Memetic search for the minmax multiple traveling salesman problem with single and multiple depots

Pengfei He and Jin-Kao Hao *

LERIA, Université d'Angers, 2 Boulevard Lavoisier, 49045 Angers, France

European Journal of Operational Research; Nov. 2022
https://doi.org/10.1016/j.ejor.2022.11.010

Abstract

The minmax multiple traveling salesman problem with single depot (the minmax mTSP) or multiple depots (the minmax multidepot mTSP) aims to minimize the longest tour among a set of tours. These two minmax problems are useful for a variety of real-life applications and typically studied separately in the literature. We propose a unified memetic approach to solving both cases of the minmax mTSP and the minmax multidepot mTSP. The proposed algorithm features a generalized edge assembly crossover to generate offspring solutions, an efficient variable neighborhood descent to ensure local optimization as well as an aggressive post-optimization for additional solution improvement. Extensive experimental results on 77 minmax mTSP benchmark instances and 43 minmax multidepot mTSP instances commonly used in the literature indicate a high performance of the algorithm compared to the leading algorithms. Additional experimental investigations are conducted to shed light on the rationality of the key algorithmic ingredients.

Keywords: Traveling salesman; Combinatorial optimization; Heuristics; Minmax; Multidepot.

1 Introduction

In the popular traveling salesman problem (TSP), a salesman needs to visit a set of cities exactly once and returns to the starting city (or the depot) while minimizing the traveling distance. The multiple traveling salesmen problem (mTSP) involves multiple salesmen who visit the given cities starting from and

* Corresponding author.
Email addresses: `pengfeihe606@gmail.com` (Pengfei He),
`jin-kao.hao@univ-angers.fr` (Jin-Kao Hao).

ending at the depot. The mTSP is a suitable model for a number of practical real-life applications related to robotics [12], transportation [2,23,24], precision agriculture [13,43] and unmanned aerial vehicles [5,31].

The mTSP can be described as follow. Let $\mathcal{G}=(\mathcal{V},\mathcal{A})$ be an edge-weighted graph, where $\mathcal{V} = \{0, 1, \ldots, n\}$ is the vertex set with 0 being the starting-ending city (depot) and $\mathcal{N} = \{1, \cdots, n\}$ representing n other cities and $\mathcal{A}$ is the set of arcs (edges). Let $\mathcal{C} = (c_{ij})$ be a non-negative cost (distance) matrix associated with $\mathcal{A}$, which satisfies the triangle inequality ($c_{ij} + c_{jk} > c_{ik}$ for all $i, j, k \in \mathcal{V}$ and $i \neq j \neq k$). The matrix $\mathcal{C}$ is said to be symmetric when $c_{ij} = c_{ji}, (i, j) \in \mathcal{A}$ and asymmetric otherwise. The basic mTSP is to partition the set of cities $\mathcal{N}$ into m distinct Hamiltonian tours $\{r_1, r_2, \ldots, r_m\}$ starting at the depot (vertex 0), such that 1) each tour r_k ($k \in \{1, 2, \ldots, m\}$) includes at least two vertices, and 2) an objective function is minimized. From an application perspective, one of the following minimization objectives is considered in the literature: 1) the minsum mTSP which minimizes the total traveling distance of the m tours [46], and 2) the minmax mTSP which minimizes the longest tour among the m tours [15].

It is known for a long time that the minsum mTSP can be conveniently transformed to the TSP [20,40]. Recently, it was shown that this transformation approach is quite powerful and able to effectively solve the existing minsum mTSP benchmark instances by leading TSP methods [17]. On the other hand, the situation is different for the minmax mTSP for which a number of dedicated methods have been developed (see the review of Section 2). In this work, we focus on the minmax mTSP including both cases of single depot and multiple depots.

The minmax mTSP with single depot can be used to formulate many applications. Meanwhile, there are other situations where multiple depots need to be considered. For example, in humanitarian logistics, several depots are deployed in different locations to ensure an efficient delivery of relief supplies to specific places [9]. The minmax multidepot vehicle routing problem was first proposed to formulate such applications, where the objective is to minimize the longest tour [10]. If the capacity constraint is ignored, the problem becomes the minmax multidepot mTSP [10,52]. Clearly, the minmax multidepot mTSP generalizes the minmax mTSP and has interesting applications such as allocating targets to unmanned vehicles [41] and allocating computer networks resources where the objective is to minimize the maximum latency between a server and a client [52].

Due to the practical relevance and computational challenge of these mTSP problems, a number of solution methods have been developed. According to the review of Section 2, the existing methods are based on general frameworks such as evolutionary algorithms, bio-inspired methods and local searches. These methods have contributed to continually improve the state-of-the-art of solving these problems. Meanwhile, their performances vary typically according to the difficulty and scale of the problem instances. Moreover, existing methods have been developed for either the minmax mTSP or the minmax multidepot mTSP. In this work, we present a unified population-based memetic algorithm (MA) able to effectively deal with both the minmax mTSP and the minmax multidepot mTSP. The contributions of this work are summarized as follows.

- The proposed MA algorithm features several complementary search components. First, it integrates a generalized edge assembly crossover to generate offspring solutions, which is inspired by the well-known EAX crossover for the TSP [34,35]. Second, MA uses an efficient variable neighborhood descent (with streamlining techniques) to improve offspring solutions. Third, the algorithm adopts an aggressive post-optimization procedure to further optimize some particularly promising offspring solutions.
- The MA algorithm reports record-breaking best results (new upper bounds) for a number of benchmark instances commonly used in the literature. These new results are useful for future research on these problems and performance assessments of new algorithms.
- We provide the code of the proposed algorithm, which can be used by researchers and practitioners to solve various problems that can be recast to the minmax mTSP or the minmax multidepot mTSP.

Next section provides a literature review of the studies on solving the minmax mTSP and the minmax multidepot mTSP. Section 3 provides a detailed description of the MA algorithm. Section 4 is dedicated to computational results on benchmark instances and comparisons with the literature. Key components of the algorithm are investigated in Section 5. Section 6 draws conclusions and discusses research perspectives.

2 Literature review

We review the representative heuristic algorithms for the minmax mTSP and the minmax multidepot mTSP. Given that the minmax mTSP was introduced much earlier than the minmax multidepot mTSP (1995 vs. 2009), there are more studies on the minmax mTSP than on the minmax multidepot mTSP.

2.1 The minmax mTSP

The minmax mTSP was introduced in 1995 by França et al. [15]. Since then many studies have been devoted to the problem. Comprehensive surveys about the applications, solution approaches and taxonomy are available in [7,12]. In this section, we focus on the most recent and representative heuristics for the problem.

Population-based metaheuristics have been presented for solving the minmax mTSP. Carter and Ragsdale [11] proposed a genetic algorithm in 2006. The algorithm was based on a two-part chromosome representation and applied classic TSP crossover operators to generate offspring solutions. One year later, Brown et al. [8] introduced another genetic algorithm, which adopted a two-part chromosome representation with real values. In 2009, Singh and Baghel [44] showed a grouping genetic algorithm, which features a new chromosome representation and a concise crossover operator such that the most promising tour (the shortest) from the parents was inherited by the offspring. In 2013, Yuan et al. [54] presented a crossover operator based on the two-part chromosome representation of [11]. In 2017, Wang et al. [53] investigated a memetic algorithm based on sequential variable neighborhood descent (MASVND) and the crossover operator of [44]. Computational experiments on 31 instances with 51-1173 cities and 3-20 tours indicated MASVND was competitive compared to other algorithms, especially for instances with a large number of cities. In 2021, Karabulut et al. [22] introduced an evolution strategy algorithm (ES), where a self-adaptive Ruin and Recreate heuristic was employed to generate offspring solutions. ES reported excellent results by improving 14 best-known solutions with 51-1173 cities and 3-30 tours among 51 minmax mTSP instances. One notices that these algorithms are based on crossover operators that focus on cities and tours, contrary to powerful TSP crossovers such as EAX [34,35] that focus on how to inherit set of edges (subtours) from parents to offspring solutions.

Swarm intelligence algorithms have been studied for solving the minmax mTSP, which showed good performances. In 2015, Pandiri and Singh [38] presented two bio-inspired algorithms (ABC and IWO). The IWO algorithm delivered excellent results and updated 12 best results reported in [8,11,44,54] for the 25 tested instances. Additional studies on swarm intelligence algorithms for the minmax mTSP were presented in [27,57]. However, they are less competitive compared to the best algorithms such as ES [22] and IWO [38].

Neighborhood-based local optimization has also been investigated for solving the minmax mTSP. In 2015, Soylu [45] presented a general variable neighborhood search algorithm based on several move operators including 2-opt and or-opt moves. Experimental results indicated a good performance of the algorithm, though it is less competitive compared to the IWO algorithm [38]. In 2022, He and Hao [17] introduced a hybrid search algorithm with neighborhood reduction (HSNR), which uses tabu search to explore the Insert and Cross-exchange neighborhoods for inter-tour optimization and the leading TSP heuristic EAX [35] for intra-tour optimization. HSNR achieved a remarkable performance by updating the best-known solutions for 15 out of the 41 popular benchmark instances (with 51-1173 cities and 3-30 tours). Additional results were reported on a new set of 36 large instances with 1379-5915 cities and 3-20 tours. Also in 2022, Zheng et al. [56] proposed an effective iterated two-stage heuristic algorithm (ITSHA), which combines a clustering-based random greedy initialization procedure and a variable neighborhood search with three move operators (2-opt, Insert and Swap). Experimental results indicated that ITSHA obtained a good performance by improving 22 upper bounds among 44 instances.

Among the reviewed studies, five algorithms (IWO [38], MASVND [53], ES [22], HSNR [17] and ITSHA [56]) hold the best-known results for the minmax mTSP on the benchmark instances. Thus, these methods serve as the main reference algorithms for our comparative study in this work.

2.2 The minmax multidepot mTSP

In 2009, Carlsson et al. [10] introduced the minmax multidepot vehicle routing problem with unbounded vehicle capacity. Interestingly, this problem is strictly equivalent to the minmax multidepot mTSP studied in this work. To solve the problem, Carlsson et al. presented a linear programming based heuristic. In 2013, Narasimha et al. [36] exposed an ant colony optimization algorithm for the problem and showed interesting computational results on 11 test instances. Later in 2015, Wang et al. [52] proposed two highly effective heuristics (MD and VNS) for solving the problem. The MD algorithm consists of three stages: (1) the multidepot problem is transformed to a single depot problem, which is then solved; (2) the longest tour is improved with TSP heuristics; (3) all tours are improved by exchanging cities between tours. The VNS algorithm combines variable neighborhood search with the powerful LKH solver [19]. Computational results on a new set of 43 instances with 10-500 cities and 3-20 tours indicated a high performance of these heuristics. Among the reviewed studies, the latest MD and VNS algorithms in [52] represent the best ones for solving the minmax multidepot mTSP (i.e., the minmax multidepot vehicle routing problem with unbounded capacity).

One notices that until now, the minmax mTSP and the minmax multidepot mTSP have been studied separately, even if they are tightly related. In this paper, we present a unified memetic search approach to handle both problems.

3 Problem solving methodology

Memetic search is a general hybrid search framework based on population-based genetic search and neighborhood-based local optimization [37]. The basic rationale behind this approach is take advantage of these two complementary search strategies [16]. Indeed, population-based search offers more facilities for exploration while local optimization provides convenient means for exploitation. A suitable combination of these two types of methods would lead to a good balance between exploitation and exploration of the search process.

Population-based evolutionary algorithms have been successfully applied to the TSP [34,35], capacitated vehicle routing problem (CVRP) [32,49] and its variants [33,50,29,51,42]. In this work, we present an original memetic algorithm (MA) for solving both the minmax mTSP and the minmax multidepot mTSP. The algorithm integrates a population initialization procedure, a generalized edge assembly crossover (mEAX), a variable neighborhood descent (VND), a post-optimization and a population management procedure. Among these search components, we identify the mEAX crossover and the post-optimization as the most innovative while VND features a streamlining techniques to accelerate its search.

The general scheme of the MA algorithm is illustrated in Algorithm 1. The algorithm starts with a population of initial solutions (or individuals) generated by the population initialization procedure (Line 2, Algorithm 1). After recording the best solution φ^* found so far (Line 3), the algorithm performs a number of generations to evolve the population (Lines 4-15). For this, it applies the dedicated mEAX crossover (Line 6) to combine two random parent solutions, yielding γ (a parameter) new offspring solutions. Then each offspring solution is first improved by the VND procedure (Line 8) and then conditionally further improved by the post-optimization (Lines 9-12). The post-optimization is applied only to elite offspring solutions with a quality better than the best recorded solution φ^*. Finally, each improved offspring solution is considered by the population management procedure to update the population (Line 13). The algorithm stops and returns the best solution found φ^* when a predefined stopping condition is reached, which is either a maximum cutoff time or a maximum number of iterations. In the later case, one iteration corresponds to one call to the (expensive) VND procedure at Line 8 of Algorithm 1.

3.1 Generation of the initial population

The MA algorithm starts its search from a population $\mathcal{P}$ of μ initial solutions. The construction process of each solution is composed of three steps. First, m tours are initialized with the depot. For the minmax multidepot mTSP, each salesman is located at one of the depots, and the tour is initialized by its corresponding depot. Second, a random unassigned city is selected and inserted into the shortest tour at the position with the least length increase of this tour. When all cities are assigned, a feasible solution is obtained. Finally, the solution is improved by the VND procedure (Section 3.3) and then added into the population. The initialization procedure stops when μ solutions are obtained.

Algorithm 1: Pseudo code of the memetic algorithm

```
Input: Problem instance $I$ with a minimization objective $f$, population
       size $\mu$, number of offspring $\gamma$;
Output: The best solution $\varphi^*$ found;
1  begin
2      $\mathcal{P} = \{\varphi_1, \varphi_2, \cdots, \varphi_\mu\} \leftarrow PopulationInitial\ (I)$; /* Section 3.1 */
3      $\varphi^* \leftarrow \arg\min \{f(\varphi_i) : i = 1, 2, \cdots, \mu\}$; /* $\varphi^*$ records the best
         solution found */
4      while Stopping condition is not met do
5          $\{\varphi_A, \varphi_B\} \leftarrow ParentSelection(\mathcal{P})$; /* Random parent selection
             */
6          $\{\varphi_O^1, \varphi_O^2, \cdots, \varphi_O^\gamma\} \leftarrow mEAX(\varphi_A, \varphi_B, \gamma)$; /* To generate $\gamma$
             offspring solutions, Section 3.2 */
7          for $i = 1$ to $\gamma$ do
8              $\varphi_O^i \leftarrow VND(\varphi_O^i)$; /* To improve each offspring
                 solution, Section 3.3 */
9              if $f(\varphi_O^i) < f(\varphi^*)$ then
10                 $\varphi_O^i \leftarrow PostOptimization(\varphi_O^i)$; /* To further improve
                     each elite offspring solution, Section 3.4 */
11                 $\varphi^* \leftarrow \varphi_O^i$;
12             end
13             $\mathcal{P} \leftarrow PoolUpdating(\mathcal{P}, \varphi_O^i)$; /* Section 3.5 */
14         end
15     end
16     return $\varphi^*$;
17 end
```

3.2 The mEAX crossover based on edge assembly

The conventional edge assembly crossover operator (EAX) was first presented for solving the TSP [34,35]. It was subsequently applied to the CVRP [32] and the vehicle routing problem with time windows (VRPTW) [33]. In this work, we introduce mEAX, which generalizes the idea of EAX to the minmax mTSP and the minmax multidepot mTSP. It is worth noting that these mTSPs are different from the TSP, CVRP and VRPTW. As such, the proposed mEAX crossover must consider the specific features of the minmax mTSP problems.

Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, a candidate solution φ for the minmax mTSP or minmax multidepot mTSP corresponds to a partial graph $\mathcal{G}_\varphi = (\mathcal{V}, \mathcal{E}_\varphi)$, where $\mathcal{E}_\varphi$ is the set of edges traversed by φ. Let φ_A and φ_B be two parent solutions. Let $\mathcal{G}_A = (\mathcal{V}, \mathcal{E}_A)$ and $\mathcal{G}_B = (\mathcal{V}, \mathcal{E}_B)$ be the corresponding partial graphs. The proposed mEAX crossover consists of the following six steps (see Algorithm 2 for the general procedure and Fig. 1 for an illustrative example).

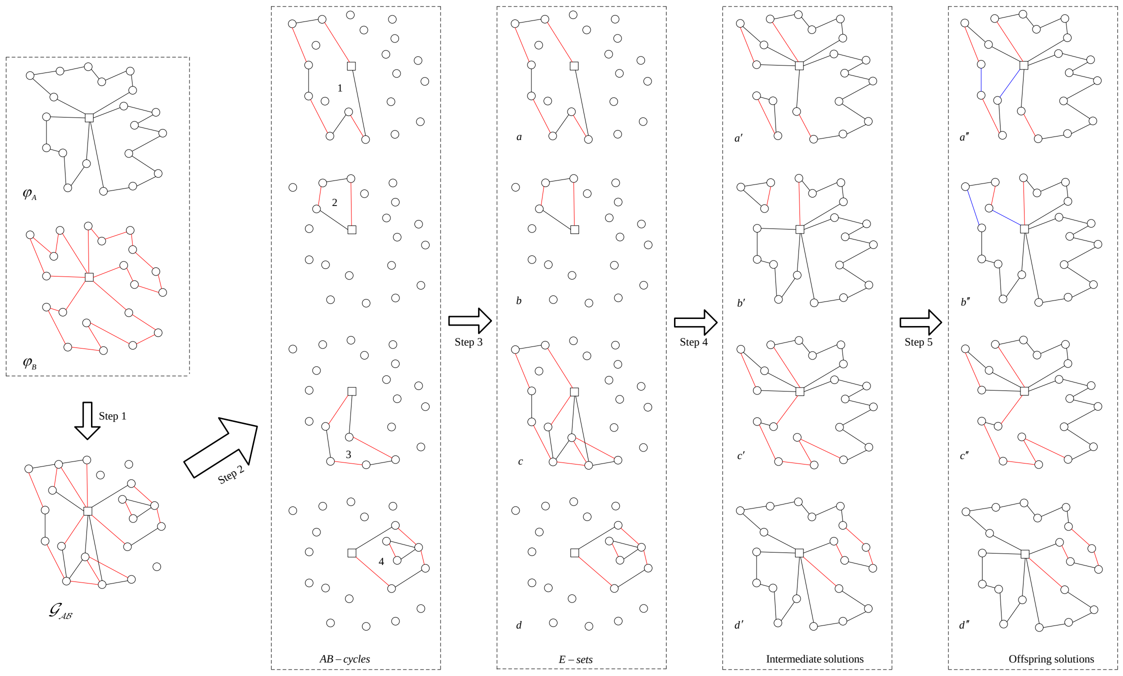


Fig. 1. Illustration of the mEAX crossover steps of the minmax mTSP

```
Algorithm 2: Procedures of mEAX for the minmax mTSP
Input: Parent $\varphi_A$ and $\varphi_B$, parameter $\gamma$;
Output: $\gamma$ offspring solutions;
begin
    Construct $\mathcal{G}_{\mathcal{AB}} = (\mathcal{V}, (\mathcal{E}_{\mathcal{A}} \cup \mathcal{E}_{\mathcal{B}}) \backslash (\mathcal{E}_{\mathcal{A}} \cap \mathcal{E}_{\mathcal{B}}))$;
    Generate AB-cycles by decomposing $\mathcal{G}_{\mathcal{AB}}$;
    Construct E-sets from AB-cycles with the block strategy;
    Generate intermediate solutions according to E-sets and a basic
      solution;
    Split giant tours in intermediate solutions for the minmax multidepot
      mTSP;
    Eliminating isolated subtours in intermediate solutions to generate
      feasible solutions;
    Select $\gamma$ best feasible solutions;
    return $\gamma$ offspring solutions;
end
```

(1) Creation of a joint graph $\mathcal{G}_{\mathcal{AB}}$. From the parent solutions φ_A and φ_B, the joint graph $\mathcal{G}_{\mathcal{AB}} = (\mathcal{V}, (\mathcal{E}_{\mathcal{A}} \cup \mathcal{E}_{\mathcal{B}}) \backslash (\mathcal{E}_{\mathcal{A}} \cap \mathcal{E}_{\mathcal{B}}))$ is constructed by the symmetric difference of $\mathcal{E}_{\mathcal{A}}$ and $\mathcal{E}_{\mathcal{B}}$.

(2) Generation of *AB-cycles*. Given the joint graph $\mathcal{G}_{\mathcal{AB}}$, a number of *AB-cycles* are generated where each new *AB-cycle* is constructed as follows. A random vertex associated with its edges from $\mathcal{G}_{\mathcal{AB}}$ is selected to initialize an *AB-cycle*. Then the edges of $\mathcal{E}_{\mathcal{A}}$ and $\mathcal{E}_{\mathcal{B}}$ are traced alternatively

to extend the ongoing *AB-cycle*. When the add of a new edge leads to a closed cycle and the number of edges is even, the *AB-cycle* is formed successfully. All the edges belonging to the *AB-cycle* are removed from $\mathcal{G}_{\mathcal{AB}}$ before building the next *AB-cycle*. This process continues until $\mathcal{G}_{\mathcal{AB}} = \emptyset$ and returns all *AB-cycles* obtained.

(3) Generation of *E-sets*. From the set of *AB-cycles*, the block strategy is used to generate the so-called *E-sets*. If two *AB-cycles* share at least one vertex (e.g., *AB-cycles* 1 and 3 in Fig. 1), these two cycles are combined to generate the *E-set*. In the example of Fig. 1, the four *AB-cycles* should be combined to form one single *E-set* since the depot is shared. However, for illustrative purpose of steps 4 and 5 blow, we suppose there are four *E-sets* as showed in Fig. 1.

(4) Generation of intermediate solutions. For each *E-set* (say $\mathcal{E}_i$), an intermediate solution φ'_i is created based on φ_A by removing from it the edges of $\mathcal{E}_{\mathcal{A}}$ shared with $\mathcal{E}_i$ and adding the edges of $\mathcal{E}_{\mathcal{B}}$ shared with $\mathcal{E}_i$, i.e., $\varphi'_i = (\mathcal{E}_{\mathcal{A}} \setminus (\mathcal{E}_i \cap \mathcal{E}_{\mathcal{A}})) \cup (\mathcal{E}_i \cap \mathcal{E}_{\mathcal{B}})$. This strategy ensures the preservation of all common edges of φ_A and φ_B in the intermediate solution. Furthermore, all edges in an intermediate solution exclusively come from the parent solutions.

(5) Elimination of giant tours. For the minmax multidepot mTSP, giant tours that visit more than one depot, may occur in intermediate solutions (e.g., the tour in the intermediate solution in Fig. 2 includes two depots represented by squares). These giant tours are split by the 2-opt* operator [39]. If a giant tour visits k depots, two new Hamiltonian tours are first generated by the 2-opt* operator, where one of the two new tours only visits one single depot while the other tour visits the remaining k-1 depots. We repeat this split operations k-1 times until k new Hamiltonian tours are generated. During the split process, the objective is to make the new tours have similar length and avoid too longer tours. For the giant tour with two depots in Fig. 2 (lower part of the intermediate solution), it includes two segments (each segment refers to the set of cities between two depots). The 2-opt* operator works as follows. Two edges (dash lines) from the two segments based on the α-nearness technique (Section 3.3.1) are replaced to create two new single depot tours such that the length of the new shorter tour is as close as possible half of the giant tour. We thus obtain two feasible tours which have similar lengths.

(6) Elimination of isolated subtours. Isolated subtours may appear in intermediate solutions (e.g., the two triangle tours in intermediate solutions a' and b' in Fig. 1). We apply the 2-opt* approach to accommodate the particular feature of our problem as follows. For each isolated subtour, its adjacency tours are defined if a vertex u is an α neighbor (Section 3.3.1) of vertex v visited by the subtour. Then, the merges of the subtour into its adjacency tours are evaluated by 2-opt* and the best merge leading to the shortest tour is performed. Once all isolated subtours are eliminated, a feasible offspring solution composed of m distinct Hamiltonian tours is

obtained (see the last sub-figure in Fig. 1).

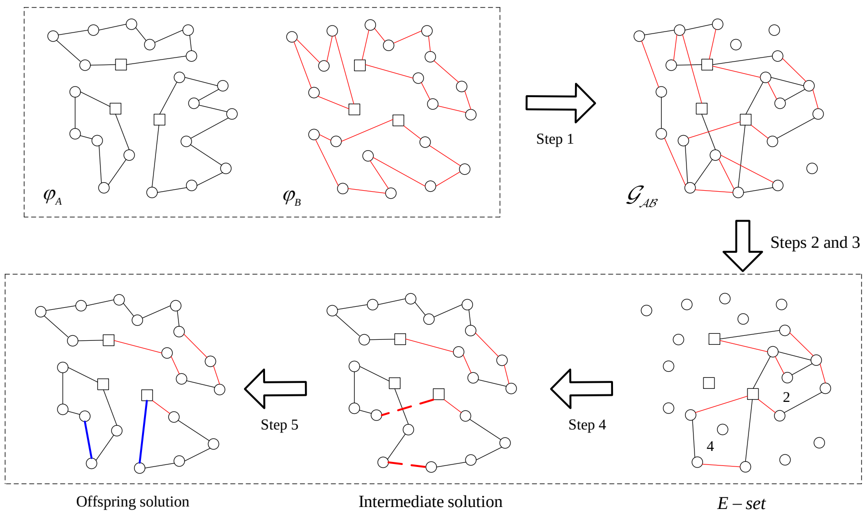


Fig. 2. Illustration of the mEAX crossover steps of the minmax multidepot mTSP

One notes that mEAX differs from EAX by the last two steps because contrary to the TSP and the CVRP, giant tours may appear in the case of the minmax multidepot mTSP.

The above mEAX process typically generates numerous offspring solutions, many of them being of bad quality and thus uninteresting. Given that the subsequent VND local optimization (Section 3.3) is time consuming, we filter out non-promising offspring solutions with a mediocre quality to retain only the γ (a parameter) best offspring solutions for solution improvement.

The mEAX crossover for the minmax mTSP and minmax multidepot mTSP follows the idea of the EAX operator initially designed for the TSP [34,35]. Meanwhile, adaptations are necessary to take into account the particular features concerning the minmax objective and the presence of possible multiple depots. The main adaptations concern the processing of giant tours and isolated subtours in intermediate solutions (steps (5) and (6)).

Our way of handling isolated subtours is similar to the technique presented in [32] where EAX is adapted to the CVRP. In [32], isolated subtours are eliminated by testing all possible combinations with the 2-opt* heuristic [39] and performing the best combination which minimizes the total traveling distance. In mEAX, since the objective is to minimize the longest tour instead of total traveling distance, the 2-opt* heuristic is applied with this specific minimization objective. Furthermore, for the minmax multidepot mTSP, giant tours which include two or more depots may occur in intermediate solutions due to the presence of multiple depots. This feature cannot be resolved by the conventional EAX operator. In mEAX, the 2-opt* based splitting strategy is introduced to split each giant tour into feasible tours while keeping all tours are as similar in length as possible. In sum, the mEAX crossover renders the idea of the EAX operator applicable to routing problems with the minmax objective.

Since a minmax mTSP solution contains $n+m-1$ edges, the space complexity of mEAX is bounded by $\mathcal{O}(n+m)$. During the first four steps, $2 \times (n+m-1)$ edges are involved, and the time complexity is bounded by $\mathcal{O}(n+m)$. In step 5, suppose that there are g giant tours and the cycle with the largest number of edges includes $|\mathcal{E}_g|$ edges, the time complexity is bounded by $\mathcal{O}(|\mathcal{E}_g| \times \alpha)$. Furthermore, suppose that there are h isolated tours and the longest tour includes $|\mathcal{E}_h|$ edges, the time complexity of step 6 is bounded by $\mathcal{O}(|\mathcal{E}_h| \times \alpha)$.

3.3 Variable neighborhood descent

Variable neighborhood descent (VND) [30] is a general local search approach which has been applied successfully to a number of routing and TSP-like problems [21,45,48,53]. VND explores local optima with several ordered neighborhoods N_θ ($\theta = 1, 2, \cdots, \theta_{max}$). VND starts its descent search from the first neighborhood and switches to the next neighborhood once a local optimum is reached. When neighborhood N_θ is examined, VND switches to the first neighborhood N_1 immediately if a better solution is found; otherwise when neighborhood N_θ ($\theta > 1$) is exhausted, VND moves to the next neighborhood $N_{\theta+1}$. Once the last neighborhood $N_{\theta_{max}}$ is exhausted and no better solution can be found, VND stops and returns the last local optimum. In this work, we use VND to exploit six neighborhoods, where two neighborhoods (2-opt* and κ-opt) are employed to solve the minmax mTSP and the minmax multidepot mTSP for the first time. To speed up neighborhood examination, two new data structures are introduced to accelerate the search process of VND.

3.3.1 Neighborhoods

The six neighborhoods adopted in this work include five inter-tour neighborhoods and one intra-tour neighborhood. Let $r(\pi)$ denote the tour containing vertex π in the incumbent solution. Let vertex δ be a neighbor of vertex π, and vertices x and y the successor of π in $r(\pi)$ and δ in $r(\delta)$, and (π_a, π_b) a substring from π_a to π_b. To avoid the examination of non-promising candidate solutions, we use the α-nearness technique [17,19] and consider, for a vertex π, only α neighbor vertices. The six neighborhoods are given by the following move operators M1-M6.

M1: If $r(\pi) \neq r(\delta)$ and $r(\pi)$ is the longest tour r_l, then remove π and place it after δ.

M2: If $r(\pi) \neq r(\delta)$ and one of them is the longest tour r_l, then, swap π and δ.

M3: If $r(\pi) \neq r(\delta)$ and one of them is the longest tour r_l, then replace (π, x) and (δ, y) by (π, y) and (δ, x).

M4: If $r(\pi) \neq r(\delta)$ and one of them is the longest tour r_l, then swap two sequencing substrings (π_a, π_b) and (δ_a, δ_b).

M5: If $r(\pi) \neq r(\delta)$ and one of them is the longest tour r_l, then swap a sequencing substring (π_a, π_b) and a reversing substring (δ_b, δ_a).

M6: This is an intra-tour optimization operator to improve a standard TSP tour. Each tour is refined by the κ-opt heuristic [25], which was previously used in several best heuristics for related routing problems [3,4,28]. In this work, the upper limit of κ is set to four.

M1 corresponds to *insertion* or *relocation*, while M2 is called *swap*. M3 is the 2-opt* inter-tour move [39]. M4 and M5 correspond to the cross-exchange operator, where two substrings from two tours are exchanged [47,17]. The cross-exchange operator generalizes M1 and M2, and has been successfully used to solve the minmax mTSP [18]. In this work, we limit the maximum length of each substring in M4 and M5 to β (a parameter).

It is worth mentioning that M6 is used for the first time in this work and M3 was independently used in [56], while the other moves were previously applied to the minmax mTSP (e.g., [18,45,53,22,52,56]). For the minmax multidepot mTSP, it is to be noted that M3 cannot be used because each salesman must start and end at the same depot. Therefore, when solving the minmax multidepot mTSP, M3 is disabled from the VND procedure. Furthermore, this is the first time that M4-M6 are adopted to solve the minmax multidepot mTSP.

3.3.2 Auxiliary data structures

In order to enhance the computational efficiency of our VND procedure, we introduce two auxiliary arrays to store useful information regarding each city.

$A1$: A one-dimensional array of length n. It stores the variation of distance of the current tour when a vertex is removed from the tour. For example, $A1[\pi]$=-100 means that if vertex π is removed from tour r_a, the length of tour r_a is shortened by 100.

$A2$: A two-dimensional array of size $n \times n$. It stores the variation of distance of the tour when vertex π is inserted after vertex δ. For example, $A2[\pi][\delta]$=100 indicates that if π is placed after δ in tour r_a, the length of tour r_a is increased by 100.

In general, a neighboring solution can be obtained from the incumbent solution by exchanging several edges. Therefore, most edges in the incumbent solution are common with its neighboring solutions. This insight has been used to design static move descriptors for several vehicle routing problems [1,6,55]. For the minmax mTSP, these two auxiliary arrays ($A1$ and $A2$) enable the VND procedure to avoid unnecessary redundant calculations. As shown in Fig. 3, city δ_b is removed from tour r_a and placed after δ_a in tour r_b. Therefore, we can easily compute the length of r'_a and r'_b as follows: $f(r'_a) = f(r_a) + A1[\delta_b]$ and $f(r'_b) = f(r_b) + A2[\delta_b][\delta_a]$. After placing δ_b after δ_a in tour r_b, only five values in $A1$ and $3 \times n$ values in $A2$ need to be updated, respectively. In general, the time complexity of updating $A1$ and $A2$ is $\mathcal{O}(n)$.

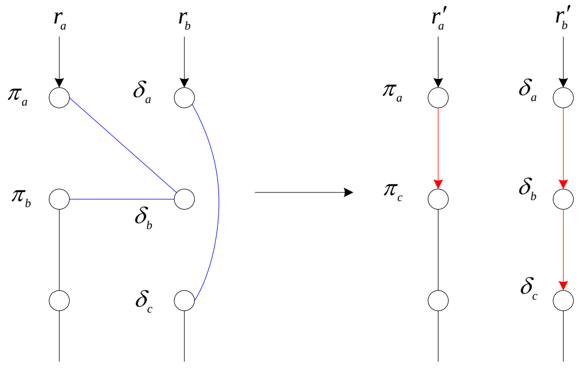


Fig. 3. Illustration of M1 move

In the VND procedure, these two auxiliary arrays are used to speed up the calculations of M1 and M2. Furthermore, the ejection chain operator, introduced in Section 3.4, also benefits from these data structures to accelerate the neighborhood examination.

3.4 Post-optimization

In addition to the above mEAX crossover and the VND procedure, the proposed MA algorithm includes an original post-optimization phase to further improve the quality of each global best offspring solution. The main purpose of the post-optimization is to perform an intensified search around each elite offspring solution to find possible still better solutions. This post-optimization phase is ensured jointly by an ejection chain operator (EC) and the conventional EAX heuristic for the TSP (denoted by EAX-TSP hereafter) [34,35].

As shown in Algorithm 3, the post-optimization applies first the EC operator to improve the solution by displacing cities among different tours. A binary array T is employed to record the tours that are modified during the EC phase, such that $T[i] = 1$ $(i = 1, ..., m)$ if the ith tour is changed by EC. Then for each modified tour, the EAX-TSP heuristic is applied to shorten its distance. When neither EC nor EAX-TSP can improve the solution, the post-optimization stops and returns the best solution.

The ejection chain approach has been used to perform inter-tour optimization

Algorithm 3: Pseudo code of the post-optimization procedure

```
Input: A solution $\varphi$;
Output: The best solution $\varphi$ found;
begin
    $fit \leftarrow$ M; /* M is a big number */
    while $fit > f(\varphi)$ do
        $fit \leftarrow f(\varphi)$;
        $< \varphi, T > \leftarrow$ EC($\varphi$); /* Ejection chain, */
        $\varphi \leftarrow$ EAX-TSP($\varphi, T$);
    end
    return $\varphi$;
end
```

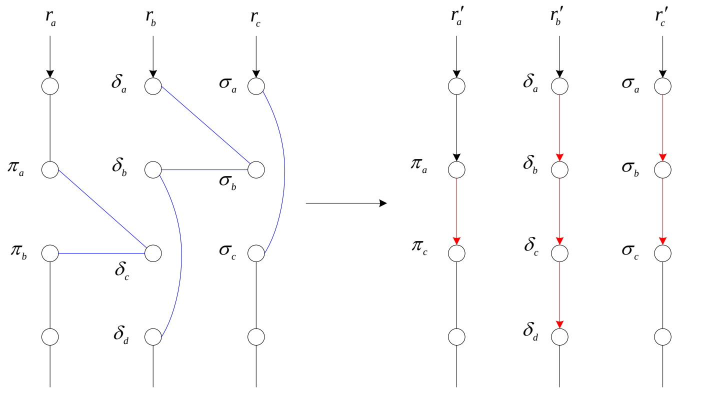


Fig. 4. Illustration of the ejection chain with two relocations

for the CVRP [1,4]. We adopt the same approach for the first time to handle the minmax mTSP. Using Fig. 4 where the incumbent solution is composed of three tours, we illustrate the EC process as follows. EC starts by greedily relocating a city δ_c from the longest tour r_a into another tour r_b. This relocation operation is followed by the relocation of another city σ_b from the extended tour r_b into another tour r_c, where r_a and r_c may be same. This process continues until a maximum number of relocation moves is reached.

The EC approach is based on the following observation. Single relocation moves between two tours may increase the length of the longest tour. For example, relocating a city from the longest tour r_a into r_b shortens r_a, but may increase tour r_b such that r_b becomes the longest tour with a distance longer than r_a. However, if we perform immediately another move to relocate a city from tour r_b into tour r_c, then it is possible that the longest tour of the solution is definitively shorten.

Eq. (1) illustrates the calculation of the move gain of an EC move based on the two auxiliary arrays introduced in Section 3.3.2, where b, c, s, t and q are indexes of r_b, r_c, the second, third and fourth longest tours, respectively.

$$\begin{aligned}
&\Delta = \max\{f(r_a^{'}), f(r_b^{'}), f(r_c^{'}), f(r_s)\} - f(r_a),\ if\ \{b,c\} \cap \{s,t\} = \varnothing \\
&\Delta = \max\{f(r_a^{'}), f(r_b^{'}), f(r_c^{'}), f(r_t)\} - f(r_a),\ if\ \{b,c\} \cap \{s,t\} = \{s\} \\
&\Delta = \max\{f(r_a^{'}), f(r_b^{'}), f(r_c^{'}), f(r_q)\} - f(r_a),\ if\ \{b,c\} \cap \{s,t\} = \{s,t\} \\
&f(r_a^{'}) = f(r_a) + A1[\delta_c] \\
&f(r_b^{'}) = f(r_b) + A2[\delta_c][\delta_b] + A1[\sigma_b] \\
&f(r_c^{'}) = f(r_c) + A2[\sigma_b][\sigma_a]
\end{aligned} \tag{1}$$

Based on the M1 move introduced in Section 3.3.1, if the number of relocation is 1, the time complexity is $\mathcal{O}(n \times \alpha)$. When we continue the EC move by performing the second relocation, the time complexity becomes $\mathcal{O}((n \times \alpha)^2)$. To keep the time complexity at an acceptable level, we limit the number of relocations to 2 in this work.

One notes two differences between the EC move applied to the CVRP [1,4] and the EC move applied in this study. First, the EC operator in our case does not need to consider the capacity constraint. Second and more importantly, even if the move gain of an EC move can be obtained in $\mathcal{O}(1)$ time in both cases, the practical computation in our case is more complicated. Indeed, for the CVRP, the move gain is simply obtained by adding up the values of $A1$ and $A2$, which themselves can be computed efficiently with the static move descriptor technique [1]. In our case, the static move descriptor is no more available and furthermore as shown in Eq. (1), the EC move gain evaluation needs to consider the second, third and fourth longest tours.

After the EC phase, the EAX-TSP heuristic[1] is triggered to optimize each individual tour that has been modified by the EC procedure. Each EAX-TSP optimization stops when the difference between the fitness of the best solution and the average fitness of individuals in the population is less than 0.01. The reason to choose the EAX-TSP heuristic is that it can effectively optimize each tour to being optimal or near-optimal in a very short time.

3.5 Population updating

The population updating mechanism is known to be a key component of successful memetic algorithms [16]. The proposed algorithm adopts the variable population scheme presented in [49,51].

[1] The code of EAX-TSP is available at: https://github.com/sugia/GA-for-TSP

The population $\mathcal{P}$ contains between μ and $\mu + \lambda$ individuals, where parameter μ is the minimum size and parameter λ is the generation size. Unlike [51], clone solutions are not permitted to join the population. In each generation of MA, offspring solutions φ_O^i are progressively added to the population (Line 13, Algorithm 1). Once the population reaches $\mu + \lambda$ individuals, the survivors selection is used to eliminate λ individuals based on their contributions to the diversity of the population. The biased fitness of each individual is calculated with respect to its fitness and diversity rank in the population.

Furthermore, if the global best solution is not improved during η consecutive iterations, the algorithm is considered to be stagnating in deep local optima. In this case, diversity is introduced into the population as follows. The survivors selection phase is triggered to reduce the number of individuals in $\mathcal{P}$ to μ individuals. Then, $\mu/2$ individuals of the population are randomly and uniformly selected and replaced by new solutions generated by the initial population procedure of Section 3.1.

4 Computational Results and Comparisons

This section is dedicated to an extensive performance assessment of the MA algorithm on popular benchmark instances.

4.1 Benchmark instances

Three sets of benchmark instances are used in our experiments: Sets $\mathbb{S}$ and $\mathbb{L}$ for the minmax mTSP and Set $\mathbb{M}$ for the minmax multidepot mTSP.

Set $\mathbb{S}$: This set includes 41 small and medium-sized instances with 51-1173 cities and 3-30 tours. These instances were introduced in [53,8,11] and used in [17,22,38,53,56].

Set $\mathbb{L}$: This set consists of 36 large-sized instances with 1379-5915 cities and 3-20 tours, which were introduced in [17].

Set $\mathbb{M}$: This set includes 43 instances with 10-500 cities and 3-20 tours, which were introduced in [52] [2].

These benchmark instances and the solution certificates for them obtained by the MA algorithm are available online [3].

[2] The benchmark instances of the minmax multidepot mTSP is available at https://drum.lib.umd.edu/handle/1903/18710

[3] https://github.com/pengfeihe-angers/minmax-mTSP.git

4.2 Experimental protocol and reference algorithms

Parameter setting. The MA algorithm has six parameters: population size μ, generation size λ, number of the best offspring solutions γ, neighborhood reduction parameter α, substring size β, maximum consecutive iterations (η) without an improvement. To calibrate these parameters, we employed the automatic parameters tuning package Irace [26]. The tuning was performed on 8 instances with 150-1655 cities for the minmax mTSP and 10 instances with 100-500 cities for the minmax multidepot mTSP. The tunning budget was set to be 2000 runs. Table 1 shows, for each parameter, the interval of values tested by Irace, and the best value returned by the method. For the experiments presented hereafter, we used consistently these parameter values, which can be considered to be the default setting of the MA algorithm.

Table 1
Parameters tuning results.

Parameters	Section	Description	Considered values	Final values	
				mTSP	multidepot mTSP
μ	3.1	population size	{10,15,20,25,30}	30	30
λ	3.5	generation size in $\mathcal{P}$	{0,5,15,20,25,30}	20	15
γ	3.2	number of the best offspring	{1,2,3,4,5,6,7}	1	5
α	3.3.1	neighborhood reduction	{10,15,20,25,30}	15	10
β	3.3.1	substring size	{1,2,3,4,5,6,7}	4	7
η	3.5	maximum iterations without improvement	{2000,4000,6000,8000,10000,12000}	4000	2000

Reference algorithms. For the minmax mTSP, five algorithms (IWO [38], MASVND [53], ES [22], HSNR [17] and ITSHA [56]) represent the state-of-the-art for solving the problem. In [17], the authors thoroughly assessed HSNR, IWO and MASVND on the same computing platform as used in this work. The executable code of ES [22] and the source code of ITSHA [56] were kindly provided by their authors. For the minmax multidepot mTSP, the MD and VNS algorithms from [52] are the leading algorithms in the literature (their codes are unavailable). Thus, the results of these algorithms (obtained on a computer with an Intel Pentium CPU of a 2.2 GHz processor) are used as reference values to evaluate the performance of the MA algorithm. According to [52], both MD and VNS terminate after five consecutive iterations without an improvement.

Experimental setting and stopping condition. The MA algorithm was written in C++ and compiled using the g++ complier with the -O3 option [4]. All experiments were conducted, like [17], on a computer with a Xeon E5-2670 v2 processor of 2.5GHz CPU and 8GB RAM running Linux.

To make the comparison as fair as possible, for the minmax mTSP, we ran 20 times our MA algorithm and the codes of the reference algorithms ES and ITSHA on our machine to solve each instance under the cutoff limit of

[4] https://github.com/pengfeihe-angers/minmax-mTSP.git

$(n/100)\times 4$ minutes per run (this is the same stopping condition used in [17] to assess IWO, MASVND and HSNR). For the other reference algorithms (IWO, MASVND, HSNR), we cite the results reported in [17], which were obtained on the same computer as used in this work. For the minmax multidepot mTSP, MA terminates when it reaches a maximum of 30,000 iterations.

4.3 Computational results and comparison

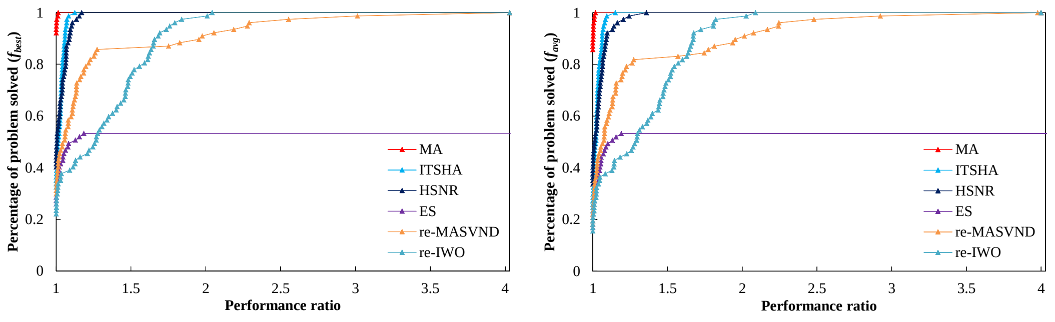
To compare MA and the reference algorithms, we report a summary of the results in Table 2 and the detailed results in the Appendix. The 'BKS' values show the best-known results compiled from the literature. To check the statistically significant difference between MA and each reference algorithm, the Wilcoxon signed-rank test is applied. With a confidence level of 0.05, a *p-value* lower than 0.05 indicates a significant difference. Furthermore, a popular benchmark tool, performance profile [14], is used to compare the algorithms in a visual way. Given a set of algorithms $\mathcal{S}$ over a set of instances $\mathcal{Q}$, the performance ratio is defined by $r_{s,q} = \frac{f_{s,q}}{min\{f_{s,q:s\in\mathcal{S}}\}}$, which represents the performance of algorithm s on instance q compared to the best performance by any approach on q. If algorithm s fails to solve an instance q, $r_{s,q} = +\infty$. The performance function of algorithm s is defined by $\mathcal{Q}_s(\tau) = \frac{|q\in\mathcal{Q}|r_{s,q}\le\tau|}{|\mathcal{Q}|}$, which calculates the fraction of instances that algorithm s can reach with at most τ many times the cost of the best algorithm.

Table 2
Summary of comparative results between MA and reference algorithms on the three sets of 120 instances. Sets $\mathbb{S}$ and $\mathbb{L}$ for the minmax mTSP and Set $\mathbb{M}$ for the minmax multidepot mTSP.

Instances	Pair algorithms	f_{best}				f_{avg}			
		#Wins	#Tiers	#Losses	*p-value*	#Wins	#Tiers	#Losses	*p-value*
Set $\mathbb{S}$ (41)	MA vs. BKS	16	23	2	1.37E-03	-	-	-	-
	MA vs. ITSHA [56]	19	21	1	2.93E-04	23	13	5	2.25E-04
	MA vs. HSNR [17]	18	22	1	8.37E-04	22	13	6	7.51E-04
	MA vs. ES [22]	20	20	1	9.22E-05	28	9	4	7.61E-06
	MA vs. re-MASVND	20	20	1	1.23E-04	27	13	1	6.53E-06
	MA vs. re-IWO	24	17	0	1.82E-05	29	12	0	3.52E-06
Set $\mathbb{L}$ (36)	MA vs. BKS	28	3	5	2.50E-06	-	-	-	-
	MA vs. ITSHA [56]	32	3	1	5.91E-07	36	0	0	1.68E-07
	MA vs. HSNR [17]	28	3	5	2.50E-06	29	2	5	3.18E-06
	MA vs. re-MASVND	33	3	0	5.39E-07	35	1	0	2.48E-07
	MA vs. re-IWO	36	0	0	1.68E-07	36	0	0	1.68E-07
Set $\mathbb{M}$ (43)	MA vs BKS	39	1	3	3.15E-08	-	-	-	-
	MA vs. MD [52]	40	1	2	2.28E-08	-	-	-	-
	MA vs. VNS [52]	41	0	2	2.04E-08	-	-	-	-

4.3.1 Results on the minmax mTSP

The comparative results on the 77 instances of Sets $\mathbb{S}$ and $\mathbb{L}$ for the minmax mTSP are shown in Tables A.1 and A.2 with the summary information in Table 2, where re-IWO and re-MASVND are the re-implemented IWO [38] and MASVND [53] algorithms in [17]. According to these tables, the MA algorithm outperforms the five reference algorithms by achieving the best result for the vast majority of the instances. MA improves the best-known solutions of 44 instances, and matches the best-known solutions of 26 other instances. Furthermore, in terms of the average result, MA also outperforms the reference algorithms. Specifically, for $n \leq 100$, MA and the reference algorithms perform similarly in terms of f_{best}. For $n \geq 150$, MA outperforms the other algorithms (improvement gap up to 8.72%). As the number of cities increases, the difference becomes more significant, especially for the instances with few tours (e.g., $m = 3, 5$). The small *p-values* from the Wilcoxon signed-rank test confirm the statistically significant difference between MA and the reference algorithms for the best and average values.



(a) Best (b) Avg.

Fig. 5. The minmax mTSP: performance profiles of MA and the five reference algorithms on the 77 instances of Sets $\mathbb{S}$ and $\mathbb{L}$.

In Fig. 5, the average gap of MA and the five reference algorithms are analyzed through their performance profiles. Intuitively, MA dominates the reference algorithms in terms of both the best and average results. Indeed, MA has a much higher $\mathcal{Q}_s(1)$, meaning that it finds better or equal results for nearly all instances. Furthermore, MA reaches 1 firstly, which indicates MA has a higher robustness.

4.3.2 Results on the minmax multidepot mTSP

Tables 2 and A.3 show the results of MA as well as the two reference algorithms (MD [52] and VNS [52]) on the 43 instances of Set $\mathbb{M}$. According to the results, MA dominates the reference algorithms by providing 39 new best-known solutions. Only for three instances, MA obtains slightly worse results. The small *p-values* ($\ll 0.05$) also confirm the statistically significant differences between MA and the compared algorithms. The performance profiles in Fig. 6 illustrate that MA has a much higher $\mathcal{Q}_s(1)$ and $\mathcal{Q}_s(\tau)$ reaches 1 first. Therefore, MA competes very favorably with the best existing algorithms for solving the minmax multidepot mTSP.

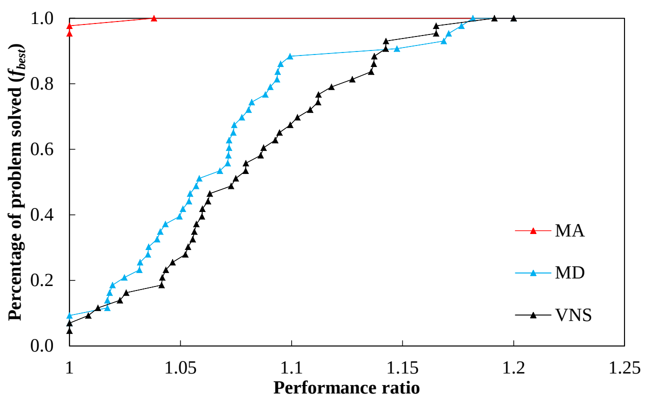


Fig. 6. The minmax multidepot mTSP: performance profiles of the MA and two reference algorithms on 43 instances of Set $\mathbb{M}$.

Given that MA, MD and VNS were run on different computers and reported results of different qualities, it is not straightforward to make a fair comparison of their computation time. One observes that for the 18 instances where the time information is available for the compared algorithms, MA is able to reach the best-known results with a time of the same order of magnitude compared to MD and VNS, and then continue to improve these results during the rest of its execution.

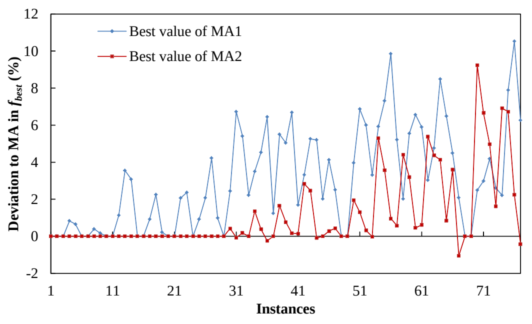
According to the results of Sections 4.3.1 and 4.3.2, we conclude that the MA algorithm is highly effective for solving the minmax mTSP and the minmax multidepot mTSP compared to the best performing algorithms.

5 Additional experiments

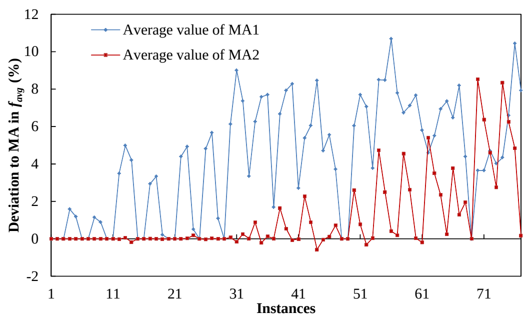
The computational results and comparisons with the existing algorithms on three sets of instances illustrated the high effectiveness and efficiency of the MA algorithm. In this section, we assess the contributions of two key components: the mEAX crossover, the post-optimization and two new neighborhood operators. Experiments are performed to compare MA and its variants where the assessed components are disabled. Furthermore, we investigate the long-term convergence behavior of the MA algorithm under a relaxed timing condition. The experiments reported in this section are based on the minmax mTSP.

5.1 Benefits of the mEAX crossover and the post-optimization procedure

To study the benefits of the mEAX operator and the post-optimization procedure, we created two MA variants MA1 and MA2 as follows. For MA1, we removed the mEAX operator (i.e., lines 5 and 6) in Algorithm 1 and replaced γ by μ in line 7. To make sure that MA1 consumes the given time budget effectively like MA, we repetitively re-start the algorithm until the

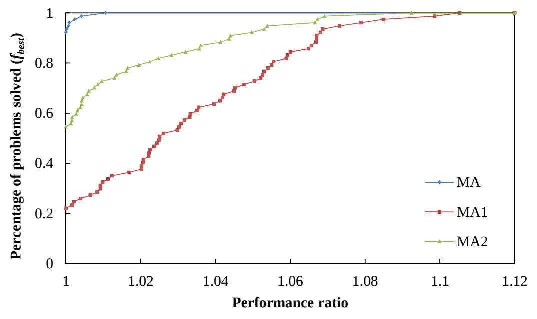


(a) Best

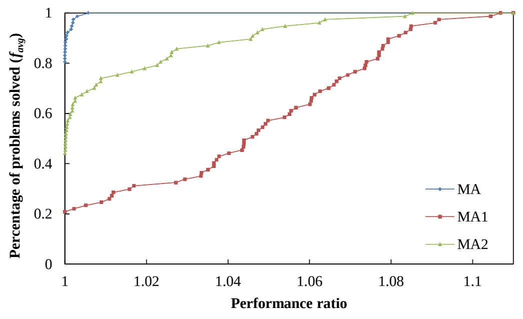


(b) Avg.

Fig. 7. Comparative results of MA with the variants MA1 (without mEAX) and MA2 (without the post-optimization) on the 77 instances of Sets $\mathbb{S}$ and $\mathbb{L}$.



(a) Best



(b) Avg.

Fig. 8. Performance profiles of MA and its variants MA1 and MA2 on the 77 instances of Sets $\mathbb{S}$ and $\mathbb{L}$.

time limit is reached. In other words, MA1 uses the VND procedure and the post-optimization to improve the solutions of the population within the given time limit. For the variant MA2, we just removed the post-optimization (i.e., lines 9-12) in Algorithm 1.

We ran MA1 and MA2 under the same condition of Section 4.2 to solve the 77 instances of Sets $\mathbb{S}$ and $\mathbb{L}$. The results are summarized in Figs. 7 and 8.

Fig. 7 shows the deviations of the two variants MA1 and MA2 compared to MA (the reference line) in terms of the best results and the average results. From these figures, we can make the following observations.

First, the results of MA1 indicate that removing mEAX deteriorates considerably the performance of the MA algorithm on a large majority of the tested instances in terms of the best and average results. The deterioration is more significant on large instances than on small instances. These results confirm the critical role of the proposed mEAX crossover.

Second, the results of MA2 indicate that the post-optimization doesn't really impact the performance of the MA algorithm on the first 29 small instances ($n \leq 318$). However, disabling this component deteriorates much MA's performance on many larger instances with $n > 318$. These results demonstrate the positive contributions of the post-optimization for solving large (and hard) instances.

Third, though both mEAX and post-optimization contribute to the high performance of the MA algorithm, the mEAX crossover plays a more general and more significant role compared to the post-optimization component.

To further study the MA1 and MA2 variants, Fig. 8 shows the performance profiles of MA, MA1 and MA2 based on their best results and their average results. We observe that MA dominates its two variants in terms of the best and average values. MA has a much higher $\mathcal{Q}_s(1)$ compared with MA1 and MA2. Indeed, MA reaches $\mathcal{Q}_s(\tau) = 1$ firstly, much earlier than the two variants, which indicates a higher robustness of the MA algorithm. In summary, these experiments confirm that both the mEAX crossover and the post-optimization contribute positively to the performance of MA, while the post-optimization component is especially useful for solving large instances.

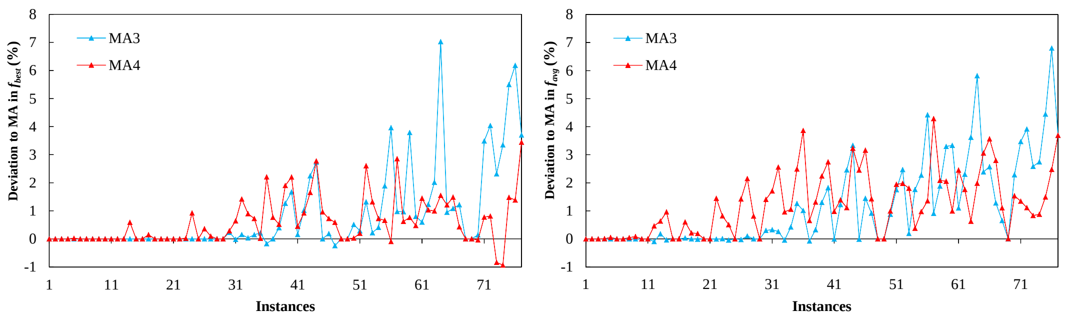
5.2 Benefits of the new neighborhood operators

Six neighborhood operators are applied in the local search to ameliorate offspring solutions. We assess the contributions of the two new neighborhood operators: M3 independently used in [56] and M6 introduced in this work. For this purpose, two MA variants, MA3 (without M3) and MA4 (without M6), are compared, along with the standard MA associated with all neighborhood operators. To ensure a fair comparison, we ran MA3 and MA4 under the same condition of Section 4.2 to conduct the experiments. The results are summarized in Table 3 and illustrated in Fig. 9.

Table 3
Summary of comparative results between MA and two variants.

Pair algorithms	f_{best}				f_{avg}			
	#Wins	#Tiers	#Losses	*p-value*	#Wins	#Tiers	#Losses	*p-value*
MA vs. MA3	39	35	3	3.90E-08	47	20	10	4.42E-09
MA vs. MA4	47	26	4	9.15E-08	63	14	0	5.17E-12

According to Table 3, the two operators are critical to ensure the performance of the MA algorithm (confirmed by the small *p-value* $\ll 0.05$). Indeed, disabling them significantly worsens the results in terms of both the best and average values. Moreover, as shown in Fig. 9, disabling the M3 operator deteriorates MA's performance more than disabling the M6 operator on many large instances with $n > 2152$. These results demonstrate the positive contributions of the M3 operator for solving large instances. Finally, both neighborhood operators have marginal contributions when solving small and medium-sized instances ($n \leq 532$) in terms of the best results.



(a) Best

(b) Avg.

Fig. 9. Comparative results of MA with two variants MA3 (without operator M3) and MA4 (without operator M6) on the 77 instances of Sets $\mathbb{S}$ and $\mathbb{L}$.

5.3 *Convergence analysis of the MA algorithm*

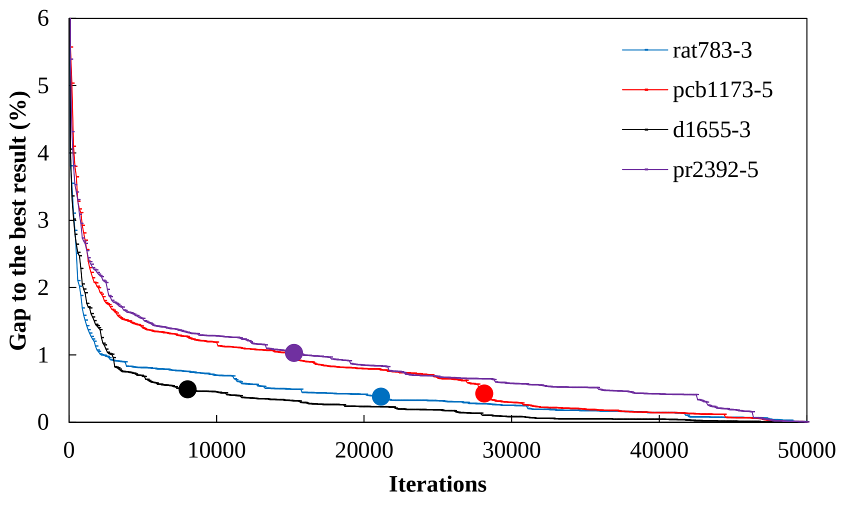


Fig. 10. Running profiles of the MA on four representative instances

In Section 4.3.1, the stopping condition for solving the minmax mTSP was set to the maximum time of $(n/100) \times 4$ minutes in line with the literature. This section aims to verify the convergence behavior of the MA algorithm in the long run by using a relaxed stopping condition of 50,000 iterations. Four representative instances (rat783-3, pcb1173-5, d1655-3, pr2392-5) with different sizes (n from 783 to 2392, m from 3 to 5) were selected and each instance was solved 20 times while the best objective values are recorded during the search. Fig. 10 shows the evolution of the gap between the current value and the best value along the iterations. The four colored dots indicate the average objective values obtained at the end of the standard cutoff time of $(n/100)\times 4$ minutes for the four instances. For these instances, 50,000 iterations lead to 4061.67, 4058.71, 16858.6, 13983.4 seconds, respectively.

From Fig. 10, one observes that with a higher time budget, MA is able to further improve its results reached at the end of the standard cutoff time

$(n/100)\times 4$ minutes). Specifically, the best result can be even improved by 1.19% while the average result can be improved by 1.03%. This experiment demonstrates that the MA algorithm has a highly desirable long-term search behavior and can effectively take advantage of a prolonged cutoff time limit to discover still better solutions.

6 Conclusions

We introduced a unified memetic algorithm for solving both the minmax mTSP and the minmax multidepot mTSP. The proposed algorithm integrates a dedicated edge assembly crossover operator (mEAX), an efficient variable neighborhood descent and an aggressive post-optimization procedure. By properly inheriting edges from high-quality parent solutions, mEAX contributes to propagate favorable characteristics from elite parent solutions to offspring. The variable neighborhood descent is able to locate local optimal solutions effectively. The post-optimization procedure takes full advantage of the ejection chain method and a leading TSP heuristic to further improve the quality of new elite solutions.

The performance of the algorithm was evaluated on two sets of 77 minmax mTSP instances and one set of 43 minmax multidepot mTSP instances. The computational results indicated that the algorithm reaches a high performance compared to the reference algorithms for both problems. Specifically, it reports 44 and 39 new upper bounds for the minmax mTSP and the minmax multidepot mTSP, respectively. We performed additional experiments to assess the contributions of the two key algorithmic components (i.e., mEAX and post-optimization). We also conducted a long term convergence analysis of the algorithm to illustrate its capacity of finding still better solutions if more time is allowed.

For further work, several directions can be envisaged. First, improved local search techniques can be investigated to identify promising neighborhoods and speed up neighborhood examinations. Second, given the contributions of the mEAX operator for the studied problems, it would be interesting to investigate its use or adaptations to solve other routing problems, such as the multidepot capacitated vehicle routing problem. Finally, efficient exact algorithms are still missing for the minmax mTSP and minmax mutlidepot mTSP. Research in this direction is thus quite valuable.

Acknowledgments

We are grateful to the reviewers for their insightful and constructive comments, which helped us to improvement the paper. We also would like to thank authors of [22,52,53,56]: Dr. Jiongzhi Zheng for sharing their source code; Dr.

Korhan Karabulut and Prof. M. Fatih Tasgetiren for sharing their executable code; Prof. Xingyin Wang and Dr. Yongzhen Wang for providing their test problems and answering our questions. Support from the China Scholarship Council (CSC, No. 201906850087) for the first author is also acknowledged.

References

[1] L. Accorsi, D. Vigo, A fast and scalable heuristic for the solution of large-scale capacitated vehicle routing problems, Transportation Science 55 (4) (2021) 832–856.

[2] D. Applegate, W. Cook, S. Dash, A. Rohe, Solution of a min-max vehicle routing problem, INFORMS Journal on Computing 14 (2) (2002) 132–143.

[3] F. Arnold, M. Gendreau, K. Sörensen, Efficiently solving very large-scale routing problems, Computers & Operations Research 107 (2019) 32–42.

[4] F. Arnold, K. Sörensen, Knowledge-guided local search for the vehicle routing problem, Computers & Operations Research 105 (2019) 32–46.

[5] T. Bai, D. Wang, Cooperative trajectory optimization for unmanned aerial vehicles in a combat environment, Science China Information Sciences 62 (1) (2019) 1–3.

[6] O. Beek, B. Raa, W. Dullaert, D. Vigo, An efficient implementation of a static move descriptor-based local search heuristic, Computers & Operations Research 94 (2018) 1–10.

[7] T. Bektas, The multiple traveling salesman problem: an overview of formulations and solution procedures, Omega 34 (3) (2006) 209–219.

[8] E. C. Brown, C. T. Ragsdale, A. E. Carter, A grouping genetic algorithm for the multiple traveling salesperson problem, International Journal of Information Technology & Decision Making 6 (02) (2007) 333–347.

[9] A. M. Campbell, D. Vandenbussche, W. Hermann, Routing for relief efforts, Transportation Science 42 (2) (2008) 127–145.

[10] J. Carlsson, D. Ge, A. Subramaniam, A. Wu, Y. Ye, Solving min-max multi-depot vehicle routing problem, Lectures on Global Optimization 55 (2009) 31–46.

[11] A. E. Carter, C. T. Ragsdale, A new approach to solving the multiple traveling salesperson problem using genetic algorithms, European Journal of Operational Research 175 (1) (2006) 246–257.

[12] O. Cheikhrouhou, I. Khoufi, A comprehensive survey on the multiple traveling salesman problem: Applications, approaches and taxonomy, Computer Science Review 40 (2021) 100369.

[13] J. Conesa-Muñoz, G. Pajares, A. Ribeiro, Mix-opt: A new route operator for optimal coverage path planning for a fleet in an agricultural environment, Expert Systems with Applications 54 (2016) 364–378.

[14] E. D. Dolan, J. J. Moré, Benchmarking optimization software with performance profiles, Mathematical Programming 91 (2) (2002) 201–213.

[15] P. M. França, M. Gendreau, G. Laporte, F. M. Müller, The m-traveling salesman problem with minmax objective, Transportation Science 29 (3) (1995) 267–275.

[16] J.-K. Hao, Memetic algorithms in discrete optimization, in: F. Neri, C. Cotta, P. Moscato (eds.), Handbook of Memetic Algorithms, vol. 379 of Studies in Computational Intelligence, Springer, 2012, pp. 73–94.

[17] P. He, J.-K. Hao, Hybrid search with neighborhood reduction for the multiple traveling salesman problem, Computers & Operations Research 142 (2022) 105726.

[18] P. He, J.-K. Hao, Q. Wu, Grouping memetic search for the colored traveling salesmen problem, Information Sciences 570 (2021) 689–707.

[19] K. Helsgaun, An effective implementation of the lin–kernighan traveling salesman heuristic, European Journal of Operational Research 126 (1) (2000) 106–130.

[20] S. Hong, M. W. Padberg, A note on the symmetric multiple traveling salesman problem with fixed charges, Operations Research 25 (5) (1977) 871–874.

[21] A. Imran, S. Salhi, N. A. Wassan, A variable neighborhood-based heuristic for the heterogeneous fleet vehicle routing problem, European Journal of Operational Research 197 (2) (2009) 509–518.

[22] K. Karabulut, H. Öztop, L. Kandiller, M. F. Tasgetiren, Modeling and optimization of multiple traveling salesmen problems: An evolution strategy approach, Computers & Operations Research 129 (2021) 105192.

[23] P. Kitjacharoenchai, M. Ventresca, M. Moshref-Javadi, S. Lee, J. M. Tanchoco, P. A. Brunese, Multiple traveling salesman problem with drones: Mathematical model and heuristic approach, Computers & Industrial Engineering 129 (2019) 14–30.

[24] F. Lehuédé, O. Péton, F. Tricoire, A lexicographic minimax approach to the vehicle routing problem with route balancing, European Journal of Operational Research 282 (1) (2020) 129–147.

[25] S. Lin, B. W. Kernighan, An effective heuristic algorithm for the traveling-salesman problem, Operations Research 21 (2) (1973) 498–516.

[26] M. López-Ibáñez, J. Dubois-Lacoste, L. P. Cáceres, M. Birattari, T. Stützle, The irace package: Iterated racing for automatic algorithm configuration, Operations Research Perspectives 3 (2016) 43–58.

[27] L.-C. Lu, T.-W. Yue, Mission-oriented ant-team aco for min–max mtsp, Applied Soft Computing 76 (2019) 436–444.

[28] Y. Lu, U. Benlic, Q. Wu, A highly effective hybrid evolutionary algorithm for the covering salesman problem, Information Sciences 564 (2021) 144–162.

[29] A. Maskooki, K. Deb, M. Kallio, A customized genetic algorithm for bi-objective routing in a dynamic network, European Journal of Operational Research 297 (2) (2022) 615–629.

[30] N. Mladenović, P. Hansen, Variable neighborhood search, Computers & Operations Research 24 (11) (1997) 1097–1100.

[31] C. C. Murray, R. Raj, The multiple flying sidekicks traveling salesman problem: Parcel delivery with multiple drones, Transportation Research Part C: Emerging Technologies 110 (2020) 368–398.

[32] Y. Nagata, O. Bräysy, Edge assembly-based memetic algorithm for the capacitated vehicle routing problem, Networks: An International Journal 54 (4) (2009) 205–215.

[33] Y. Nagata, O. Bräysy, W. Dullaert, A penalty-based edge assembly memetic algorithm for the vehicle routing problem with time windows, Computers & Operations Research 37 (4) (2010) 724–737.

[34] Y. Nagata, S. Kobayashi, Edge assembly crossover: A high-power genetic algorithm for the travelling salesman problem, in: T. Bäck (ed.), Proceedings of the 7th International Conference on Genetic Algorithms, East Lansing, MI, USA, July 19-23, 1997, Morgan Kaufmann, 1997.

[35] Y. Nagata, S. Kobayashi, A powerful genetic algorithm using edge assembly crossover for the traveling salesman problem, INFORMS Journal on Computing 25 (2) (2013) 346–363.

[36] K. V. Narasimha, E. Kivelevitch, B. Sharma, M. Kumar, An ant colony optimization technique for solving min–max multi-depot vehicle routing problem, Swarm and Evolutionary Computation 13 (2013) 63–73.

[37] F. Neri, C. Cotta, P. Moscato (eds.), Handbook of Memetic Algorithms, vol. 379 of Studies in Computational Intelligence, Springer, 2012.

[38] V. Pandiri, A. Singh, Two metaheuristic approaches for the multiple traveling salesperson problem, Applied Soft Computing 26 (2015) 74–89.

[39] J.-Y. Potvin, J.-M. Rousseau, An exchange heuristic for routeing problems with time windows, Journal of the Operational Research Society 46 (12) (1995) 1433–1446.

[40] M. Rao, A note on the multiple traveling salesmen problem, Operations Research 28 (3-part-i) (1980) 628–632.

[41] S. Rasmussen, P. Chandler, J. Mitchell, C. Schumacher, A. Sparks, Optimal vs. heuristic assignment of cooperative autonomous unmanned air vehicles, in: AIAA Guidance, Navigation, and Control Conference and Exhibit, 2003.

[42] J. Ren, J.-K. Hao, F. Wu, Z.-H. Fu, An effective hybrid search algorithm for the multiple traveling repairman problem with profits, European Journal of Operational Research (2022) (in press).
URL `https://doi.org/10.1016/j.ejor.2022.04.007`

[43] H. Seyyedhasani, J. S. Dvorak, E. Roemmele, Routing algorithm selection for field coverage planning based on field shape and fleet size, Computers and Electronics in Agriculture 156 (2019) 523–529.

[44] A. Singh, A. S. Baghel, A new grouping genetic algorithm approach to the multiple traveling salesperson problem, Soft Computing 13 (1) (2009) 95–101.

[45] B. Soylu, A general variable neighborhood search heuristic for multiple traveling salesmen problem, Computers & Industrial Engineering 90 (2015) 390–401.

[46] J. A. Svestka, V. E. Huckfeldt, Computational experience with an m-salesman traveling salesman algorithm, Management Science 19 (7) (1973) 790–799.

[47] É. Taillard, P. Badeau, M. Gendreau, F. Guertin, J.-Y. Potvin, A tabu search heuristic for the vehicle routing problem with soft time windows, Transportation Science 31 (2) (1997) 170–186.

[48] R. Todosijević, S. Hanafi, D. Urošević, B. Jarboui, B. Gendron, A general variable neighborhood search for the swap-body vehicle routing problem, Computers & Operations Research 78 (2017) 468–479.

[49] T. Vidal, Hybrid genetic search for the cvrp: Open-source implementation and swap* neighborhood, Computers & Operations Research 140 (2022) 105643.

[50] T. Vidal, T. G. Crainic, M. Gendreau, C. Prins, A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows, Computers & Operations Research 40 (1) (2013) 475–489.

[51] T. Vidal, T. G. Crainic, M. Gendreau, C. Prins, A unified solution framework for multi-attribute vehicle routing problems, European Journal of Operational Research 234 (3) (2014) 658–673.

[52] X. Wang, B. Golden, E. Wasil, The min-max multi-depot vehicle routing problem: Heuristics and computational results, Journal of the Operational Research Society 66 (9) (2015) 1430–1441.

[53] Y. Wang, Y. Chen, Y. Lin, Memetic algorithm based on sequential variable neighborhood descent for the minmax multiple traveling salesman problem, Computers & Industrial Engineering 106 (2017) 105–122.

[54] S. Yuan, B. Skinner, S. Huang, D. Liu, A new crossover approach for solving the multiple travelling salesmen problem using genetic algorithms, European Journal of Operational Research 228 (1) (2013) 72–82.

[55] E. E. Zachariadis, C. T. Kiranoudis, A strategy for reducing the computational complexity of local search-based methods for the vehicle routing problem, Computers & Operations Research 37 (12) (2010) 2089–2105.

[56] J. Zheng, Y. Hong, W. Xu, W. Li, Y. Chen, An effective iterated two-stage heuristic algorithm for the multiple traveling salesmen problem, Computers & Operations Research 143 (2022) 105772.

[57] H. Zhou, M. Song, W. Pedrycz, A comparative study of improved ga and pso in solving multiple traveling salesmen problem, Applied Soft Computing 64 (2018) 564–580.

Appendix

A Detailed computational results

This appendix presents detailed computational results of the proposed MA algorithm together with the results of reference algorithms: re-IWO, re-MASVND, ES [22], HSNR [17] and ITSHA [56]. In the tables presented hereafter, column 'Instances' indicates the name of the benchmark instance; column 'BKS' shows the best-known solution summarized from the literature. For the min-max mTSP, the starred BKS values are optimal values. 'f_{best}' and 'f_{avg}' are the best and average solution found by the algorithm in the column header, respectively. 'Gap' is calculated as $Gap = 100\times(f_{best}\text{-}f_{bk})/f_{bk}$ where f_{best} and f_{bk} are the best objective value of MA and the best objective value from all reference algorithms (including BKS), respectively. Since both problems have a minimization objective, a negative Gap indicates an improvement over the BKS value (i.e., a new upper bound). Furthermore, the dark gray color indicates that the algorithm obtains the best result among the compared algorithms on the corresponding instance; the medium gray color displays the second best result, and so on. We provide additionally information for each algorithm in terms of the best and average value. 'Average' is the average value over the instances of a benchmark set.

As shown in Table A.3, the time information in Table A.3 is provided only for indicative purposes (The '-' symbol indicates the time information is unavailable for MD and VNS or non-applicable for MA). The time (in seconds) for MD and VNS corresponds to the average time of one run under the stopping conditions indicated in Section 4.2. For the MA algorithm, 'TTB' indicates the average time in seconds needed for MA to hit the BKS values, while 'AT' is the average time of one run.

Table A.1. The minmax mTSP: comparative results of MA with five reference algorithms on the 41 instances of Set $\mathbb{S}$.

Instances	BKS	f_{best}						Gap(%)	f_{avg}					
		re-IWO	re-MASVND	ES [22]	HSNR [17]	ITSHA[56]	MA		re-IWO	re-MASVND	ES [22]	HSNR [17]	ITSHA[56]	MA
mtsp51-3	159.57	159.57	159.57	159.57	159.57	159.57	159.57	0.00	159.57	159.57	159.57	159.85	159.57	159.57
mtsp51-5	118.13	118.13	118.13	118.13	118.13	118.13	118.13	0.00	118.13	118.13	118.13	118.13	118.13	118.13
mtsp51-10	112.07*	112.07	112.07	112.07	112.07	112.07	112.07	0.00	112.07	112.07	112.07	112.07	112.07	112.07
mtsp100-3	8509.16	8509.16	8509.16	8509.16	8509.16	8509.16	8509.16	0.00	8510.86	8602.30	8649.75	8513.75	8509.16	8509.16
mtsp100-5	6765.73	6780.37	6767.02	6767.02	6765.73	6767.82	6765.73	0.00	6833.45	6801.75	6832.74	6770.67	6772.95	6766.78
mtsp100-10	6358.49*	6358.49	6358.49	6358.49	6358.49	6358.49	6358.49	0.00	6358.49	6358.49	6358.49	6358.49	6358.49	6358.49
mtsp100-20	6358.49*	6358.49	6358.49	6358.49	6358.49	6358.49	6358.49	0.00	6358.49	6358.49	6358.49	6358.49	6358.49	6358.49
rand100-3	3031.95	3031.95	3031.95	3031.95	3031.95	3031.95	3031.95	0.00	3031.95	3047.71	3084.49	3032.67	3033.65	3031.95
rand100-5	2409.63	2409.90	2409.63	2409.63	2411.68	2412.35	2409.63	0.00	2429.50	2428.35	2422.41	2415.00	2414.65	2409.64
rand100-10	2299.16*	2299.16	2299.16	2299.16	2299.16	2299.16	2299.16	0.00	2299.16	2299.16	2299.16	2299.16	2299.16	2299.16
rand100-20	2299.16*	2299.16	2299.16	2299.16	2299.16	2299.16	2299.16	0.00	2299.16	2299.16	2299.16	2299.16	2299.16	2299.16
mtsp150-3	13075.80	13078.40	13234.10	13303.80	13075.80	13088.74	13038.30	**-0.29**	13259.97	13411.26	13526.70	13169.37	13210.69	13079.74
mtsp150-5	8466.00	8477.96	8493.62	8563.08	8477.96	8492.97	8417.02	**-0.58**	8650.11	8686.61	8757.22	8538.83	8572.77	8453.15
mtsp150-10	5557.00	5751.41	5666.45	5625.32	5590.64	5593.56	5557.41	0.00	5851.79	5763.28	5718.45	5604.92	5608.50	5588.86
mtsp150-20	5246.49*	5246.49	5246.49	5246.49	5246.49	5246.49	5246.49	0.00	5247.62	5246.49	5247.21	5246.49	5246.49	5246.49
mtsp150-30	5246.49*	5246.49	5246.49	5246.49	5246.49	5246.49	5246.49	0.00	5246.49	5246.49	5246.49	5246.49	5246.49	5246.49
gtsp150-3	2407.34	2407.34	2433.80	2423.17	2407.34	2407.34	2401.63	**-0.24**	2421.40	2468.12	2491.00	2435.49	2416.87	2401.86
gtsp150-5	1741.61	1744.57	1744.26	1751.85	1741.71	1741.13	1741.13	0.00	1777.90	1779.32	1797.71	1743.48	1752.06	1741.13
gtsp150-10	1554.64*	1554.64	1554.64	1554.64	1554.64	1554.64	1554.64	0.00	1557.22	1559.10	1554.64	1554.64	1554.64	1554.76
gtsp150-20	1554.64*	1554.64	1554.64	1554.64	1554.64	1554.64	1554.64	0.00	1554.64	1554.64	1554.64	1554.64	1554.64	1554.64
gtsp150-30	1554.64*	1554.64	1554.64	1554.64	1554.64	1554.64	1554.64	0.00	1554.64	1554.64	1554.64	1554.64	1554.64	1554.64
kroA200-3	10748.10	10801.30	10833.60	10883.30	10748.10	10700.57	10691.00	**-0.09**	10965.59	11136.70	11174.70	10987.69	10819.85	10691.41
kroA200-5	7415.54	7497.21	7484.17	7536.91	7418.87	7449.22	7412.12	**-0.05**	7697.45	7634.61	7770.43	7494.44	7513.67	7414.21
kroA200-10	6223.22*	6223.22	6223.22	6223.22	6223.22	6223.22	6223.22	0.00	6255.37	6266.44	6240.52	6223.22	6223.22	6249.10
kroA200-20	6223.22*	6223.22	6223.22	6223.22	6223.22	6223.22	6223.22	0.00	6223.22	6223.22	6223.22	6223.22	6223.22	6223.22
lin318-3	15902.50	16133.40	16551.60	16349.60	15902.50	15930.04	15663.50	**-1.50**	17006.92	16886.01	16797.80	16207.05	16088.56	15699.92
lin318-5	11295.20	12291.60	11741.60	11619.60	11295.20	11430.65	11276.80	**-0.16**	12882.38	12023.74	11907.90	11596.35	11601.67	11291.59
lin318-10	9731.17*	9861.64	9731.17	9731.18	9731.17	9731.17	9731.17	0.00	9963.31	9797.38	9736.17	9731.17	9731.17	9731.17
lin318-20	9731.17*	9731.17	9731.17	9731.17	9731.17	9731.17	9731.17	0.00	9731.18	9731.17	9731.18	9731.17	9731.17	9731.17
att532-3	10231.00	11258.00	10566.00	10585.00	10231.00	10158.00	9966.00	**-1.89**	11525.50	10853.05	10953.90	10565.30	10344.50	10064.00
att532-5	7067.00	8518.00	7279.00	7344.00	7067.00	7067.00	6986.00	**-1.15**	8895.80	7429.50	7463.50	7334.00	7156.80	7070.95
att532-10	5709.00	6427.00	5745.00	5761.00	5709.00	5731.00	5770.00	1.07	6552.90	5809.00	5806.75	5738.90	5787.50	5796.75
att532-20	5580.00*	5745.00	5580.00	5580.00	5580.00	5583.00	5580.00	0.00	5836.90	5582.90	5580.05	5580.00	5601.75	5589.35
rat783-3	3158.34	3688.79	3295.90	3444.20	3187.90	3131.99	3052.41	**-2.54**	3786.16	3364.20	3485.74	3237.29	3180.79	3083.52
rat783-5	2006.46	2627.74	2120.74	2125.53	2006.46	2018.44	1961.12	**-2.26**	2781.71	2145.38	2189.92	2044.32	2058.65	1989.68
rat783-10	1334.76	1692.31	1396.92	1373.46	1334.76	1357.65	1313.01	**-1.63**	1718.75	1424.76	1396.78	1345.88	1381.69	1325.54
rat783-20	1231.69*	1371.32	1237.97	1231.69	1231.69	1231.69	1231.69	0.00	1390.09	1244.26	1231.69	1231.69	1231.84	1235.37
pcb1173-3	20292.61	25557.90	22255.20	23193.10	20813.80	20288.75	19569.50	**-3.55**	26439.83	22941.19	23640.00	21144.92	20473.45	19858.77
pcb1173-5	12952.97	18703.50	14088.40	14333.00	13032.30	12816.55	12406.60	**-3.20**	19226.82	14305.57	14601.30	13216.99	13045.14	12639.49
pcb1173-10	7758.26	11170.00	8452.28	8222.40	7758.26	7801.18	7623.59	**-1.74**	11388.41	8637.95	8352.07	7897.20	7910.09	7745.00
pcb1173-20	6528.86*	8132.08	6549.14	6549.14	6528.86	6528.86	6528.86	0.00	8356.53	6623.91	6577.59	6528.86	6534.75	6548.87
Average	5998.71	6553.84	6152.15	6177.75	6015.33	6000.98	5943.29	-	6689.21	6241.85	6268.40	6076.73	6043.73	5971.30

Table A.2. The minmax mTSP: comparative results of MA with four reference algorithms on the 36 instances of Set $\mathbb{L}$.

Instances	BKS	f_{best}					Gap(%)	f_{avg}				
		re-IWO	re-MASVND	HSNR [17]	ITSHA[56]	MA		re-IWO	re-MASVND	HSNR [17]	ITSHA[56]	MA
nrw1379-3	20495.90	24401.20	22236.40	20495.90	19871.21	19222.10	**-3.27**	25204.30	23349.12	20765.70	20085.29	19472.39
nrw1379-5	12416.50	17636.70	13368.80	12416.50	12218.09	11913.40	**-2.49**	18019.88	13847.14	12652.56	12493.99	12203.76
nrw1379-10	7114.71	10145.40	7583.59	7114.71	7008.61	6846.27	**-2.32**	10404.29	7748.69	7212.24	7136.56	6987.31
nrw1379-20	5370.82*	7082.11	5495.31	5370.82	5370.82	5381.59	0.20	7310.74	5571.01	5371.08	5402.42	5388.99
fl1400-3	9192.38	9860.63	9562.25	9192.38	8310.34	7854.97	**-5.48**	10140.92	10094.49	9621.59	8482.77	7989.25
fl1400-5	6268.25	8422.09	6803.42	6268.25	6360.45	6116.28	**-2.42**	8590.95	7134.69	6783.62	6572.53	6185.40
fl1400-10	5763.26*	7359.74	5763.26	5763.26	5763.26	5763.26	0.00	7485.43	5763.74	5763.26	5785.69	5763.26
fl1400-20	5763.26*	6687.79	5763.26	5763.26	5763.26	5763.26	0.00	6819.01	5763.26	5763.26	5763.26	5763.26
d1655-3	25229.30	32293.30	30143.30	25229.30	25403.26	23921.90	**-5.18**	33051.92	42813.48	25635.98	25804.06	24149.72
d1655-5	17181.20	24146.80	18719.10	17181.20	17502.88	16512.20	**-3.89**	24854.66	19376.15	17454.32	17824.61	16754.81
d1655-10	11660.00	15868.50	12454.00	11660.00	11814.34	11320.10	**-2.92**	16569.21	12623.92	11816.04	11971.60	11528.33
d1655-20	9598.94	12165.20	9893.04	9598.94	9910.12	9627.28	0.30	12605.33	10120.03	9607.73	10172.24	9669.97
u2152-3	24207.40	32354.70	43724.70	24207.40	23295.73	22127.80	**-5.01**	33246.73	44187.24	24747.01	23746.41	22629.25
u2152-5	15055.10	23356.00	17653.10	15055.10	14778.56	14094.00	**-4.63**	23534.98	18404.22	15394.85	15236.58	14442.21
u2152-10	8624.61	13454.40	9458.60	8624.61	8763.71	8332.12	**-3.39**	13985.98	9600.79	8780.91	9018.30	8499.64
u2152-20	6171.89	9223.98	6550.73	6171.89	6605.48	6253.35	1.32	9532.87	6727.13	6225.82	6676.91	6339.15
pr2392-3	141627.00	186013.00	254034.00	141627.00	135763.02	130015.00	**-4.23**	190584.70	256052.65	143703.00	137589.12	132228.70
pr2392-5	88083.20	133780.00	104977.00	88083.20	87465.60	82408.50	**-5.78**	135073.30	132626.10	89582.83	88179.42	84448.64
pr2392-10	51085.30	80135.10	55337.60	51085.30	50514.84	49033.60	**-2.93**	83131.04	56650.63	52100.80	50929.73	49985.84
pr2392-20	35325.30	56941.00	38175.60	35325.30	35999.41	35455.50	0.37	58490.18	39420.33	35709.02	36546.00	36107.77
pcb3038-3	51049.90	66159.20	85795.40	51049.90	48351.41	46994.60	**-2.81**	68931.99	86481.39	51582.38	49081.79	47686.85
pcb3038-5	31140.20	46465.70	66560.40	31140.20	30089.85	29223.00	**-2.88**	46938.10	67071.90	31495.59	30603.78	29864.61
pcb3038-10	16949.90	26954.20	19198.20	16949.90	16878.69	16031.70	**-5.02**	27659.07	19620.41	17450.44	16645.10	16509.95
pcb3038-20	10835.00	17772.50	12012.20	10835.00	10827.78	10769.60	**-0.54**	18323.59	12643.54	11004.40	11196.47	10961.26
fl3795-3	11971.00	16611.70	22444.50	11971.00	12290.18	10927.40	**-8.72**	17207.03	22801.50	12815.54	13022.12	11321.19
fl3795-5	7923.71	13391.00	19698.50	7923.71	8151.43	7715.36	**-2.63**	13809.93	19877.81	8610.84	8657.18	8014.97
fl3795-10	5763.26*	10132.30	6715.07	5763.26	5824.14	5764.85	0.03	10500.27	7120.46	5823.89	5990.03	5810.07
fl3795-20	5763.26*	8519.26	5763.26	5763.26	5763.26	5763.26	0.00	8679.69	5763.75	5763.26	5766.51	5763.74
fnl4461-3	66903.70	90062.10	108622.00	66903.70	64140.70	62016.70	**-3.31**	91143.24	109798.50	67971.34	65268.41	62894.65
fnl4461-5	40721.20	59532.70	83650.40	40721.20	39839.40	38265.50	**-3.95**	60170.68	84430.87	41777.11	39839.40	38935.22
fnl4461-10	22041.50	34068.20	25385.20	22041.50	22145.65	20671.60	**-6.22**	34741.06	43581.63	22891.45	22954.31	20996.25
fnl4461-20	12630.10	22142.80	14611.30	12630.10	12789.54	12347.80	**-2.24**	22852.80	15262.97	13046.38	12987.65	12542.14
rl5915-3	213864.00	328020.00	443748.00	213864.00	210056.49	193879.00	**-7.70**	332327.15	445979.00	226819.75	215466.06	198796.00
rl5915-5	133457.00	221495.00	362776.00	133457.00	125537.65	120418.00	**-4.08**	225566.65	364717.65	145173.07	132524.74	124718.05
rl5915-10	76585.20	133266.00	267295.00	76585.20	70853.30	66329.40	**-6.38**	137737.55	270354.45	84459.02	71353.13	67961.45
rl5915-20	48958.50	88081.70	51115.20	48958.50	44716.69	43121.00	**-3.57**	92716.25	53066.77	60306.22	46080.96	44373.90
Average	35077.55	52611.17	63141.32	35077.55	34076.09	32450.03	-	53831.71	65456.87	36713.40	34801.53	33158.00

Table A.3
The minmax multidepot mTSP: comparative results of MA with two reference algorithms on the 43 instances of Set $\mathbb{M}$.

Instances	BKS	MD [52]		VNS [52]		MA				
		f_{best}	Time (s)	f_{best}	Time (s)	f_{best}	f_{avg}	Gap(%)	TTB (s)	AT (s)
MM1	170.109	170.909	1	170.909	1	170.908	170.908	0.470	-	2
MM2	130.8	130.800	11	131.497	6	124.067	125.8095	**-5.148**	18	277
MM3	238.973	238.973	18	240.397	13	230.821	231.9083	**-3.411**	28	371
MM4	479.676	479.676	18	481.595	340	438.039	442.2707	**-8.680**	52	1446
MM5	315.889	315.889	33	333.376	18	299.751	299.9226	**-5.109**	13	1546
MM6	82.187	82.226	44	82.226	16	85.356	86.94082	3.856	-	306
MM7	189.016	189.016	2	189.016	5	189.017	189.017	0.001	-	9
MM8	217.383	217.383	30	231.493	27	203.585	204.2455	**-6.347**	12	460
MM9	152.504	152.504	112	156.972	57	142.355	143.4553	**-6.655**	55	840
MM10	182.926	197.390	4	182.926	9	181.382	181.382	**-0.844**	1	33
MM11	102.346	102.346	3	103.663	8	102.346	102.346	0.000	12	99
MM12	78.903	78.903	3	80.828	5	72.921	73.23033	**-7.581**	1	83
MM13	120.688	121.872	5	120.688	11	117.681	117.7794	**-2.492**	13	215
MM14	134.613	134.613	8	137.219	11	125.585	126.0707	**-6.707**	11	262
MM15	96.524	99.805	5	96.524	7	90.787	91.27053	**-5.943**	11	221
MM16	101.68	101.680	23	103.696	28	96.068	98.70936	**-5.519**	204	548
MM17	248.588	248.588	235	259.255	28	236.859	238.7844	**-4.718**	98	1038
MM18	390.16	390.160	619	400.269	58	383.617	385.4731	**-1.677**	324	985
MM19	365.657	365.657	616	395.371	159	339.333	344.665	**-7.199**	6	1046
MM20	339.92	339.920	360	356.176	152	311.737	315.116	**-8.291**	10	1213
MM21	259.14	259.140	-	274.100	-	245.165	246.5903	**-5.393**	18	551
MM22	400.6	400.600	-	413.270	-	390.934	393.2435	**-2.413**	61	510
MM23	374.97	374.970	-	378.710	-	363.504	363.5538	**-3.058**	22	280
MM24	204	204.000	-	206.220	-	195.992	198.7478	**-3.925**	146	801
MM25	272.61	272.610	-	274.840	-	230.690	233.9735	**-15.377**	4	746
MM26	364.56	364.560	-	369.100	-	349.459	351.2541	**-4.142**	40	1047
MM27	290.37	290.370	-	298.460	-	285.220	286.3728	**-1.774**	75	568
MM28	354.31	354.310	-	367.720	-	348.377	351.8489	**-1.675**	556	1343
MM29	364.01	364.010	-	376.180	-	357.100	359.0724	**-1.898**	246	1001
MM30	140.34	140.340	-	149.540	-	128.349	130.7342	**-8.544**	85	995
MM31	112.52	124.320	-	112.520	-	106.189	107.7453	**-5.627**	29	415
MM32	98.45	103.150	-	98.450	-	96.260	96.35376	**-2.225**	1	50
MM33	97.56	97.560	-	100.930	-	92.820	92.8197	**-4.859**	1	469
MM34	84.64	84.640	-	85.580	-	78.796	80.94745	**-6.905**	168	703
MM35	107.86	109.300	-	107.860	-	99.928	99.92959	**-7.354**	2	282
MM36	153.27	155.990	-	153.270	-	135.947	136.9038	**-11.302**	8	1187
MM37	151.19	156.410	-	151.190	-	132.937	132.937	**-12.073**	1	744
MM38	155.46	155.460	-	166.300	-	149.556	149.764	**-3.798**	10	331
MM39	209.85	209.850	-	223.670	-	195.788	198.0532	**-6.701**	28	676
MM40	243.47	243.470	-	250.680	-	235.961	236.2019	**-3.084**	15	213
MM41	255.27	257.160	-	255.270	-	237.959	241.4352	**-6.781**	158	1312
MM42	357.17	367.440	-	357.170	-	314.451	318.7805	**-11.960**	23	1046
MM43	375.16	375.160	-	375.550	-	349.380	351.95	**-6.872**	32	811
Average	222.449	223.794	-	227.923	-	211.132	212.964	-	-	-