Computers & Operations Research 142 (2022) 105726

Contents lists available at ScienceDirect

Computers and Operations Research

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

Hybrid search with neighborhood reduction for the multiple traveling salesman problem

Pengfei He, Jin-Kao Hao *

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

ARTICLE INFO

Keywords:
Traveling salesman
Multiple traveling salesman
Hybrid heuristic
Neighborhood reduction

ABSTRACT

We present an effective hybrid algorithm with neighborhood reduction for solving the multiple traveling salesman problem (mTSP). This problem aims to optimize one of the two objectives: to minimize the total traveling distance (the minsum mTSP) or to minimize the longest tour (the minmax mTSP). The proposed algorithm hybridizes inter-tour optimization with an efficient neighborhood search based on tabu search and intra-tour optimization using the traveling salesman heuristic EAX. A dedicated neighborhood reduction strategy is introduced to avoid the examination of non-promising candidate solutions and thus speed up the neighborhood search. Results of extensive computational experiments are shown on 41 popular instances from several sources and 36 new large instances. Comparisons with five state-of-the-art methods in the literature demonstrate a high competitiveness of the proposed algorithm. Additional experiments on applying a classical TSP heuristic to the minsum mTSP instances show excellent results.

1. Introduction

The multiple traveling salesman problem (mTSP) generalizes the popular NP-hard traveling salesman problem (TSP) with multiple salespersons. Formally, the mTSP is the following graph theoretic problem. Let $G=(V,\ A)$ be a graph with vertex set $V = \{0, 1, \ldots, n\}$ and a set of arcs A, where 0 of V is the depot and the remaining vertices $N = \{1, \ldots, n\}$ represent n cities. Let $C = (c_{ij})$ be a non-negative cost (distance) matrix associated with A, which satisfies the triangle inequality ($c_{ij} + c_{jk} > c_{ik}$ for any $i, j, k \in V$ and $i \neq j \neq k$). The matrix C is said to be symmetric when $c_{ij} = c_{ji}, (i, j) \in A$ and asymmetric otherwise. A *feasible* solution is a partition of the set of cities N into m distinct Hamiltonian tours $\{r_1, r_2, \ldots, r_m\}$, such that each tour r_k ($k \in \{1, \ldots, m\}$) starts and ends at the depot, and includes at least one city. The minsum mTSP, first proposed in Svestka and Huckfeldt (1973), is to minimize the total traveling tour-length of a given mTSP instance and can be described by the following mathematical model (Cheikhrouhou and Khoufi, 2021).

$$\begin{array}{ll} \text{(minsum mTSP)} & min\ F(\varphi) = \sum_{k=1}^{m} TSP(r_k) \\ subject\ to & \cup_{k=1}^{m} r_k = V \\ & r_k \cap r_{k'} = \{0\}, k \neq k', 1 \leq k, k' \leq m \end{array} \tag{1}$$

where $\varphi = \{r_1, r_2, \ldots, r_m\}$ is a feasible solution with r_k ($k \in \{1, \ldots, m\}$) representing the kth tour composed of the vertices visited by the kth salesman, and $TSP(r_k)$ is the length of the tour r_k. It is easy to observe that the minsum mTSP becomes the conventional TSP when $m = 1$ (only one salesman).

By minimizing the total tour-length of all the salesmen, the minsum mTSP aims to optimize the total efficiency of a solution. In some contexts, it is useful to consider the equity criterion by avoiding excessive tour-length differences among the salesmen. To this end, the minmax mTSP was introduced in França et al. (1995), which minimizes the longest tour and can be formulated by the mathematical model as follows (Cheikhrouhou and Khoufi, 2021).

$$\begin{array}{ll} \text{(minmax mTSP)} & min\ F(\varphi) = max_{k \in \{1, \ldots, m\}} \{TSP(r_k)\} \\ subject\ to & \cup_{k \in \{1, \ldots, m\}} r_k = V \\ & r_k \cap r_{k'} = \{0\}, k \neq k', 1 \leq k, k' \leq m \end{array} \tag{2}$$

From an application perspective, these mTSP models are useful for a number of real problems that cannot be formulated conveniently with the classical TSP model (Cheikhrouhou and Khoufi, 2021). Representative examples include news paper delivery (Whizzkids'96, 1996), hot rolling scheduling (Tang et al., 2000), 3D path planning (Ergezer and Leblebicioğlu, 2014), multi-unit service scheduling (Carter and Ragsdale, 2009), path planning for robot and UAV (Zhan and Zeng, 2019; Koubâa et al., 2017), container drayage services (Zhang et al., 2018; Shiri and Huynh, 2016), and harvesters scheduling (He et al., 2018, 2019). Additional practical problems can be formulated by extended mTSP variants (Bektaş, 2012; Lu et al., 2019; Paydar et al., 2010).

* Corresponding author.
E-mail addresses: pengfeihe606@gmail.com (P. He), jin-kao.hao@univ-angers.fr (J.-K. Hao).

https://doi.org/10.1016/j.cor.2022.105726
Received 7 June 2021; Received in revised form 17 January 2022; Accepted 17 January 2022
Available online 10 February 2022

On the other hand, as a generalization of the NP-hard TSP problem, the mTSP is computationally challenging from the perspective of optimization.

Due to its theoretical and practical interest, the mTSP has received much attention from various fields including engineering, operations research and computer science. There are exact algorithms for the minsum mTSP, including a branch-and-bound algorithm (Gavish and Srikanth, 1986) and a cutting plane algorithm (Laporte and Nobert, 1980). Optimal results were reported on instances with up to 500 vertices and 10 salesmen. There are also exact algorithms for variants of the minmax mTSP. For example, a branch-and-cut algorithm (Applegate et al., 2002) was presented to solve a minmax vehicle routing problem on instances up to 120 cities and 4 vehicles. Benders decomposition algorithms (Bektaş, 2012) were proposed to optimally solve the mTSP with load balancing on instances with up to 171 cities and 10 salesmen. Given the NP-hard nature of the problem, a number of heuristic and metaheuristic algorithms have been developed to find suboptimal solutions for large instances that cannot be optimally solved, as reviewed in Section 2.

We observe that computational results have been improved continually with the introduction of new solution approaches and algorithms. Meanwhile, our literature review (see Section 2) indicates that existing methods lack stability and their performances typically degrade when large instances are solved (e.g. $n > 1000$). Moreover, some algorithms were designed only for one mTSP objective (minsum or minmax).

In this work, we aim to advance the state-of-the-art of solving large-scale instances of the mTSP for both objectives. For this purpose, we introduce an effective hybrid search algorithm that performs well especially on large mTSP instances. The proposed algorithm benefits from the symbiosis of inter-tour optimization and intra-tour optimization. The inter-tour optimization uses neighborhood search to improve the solution by exchanging information between two tours (via the *insert* and *cross-exchange* operators). The intra-tour optimization applies a TSP method (the EAX heuristic Nagata and Kobayashi, 2013) to keep each individual tour as short as possible. We carry out extensive experiments to show the competitiveness of the proposed algorithm. We perform additional experiments to assess the usefulness of its key ingredients. Finally, we present for the first time computational experiments of applying the TSP heuristic EAX to the minsum mTSP, and draw conclusions regarding the effectiveness of this approach.

The remainder of this paper is organized as follows. Section 2 provides a literature review on heuristic algorithms for the mTSP. Section 3 presents the details of the proposed algorithm. Section 4 shows computational results and comparisons. Section 5 investigates key ingredients of the proposed algorithm. Section 6 draws conclusions with research perspectives.

2. Literature review

In this section, we provide a literature review of the most representative heuristic algorithms for the mTSP. These algorithms are divided into three categories: population-based evolutionary algorithms, swarm intelligence algorithms and neighborhood-based local optimization. The reviewed algorithms are summarized in Table 1, where "both" means the corresponding algorithm solves both the minsum and minmax mTSP. For a comprehensive survey of exact and heuristic methods, the reader is referred to Bektas (2006) and Cheikhrouhou and Khoufi (2021).

Various population-based evolutionary algorithms have been proposed for solving the mTSP. In 2006, Carter and Ragsdale (2006) presented a grouping genetic algorithm for the mTSP using a two-part chromosome to represent a solution. Compared to two previous chromosome representations, the two-part chromosome representation avoids redundant solutions and thus reduces the solution space. This work also introduced a set of benchmark instances with 50–150 cities and 3–30 salesmen, and showed comparisons with genetic algorithms using other representations. Similarly, in 2007, Brown et al. (2007) showed a follow-up study (Carter and Ragsdale, 2006) of using another two-part chromosome representation where both real-valued genes and integer-valued genes are used. Another group of benchmark instances was proposed for their computational studies. Subsequently, in 2009, Singh and Baghel (2009) presented another grouping genetic algorithm with the so-called m-tour chromosome representation, where each tour is represented by an array and no ordering is imposed among tours. This algorithm employed a steady-state population replacement method, and outperformed the genetic algorithms of Carter and Ragsdale (2006), Brown et al. (2007) in terms of the minsum mTSP and the minmax mTSP. In 2013, Yuan et al. (2013) investigated a specific crossover operator (called TCX) based on the two-part chromosome of Carter and Ragsdale (2006). The proposed crossover aims to better preserve building block information during solution recombination while ensuring a good diversity. They showed a superior performance of their TCX-based genetic algorithm over genetic algorithms using three other crossover operators including the algorithm of Carter and Ragsdale (2006). In 2017, Wang et al. (2017) designed a memetic algorithm (MASVND) for the minmax mTSP. The algorithm employs recombination and mutation operators based on spatial distribution (Pandiri and Singh, 2015) and incorporates four neighborhood search operators (one-point move, Or-opt$_2$ move, Or-opt$_3$ move and Or-opt$_4$ move) for the variable neighborhood descent. They introduced a new set of (large) benchmark instances and assessed MASVND for the minmax mTSP compared to ABC (Pandiri and Singh, 2015), IWO (Pandiri and Singh, 2015) and GVNS (Soylu, 2015). The results indicated that MASVND outperforms its competitors on large instances (with 532–1173 cities), but performs worse than IWO on small instances (with 51–318 cities). In 2021, Karabulut et al. (2021) proposed an evolution strategy (ES) approach for solving the mTSP and multi-depots mTSP with non-predetermined depots. This approach adopts a self-adaptive Ruin and Recreate heuristic to generate offspring solutions, and a local search, including 3-opt, to further enhance the solution quality. The computational experiments showed the competitiveness of this approach on the minsum and minmax mTSP instances.

Another popular approach for solving the mTSP concerns swarm intelligence methods. In 2006, Pan and Wang (2006) presented a basic ant colony optimization (ACO) algorithm and showed a limited comparison with a genetic algorithm. In 2009, Liu et al. (2009) exposed another ACO algorithm which integrates local search for search intensification. They showed competitive results for the minsum mTSP and the minmax mTSP compared to a genetic algorithm on some benchmark instances. In 2019, Lu and Yue (2019) introduced a mission-oriented ant-team ACO algorithm and reported comparative studies with previous algorithms on the instances of Carter and Ragsdale (2006). In 2015, Pandiri and Singh (2015) presented several algorithms based on artificial bee colony (ABC) and invasive weed optimization (IWO) for the minsum mTSP and the minmax mTSP, which use local search for the post-optimization. There are two versions of the ABC algorithm, where neighboring solutions are generated from the original solution based on different distance strategies. IWO can be considered as a reinforced ABC algorithm because it generalizes ABC, by visiting more neighboring solutions at each generation. These algorithms showed excellent performances and updated a majority of the best results of previous algorithms for the benchmark instances of Carter and Ragsdale (2006), Brown et al. (2007), Singh and Baghel (2009).

Compared to the aforementioned approaches, there are relatively few studies using neighborhood-based local optimization to solve the mTSP, among which the general variable neighborhood search heuristic (GVNS) presented by Soylu (2015) is a representative example. Based on the m-tour solution representation, this algorithm applies six neighborhood search operators (one-point move, two types of Or-opt move, two-point move and three-point move, as well as 2-opt) to find local optima and uses a random shaking method to escape local optimum traps.

Table 1
Summary and taxonomy of representative heuristic algorithms for the mTSP.

Algorithm	Population-based evolutionary algorithms	Swarm intelligence algorithms	Neighborhood-based local search	Problem solved
Carter and Ragsdale (2006)	✓			both
Brown et al. (2007)	✓			both
Singh and Baghel (2009)	✓			both
Yuan et al. (2013)	✓			both
Wang et al. (2017)	✓			minmax
Karabulut et al. (2021)	✓			both
Pan and Wang (2006)		✓		both
Liu et al. (2009)		✓		both
Pandiri and Singh (2015)		✓		both
Lu and Yue (2019)		✓		minmax
Soylu (2015)			✓	both
Penna et al. (2013)			✓	minsum
Uchoa et al. (2017)			✓	minsum

Experimental results indicated that the algorithm globally competes well with previous methods, except IWO (Pandiri and Singh, 2015) which showed superior results on the instances of Carter and Ragsdale (2006).

One notices that iterated local search (ILS) algorithms were designed for the related capacitated vehicle routing problem (CVRP), that becomes the minsum mTSP when the capacity is set to 1. In particular, Penna et al. (2013) proposed an ILS algorithm which uses a variable neighborhood descent procedure, with a random neighborhood ordering, in the local search phase. Uchoa et al. (2017) tested an ILS-based matheuristic algorithm on a set of new CVRP benchmark instances and reported several good results for the CVRP with capacity of 1, which is equivalent to the minsum mTSP. Local search algorithms were also proposed for the balanced mTSP (Garn, 2020) and balanced dynamic mTSP (Garn, 2021).

Among the reviewed studies, the following algorithms hold the best-known results on the commonly used mTSP benchmark instances introduced in Carter and Ragsdale (2006), Brown et al. (2007), Wang et al. (2017): ABC(VC), IWO (Pandiri and Singh, 2015), GVNS (Soylu, 2015), MASVND (Wang et al., 2017) (for the minmax mTSP only) and ES (Karabulut et al., 2021). Thus they can be considered to be the state-of-the-art methods for solving the mTSP, and are used as the main reference algorithms for the computational studies in this work. Nevertheless, none of the existing mTSP algorithms can be considered as the most effective for all benchmark instances for both the minsum and minmax objectives of the mTSP.

According to the reviewed studies, we observe that most existing mTSP algorithms are based on population-based and swarm intelligence approaches. These algorithms have fast convergences, and typically performed well on small instances. However, they showed inferior performances on large instances (Wang et al., 2017; Karabulut et al., 2021). To advance the state-of-the-art of solving the mTSP, especially on large instances, this work introduces a hybrid algorithm that combines an efficient neighborhood search (for inter-tour optimization) and a traveling salesman heuristic (for intra-tour optimization).

Finally, it is known that the minsum mTSP can be conveniently transformed to the conventional TSP (Hong and Padberg, 1977; Rao, 1980). For a minsum mTSP instance G with n vertices and m tours, this transformation leads to an equivalent TSP instance G^T with $n+m-1$ vertices. G^T is an extension of G with $m-1$ additional vertices such that each new vertex is a duplicate of the depot in G and each pair of depots have a large enough (e.g., infinite) distance between them. Then a mTSP solution of G with m tours ($m>1$) can be obtained from a TSP solution of G^T (one single tour) by splitting the TSP solution of G^T with each depot as the delimiter. As the result, the minsum mTSP can be solved by any TSP algorithm in principle. However, this approach has not been investigated experimentally in the literature. We fill the gap in this study by reporting the first computational results obtained by a TSP heuristic algorithm. We also use these results as additional references to assess our algorithm on the minsum mTSP instances.

3. Hybrid search with neighborhood reduction

This section introduces the hybrid search algorithm with neighborhood reduction (HSNR) designed to solve the minsum mTSP and the minmax mTSP. The general procedure is first exposed, followed by the detailed presentation of the search components.

3.1. General procedure

HSNR is a hybrid algorithm combining inter-tour optimization by exchanging information between tours and intra-tour optimization by optimizing individual tours. The inter-tour optimization component aims to improve the solution by relocating cities among different tours, while the intra-tour optimization component tries to improve an individual tour by considering it as a TSP tour. By alternating these two complementary optimization components, the algorithm is offered the promise of exploring the search space effectively. To ensure a high computational efficiency, HSNR additionally adopts a specific neighborhood reduction technique to accelerate the examination of candidate solutions.

As shown in Algorithm 1, starting from a feasible solution given by the initialization procedure (Section 3.2) (line 2), the algorithm performs a number of iterations to improve the current solution (φ) (lines 4–8). At each iteration, the solution φ is first improved by tabu search (Section 3.3.4) with the *insert* operator (Section 3.3.1) and the *cross-exchange* operator (Section 3.3.2), where cities are displaced among different tours. Once this insert and cross-exchange based inter-tour optimization is exhausted, the intra-tour optimization using the TSP heuristic EAX (Section 3.4) is triggered to improve each individual tour that was previously modified by insert and cross-exchange during inter-tour optimization. The above steps are then iterated until the stopping condition (typically a cutoff time limit) is met. During the search process, the best solution found (φ^*) is updated whenever it is needed and finally returned at the end of the algorithm.

3.2. Initial solution

The initialization procedure of HSNR first constructs μ good candidate solutions and then selects the best one as the starting solution of the HSNR algorithm. To generate each of these μ solutions, the depot 0 and a random unassigned city in N are used to initiate each of the m tours of the solution. Then the remaining cities (denoted by N^-) are added one by one and in a random order into the solution according to a greedy heuristic such that each city is inserted at the best position that increases the least either the total tour-length (for the minsum mTSP) or the current shortest tour (for the minmax mTSP).

Specifically, in the case of the minsum mTSP, a random tour r_k is picked first among the m initial tours including only the depot and another city. Then the unassigned cities in N^- are randomly considered

Algorithm 1: Main framework of HSNR for the mTSP

Input: Instance I, number of initial solutions μ, parameter τ, depth of tabu search γ, tabu tenure parameter β;
Output: The best solution φ^* found so far;

begin
  $\varphi \leftarrow Initialization(I, \mu)$; /* Generate an initial solution, Section 3.2 */
  $\varphi^* \leftarrow \varphi$; /* φ^* records the best solution found so far */
  while *Stopping condition is not met* **do**
    $< \varphi, \varphi^*, R > \leftarrow Insert\_based\_TS(\varphi, \varphi^*, \gamma, \beta)$; /* Inter-tour optimization by tabu search with the insert operator, Sections 3.3.1 and 3.3.4 */
    $< \varphi, \varphi^*, R > \leftarrow CrossExchange\_based\_TS(\varphi, \varphi^*, \gamma, \beta, \tau)$; /* Inter-tour optimization by tabu search with the cross-exchange operator, Sections 3.3.2 and 3.3.4 */
    $\varphi \leftarrow EAX(\varphi, R)$; /* Intra-tour optimization with the TSP heuristic EAX, Section 3.4 */
  end
  return φ^*;
end

one after the other and each selected city is greedily inserted into the tour r_k at the position that leads to the smallest increase of the minsum objective. For the minmax mTSP, the unassigned cities are also randomly considered one by one. However, given that its objective is to minimize the longest tour, each selected city is inserted into the *current shortest* tour r_{cs} at the position with the least increase of this shortest tour r_{cs}. It is worth noting that for the minsum mTSP, the same tour r_k is used to host all the unassigned cities in N^-, while for the minmax mTSP, the shortest tour r_{cs} used for each city insertion could change between two successive iterations.

Finally, when all cities are assigned, a feasible solution is obtained. To raise its quality, the solution is improved by the best improvement descent based on the insert and cross-exchange operators (Sections 3.3.1 and 3.3.2), followed by the optimization with the TSP heuristic EAX (Section 3.4).

3.3. Inter-tour optimization with insert and cross-exchange

The inter-tour optimization component of HSNR relies on the insert and cross-exchange operators, which are popular for solving a variety of vehicle routing problems (e.g., Arnold et al. (2019), Taillard et al. (1997), Vidal et al. (2013)). For the mTSP, the insert operator was previously used in the GVNS algorithm (Soylu, 2015) as one of its six move operators and the MASVND algorithm (Wang et al., 2017) one of the four move operators. In this work, in addition to the basic insert operator, we adopt for the first time the cross-exchange operator for solving the mTSP. Compared to insert, cross-exchange is a large neighborhood operator, which may help the algorithm to attain solutions that cannot be accessed with the insert operator.

3.3.1. Insert

Let $\varphi = \{r_1, r_2, \dots, r_m\}$ be a candidate solution composed of m tours where r_k ($k \in \{1, \dots, m\}$) represents the kth tour including the cities visited by the kth salesman. For each city, the insert operator looks for the best alternative position for the city with the minimal move gain (i.e., objective variation). When all cities are examined, the best move involving a pair of cities a and π_b is identified. Then the insert operator removes city a from tour r_a and reinserts a after city π_b in r_b ($r_a \neq r_b$). After that, tour r_a is reconnected by linking the city preceding

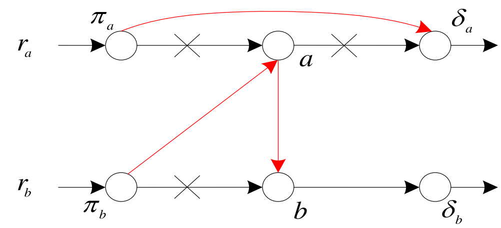

Fig. 1. Illustrative example of the insert operator. Removed links are marked with a cross and new links are marked in red.

a and the city succeeding a, while tour r_b is updated by removing the link between the city preceding b and b. Fig. 1 illustrates one insert operation with the reconnection of the two impacted tours r_a and r_b.

Let φ' be the neighboring solution that is obtained by applying the insert operator to φ and $N_I(\varphi)$ be the induced neighborhood that comprises all the neighboring solutions of φ. $N_I(\varphi)$ is bounded by $O(n^2)$ in size in the general case because there are n^2 pairs of cities.

For the minsum mTSP, this neighborhood is directly exploited by our algorithm. However, for the minmax mTSP, given that the goal is to minimize the longest tour, we limit the candidate cities to be moved by the insert operator to those of the longest tour in φ. This naturally reduces the general neighborhood $N_I(\varphi)$ to a much smaller neighborhood. In the HSNR algorithm, this reduced $N_I(\varphi)$ neighborhood is used in the case of the minmax mTSP.

Given the solution φ and a neighboring solution φ' generated by displacing city a from tour r_a to tour r_b, and the move gain $\Delta = F(\varphi) - F(\varphi')$ (F is the minsum or minmax objective function) is calculated as follows. For the minsum mTSP, the move gain Δ is computed by Eq. (3) in $O(1)$ time.

$$\Delta = c_{\pi_a \delta_a} + c_{\pi_b a} + c_{ab} - c_{\pi_a a} - c_{a \delta_a} - c_{\pi_b b} \tag{3}$$

where π_a and δ_a are the city preceding and succeeding a in tour r_a, respectively, while π_b and δ_b are the city preceding and succeeding b in tour r_b, respectively.

For the minmax mTSP, Δ is also obtained in constant time by Eq. (4).

$$\begin{aligned}
&\Delta = max\{F'(r_a), F'(r_b)\} - F(r_a),\ if\ r_b = r_s \\
&\Delta = max\{F'(r_a), F'(r_b), F(r_s)\} - F(r_a),\ if\ r_b \neq r_s \\
&F'(r_a) = F(r_a) + c_{\pi_a \delta_a} - c_{\pi_a a} - c_{a \delta_a} \\
&F'(r_b) = F(r_b) + c_{\pi_b a} + c_{ab} - c_{\pi_b b}
\end{aligned} \tag{4}$$

where r_a and r_s are the longest tour and the second longest tour, respectively

3.3.2. Cross-exchange

Given a solution $\varphi = \{r_1, \dots, r_m\}$, the cross-exchange operator modifies two tours (say r_a and r_b) to generate a neighboring solution by removing four arcs in r_a and r_b, and then adding four other arcs (see Fig. 2). Equivalently, a cross-exchange operation can be viewed as exchanging a substring $\hat{r}_a = (a, \dots, \sigma_a)$ from r_a and a substring $\hat{r}_b = (b, \dots, \sigma_b)$ from another tour r_b. Besides, one of the two substrings is reversible when they are exchanged, as shown in Fig. 2 (right) where the substring $\hat{r}_a = (a, \dots, \sigma_a)$ is reversed. Clearly, without any additional condition, this operator can lead to an extremely large neighborhood (denoted by N_{CE}) due to the size of the two exchanged substrings, making its exploration highly time-consuming.

To reduce the cross-exchange neighborhood to a reasonable size, we follow the idea of Taillard et al. (1997) developed for the vehicle routing problem (VRP) and limit the number of cities (the substring size) of the two candidate substrings $\hat{r}_a$ and $\hat{r}_b$ to τ cities at most (i.e., $|\hat{r}_a| \leq \tau$ and $|\hat{r}_b| \leq \tau$) where τ is a parameter. With this constraint, the cardinality of $N_{CE}(\varphi)$ is bounded by $O(n^2 \times \tau^2)$ in the general case.

Specifically, as shown in Fig. 2 (left), given a city a, a new neighbor in another tour needs to be found. Let π_b be such a neighbor. Suppose

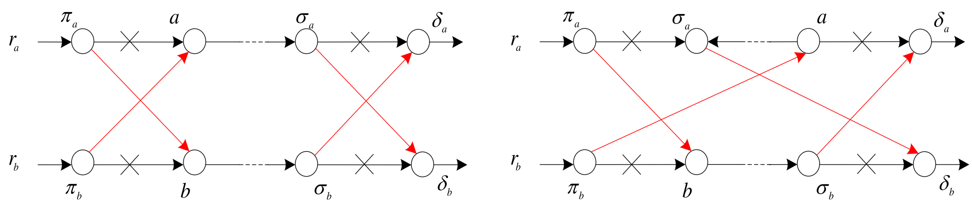

Fig. 2. Illustrative example of the cross-exchange operator. The removed arcs are marked with a cross and the added arcs are marked in red.

that (π_b, a) is added as a new edge and the edge (π_a, a) needs to be removed, since vertex a can only have two adjacent vertices. For each determined pair of vertices a and π_b, the corresponding substrings $\hat{r}_a$ and $\hat{r}_b$ can consist of at most τ consecutive cities (i.e., $1 \le |\hat{r}_a| \le \tau$ and $1 \le |\hat{r}_b| \le \tau$). For a given pair of vertices, there are τ^2 neighborhood solutions which need to be evaluated. For the specific case where the substring $\hat{r}_a$ only consists of a city ($|\hat{r}_a|$=1), the size of $\hat{r}_b$ can vary from 1 to τ ($1 \le |\hat{r}_b| \le \tau$), and thus τ neighborhood solutions need to be evaluated. Similarly, the size of substring $\hat{r}_a$ can also vary from 1 to τ. Therefore, once a pair of vertices is given, the two corresponding substrings have τ^2 combinations, leading to τ^2 neighborhood solutions needed to be evaluated. Furthermore, given that there are n^2 pairs of vertices, $N_{CE}(\varphi)$ is thus bounded by $O(n^2 \times \tau^2)$ in size. To explore the neighborhood $N_{CE}(\varphi)$, the cross-exchange operator needs to identify, among all pairs of cities, the best pair of cities, and then exchanges their corresponding substrings.

For the minsum mTSP, the move gain Δ is computed by Eq. (5).

$$\Delta = c_{\pi_a b} + c_{\pi_b a} + c_{\sigma_a \delta_b} + c_{\sigma_b \delta_a} - c_{\pi_a a} - c_{\pi_b b} - c_{\sigma_a \delta_a} - c_{\sigma_b \delta_b} \tag{5}$$

For the minmax mTSP whose objective is to minimize the longest tour, one of the two substrings is always selected from the longest tour. Let r_a be the longest tour. We first determine the start of substring $\hat{r}_a$ as city a. Then, we determine the start of the substring $\hat{r}_b$ in another tour r_b. Finally, the length of each substring based on the minimal move gain Δ is determined by Eq. (6), where r_s and r_t are the second and third longest tours, respectively.

$$\begin{aligned}
&\Delta = max\{F'(r_a), F'(r_b), F(r_s)\} - F(r_a),\ if\ r_b \neq r_s \\
&\Delta = max\{F'(r_a), F'(r_b), F(r_t)\} - F(r_a),\ if\ r_b = r_s \\
&F'(r_a) = F(r_a) + c_{\pi_a b} + F(\hat{r_b}) + c_{\sigma_b \delta_a} - c_{\pi_a a} - F(\hat{r_a}) - c_{\sigma_a \delta_a} \\
&F'(r_b) = F(r_b) + c_{\pi_b a} + F(\hat{r_a}) + c_{\sigma_a \delta_b} - c_{\pi_b b} - F(\hat{r_b}) - c_{\sigma_b \delta_b}
\end{aligned} \tag{6}$$

It is obvious that the move gain Δ can be calculated in $O(1)$ time for both the minsum and minmax objectives.

By limiting the number of cities in the two candidate substrings using the τ parameter, the cross-exchange neighborhood is reduced to the size of $O(n^2 \times \tau^2)$. However, such a neighborhood is still too large to be efficiently explored for high n values. To an ensure high computational efficiency of the proposed algorithm, we introduce in Section 3.3.3 an additional neighborhood reduction technique that allows to reduce drastically the neighborhood without scarifying the search capacity of the algorithm. This technique is also applicable to the insert neighborhood.

3.3.3. Neighborhood reduction

The difficulty of exploring the large cross-exchange neighborhood has been recognized in the VRP communities for a long time. To cope with the difficulty related to large neighborhoods, neighborhood pruning techniques have been introduced for the VRP, such as δ-nearest neighbors (Arnold and Sörensen, 2019) and granular neighborhoods (Toth and Vigo, 2003). Rather than examining the entire neighborhood, pruning techniques limit the considered neighboring solutions to specifically identified (promising) solutions. Similar neighborhood reduction techniques have been proposed to accelerate TSP algorithms for solving large instances. One popular technique is the α-nearness strategy (Helsgaun, 2000) that was designed to improve the computational efficiency of the well known Lin–Kernighan (LK) heuristic for the TSP (Lin and Kernighan, 1973) and was also applied to the VRP (Arnold et al., 2019).

The α-nearness strategy is developed by Helsgaun (2000) based on sensitivity analysis using minimum spanning 1-trees and showing a high similarity between a minimum 1-tree and an optimal TSP solution (they typically have 70% to 80% of edges in common). In other words, edges that belong to a minimum 1-tree stand a good chance of also belonging to an optimal tour and vice versa. Based on this, the α-nearness strategy uses minimum 1-trees to identify a set of promising edges S that are more likely involved in the optimal TSP solution. Given that the mTSP is an extension of the TSP, it is reasonable to use minimum 1-trees as a nearness measure for the mTSP as well. As such, the edges belonging to minimum 1-trees will be considered as promising in the sense that they are highly probably part of the optimal solution of the mTSP. Therefore, the set of promising edges S identified by the α-nearness strategy (Helsgaun, 2000) can be beneficially adopted for solving the mTSP.

In this work, we explore for the first time the idea of using the α-nearness to accelerate the insert and cross-exchange operations for the mTSP and show its practical effectiveness especially for handling large instances. The basic rationale is that one can ignore many neighboring solutions of low quality induced by the insert and cross-exchange operators and focus only on promising neighboring solutions. Consider the insert operator shown in Fig. 1 and let S be the set of promising edges identified by the α-nearness as explained next. If an edge (say (π_b, a)) belongs to S, then the corresponding move gain Δ is evaluated; otherwise, the corresponding neighboring solution is ignored. When all the edges of S are considered and the corresponding move gains are evaluated, the best neighboring solution is selected. Because the time complexity of evaluating a move gain is $O(1)$ and $|S|$ neighboring solutions are evaluated, the time complexity of evaluating the insert neighborhood is reduced to $O(|S|)$. Similarly, for the cross-exchange operator shown in Fig. 2, if an arc (say (π_b, a)) belongs to S, then the corresponding τ^2 move gains need to be evaluated. When all the edges of the set S are considered, the best neighboring solution is acquired. Therefore, the time complexity of exploring the N_{CE} neighborhood is reduced to $O(|S| \times \tau^2)$.

We now explain how the set of promising edges S is identified with the α-nearness technique based on the notion of 1-tree. As shown in Algorithm 2, the minimum 1-tree (T) for a graph $G = (V, A)$ is a minimum spanning tree covering the cities of N together with two edges of A incident to the depot 0 (lines 3–4). By inserting a new edge (i, j) to T, a cycle containing edge (i, j) in the spanning tree is generated (line 7). Then, a new 1-tree is obtained by removing the longest edge on the cycle (line 8). When all edges from V incident to vertex i are considered, the α edges (α is a parameter) corresponding to the α shortest 1-trees (T^+) are saved in the set S (lines 11–12). This process continues until all the vertices in V are considered, and then the set of promising edges S is obtained. Based on the implementation techniques in Helsgaun (2000), building the set S with the α-nearness technique requires $O(n^2)$ time.

It is worth mentioning that no neighborhood reduction technique was employed in the existing mTSP algorithms including the neighborhood search algorithm GVNS (Soylu, 2015). As we show in Section 5.1, the α-nearness technique contributes positively to the performance of the HSNR algorithm.

Algorithm 2: Generation of the set of promising edges S by the α-nearness technique

Input: Input graph $G = (V, A)$, parameter α;
Output: The set of promising edges S;
begin
 $S \leftarrow \emptyset$;
 Generate a minimum spanning tree (T^-) for the cities of N; /* Prim's algorithm */
 Generate a minimum 1-tree (T); /* By adding to T^- two shortest edges of A incident to the depot 0 */
 for $i = 0$ *to* n **do**
  for $j = 0$ *to* n **do**
   Add edge (i, j) to T;
   Generate a new 1-tree (T^+) /* By deleting the longest edge from the new cycle containing edge (i,j) in the tree (T) */
   Calculate the length of T^+;
  end
  Get the α shortest 1-trees from n 1-trees;
  Get the α edges (E) corresponding to the α shortest 1-trees;
  $S \leftarrow S \cup E$;
 end
 return S;
end

Algorithm 3: General tabu search

Input: Input solution φ, best solution φ^*, neighborhood N, depth of tabu search γ, tabu tenure parameter β;
Output: Updated best solution φ^*, ending solution φ, set of modified tours R;
begin
 $i \leftarrow 0$;
 $R \leftarrow \emptyset$;
 Initialize tabu list L;
 while $i \leq \gamma$ **do**
  Choose the best eligible neighboring solution $\varphi' \in N(\varphi)$;
  $\varphi \leftarrow \varphi'$;
  Update L and R; /* Udpdate the tabu list and set of modified tours */
  if $F(\varphi) < F(\varphi^*)$ **then**
   $\varphi^* \leftarrow \varphi$; /* Update the best solution φ^* */
   $i \leftarrow 0$;
  else
   $i \leftarrow c + 1$;
  end
 end
 return $<\varphi, \varphi^*, R>$;
end

3.3.4. Neighborhood exploration with tabu search

To examine candidate solutions of a mTSP instance, HSNR employs the well-known tabu search (TS) metaheuristic (Glover and Laguna, 1997). One notices that TS is a popular method for solving routing problems (e.g., Taillard et al. (1997), Toth and Vigo (2003)), that are more general models than the mTSP. In our case, we design the first tabu search procedure to explore the insert neighborhood N_I and the cross-exchange neighborhood N_{CE} that are reduced by the α-nearness technique of Section 3.3.3.

As described in Algorithm 3, the TS procedure starts by the initialization of the tabu list L and the set R containing the tours that are modified by the insert and cross-exchange operations. Then it performs a number of iterations until the best solution φ^* cannot be improved during γ consecutive iterations. At each iteration, tabu search identifies within the given neighborhood, the *best eligible* neighboring solution φ' according to the mTSP objectives and uses φ' to replace the current solution φ. A neighboring solution is qualified eligible if it is not forbidden by the tabu list or its quality is better than the best solution found so far φ^*. After each solution transition, the two modified tours are recorded in R and the underlying insert or cross-exchange move leading to the new solution φ' is added in the tabu list L to avoid revisiting the replaced solution. For the tabu list, we use the following mechanism. For a neighboring solution φ' where the city a is displaced from the tour r_a to another tour, a is recorded in L and not allowed to join the tour r_a again for the next t iterations, where t (called tabu tenure) is set to $\beta + rand(\beta)$ with $rand(\beta)$ being a random integer number in $\{0, \dots, \beta\}$.

During the tabu search, if its best solution found (φ^*) is not updated during γ consecutive iterations, the search is judged to be exhausted and terminates while returning the best solution found, the current solution (φ) and the set of modified tours (R).

3.4. Intra-tour optimization with the TSP heuristic EAX

Given a candidate solution $\varphi = \{r_1, \dots, r_m\}$, it is easy to observe that each individual tour r_k can be considered as a TSP tour. As the result, existing TSP algorithms (e.g., 2-opt and LK) can directly be used to optimize the mTSP objectives by minimizing an individual tour without the need for designing new optimization methods. Indeed, this idea proved to be quite effective for several VRPs (Arnold et al., 2019; Arnold and Sörensen, 2019) and has been used in the GVNS algorithm for the mTSP (with the 2-opt heuristic) (Soylu, 2015) as well. In this work, the EAX heuristic (Nagata and Kobayashi, 2013),[1] which is among the best TSP heuristics, is adopted for intra-tour optimization.

Specifically, for each tour r_k in the set R (It records the tours modified by the insert and cross-exchange operators during tabu search), EAX is applied to minimize the tour as follows. First, the tour r_k is mapped to a standard TSP tour, by renaming the cities of the tour with consecutive numbers. Second, EAX is run to optimize the TSP tour. Given that the number of cities in a tour is relatively small (typically from several tens to several hundreds of cities for the mTSP benchmark instances), EAX needs a short time to make the TSP tour optimal or close-to-optimal. Third, we map the optimized TSP tour back to the corresponding mTSP tour. Experiments showed that the intra-tour optimization using EAX contributes favorably to the performance of the HSNR algorithm.

The EAX heuristic firstly constructs randomly a population of solutions by using the coordinates of the cities and then performs a number of generations to improve the tour length. At each generation, two parents solutions are selected randomly and recombined to generate offspring solutions. Let p_A and p_B be the parent solutions, and let E_A and E_B be the sets of edges in p_A and p_B. An offspring solution is created according to the following steps.

(1) Define the undirected multigraph $G_{AB} = (V, E_A \cup E_B)$ from edge sets E_A and E_B;
(2) Partition the edges of $E_A \cup E_B$ into *AB-cycles*, where an *AB-cycle* is a cycle in G_{AB}, such that edges of E_A and edges of E_B are alternatively linked;
(3) Build an E_{set} by selecting some *AB-cycles* according to a selection criterion;

[1] https://github.com/sugia/GA-for-TSP

Table 2
Parameters tuning results.

Parameters	Section	Description	Considered values	Final value	
				Minsum	Minmax
μ	3.2	candidate initial solutions	$\{1,5,10,15,20\}$	15	20
α	3.3.3	α-nearness in 1-tree	$\{5,10,15,20,25,30\}$	20	10
τ	3.3.2	substring size	$\{2,3,4,5,6,7\}$	4	7
γ	3.3.4	depth of tabu search	$\{10,30,50,70,90,100\}$	10	50
β	3.3.4	tabu tenure parameter	$\{20,40,60,80,100\}$	60	20

(4) Build an intermediate solution E_C from p_A by removing the edges of E_A that appear in E_{set} and adding the edges of E_B that appear in E_{set}, i.e., $E_C := (E_A \backslash (E_{set} \cap E_A)) \cup (E_{set} \cap E_B)$;
(5) Generate an offspring solution by connecting all subtours of E_C to obtain a single tour.

As we show in Section 5.1, the EAX heuristic is quite beneficial for the proposed algorithm. This is the first application of this TSP heuristic within a mTSP algorithm.

4. Computational results and comparisons

This section assesses the proposed algorithm for solving both the minsum mTSP and the minmax mTSP. We show computational results on benchmark instances and comparisons with the state-of-the-art algorithms.

4.1. Benchmark instances

Our experiments are based on two sets of 77 instances covering small, medium and large instances (available from the link of footnote 4.2).

Set I (41 instances): These instances were introduced in Carter and Ragsdale (2006), Brown et al. (2007), Wang et al. (2017). Carter and Ragsdale (2006) presented 12 instances using 3 TSP graphs (with 51, 100, 150 cities and 3, 5, 10, 20 and 30 tours), while Brown et al. (2007) also defined 12 instances using 3 TSP graphs (from 51 cities and 3 tours up to 150 cities and 30 tours). Note that among these 3 graphs adopted in Brown et al. (2007), only one graph (*gtsp150*) is not used in Carter and Ragsdale (2006). Therefore, most of the instances in Carter and Ragsdale (2006) and Brown et al. (2007) share the same features. We thus exclude the redundant instances and keep 17 distinct instances out of these 24 instances. For these 17 instances, the best-known objective values are available in the literature for both mTSP objectives. Wang et al. (2017) defined 31 instances using 8 graphs (with 51–1173 cities and 3–20 tours) and tested them only for the minmax mTSP. Among the 8 used graphs, one is a graph used in Carter and Ragsdale (2006) and one is a graph used in Brown et al. (2007). By eliminating these redundant instances, we retain 24 instances out of the 31 instances. For these instances, the best-known objective values are available only for the minmax mTSP. The instances of Set I are limited to 1173 cities and 30 tours and their optimal values are still unknown in the literature.

Set II (36 instances): This is a new set of large instances with 1379–5915 cities and 3–20 tours introduced in this study. Like previous benchmark instances, these instances were generated from 9 TSP graphs in TSPlib[2] (*nrw1379, fl1400, d1655, u2152, pr2392, pcb3038, fl3795, fnl4461, rl5915*), which come from different practical problems. The optimal values for these instances are unknown.

Note that most of these instances involve distance matrices whose values are real numbers. Our HSNR algorithm operates directly with these real number distances and reports its results in real numbers.

4.2. Experimental protocol and reference algorithms

Parameter setting. HSNR has 5 parameters: number of candidate solutions for initialization μ, neighborhood reduction parameter α, substring size τ, depth of tabu search γ and tabu tenure parameter β. In order to calibrate these parameters, the "IRACE" package (López-Ibáñez et al., 2016) was used to automatically identify a set of suitable values. The tuning was performed on 8 representative instances (with 150–1173 cities). For the experiment, the tuning budget was set to 1080 runs, with a cutoff time of $n/100$ minutes. The candidate values of these parameters and their final values given by IRACE are shown in Table 2.

Reference algorithms. According to the literature, five algorithms (IWO & ABC(VC) (Pandiri and Singh, 2015), GVNS (Soylu, 2015), MASVND (Wang et al., 2017) and ES (Karabulut et al., 2021)) represent the state-of-the-art for solving the mTSP (MASVND for the minmax mTSP only). Thus these algorithms are adopted as the main references for our comparative studies. Given that only one code is available (an executable code of ES kindly provided by its authors), we faithfully re-implemented ABC(VC), IWO, GVNS and MASVND (denoted by re-ABC(VC), re-IWO, re-GVNS and re-MASVND) and verified that our implementations were able to match the results reported in Pandiri and Singh (2015), Soylu (2015), Wang et al. (2017).

Finally, as indicated in Section 2, the minsum mTSP can be transformed to the standard TSP. We provide the results obtained by the TSP heuristic EAX (Nagata and Kobayashi, 2013) in Appendix A.2.

Experimental setting. HSNR and the re-implemented reference algorithms were programmed in C++[3] and complied with the g++ compiler with the -O3 option. All the experiments were conducted on a computer with an Intel Xeon E5-2670 processor of 2.5 GHz CPU and 6 GB RAM running Linux. Given the stochastic nature of the compared algorithms, each algorithm was run 20 times on each instance with different random seeds. We used the default parameter setting of Table 2 to run HSNR, while for the reference algorithms, we adopted their default parameter settings given in Pandiri and Singh (2015), Soylu (2015), Wang et al. (2017).

Stopping condition. Each run of the compared algorithm was given the same cutoff time of $(n/100) \times 4$ minutes. This cutoff time allows all the compared algorithms to converge to their best possible solutions. Additional results under shorter cutoff conditions are reported in Appendix A.1.

4.3. Computational results and comparison

This section reports the comparative results between the proposed HSNR algorithm and the reference algorithms for the minsum mTSP and the minmax mTSP. The results are obtained according to the experimental protocol above and reported for the two sets of 77 benchmark instances (listed in increasing order of numbers of cities). Note that the executable code of ES failed to run on the instances of Set II due to unknown reasons. So its results are ignored as far as Set II is concerned.

For each instance, we show the best-known objective value BKS ever reported in the literature (when it is available), the best objective

[2] http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/index.html

[3] The source codes of these algorithms and the instances will be available at https://github.com/pengfeihe-angers/mTSP

Table 3
The minsum mTSP: comparative results between HSNR and four reference algorithms on the 41 instances of Set I with a cutoff time of $(n/100) \times 4$ minutes.

Instance	BKS	re-ABC(VC) (2015)		re-IWO (2015)		re-GVNS (2015)		ES (2021)		HSNR (this work)		
		Best	*Avg.*	*Best*	*Avg.*	*Best*	*Avg.*	*Best*	*Avg.*	*Best*	*Avg.*	*Gap(%)*
mtsp51–3	446	445.99	445.99	445.99	445.99	445.99	445.99	449.01	452.18	445.99	445.99	0.00
mtsp51–5	472	471.69	471.69	471.69	471.69	471.69	471.69	474.75	476.17	471.69	471.69	0.00
mtsp51–10	580	580.72	580.72	580.72	580.72	580.72	580.72	585.47	588.00	580.72	580.72	0.00
mtsp100–3	21798	21797.60	21825.66	21797.60	21920.60	21797.60	21807.54	21797.60	21952.94	21797.60	21797.60	0.00
mtsp100–5	23175	23256.10	23334.94	23174.90	23282.45	23174.90	23203.09	23188.70	23325.02	23174.90	23174.90	0.00
mtsp100–10	26927	27474.70	27775.28	26961.10	27065.57	27074.80	27227.84	27137.60	27235.18	26926.60	26983.51	0.00
mtsp100–20	38245	39581.30	40876.66	38529.60	39165.78	38844.90	39105.67	38603.80	38603.80	38245.10	38259.98	0.00
rand100–3	–	8012.13	8033.16	8012.13	8018.91	8012.13	8046.32	8012.13	8186.56	8012.13	8012.13	0.00
rand100–5	–	8257.84	8337.79	8223.91	8252.37	8232.91	8331.35	8308.88	8450.00	8223.91	8223.91	0.00
rand100–10	–	9635.39	9762.62	9366.80	9485.79	9377.28	9550.08	9366.80	9412.17	9366.80	9366.80	0.00
rand100–20	–	14175.00	14450.33	13529.20	13627.89	13587.70	13747.45	13487.80	13522.68	13404.10	13404.10	**−0.62**
mtsp150–3	37957	38182.30	38307.72	38025.40	38251.43	38476.60	38753.09	38206.00	38810.89	37910.70	37910.70	**−0.12**
mtsp150–5	38714	39173.30	39472.84	38813.90	39155.66	39456.20	39743.72	39132.10	39470.06	38714.40	38722.24	0.00
mtsp150–10	42203	43429.10	43598.38	42482.30	42897.26	43028.60	43415.44	42690.70	42809.47	42234.30	42310.82	0.07
mtsp150–20	53343	55059.60	55635.92	53902.90	54377.61	54797.80	55087.56	53902.30	54078.21	53351.30	53483.13	0.02
mtsp150–30	68541	70669.30	71230.93	69052.80	69757.77	69841.40	70281.72	68955.10	69186.83	68455.90	68539.07	**−0.12**
gtsp150–3	6590	6606.75	6636.44	6595.60	6661.47	6663.19	6732.97	6615.62	6673.44	6574.20	6574.52	**−0.24**
gtsp150–5	6652	6768.64	6815.25	6683.13	6765.03	6744.64	6846.12	6688.90	6746.81	6655.11	6655.11	0.05
gtsp150–10	7342	7589.85	7730.54	7401.76	7437.68	7413.19	7637.65	7360.13	7401.72	7332.11	7332.11	**−0.13**
gtsp150–20	9525	10114.90	10480.57	9625.15	9875.01	9891.21	10048.82	9535.07	9562.90	9512.23	9513.38	**−0.13**
gtsp150–30	12976	13870.50	14289.75	13180.80	13474.55	13576.80	13767.78	12980.80	13062.24	12966.50	12969.05	**−0.07**
kroA200–3	–	29735.10	29959.45	29584.90	29644.31	30114.10	30616.59	29649.00	29921.15	29539.50	29539.50	**−0.15**
kroA200–5	–	30807.10	31062.41	29982.40	30469.71	30314.20	31188.11	30213.20	30410.36	29916.20	29916.20	**−0.22**
kroA200–10	–	33971.50	34927.99	33077.00	33499.48	33558.50	33958.52	32901.70	33149.47	32613.40	32613.40	**−0.88**
kroA200–20	–	45590.30	46797.22	43290.60	44201.81	43253.50	44081.32	41686.60	41997.94	41439.20	41522.45	**−0.59**
lin318–3	–	43514.90	43823.48	42447.60	42792.41	44388.40	46422.39	43181.20	43643.25	42404.60	42404.60	**−0.10**
lin318–5	–	45000.30	45681.29	43553.70	43949.98	46172.40	47684.53	44236.50	44595.94	43315.00	43315.00	**−0.55**
lin318–10	–	52335.30	53399.17	48459.80	49744.27	50409.50	53336.21	48389.70	48798.79	47325.50	47333.21	**−2.20**
lin318–20	–	75819.00	81661.46	68883.50	73448.10	65757.80	68530.19	60566.00	61204.82	59893.20	60416.35	**−1.11**
att532–3	–	29621.00	29750.20	29295.00	29517.35	29931.00	30835.25	29321.00	29634.40	28242.00	28242.00	**−3.59**
att532–5	–	30692.00	30962.90	30393.00	30695.75	30829.00	31903.50	30253.00	30523.10	28945.00	28945.00	**−4.32**
att532–10	–	35159.00	35715.75	34234.00	34883.75	33946.00	35024.30	32422.00	32573.90	31001.00	31038.80	**−4.38**
att532–20	–	47480.00	48333.25	45672.00	46831.95	39706.00	41529.90	37813.00	38127.10	36305.00	36696.65	**−3.99**
rat783–3	–	9755.60	9786.55	9698.51	9768.69	9569.58	9807.83	9728.67	9792.58	8880.03	8880.64	**−7.21**
rat783–5	–	9971.01	10011.91	9922.56	9982.26	9838.55	10156.87	9815.50	9898.41	8964.80	8964.90	**−8.67**
rat783–10	–	10745.80	10862.35	10643.70	10771.53	10158.40	10582.99	10056.80	10202.08	9265.64	9275.16	**−7.87**
rat783–20	–	13659.60	13914.47	13186.90	13747.13	11104.90	11962.38	10741.80	10926.51	10172.60	10272.95	**−5.30**
pcb1173–3	–	64553.70	64846.11	64533.50	64739.49	61929.50	63828.01	65007.20	65664.84	57167.20	57174.12	**−7.69**
pcb1173–5	–	65845.60	66231.05	65827.00	66105.02	64154.60	66595.52	64818.40	66792.82	57628.80	57654.20	**−10.17**
pcb1173–10	–	70907.60	71683.97	69994.80	71198.91	65816.90	68882.00	66611.60	67600.29	59241.90	59299.07	**−9.99**
pcb1173–20	–	85807.50	88122.07	85228.60	88136.83	73482.80	76727.24	71489.70	73905.52	64063.60	65102.08	**−10.39**
AVG	–	31124.99	31649.42	30360.16	30856.10	29900.63	30694.79	29423.95	29740.75	28309.28	28374.09	–
BKS#	–	0	0	0	0	0	0	0	0	**27**	**38**	–
p-value	–	1.68E−07	3.57E−07	5.39E−07	5.26E−07	2.48E−07	7.74E−08	7.74E−08	2.42E−08	–	–	–

value obtained by an algorithm *Best* and the average objective value *Avg.*. For our HSNR algorithm, we additionally report the gap of its best objective value to the previous best objective value calculated as $Gap(\%) = 100(Best - AllBest)/AllBest$ with $Best$ and $AllBest$ being respectively the best objective value of HSNR and the best objective value from all reference algorithms (including those published in the literature). Given that the mTSP is a minimization problem, a negative gap indicates an improved best result. The background of the top results for each instance is highlighted in dark gray; the second best results in medium gray; and the worst results in the lightest gray. Note that in the literature, the results are rounded to the nearest integers, and we report our results in more precise real values.

For each set of instances, we additionally report the following information. For the best and average objective values of each algorithm, AVG is the average value over the instances of one benchmark set. For each algorithm, BKS# indicates the number of instances out of all the instances of the set for which the algorithm reports the best objective value.

Finally, to assess the statistically significant difference between the results of the HSNR algorithm and the results of each reference algorithm, we show the *p-values* from the Wilcoxon signed-rank test applied to the best and average objective values with a confidence level of 0.05. A *p-value* smaller than 0.05 rejects the null hypothesis.

4.3.1. Results for the minsum mTSP

Tables 3 and 4 show the comparative results of the compared algorithms for the 77 instances of Set I and Set II, respectively.

From Table 3, we can make the following comments about the instances of Set I. First, for the 17 instances for which the best-known results (BKS) are available, HSNR finds 6 improved results (with an improvement gap up to −0.24%), 7 equal results and 4 worse results. Second, for the remaining 24 instances of Set I, HSNR clearly outperforms the reference algorithms both in terms of the best and average results, with more important improvements for the largest instances with at least 200 cities (improvement gap up to 10.39% for the largest instance). Also, even the average results of HSNR are better than the best results of the reference algorithms. Third, the small *p-values* from the Wilcoxon signed-rank tests confirm the statistical difference between the HSNR algorithm and the reference algorithms in terms of the best and average results.

From Table 4 on the large instances of Set II, we observe that the dominance of the HSNR algorithm over the reference algorithms

Table 4
The minsum mTSP: comparative results between HSNR and three reference algorithms on the 36 instances of Set II with a cutoff time of $(n/100) \times 4$ minutes.

Instance	re-ABC(VC) (2015)		re-IWO (2015)		re-GVNS (2015)		HSNR (this work)		
	Best	*Avg.*	*Best*	*Avg.*	*Best*	*Avg.*	*Best*	*Avg.*	*Gap(%)*
nrw1379–3	62099.80	62413.66	62211.90	62384.95	62449.60	63614.69	56775.70	56775.70	**−8.57**
nrw1379–5	62853.40	63036.26	62788.40	63011.51	63593.80	65998.39	56992.60	56999.16	**−9.23**
nrw1379–10	64985.10	65396.08	65147.40	65392.47	65011.90	69268.91	57636.20	57795.15	**−11.31**
nrw1379–20	72415.90	73267.10	71915.30	73075.37	69900.30	74382.44	59618.40	60278.03	**−14.71**
fl1400–3	21733.90	21819.77	21682.60	21771.70	24456.90	25566.53	21169.40	21169.47	**−2.37**
fl1400–5	23051.40	23179.70	22841.20	23068.25	24030.00	26993.65	22066.20	22238.10	**−3.39**
fl1400–10	27960.10	28563.58	27556.10	27933.99	28276.70	30150.92	24373.90	25069.65	**−11.55**
fl1400–20	44588.20	47458.31	44715.00	45981.11	32713.30	34886.35	29579.20	31966.86	**−9.58**
d1655–3	76672.20	77095.10	76471.40	76887.31	78155.30	79462.89	68364.40	68370.50	**−10.60**
d1655–5	83908.00	84208.31	83221.80	83962.59	86806.30	89456.39	74273.50	74292.65	**−10.75**
d1655–10	102457.00	103865.80	102268.00	103386.30	100732.00	105478.45	89262.50	89856.83	**−11.39**
d1655–20	146870.00	149739.75	147454.00	149130.20	134860.00	143426.30	121373.00	124263.45	**−10.00**
u2152–3	75107.40	75322.56	74957.90	75399.52	73757.10	75777.34	65064.90	65072.31	**−11.78**
u2152–5	75533.50	76109.51	75686.10	76083.68	74271.40	78510.40	65201.70	65219.93	**−12.21**
u2152–10	78836.20	79676.56	78726.40	79471.17	75482.90	83485.66	65762.50	66291.71	**−12.88**
u2152–20	89564.50	91776.90	89331.80	91322.73	80486.60	85760.90	67993.10	71115.74	**−15.52**
pr2392–3	428886.00	430482.05	428802.00	429994.15	423607.00	433789.50	378661.00	378661.00	**−10.61**
pr2392–5	433633.00	437696.40	435449.00	438130.75	426073.00	444213.90	380061.00	380069.40	**−10.80**
pr2392–10	462078.00	465864.35	458177.00	465361.70	441436.00	476382.30	387498.00	389012.85	**−12.22**
pr2392–20	539219.00	549174.10	542251.00	549066.05	459442.00	502937.95	417424.00	421532.30	**−9.15**
pcb3038–3	156742.00	157141.25	156844.00	157227.80	153338.00	155312.45	137916.00	137925.00	**−10.06**
pcb3038–5	158160.00	158614.05	157607.00	158559.90	156678.00	159923.10	138121.00	138123.20	**−11.84**
pcb3038–10	162709.00	164019.75	163743.00	164470.35	156525.00	162016.80	139142.00	139379.85	**−11.11**
pcb3038–20	181677.00	183532.75	181894.00	183531.15	153084.00	170283.40	144295.00	146491.65	**−5.74**
fl3795–3	32749.00	32983.87	32678.10	32817.07	34634.30	37772.26	29589.90	29823.75	**−9.45**
fl3795–5	33924.60	34497.01	33833.20	34198.05	37162.40	40342.25	30480.80	31048.26	**−9.91**
fl3795–10	39470.20	40288.27	38864.50	39779.70	36823.70	41088.57	32729.60	35467.72	**−11.12**
fl3795–20	53852.70	55606.56	53723.40	55121.13	41337.00	45838.94	39083.80	45437.27	**−5.45**
fnl4461–3	204334.00	204844.15	204490.00	204833.45	203756.00	206706.75	182888.00	182890.85	**−10.24**
fnl4461–5	205639.00	206196.00	205745.00	206132.15	207600.00	212214.50	183074.00	183076.50	**−10.97**
fnl4461–10	210341.00	211064.95	210158.00	210906.80	215447.00	224158.65	183808.00	184811.75	**−12.54**
fnl4461–20	224749.00	225855.50	223448.00	225219.15	221402.00	236283.55	191025.00	193356.10	**−13.72**
rl5915–3	676316.00	678576.60	676268.00	679179.35	666852.00	707708.75	565949.00	566066.70	**−15.13**
rl5915–5	678177.00	680809.90	673768.00	680248.85	703003.00	746016.20	566626.00	566780.55	**−15.90**
rl5915–10	692109.00	694947.55	689402.00	694087.15	783210.00	811408.35	569619.00	573689.20	**−17.37**
rl5915–20	744400.00	752084.65	742284.00	750748.75	777638.00	861515.54	597878.00	609385.79	**−19.45**
AVG	206327.84	207978.02	206011.24	207718.79	204834.24	216892.61	173371.56	174716.80	–
BKS#	0	0	0	0	0	0	**36**	**36**	–
p-value	1.68E−07	1.68E−07	1.68E−07	1.68E−07	1.68E−07	1.68E−07	–	–	–

is even more significant. Indeed, HSNR consistently reports better results in terms of the best and average values, with improvement gaps from 2.37% to 19.45% compared to the best results of the reference algorithms. Once again, even the average results of HSNR are far better than the best results of the compared algorithms. Finally, the Wilcoxon signed-rank tests confirm the statistical difference of these comparisons.

To further assess the compared algorithms, we also present the performance profiles (Dolan and Moré, 2002) to visually illustrate the performance of each algorithm. Performance profiles rely on a specific performance metric (in our case, we use f_{best} and f_{avg}). To compare a set of algorithms S over a set of problems Q, the performance ratio is defined by $r_{s,q} = \frac{f_{s,q}}{min\{f_{s,q}: s \in S, q \in Q\}}$. If an algorithm does not report result for a problem q, $r_{s,q} = +\infty$. The performance function of an algorithm s is computed by $Q_s(\tau) = \frac{|q \in Q | r_{s,q} \leq \tau|}{|Q|}$. The value $Q_s(\tau)$ computes the fraction of problems that algorithm s can solve with at most τ many times the cost of the best algorithm. For example, $Q_s(1)$ equals the number of problems that algorithm s solved better than, or as good as the other algorithms in Q. Similarly, the value $Q_s(r_f)$ is the maximum number of problems that algorithm s solved. Therefore, $Q_s(1)$ and $Q_s(r_f)$ represent the efficiency and robustness of algorithm s. Fig. 3 visually illustrates the competitiveness of HSNR in terms of the best and average values on the benchmark 77 instances. Indeed, HSNR has a much higher $Q_s(1)$ value compared to the reference algorithms, by finding better or equal results for nearly all instances. Furthermore, HSNR also reaches $Q_s(r_f)$ first, indicating a high robustness of our approach. In brief, compared with the reference algorithms, HSNR is the best solution approach for the minsum mTSP on both small and large scale instances.

Finally, since the minsum mTSP can be transformed to the TSP, we show in Appendix A.2 the results obtained by the effective TSP heuristic EAX (Nagata and Kobayashi, 2013).

4.3.2. Results for the minmax mTSP

We now assess the performance of the HSNR algorithm for the minmax mTSP. For this problem, ABC(VC) (Pandiri and Singh, 2015), IWO (Pandiri and Singh, 2015), GVNS (Soylu, 2015), MASVND (Wang et al., 2017) and ES (Karabulut et al., 2021) are the state-of-the-art algorithms, which are used for our comparative study. Note that for three graphs *kroA200, lin318, att532*, the initial solutions of HSNR are generated in such a way that each city is greedily inserted in an arbitrary random tour, not limited to the shortest tour.

Tables 5 and 6 report the computational results of the compared algorithms on Set I and Set II. From the tables, we observe that in terms of the best objective values, HSNR reaches the best results on 48 out of the 77 instances and matches the best results of the compared algorithms on 25 instances. Only for four instances, HSNR reports a slightly worse result with a gap to the best objective value no larger than 0.61%. In terms of the average objective value, HSNR reports 54 dominating values. It is worth noting that the average results of HSNR are better than the best results of the reference algorithms. Third, the dominance of HSNR over the reference algorithms is better demonstrated on the large instances of Set II with up to 32.81% improvements of their best results. Finally, the small *p-values* ($\ll 0.05$) confirm the statistically significant differences between HSNR and the reference algorithms for the best and average results.

Table 5
The minmax mTSP: comparative results of HSNR and five reference algorithms on Set I with a cutoff time of $(n/100) \times 4$ minutes.

Instance	BKS	re-ABC(VC) (2015)		re-IWO (2015)		re-GVNS (2015)		ES (2021)		re-MASVND (2017)		HSNR (this work)		
		Best	*Avg.*	*Best*	*Avg.*	*Best*	*Avg.*	*Best*	*Avg.*	*Best*	*Avg.*	*Best*	*Avg.*	*Gap*(%)
mtsp51–3	159.57	159.57	159.57	159.57	159.57	159.57	159.85	159.57	159.57	159.57	159.57	159.57	159.85	0.00
mtsp51–5	118.13	118.13	118.13	118.13	118.13	118.13	118.13	118.13	118.13	118.13	118.13	118.13	118.13	0.00
mtsp51–10	112.07	112.07	112.07	112.07	112.07	112.07	112.07	112.07	112.07	112.07	112.07	112.07	112.07	0.00
mtsp100–3	8509.00	8544.69	8590.07	8509.16	8510.86	8544.34	8590.97	8509.16	8649.75	8509.16	8602.30	8509.16	8513.75	0.00
mtsp100–5	6766.00	6819.80	6921.35	6780.37	6833.45	6767.82	6776.61	6767.02	6832.74	6767.02	6801.75	6765.73	6770.67	0.00
mtsp100–10	6358.00	6358.49	6360.86	6358.49	6358.49	6358.49	6358.49	6358.49	6358.49	6358.49	6358.49	6358.49	6358.49	0.00
mtsp100–20	6358.00	6358.49	6358.49	6358.49	6358.49	6358.49	6358.49	6358.49	6358.49	6358.49	6358.49	6358.49	6358.49	0.00
rand100–3	–	3032.58	3044.01	3031.95	3031.95	3057.83	3074.55	3031.95	3084.49	3031.95	3047.71	3031.95	3032.67	0.00
rand100–5	–	2438.19	2462.75	2409.90	2429.50	2413.57	2427.75	2409.63	2422.41	2409.63	2428.35	2411.68	2415.00	0.09
rand100–10	–	2299.16	2299.16	2299.16	2299.16	2299.16	2299.16	2299.16	2299.16	2299.16	2299.16	2299.16	2299.16	0.00
rand100–20	–	2299.16	2299.16	2299.16	2299.16	2299.16	2299.16	2299.16	2299.16	2299.16	2299.16	2299.16	2299.16	0.00
mtsp150–3	13151.00	13497.20	13645.64	13078.40	13259.97	13595.30	13730.70	13303.80	13526.70	13234.10	13411.26	13075.80	13169.37	**−0.02**
mtsp150–5	8466.00	8895.22	9329.95	8477.96	8650.11	8928.35	8984.99	8563.08	8757.22	8493.62	8686.61	8477.96	8538.83	0.14
mtsp150–10	5557.00	6020.34	6189.74	5751.41	5851.79	5825.83	5862.65	5625.32	5718.45	5666.45	5763.28	5590.64	5604.92	0.61
mtsp150–20	5246.00	5262.55	5307.20	5246.49	5247.62	5246.49	5246.49	5246.49	5247.21	5246.49	5246.49	5246.49	5246.49	0.00
mtsp150–30	5246.00	5246.49	5246.49	5246.49	5246.49	5246.49	5246.49	5246.49	5246.49	5246.49	5246.49	5246.49	5246.49	0.00
gtsp150–3	2407.34	2445.92	2486.43	2407.34	2421.40	2443.10	2467.07	2423.17	2491.00	2433.80	2468.12	2407.34	2435.49	0.00
gtsp150–5	1742.00	1858.94	1894.78	1744.57	1777.90	1795.75	1807.36	1751.85	1797.71	1744.26	1779.32	1741.71	1743.48	**−0.02**
gtsp150–10	1554.00	1562.13	1578.49	1554.64	1557.22	1554.64	1556.10	1554.64	1554.64	1554.64	1559.10	1554.64	1554.64	0.00
gtsp150–20	1554.00	1554.64	1554.64	1554.64	1554.64	1554.64	1554.64	1554.64	1554.64	1554.64	1554.64	1554.64	1554.64	0.00
gtsp150–30	1554.00	1554.64	1554.64	1554.64	1554.64	1554.64	1554.64	1554.64	1554.64	1554.64	1554.64	1554.64	1554.64	0.00
kroA200–3	10768.10	11223.30	11511.69	10801.30	10965.59	11061.60	11256.60	10883.30	11174.70	10833.60	11136.70	10748.10	10987.69	**−0.19**
kroA200–5	7415.54	8417.81	8689.64	7497.21	7697.45	7693.65	7763.61	7536.91	7770.43	7484.17	7634.61	7418.87	7494.44	0.04
kroA200–10	6223.22	6299.77	6456.58	6223.22	6255.37	6224.39	6270.38	6223.22	6240.52	6223.22	6266.44	6223.22	6223.22	0.00
kroA200–20	6223.22	6223.22	6223.22	6223.22	6223.22	6223.22	6223.22	6223.22	6223.22	6223.22	6223.22	6223.22	6223.22	0.00
lin318–3	16088.73	17339.30	18119.72	16133.40	17006.92	16362.30	16532.08	16349.60	16797.80	16551.60	16886.01	15902.50	16207.05	**−1.16**
lin318–5	11524.29	13893.60	14197.25	12291.60	12882.38	11903.00	12069.83	11619.60	11907.90	11741.60	12023.74	11295.20	11596.35	**−1.99**
lin318–10	9731.17	10444.40	10588.94	9861.64	9963.31	9742.98	9754.76	9731.18	9736.17	9731.17	9797.38	9731.17	9731.17	0.00
lin318–20	9731.17	9750.68	9787.11	9731.17	9731.18	9731.17	9731.17	9731.17	9731.18	9731.17	9731.17	9731.17	9731.17	0.00
att532–3	–	12011.00	12169.15	11258.00	11525.50	10656.00	10762.95	10585.00	10953.90	10566.00	10853.05	10231.00	10565.30	**−3.17**
att532–5	–	8899.00	9215.85	8518.00	8895.80	8019.00	8146.85	7344.00	7463.50	7279.00	7429.50	7067.00	7334.00	**−2.91**
att532–10	–	6696.00	6829.10	6427.00	6552.90	6449.00	6650.00	5761.00	5806.75	5745.00	5809.00	5709.00	5738.90	**−0.63**
att532–20	–	5912.00	6008.15	5745.00	5836.90	5991.00	6090.70	5580.00	5580.05	5580.00	5582.90	5580.00	5580.00	0.00
rat783–3	3272.95	3821.06	3948.67	3688.79	3786.16	3359.95	3406.72	3444.20	3485.74	3295.90	3364.20	3187.90	3237.29	**−2.60**
rat783–5	2092.77	2748.91	2824.59	2627.74	2781.71	2252.93	2289.48	2125.53	2189.92	2120.74	2145.38	2006.46	2044.32	**−4.12**
rat783–10	1360.89	1725.45	1766.78	1692.31	1718.75	1440.55	1457.26	1373.46	1396.78	1396.92	1424.76	1334.76	1345.88	**−1.92**
rat783–20	1231.69	1386.96	1416.13	1371.32	1390.09	1235.21	1240.96	1231.69	1231.69	1237.97	1244.26	1231.69	1231.69	0.00
pcb1173–3	22252.31	27011.10	27466.52	25557.90	26439.83	21781.10	22260.07	23193.10	23640.00	22255.20	22941.19	20813.80	21144.92	**−4.44**
pcb1173–5	14099.50	18692.20	19292.07	18703.50	19226.82	14566.20	14853.24	14333.00	14601.30	14088.40	14305.57	13032.30	13216.99	**−7.50**
pcb1173–10	8160.25	11463.30	11663.99	11170.00	11388.41	8679.08	8932.06	8222.40	8352.07	8452.28	8637.95	7758.26	7897.20	**−4.93**
pcb1173–20	6549.14	8220.93	8519.74	8132.08	8356.53	6604.14	6627.65	6549.14	6577.59	6549.14	6623.91	6528.86	6528.86	**−0.31**
AVG	–	6795.57	6931.91	6553.84	6689.21	6249.03	6314.78	6177.75	6268.40	6152.15	6241.85	6015.33	6076.73	–
BKS#	3	0	0	0	4	0	0	0	0	0	0	**15**	**21**	–
p-value	–	1.17E−06	4.37E−07	3.09E−05	1.21E−05	5.61E−06	3.79E−06	6.08E−05	1.02E−06	4.03E−05	1.92E−06	–	–	–

Table 6
The minmax mTSP: comparative results of HSNR and four reference algorithms on Set II with a cutoff time of $(n/100) \times 4$ minutes.

Instance	re-ABC(VC) (2015)		re-IWO (2015)		re-GVNS (2015)		re-MASVND (2017)		HSNR (this work)		
	Best	*Avg.*	*Best*	*Avg.*	*Best*	*Avg.*	*Best*	*Avg.*	*Best*	*Avg.*	*Gap(%)*
nrw1379–3	25566.10	26173.22	24401.20	25204.30	21746.00	21946.34	22236.40	23349.12	20495.90	20765.70	**–5.75**
nrw1379–5	17765.00	18097.04	17636.70	18019.88	14105.50	14382.16	13368.80	13847.14	12416.50	12652.56	**–7.12**
nrw1379–10	10185.60	10595.08	10145.40	10404.29	8026.78	8240.14	7583.59	7748.69	7114.71	7212.24	**–6.18**
nrw1379–20	7306.79	7407.53	7082.11	7310.74	5492.36	5579.57	5495.31	5571.01	5370.82	5371.08	**–2.21**
fl1400–3	10000.80	10206.21	9860.63	10140.92	9192.65	9563.03	9562.25	10094.49	9192.38	9621.59	**0.00**
fl1400–5	8478.34	8656.84	8422.09	8590.95	6305.31	6477.52	6803.42	7134.69	6268.25	6783.62	–0.59
fl1400–10	7402.60	7554.95	7359.74	7485.43	5763.26	5763.26	5763.26	5763.74	5763.26	5763.26	0.00
fl1400–20	6564.38	6848.50	6687.79	6819.01	5763.26	5763.26	5763.26	5763.26	5763.26	5763.26	0.00
d1655–3	32743.40	33748.32	32293.30	33051.92	26503.10	27189.53	30143.30	42813.48	25229.30	25635.98	**–4.81**
d1655–5	24456.10	24852.69	24146.80	24854.66	19003.50	19369.46	18719.10	19376.15	17181.20	17454.32	**–8.22**
d1655–10	16577.80	16777.54	15868.50	16569.21	12747.10	12975.06	12454.00	12623.92	11660.00	11816.04	**–6.38**
d1655–20	12417.00	12766.90	12165.20	12605.33	9857.22	9919.59	9893.04	10120.03	9598.94	9607.73	**–2.62**
u2152–3	33688.20	34177.35	32354.70	33246.73	25949.70	26569.84	43724.70	44187.24	24207.40	24747.01	**–6.71**
u2152–5	23228.30	23571.56	23356.00	23534.98	16950.50	17387.25	17653.10	18404.22	15055.10	15394.85	**–11.18**
u2152–10	13824.90	14211.38	13454.40	13985.98	9927.97	10193.25	9458.60	9600.79	8624.61	8780.91	**–8.82**
u2152–20	9341.27	9609.48	9223.98	9532.87	6652.68	6811.78	6550.73	6727.13	6171.89	6225.82	**–5.78**
pr2392–3	192348.00	195388.25	186013.00	190584.70	151300.00	155742.30	254034.00	256052.65	141627.00	143703.00	**–6.39**
pr2392–5	134496.00	135676.40	133780.00	135073.30	102087.00	104603.90	104977.00	132626.10	88083.20	89582.83	**–13.72**
pr2392–10	82834.80	84149.45	80135.10	83131.04	58955.70	60860.42	55337.60	56650.63	51085.30	52100.80	**–7.68**
pr2392–20	56415.60	58338.68	56941.00	58490.18	39021.20	39776.00	38175.60	39420.33	35325.30	35709.02	**–7.47**
pcb3038–3	67464.20	70003.31	66159.20	68931.99	55841.90	56661.80	85795.40	86481.39	51049.90	51582.38	**–8.58**
pcb3038–5	46209.60	46858.06	46465.70	46938.10	36115.80	37126.47	66560.40	67071.90	31140.20	31495.59	**–13.78**
pcb3038–10	27294.50	27700.36	26954.20	27659.07	20280.40	20851.11	19198.20	19620.41	16949.90	17450.44	**–11.71**
pcb3038–20	18106.10	18507.84	17772.50	18323.59	12560.30	12955.65	12012.20	12643.54	10835.00	11004.40	**–9.80**
fl3795–3	17156.10	17466.60	16611.70	17207.03	13158.30	13793.36	22444.50	22801.50	11971.00	12815.54	**–9.02**
fl3795–5	13476.10	13766.95	13391.00	13809.93	9019.75	9494.51	19698.50	19877.81	7923.71	8610.84	**–12.15**
fl3795–10	10464.90	10594.86	10132.30	10500.27	5764.85	6156.61	6715.07	7120.46	5763.26	5823.89	–0.03
fl3795–20	8573.65	8708.45	8519.26	8679.69	5763.26	5763.26	5763.26	5763.75	5763.26	5763.26	0.00
fnl4461–3	90850.00	91886.08	90062.10	91143.24	76245.00	77330.30	108622.00	109798.50	66903.70	67971.34	**–12.25**
fnl4461–5	59246.10	60047.22	59532.70	60170.68	48352.60	49343.13	83650.40	84430.87	40721.20	41777.11	**–15.78**
fnl4461–10	34671.80	34942.84	34068.20	34741.06	26182.10	26871.35	25385.20	43581.63	22041.50	22891.45	**–13.17**
fnl4461–20	22113.00	22798.53	22142.80	22852.80	15810.10	16341.60	14611.30	15262.97	12630.10	13046.38	**–13.56**
rl5915–3	329296.00	334606.95	328020.00	332327.15	284176.00	314531.05	443748.00	445979.00	213864.00	226819.75	**–24.74**
rl5915–5	224206.00	226396.90	221495.00	225566.65	198641.00	201423.65	362776.00	364717.65	133457.00	145173.07	**–32.81**
rl5915–10	135096.00	137649.80	133266.00	137737.55	89353.00	98436.47	267295.00	270354.45	76585.20	84459.02	**–14.29**
rl5915–20	88081.70	92870.91	88081.70	92716.25	68724.00	70692.60	51115.20	53066.77	48958.50	60306.22	**–4.22**
AVG	53276.30	54267.03	52611.17	53831.71	42259.42	44080.18	63141.32	65456.87	35077.55	36713.40	–
BKS#	0	0	0	0	0	2	0	0	**33**	**33**	–
p-value	1.68E–07	1.68E–07	1.68E–07	1.68E–07	5.39E–07	7.79E–07	5.39E–07	1.47E–06	–	–	–

Table 7
Summary of comparative results between HSNR and the reference algorithms.

Pair	#Instances	*Best*				*Avg.*			
		#Wins	#Tiers	#Losses	*p-value*	#Wins	#Tiers	#Losses	*p-value*
Minsum									
HSNR vs. BKS	17	7	6	4	–	–	–	–	–
HSNR vs. re-ABC(VC)	77	72	5	0	1.66E–13	74	3	0	7.73E–14
HSNR vs. re-IWO	77	69	8	0	5.21E–13	74	3	0	7.73E–14
HSNR vs. re-GVNS	77	71	6	0	2.43E–13	74	3	0	7.73E–14
HSNR vs. ES	41	38	3	0	7.74E–08	41	0	0	2.42E–08
Minmax									
HSNR vs. BKS	33	12	18	3	–	–	–	–	–
HSNR vs. re-ABC(VC)	77	66	11	0	1.64E–12	67	9	1	5.69E–13
HSNR vs. re-IWO	77	57	19	1	3.69E–11	62	10	5	3.75E–12
HSNR vs. re-GVNS	77	60	17	0	1.63E–11	59	16	2	4.84E–11
HSNR vs. ES	41	21	19	1	6.08E–05	28	12	1	1.02E–06
HSNR vs. re-MASVDN	77	54	22	1	1.27E–10	63	13	2	3.74E–11

Once again, the performance profiles of Fig. 4 clearly show the competitiveness of HSNR over the compared algorithms. Indeed, HSNR has a much higher $Q_s(1)$ value compared to the reference algorithms, indicating that HSNR finds better or equal results for nearly all instances. Furthermore, HSNR reaches $Q_s(r_f)$ first, implying a high robustness of our approach. Therefore, HSNR competes favorably with the state-of-the-art algorithms for the minmax mTSP. Its competitiveness is particularly demonstrated on large instances in terms of the best and average results.

Finally, Table 7 summaries the comparative results of each pair of compared algorithms on the 77 benchmark instances, by providing the number of instances for which HSNR obtained a better (#Wins), equal (#Ties) or worse (#Losses) result compared to each reference algorithm and the BKS value.

We conclude that HSNR significantly dominates the reference algorithms for both the minsum mTSP and the minmax mTSP. Its competitiveness is even more evident on large-scale instances.

5. Analysis

The computational results and comparisons with the state-of-the-art algorithms presented in Section 4 showed high effectiveness of the

Table 8
Summary of comparative results between HSNR and the four compared algorithms.

Pair	#Instances	Best				Avg.			
		#Wins	#Ties	#Losses	*p-value*	#Wins	#Ties	#Losses	*p-value*
Minsum									
HSNR vs. HSNR1	20	2	18	0	5.00E−01	3	5	12	3.00E−03
HSNR vs. HSNR2	20	3	17	0	2.50E−01	5	5	10	7.90E−02
HSNR vs. HSNR3	20	20	0	0	8.85E−05	20	0	0	8.85E−05
HSNR vs. HSNR4	20	20	0	0	8.85E−05	19	0	1	1.20E−04
Minmax									
HSNR vs. HSNR1	20	12	6	2	5.00E−02	12	6	2	2.00E−02
HSNR vs. HSNR2	20	15	5	0	6.10E−05	15	5	0	6.10E−05
HSNR vs. HSNR3	20	12	6	2	1.00E−02	10	6	4	4.90E−01
HSNR vs. HSNR4	20	10	7	3	9.00E−02	7	6	7	6.30E−01

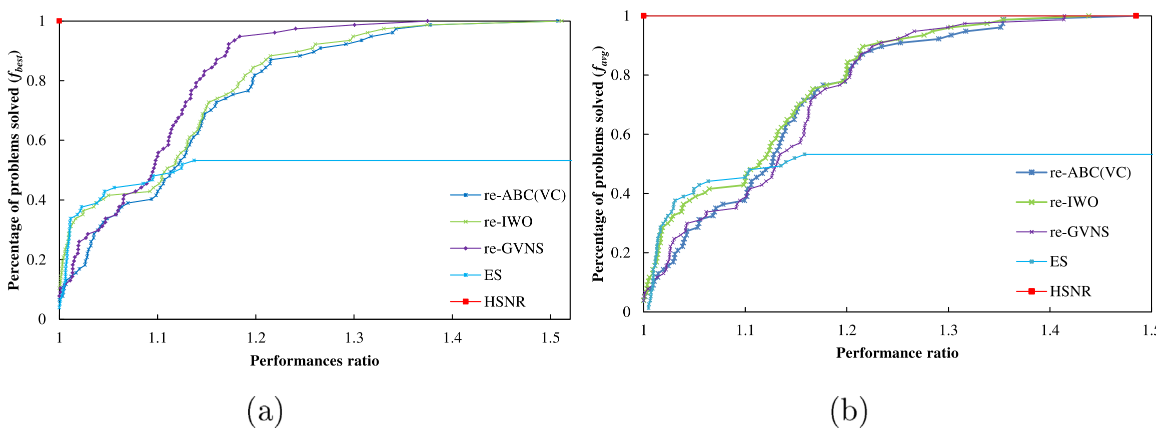

Fig. 3. The minsum mTSP: performance profiles of HSNR and four reference algorithms on all the 77 benchmark instances. The left figure corresponds to the best results while the right figure is for the average results.

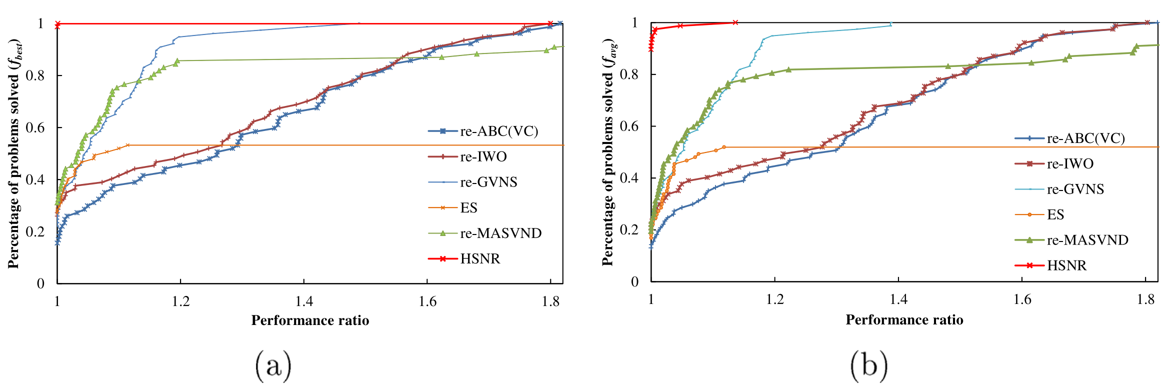

Fig. 4. The minmax mTSP: performance profiles of HSNR and five reference algorithms on all the 77 benchmark instances. The left figure corresponds to the best results while the right figure is for the average results.

HSNR algorithm. This section aims to investigate the contributions of two important ingredients of HSNR: the neighborhood reduction strategy (Section 3.3.3) for efficient neighborhood examination and the EAX heuristic (Section 3.4) for effective intra-tour optimization. For this purpose, we performed additional experiments to compare HSNR with several HSNR variants where the studied component (i.e., neighborhood reduction and EAX) was disabled and replaced by another alternative method. These experiments were based on 20 representative instances with different sizes (n from 150 to 2392, m from 3 to 20) and followed the experimental protocol of Section 4.2.

5.1. Importance of the α-nearness technique for neighborhood reduction

To study the benefit of the α-nearness pruning technique (Section 3.3.3), we compared HSNR with two alternative versions: HSNR1 where the α-nearness pruning technique was replaced by the method of δ-nearest neighbors (Arnold et al., 2019; Brandão, 2020), and HSNR2 where no pruning technique was used. As such, at each neighborhood search iteration of HSNR1, city a must be one of the δ-nearest cities of city π_b (δ was set to 40), as shown in the illustrative example of Fig. 1. For HSNR2, there is no any restriction between city a and π_b.

The experimental results of HSNR, HSNR1 and HSNR2 are summarized in Figs. 5 and 6 as well as Table 8. In the figures, the results of HSNR are used as the baseline and the results of HSNR1 and HSNR2 are showed relative to this baseline. From these results, the following observations can be made.

For the minsum mTSP, compared to HSNR2 which does not use any neighborhood pruning technique, both reductions (α-nearness pruning for HSNR and δ-nearest pruning for HSNR1) led to slightly better results in terms of the best objectives values, while the average quality was

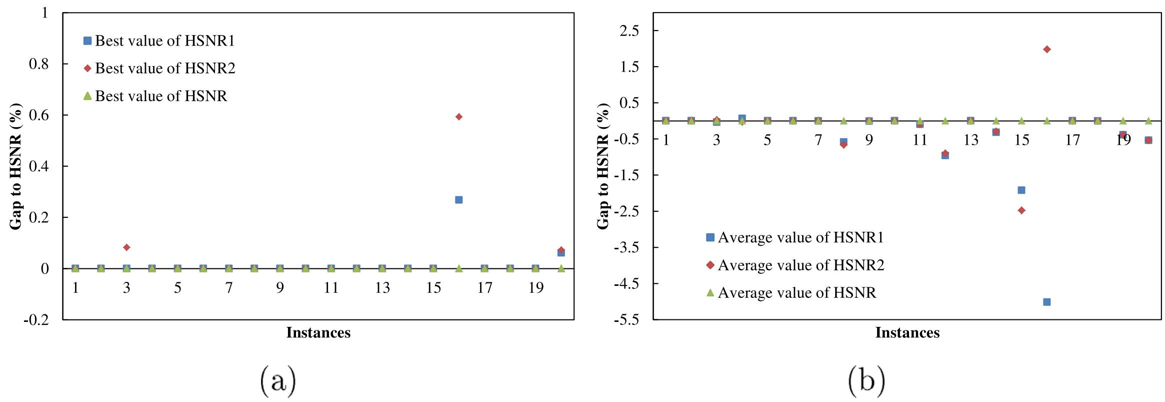

Fig. 5. Minsum mTSP: comparative results of HSNR with HSNR1 (using δ-nearest neighbors) and HSNR2 (without pruning).

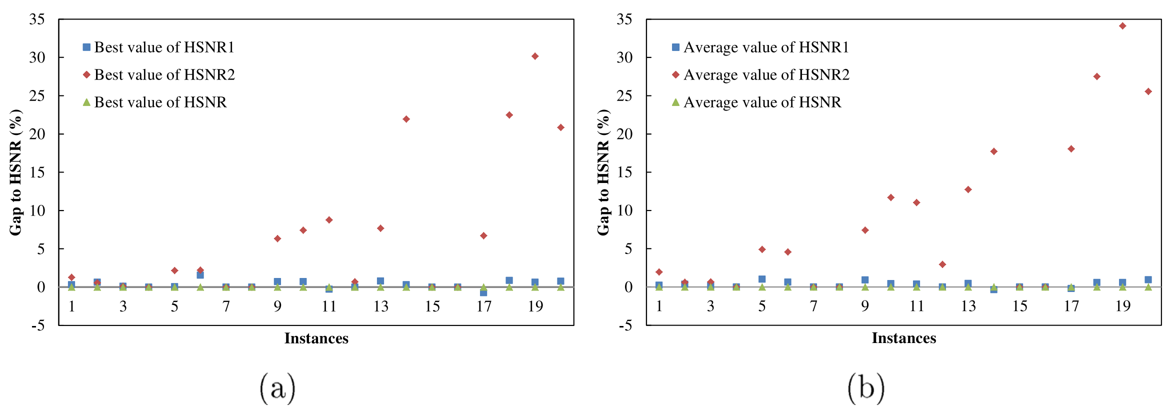

Fig. 6. Minmax mTSP: comparative results of HSNR with HSNR1 (using δ-nearest neighbors) and HSNR2 (without pruning).

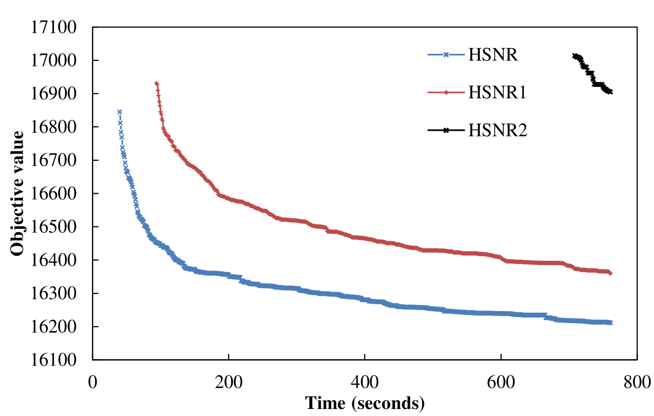

Fig. 7. Convergence chart (running profiles) of HSNR and two HSNR variants for solving instance *lin318-3* with the minmax mTSP. The results were obtained from 20 independent executions of each algorithm.

slightly scarified in several cases. The Wilcoxon signed-rank tests in Table 8, however, do not confirm statistically significant differences between the compared algorithms. For the minmax mTSP, both HSNR and HSNR1 significantly outperformed the HSNR2 variant in terms of the best and average values (confirmed by the Wilcoxon signed-rank tests). The importance of the pruning techniques is even more amplified on large instances. One also observes that HSNR using the α-nearness pruning technique consistently showed better performances than HSNR1 using the δ-nearest neighbors technique. As an example, the convergence charts shown in Fig. 7 also illustrate the usefulness of the α-nearness pruning technique on a representative instance.

This experiment confirms the interest of heuristic pruning techniques, especially the α-nearness technique adopted in the HSNR algorithm. By avoiding useless examinations of non-promising neighboring

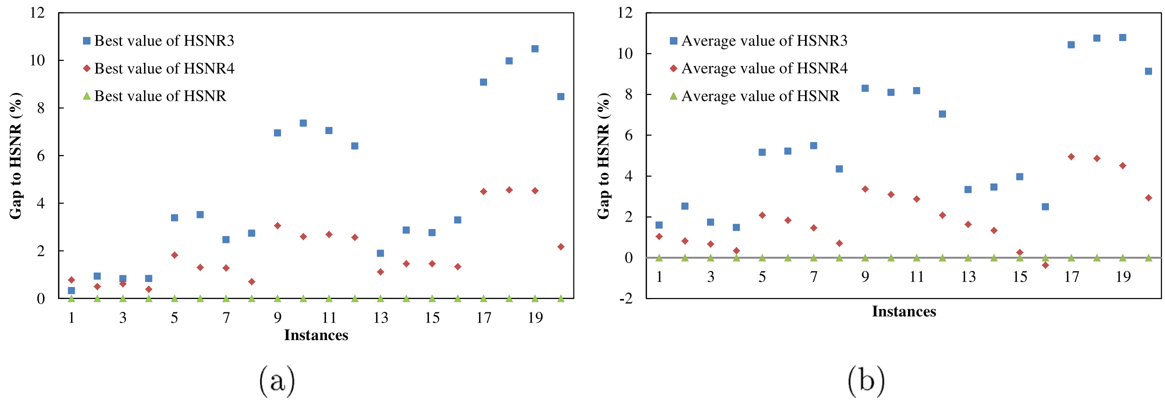

Fig. 8. Minsum mTSP: comparative results of HSNR (using EAX) with HSNR3 (using the 2-opt heuristic) and HSNR4 (using the LK algorithm).

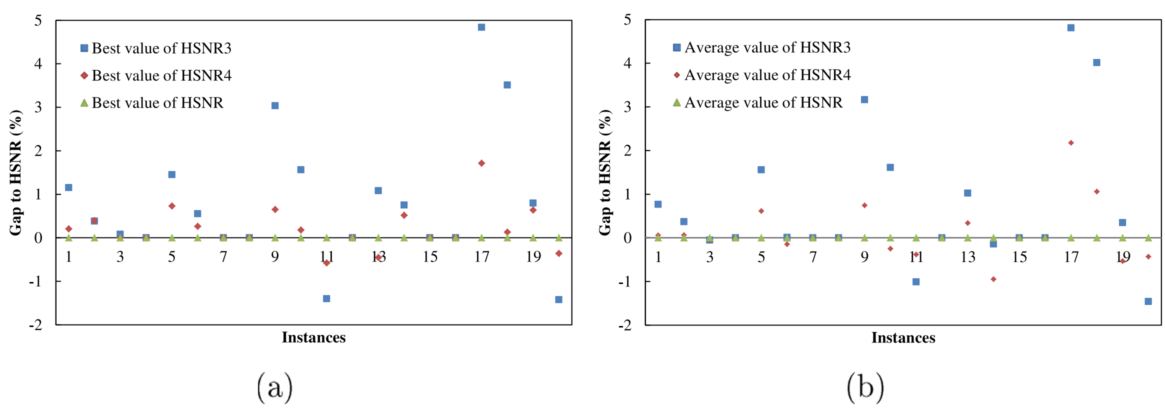

Fig. 9. Minmax mTSP: comparative results of HSNR (using EAX) with HSNR3 (using the 2-opt heuristic) and HSNR4 (using the LK algorithm).

solutions, the neighborhood reduction strategy is particularly useful for solving large instances of the minmax mTSP, even if its contribution to the minsum mTSP is less significant.

5.2. Importance of the EAX heuristic for intra-optimization

To evaluate the benefits of the EAX heuristic for intra-tour optimization (Section 3.4), we compare HSNR with two alternative algorithms: HSNR3 where EAX is replaced by the popular 2-opt heuristic, and HSNR4 where EAX is replaced by the LK algorithm (Lin and Kernighan, 1973). The comparative results are shown in Figs. 8 and 9 as well as Table 8.

For the minsum mTSP, HSNR with EAX significantly dominates its variants with the 2-opt and LK heuristics in terms of the best and average results (confirmed by the Wilcoxon signed-rank tests). For the minmax mTSP, HSNR also performs better than its competitors except for a small number of instances. This experiment demonstrates clearly the usefulness of the TSP heuristic EAX as a critical intra-tour optimization tool for the mTSP.

6. Conclusions

This work studied the multiple traveling salesman problem, which is a relevant model to formulate a number of practical applications. The presented hybrid search with neighborhood reduction algorithm combines tabu search based inter-tour optimization (with 2 complementary neighborhoods) and a TSP heuristic based intra-tour optimization. A dedicated neighborhood reduction technique was introduced, which avoids the evaluations of non-promising candidate solutions and thus speeds up the neighborhood search.

Extensive computational results on the set of 41 benchmark instances commonly tested in the literature indicate that the algorithm is highly competitive compared with the existing leading algorithms. In particular, for the minsum mTSP, the proposed algorithm reports 27 best results while matching 10 best-known results. For the minmax mTSP, the algorithm performs also well by reporting 15 best bounds. To assess the presented algorithm on still larger instances, we introduced a new set of 36 large instances and reported the first computational results, which further demonstrated the superiority of the algorithm over the reference algorithms. These new large instances and the presented results can be used to assess other mTSP algorithms.

The TSP heuristic EAX was also used for the first time to solve the minsum mTSP, based on the fact that the minsum mTSP can be conveniently transformed to the TSP. The results showed that this transformation approach performs remarkably well on most minsum mTSP instances and significantly dominates all algorithms dedicated to the minsum mTSP.

For future work, there are several perspectives. First, it would be interesting to adopt the main idea of this study (i.e., neighborhood reduction, TSP tool) to design effective heuristics for other TSP variants and routing problems, including practical problems faced in real-life applications. Second, even if the minsum mTSP can be effectively solved by popular TSP algorithms, this is not the case for the minmax mTSP. As such, more efforts are needed to design effective algorithms

Table A.1
Minsum mTSP: comparative results between HSNR and four state-of-the-art algorithms on 17 instances of Set I with the short cutoff time of $(1.36 \times n)/5$ seconds on our computer.

		ABC(VC) (Pandiri and Singh, 2015)		IWO (Pandiri and Singh, 2015)		GVNS (Soylu, 2015)		ES (Karabulut et al., 2021)		HSNR (this work)		
Instance	BKS	*Best*	*Avg.*	*Best*	*Avg.*	*Best*	*Avg.*	*Best*	*Avg.*	*Best*	*Avg.*	*Gap(%)*
mtsp51–3	446	446	448	446	448	446	449	446	446.6	446	446	0.00
mtsp51–5	472	472	475	472	478	472	474	472	472.6	472	472	0.00
mtsp51–10	580	580	581	581	583	580	580	580	580.7	580	583.4	0.00
mtsp100–3	21798	21798	21814	21798	21941	21879	22068	2179	21839	21797.6	21852.42	0.00
mtsp100–5	23175	23182	23222	23294	23319	23175	23383	2317	23252	23174.9	23195.31	0.00
mtsp100–10	26927	26961	27004	26961	27072	27008	27368	2692	26927	27026.4	27081.47	0.37
mtsp100–20	38245	38333	38397	38245	38357	3832	38867	3824	38257	38297.1	38882.61	0.14
mtsp150–3	37957	38066	38263	37957	38055	38430	38827	3807	38241.1	37910.7	37912.88	**−0.12**
mtsp150–5	38714	38979	39202	38714	38881	39171	39566	3890	39132.5	38714.4	38768.54	0.00
mtsp150–10	42203	42441	42712	42234	42462	42730	42922	4220	42428.1	42268.4	42393.11	0.15
mtsp150–20	53343	53603	53877	53475	53612	5357	53854	5334	53516.4	53608.3	54142.57	0.50
mtsp150–30	68541	68865	69046	68541	68751	6855	68804	6860	68774.7	68787.3	69224.97	0.36
gtsp150–3	6590	6590	6614	6593	6628	–	–	–	–	6574.2	6575.5	**−0.24**
gtsp150–5	6652	6708	6725	6652	6716	–	–	–	–	6655.11	6657.97	0.05
gtsp150–10	7342	7377	7414	7342	7388	–	–	–	–	7332.11	7346.09	**−0.13**
gtsp150–20	9525	9542	9596	9525	9583	–	–	–	–	9542.29	9637.45	0.18
gtsp150–30	12976	13055	13115	12976	13127	–	–	–	–	13059.8	13190.41	0.65
AVG.	23263.88	23352.82	23441.47	23282.7	23376.53	–	–	–	–	23308.6	23433.1	–
Best#	–	0	3	0	3	0	1	3	0	3	**9**	–
p-value	6.00E−02	3.00E−02	9.80E−01	2.30E−01	7.20E−01	–	–	–	–	–	–	–

for the minmax mTSP. In this regard, it is worth investigating other search framework such as memetic algorithms integrating dedicated crossover operators. Also, few exact algorithms exist for the minmax mTSP, there is much room for making progressive in this area.

CRediT authorship contribution statement

Pengfei He: Conceptualization, Methodology, Software, Experiments, Data curation, Writing – original draft. **Jin-Kao Hao:** Supervision, Conceptualization, Methodology, Validation, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

We are grateful to the reviewers for their useful comments and suggestions which helped us to significantly improve the paper. We would like to thank authors of Karabulut et al. (2021), Pandiri and Singh (2015), Wang et al. (2017), Yuan et al. (2013): Prof. K. Karabulut and Prof. M. F. Tasgetiren for sharing their executable code; Prof. A. Singh, Dr. Y. Wang, and Dr. S. Yuan for providing their test problems and answering our questions. Support from the China Scholarship Council (CSC, grant No. 201906850087) for the first author is acknowledged.

Appendix

This appendix includes computational results of two additional experiments. The first experiment concerns a comparison between the proposed HSNR algorithm and the reference algorithms under a short cutoff time for the minsum mTSP and the minmax mTSP. The second experiment is about solving the minsum mTSP by running a TSP solver, given that the minsum mTSP can be transformed to the TSP (Hong and Padberg, 1977; Rao, 1980). Even if this transformation is known for a long time, to our knowledge, this is the first study reporting extensive computational results using this approach.

A.1. Additional computational results and comparisons

We compare the results of the HSNR algorithm with the best results of the reference algorithms directly extracted from the literature. Given that the reference algorithms were coded by different persons and run on different computers under various stopping conditions, this comparison is presented for indicative purposes only. For this study, we used the following reference algorithms.

- IWO (Pandiri and Singh, 2015), which reports results on 17 instances of Set I for the minsum mTSP and the minmax mTSP. The algorithm was written in C and run on a computer with a 2.83 GHz CPU and the stopping condition is a maximum of 1000 iteration steps.
- ABC(VC) (Pandiri and Singh, 2015), which reports results on 17 instances of Set I for the minsum mTSP and the minmax mTSP. The algorithm was written in C and run on the same computer under the same stopping condition as IWO.
- GVNS (Soylu, 2015), which reports results on 12 instances of Set I for the minsum mTSP and the minmax mTSP. The algorithm was written in C++ and run on a computer with a 2.4 GHz CPU, and the stopping condition is a maximum running time of n seconds.
- MASVND (Wang et al., 2017), which is designed for the minmax mTSP only and reports results on 31 out of the 41 instances of Set I. The algorithm was written in Java and run on a computer with a 3.4 GHz CPU, and the stopping condition is a maximum running time of $n/5$ seconds.
- ES (Karabulut et al., 2021), which reports results on 12 instances of Set I for the minsum mTSP and 31 out of the 41 instances of Set I for the minmax mTSP. The algorithm was written in C++ and run on a computer with a 2.66 GHz CPU, and the stopping condition is a maximum time of n and $n/5$ seconds for the minsum mTSP and the minmax mTSP, respectively.

To make the comparison as meaningful as possible, we adopted as our stopping condition the shortest cutoff time among those used by the reference algorithms, i.e., $n/5$ seconds used in Wang et al. (2017). We used the CPU frequency to convert this cutoff time to our computer, leading to a cutoff time of $(1.36 \times n)/5$ seconds for our HSNR algorithm on our computer. Note that MASVND reports results for the minmax mTSP only, while the other reference algorithms report results for both the minsum mTSP and the minmax mTSP.

Table A.2
Minmax mTSP: comparative results of HSNR and six state-of-the-art algorithms on the instances of Set I. The cutoff time is $(1.36 \times n)/5$ seconds on our computer.

		ABC(VC) (Pandiri and Singh, 2015)		IWO Pandiri and Singh (2015)		GVNS Soylu (2015)		IWO-Wang (Wang et al., 2017)		MASVND (Wang et al., 2017)		ES (Karabulut et al., 2021)		HSNR (this work)		
Instance	BKS	*Best*	*Avg.*	*Best*	*Avg.*	*Best*	*Avg.*	*Best*	*Avg.*	*Best*	*Avg.*	*Best*	*Avg.*	*Best*	*Avg.*	*Gap*(%)
mtsp51–3	159.57	160.00	160.00	160.00	160.00	160.00	162.00	159.57	159.57	159.57	159.73	159.57	160.28	159.57	159.99	0.00
mtsp51–5	118.00	118.00	118.00	118.00	118.00	118.00	120.00	118.13	118.13	118.13	120.54	118.13	118.62	118.13	118.13	0.00
mtsp51–10	112.00	112.00	112.00	112.00	112.00	112.00	112.00	112.07	112.07	112.07	112.07	112.07	112.07	112.07	112.07	0.00
mtsp100–3	8509.00	8509.00	8574.00	8509.00	8550.00	8509.00	8571.00	–	–	–	–	8509.00	8509.00	8509.16	8578.51	0.00
mtsp100–5	6765.73	6768.00	6789.00	6767.00	6769.00	6767.00	6835.00	–	–	–	–	6766.00	6766.90	6765.73	6774.24	0.00
mtsp100–10	6358.00	6358.00	6358.00	6358.00	6358.00	6358.00	6358.00	–	–	–	–	6358.00	6358.00	6358.49	6358.49	0.00
mtsp100–20	6358.00	6358.00	6358.00	6358.00	6358.00	6358.00	6358.00	–	–	–	–	6358.00	6358.00	6358.49	6358.49	0.00
mtsp150–3	13151.0	13313.00	13761.00	13168.0	13313.00	13376.0	13628.00	–	–	–	–	13151.0	13272.20	13174.3	13352.37	0.18
mtsp150–5	8466.00	8567.00	8795.00	8479.00	8567.00	8467.00	8601.00	–	–	–	–	8466.00	8572.50	8479.60	8602.15	0.16
mtsp150–10	5557.00	5651.00	5834.00	5594.00	5654.00	5674.00	5736.00	–	–	–	–	5557.00	5609.60	5616.71	5668.59	1.07
mtsp150–20	5246.00	5246.00	5281.00	5246.00	5246.00	5246.00	5246.00	–	–	–	–	5246.00	5246.00	5246.49	5246.49	0.00
mtsp150–30	5246.00	5246.00	5247.00	5246.00	5246.00	5246.00	5246.00	–	–	–	–	5246.00	5246.00	5246.49	5246.49	0.00
gtsp150–3	2407.59	2451.00	2479.00	2408.00	2439.00	–	–	2413.24	2435.42	2429.49	2450.13	2407.59	2477.34	2425.87	2449.65	0.76
gtsp150–5	1741.61	1766.00	1775.00	1742.00	1742.00	–	–	1752.11	1761.32	1758.08	1796.86	1741.61	1792.19	1744.26	1755.13	0.15
gtsp150–10	1554.00	1557.00	1560.00	1554.00	1554.00	–	–	1554.64	1558.03	1554.64	1557.16	1554.64	1555.70	1554.64	1554.64	0.00
gtsp150–20	1554.00	1554.00	1554.00	1554.00	1554.00	–	–	1554.64	1554.64	1554.64	1554.64	1554.64	1554.64	1554.64	1554.64	0.00
gtsp150–30	1554.00	1554.00	1554.00	1554.00	1554.00	–	–	–	–	–	–	–	–	1554.64	1554.64	0.00
kroA200–3	10768.10	–	–	–	–	–	–	10814.1	10947.79	10831.6	11045.91	10768.1	11099.63	10801.80	11169.82	0.31
kroA200–5	7415.54	–	–	–	–	–	–	7493.24	7593.15	7415.54	7582.08	7572.32	7684.73	7418.87	7575.80	0.04
kroA200–10	6223.22	–	–	–	–	–	–	6237	6278.99	6223.22	6249.17	6223.22	6231.97	6223.22	6223.22	0.00
kroA200–20	6223.22	–	–	–	–	–	–	6223.22	6223.22	6223.22	6223.22	6223.22	6223.22	6223.22	6223.22	0.00
lin318–3	16088.73	–	–	–	–	–	–	16200.2	16340.3	16206.2	16477.89	16273.80	16753.24	16094.9	16482.00	0.04
lin318–5	11524.29	–	–	–	–	–	–	11730.03	11908.18	11752.4	11896.71	11604.2	11876.42	11458.2	11851.87	**−0.57**
lin318–10	9731.17	–	–	–	–	–	–	9845.72	9955.42	9731.17	9818.75	9731.17	9742.20	9731.17	9731.17	0.00
lin318–20	9731.17	–	–	–	–	–	–	9731.17	9731.17	9731.17	9731.17	9731.17	9731.17	9731.17	9731.17	0.00
rat783–3	3272.95	–	–	–	–	–	–	3457.97	3497.56	3279.16	3336.57	3369.40	3418.06	3262.52	3333.55	**−0.32**
rat783–5	2092.77	–	–	–	–	–	–	2273.8	2303.14	2092.77	2134.03	2127.99	2163.89	2066.38	2115.41	**−1.26**
rat783–10	1360.89	–	–	–	–	–	–	1542.05	1564.7	1432.34	1452.67	1360.89	1388.64	1358.06	1386.26	**−0.21**
rat783–20	1231.69	–	–	–	–	–	–	1311.3	1333.12	1260.88	1270.31	1231.69	1233.88	1231.69	1231.69	0.00
pcb1173–3	22252.31	–	–	–	–	–	–	24008.47	24300.25	22443.2	22781.61	22601.7	23095.02	21430.1	21928.16	**−3.96**
pcb1173–5	14099.50	–	–	–	–	–	–	16057.19	16274.64	14557.3	14861.4	14099.5	14346.76	13402.3	13743.67	**−4.94**
pcb1173–10	8160.25	–	–	–	–	–	–	10517.94	10667.97	9222.92	9352.28	8160.25	8260.99	8120.45	8367.34	**−0.49**
pcb1173–20	6549.14	–	–	–	–	–	–	8063.17	8207.88	7063.23	7276.69	6549.14	6592.50	6528.86	6540.13	**−0.31**
AVG.	6411.60	–	–	–	–	–	–	–	–	–	–	–	–	6365.52	6456.94	–
Best#	–	0	0	0	3	–	0	0	3	1	0	5	5	**8**	**12**	–
p-value	7.18E−01	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

Table A.3
Minsum mTSP: comparative results of HSNR and EAX on Set I with a cutoff time of $(n/100) \times 4$ minutes.

Instance	EAX (Nagata and Kobayashi, 2013)			HSNR (this work)		
	Best	*Avg.*	σ	*Best*	*Avg.*	σ
mtsp51–3	445.99	445.99	0.00	445.99	445.99	0.00
mtsp51–5	471.69	471.69	0.00	471.69	471.69	0.00
mtsp51–10	579.70	579.70	0.00	580.72	580.72	0.00
mtsp100–3	21797.60	21797.60	0.00	21797.60	21797.60	0.00
mtsp100–5	23174.90	23174.90	0.00	23174.90	23174.90	0.00
mtsp100–10	26926.60	26926.60	0.00	26926.60	26983.51	50.63
mtsp100–20	38245.10	38245.10	0.00	38245.10	38259.98	51.79
rand100–3	8012.13	8012.13	0.00	8012.13	8012.13	0.00
rand100–5	8223.91	8223.91	0.00	8223.91	8223.91	0.00
rand100–10	9366.80	9366.80	0.00	9366.80	9366.80	0.00
rand100–20	13404.10	13404.10	0.00	13404.10	13404.10	0.00
mtsp150–3	37910.70	37910.70	0.00	37910.70	37910.70	0.00
mtsp150–5	38714.40	38714.40	0.00	38714.40	38722.24	11.83
mtsp150–10	42202.80	42202.80	0.00	42234.30	42310.82	36.72
mtsp150–20	53305.90	53305.90	0.00	53351.30	53483.13	95.76
mtsp150–30	68442.90	68442.90	0.00	68455.90	68539.07	123.03
gtsp150–3	6574.20	6574.20	0.00	6574.20	6574.52	1.45
gtsp150–5	6655.11	6655.11	0.00	6655.11	6655.11	0.00
gtsp150–10	7332.11	7332.11	0.00	7332.11	7332.11	0.00
gtsp150–20	9512.23	9512.23	0.00	9512.23	9513.38	4.17
gtsp150–30	12966.50	12966.50	0.00	12966.50	12969.05	9.86
kroA200–3	29539.50	29539.50	0.00	29539.50	29539.50	0.00
kroA200–5	29916.20	29916.20	0.00	29916.20	29916.20	0.00
kroA200–10	32613.40	32613.40	0.00	32613.40	32613.40	0.00
kroA200–20	41439.20	41439.20	0.00	41439.20	41522.45	207.47
lin318–3	42404.60	42404.60	0.00	42404.60	42404.60	0.00
lin318–5	43315.00	43315.00	0.00	43315.00	43315.00	0.00
lin318–10	47325.50	47325.50	0.00	47325.50	47333.21	9.50
lin318–20	59893.20	59893.20	0.00	59893.20	60416.35	742.66
att532–3	28242.00	28242.00	0.00	28242.00	28242.00	0.00
att532–5	28945.00	28945.00	0.00	28945.00	28945.00	0.00
att532–10	31001.00	31001.00	0.00	31001.00	31038.80	88.22
att532–20	36303.00	36303.00	0.00	36305.00	36696.65	482.00
rat783–3	8880.03	8880.03	0.00	8880.03	8880.64	2.72
rat783–5	8964.80	8964.80	0.00	8964.80	8964.90	0.45
rat783–10	9265.64	9265.64	0.00	9265.64	9275.16	17.08
rat783–20	10172.10	10172.10	0.00	10172.60	10272.95	106.03
pcb1173–3	57167.20	57169.20	4.40	57167.20	57174.12	19.79
pcb1173–5	57628.80	57628.80	0.00	57628.80	57654.20	17.40
pcb1173–10	59241.90	59242.10	3.30	59241.90	59299.07	187.13
pcb1173–20	64052.00	64052.00	0.00	64063.60	65102.08	646.01
Avg.	28306.72	28306.77	–	28309.28	28374.09	–
Best#	**7**	**23**	–	0	0	–
p-value	1.95E−02	3.25E−05	–	–	–	–

A.1.1. Comparative results for the minsum mTSP

Table A.1 shows the computational results of the compared algorithms for the minsum mTSP with the same information as in Section 4.

From Table A.1, one observes that the proposed HSNR algorithm performs better than ABC(VC), GVNS, by matching more BKS values, while its performance is slightly worse than the fast IWO algorithm and ES. Interestingly, HSNR reports three new best-known results. This experiment indicates that under short stopping conditions, the fast IWO and ES algorithms perform the best for the minsum mTSP, while HSNR remains competitive by reporting three new upper bounds.

A.1.2. Comparative results for the minmax mTSP

We show in Table A.2 the computational results of the compared algorithms for the minmax mTSP with the same information as in Section 4. In this table, we included the results of IWO-Wang (Wang et al., 2017), which is a re-implementation of the IWO algorithm of Pandiri and Singh (2015).

Table A.2 indicates that HSNR performs competitively compared to the main reference algorithms, that is MASVND (Wang et al., 2017) and ES (Karabulut et al., 2021). In terms of the best objective value, HSNR updates the best upper bounds (BKS) for 9 out of 33 instances and reaches the BKS values for 17 instances. Given that the BKS values are compiled from the best results ever reported by all existing algorithms in the literature, the performance of HSNR for the minmax mTSP can be considered as remarkable. In summary, these results confirm the competitiveness of HSNR over the state-of-the-art algorithms for the minmax mTSP also under this short cutoff limit.

A.2. Computational results for the minsum mTSP with a TSP heuristic

We report computational results of running the EAX heuristic (Nagata and Kobayashi, 2013) on the TSP instances transformed from the minsum mTSP instances. Given that most of the 77 instances involve distance matrices of real numbers, we updated the data type of EAX from integer numbers to real numbers. For this experiment, we ran the EAX code with its default parameter setting under the same stopping condition as HSNR (i.e., $(n/100) \times 4$ minutes, see Section 4). Each instance was solved 20 times by EAX with difference random seeds. Note that EAX may also terminate if the gap between the average tour length and the shortest tour length in the population becomes less than 0.0001.

Tables A.3 and A.4 show the comparative results of EAX and HSNR with the same information as in Section 4.3.1. The background of the top results for each instance is highlighted in dark gray; the second best results in medium gray. The results of Tables A.3 and A.4 clearly indicate that EAX significantly dominates HSNR in terms of the best and average results for both sets of instances. Only on three large instances of Set II, HSNR reported better results. Given that HSNR performs better

Table A.4
Minsum mTSP: comparative results of HSNR and EAX on Set II with a cutoff time of $(n/100) \times 4$ minutes.

Instance	EAX (Nagata and Kobayashi, 2013)			HSNR (this work)		
	Best	*Avg.*	*σ*	*Best*	*Avg.*	*σ*
nrw1379–3	56775.70	56775.70	0.00	56775.70	56775.70	0.00
nrw1379–5	56992.60	56994.40	1.81	56992.60	56999.16	5.27
nrw1379–10	57636.10	57637.00	1.10	57636.20	57795.15	168.81
nrw1379–20	59539.80	59542.70	4.14	59618.40	60278.03	426.66
fl1400–3	21169.40	21176.40	14.74	21169.40	21169.47	0.31
fl1400–5	22066.20	22069.70	11.06	22066.20	22238.10	239.95
fl1400–10	24373.90	24380.40	14.75	24373.90	25069.65	531.24
fl1400–20	29480.40	29492.70	16.14	29579.20	31966.86	1516.54
d1655–3	68364.40	68367.70	3.61	68364.40	68370.50	8.69
d1655–5	74272.70	74273.10	1.78	74273.50	74292.65	43.66
d1655–10	89261.10	89262.40	2.03	89262.50	89856.83	717.31
d1655–20	120016.00	120019.00	5.21	121373.00	124263.45	1190.66
u2152–3	65064.90	65066.10	2.70	65064.90	65072.31	10.68
u2152–5	65197.20	65200.70	11.15	65201.70	65219.93	8.60
u2152–10	65748.30	65750.50	3.85	65762.50	66291.71	526.37
u2152–20	67493.40	67494.20	1.76	67993.10	71115.74	1344.28
pr2392–3	378661.00	378661.00	0.00	378661.00	378661.00	0.00
pr2392–5	380061.00	380061.00	0.00	380061.00	380069.40	28.64
pr2392–10	387498.00	387498.00	0.00	387498.00	389012.85	1621.15
pr2392–20	407678.00	407680.00	9.39	417424.00	421532.30	2665.82
pcb3038–3	137916.00	137917.00	2.69	137916.00	137925.00	3.08
pcb3038–5	138121.00	138122.00	2.69	138121.00	138123.20	4.51
pcb3038–10	139142.00	139142.00	0.00	139142.00	139379.85	369.30
pcb3038–20	142401.00	142402.00	3.67	144295.00	146491.65	1068.88
fl3795–3	29601.20	29661.50	72.21	29589.90	29823.75	394.67
fl3795–5	30508.20	30560.50	50.68	30480.80	31048.26	634.63
fl3795–10	32779.80	32866.60	75.61	32729.60	35467.72	1551.01
fl3795–20	37333.30	37419.10	70.10	39083.80	45437.27	3166.39
fnl4461–3	182888.00	182890.00	2.43	182888.00	182890.85	7.74
fnl4461–5	183074.00	183076.00	1.79	183074.00	183076.50	4.70
fnl4461–10	183803.00	183806.00	3.49	183808.00	184811.75	874.86
fnl4461–20	186618.00	186619.00	3.58	191025.00	193356.10	1527.51
rl5915–3	565949.00	566001.00	70.32	565949.00	566066.70	58.80
rl5915–5	566626.00	566684.00	69.02	566626.00	566780.55	100.60
rl5915–10	569619.00	569653.00	75.52	569619.00	573689.20	3457.21
rl5915–20	578212.00	578278.00	77.77	597878.00	609385.79	7492.50
Avg.	172276.16	172291.68	–	173371.56	174716.80	–
Best#	**15**	**33**	–	3	1	–
p-value	6.50E−03	5.39E−07	–	–	–	–

than the existing minsum mTSP algorithms in the literature, we can safely say that EAX dominates all existing minsum mTSP algorithms. Finally, even if we did not show detailed run-time information, we mention that EAX converges much faster than the existing algorithms (by at least one order of magnitude). EAX requires no more than 30 s for Set I and no more than 400 s for Set II.

We conclude that the transformation approach of the minsum mTSP to the TSP is particularly effective and can be considered as the current best solution method for the minsum mTSP. It is worth mentioning that this is the first study that demonstrates the high interest of solving the minsum mTSP via TSP algorithms. This finding will benefit future research on the minsum mTSP.

References

Applegate, David, Cook, William, Dash, Sanjeeb, Rohe, André, 2002. Solution of a min-max vehicle routing problem. INFORMS J. Comput. 14 (2), 132–143.

Arnold, Florian, Gendreau, Michel, Sörensen, Kenneth, 2019. Efficiently solving very large-scale routing problems. Comput. Oper. Res. 107, 32–42.

Arnold, Florian, Sörensen, Kenneth, 2019. Knowledge-guided local search for the vehicle routing problem. Comput. Oper. Res. 105, 32–46.

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

Bektaş, Tolga, 2012. Formulations and Benders decomposition algorithms for multidepot salesmen problems with load balancing. Eur. J. Oper. Res. 216 (1), 83–93.

Brandão, José, 2020. A memory-based iterated local search algorithm for the multi-depot open vehicle routing problem. Eur. J. Oper. Res. 284 (2), 559–571.

Brown, Evelyn C., Ragsdale, Cliff T., Carter, Arthur E., 2007. A grouping genetic algorithm for the multiple traveling salesperson problem. Int. J. Inf. Technol. Decis. Making 6 (02), 333–347.

Carter, Arthur E., Ragsdale, Cliff T., 2006. A new approach to solving the multiple traveling salesperson problem using genetic algorithms. Eur. J. Oper. Res. 175 (1), 246–257.

Carter, Arthur E., Ragsdale, Cliff T., 2009. Quality inspection scheduling for multi-unit service enterprises. Eur. J. Oper. Res. 194 (1), 114–126.

Cheikhrouhou, Omar, Khoufi, Ines, 2021. A comprehensive survey on the multiple Traveling Salesman Problem: Applications, approaches and taxonomy. Comp. Sci. Rev. 40, 100369.

Dolan, Elizabeth D., Moré, Jorge J., 2002. Benchmarking optimization software with performance profiles. Math. Program. 91 (2), 201–213.

Ergezer, Halit, Leblebicioğlu, Kemal, 2014. 3D path planning for multiple UAVs for maximum information collection. J. Intell. Robot. Syst. 73 (1–4), 737–762.

França, Paulo M, Gendreau, Michel, Laporte, Gilbert, Müller, Felipe M, 1995. The m-traveling salesman problem with minmax objective. Transp. Sci. 29 (3), 267–275.

Garn, Wolfgang, 2020. Closed form distance formula for the balanced multiple travelling salesmen. arXiv preprint arXiv:2001.07749.

Garn, Wolfgang, 2021. Balanced dynamic multiple travelling salesmen: Algorithms and continuous approximations. Comput. Oper. Res. 136, 105509.

Gavish, Bezalel, Srikanth, Kizhanathan, 1986. An optimal solution method for large-scale multiple traveling salesmen problems. Oper. Res. 34 (5), 698–717.

Glover, Fred W., Laguna, Manuel, 1997. Tabu Search. Kluwer.

He, Pengfei, Li, Jing, Qin, Hailong, He, Yanping, Cao, Guangqiao, 2019. Using hybrid algorithm to reduce non-working distance in intra-and inter-field logistics simultaneously for heterogeneous harvesters. Comput. Electron. Agric. 167, 105065.

He, Pengfei, Li, Jing, Wang, Xin, 2018. Wheat harvest schedule model for agricultural machinery cooperatives considering fragmental farmlands. Comput. Electron. Agric. 145, 226–234.

Helsgaun, Keld, 2000. An effective implementation of the Lin–Kernighan traveling salesman heuristic. Eur. J. Oper. Res. 126 (1), 106–130.

Hong, Saman, Padberg, Manfred W., 1977. A note on the symmetric multiple traveling salesman problem with fixed charges. Oper. Res. 25 (5), 871–874.

Karabulut, Korhan, Öztop, Hande, Kandiller, Levent, Tasgetiren, M Fatih, 2021. Modeling and optimization of multiple traveling salesmen problems: An evolution strategy approach. Comput. Oper. Res. 129, 105192.

Koubâa, Anis, Cheikhrouhou, Omar, Bennaceur, Hachemi, Sriti, Mohamed-Foued, Javed, Yasir, Ammar, Adel, 2017. Move and improve: a market-based mechanism for the multiple depot multiple travelling salesmen problem. J. Intell. Robot. Syst. 85 (2), 307–330.

Laporte, Gilbert, Nobert, Yves, 1980. A cutting planes algorithm for the m-salesmen problem. J. Oper. Res. Soc. 31 (11), 1017–1023.

Lin, Shen, Kernighan, Brian W., 1973. An effective heuristic algorithm for the traveling-salesman problem. Oper. Res. 21 (2), 498–516.

Liu, Weimin, Li, Sujian, Zhao, Fanggeng, Zheng, Aiyun, 2009. An ant colony optimization algorithm for the multiple traveling salesmen problem. In: 2009 4th IEEE Conference On Industrial Electronics And Applications. IEEE, pp. 1533–1537.

López-Ibáñez, Manuel, Dubois-Lacoste, Jérémie, Cáceres, Leslie Pérez, Birattari, Mauro, Stützle, Thomas, 2016. The irace package: Iterated racing for automatic algorithm configuration. Oper. Res. Perspect. 3, 43–58.

Lu, Yongliang, Benlic, Una, Wu, Qinghua, 2019. A population algorithm based on randomized tabu thresholding for the multi-commodity pickup-and-delivery traveling salesman problem. Comput. Oper. Res. 101, 285–297.

Lu, Li-Chih, Yue, Tai-Wen, 2019. Mission-oriented ant-team ACO for min–max MTSP. Appl. Soft Comput. 76, 436–444.

Nagata, Yuichi, Kobayashi, Shigenobu, 2013. A powerful genetic algorithm using edge assembly crossover for the traveling salesman problem. INFORMS J. Comput. 25 (2), 346–363.

Pan, Junjie, Wang, Dingwei, 2006. An ant colony optimization algorithm for multiple travelling salesman problem. In: First International Conference On Innovative Computing, Information And Control. IEEE Computer Society, pp. 210–213.

Pandiri, Venkatesh, Singh, Alok, 2015. Two metaheuristic approaches for the multiple traveling salesperson problem. Appl. Soft Comput. 26, 74–89.

Paydar, Mohammad Mahdi, Mahdavi, Iraj, Sharafuddin, Iman, Solimanpur, Maghsud, 2010. Applying simulated annealing for designing cellular manufacturing systems using MDmTSP. Comput. Ind. Eng. 59 (4), 929–936.

Penna, Puca Huachi Vaz, Subramanian, Anand, Ochi, Luiz Satoru, 2013. An iterated local search heuristic for the heterogeneous fleet vehicle routing problem. J. Heuristics 19 (2), 201–232.

Rao, M.R., 1980. A note on the multiple traveling salesmen problem. Oper. Res. 28 (3-part-i), 628–632.

Shiri, Samaneh, Huynh, Nathan, 2016. Optimization of drayage operations with time-window constraints. Int. J. Prod. Econ. 176, 7–20.

Singh, Alok, Baghel, Anurag Singh, 2009. A new grouping genetic algorithm approach to the multiple traveling salesperson problem. Soft Comput. 13 (1), 95–101.

Soylu, Banu, 2015. A general variable neighborhood search heuristic for multiple traveling salesmen problem. Comput. Ind. Eng. 90, 390–401.

Svestka, Joseph A., Huckfeldt, Vaughn E., 1973. Computational experience with an m-salesman traveling salesman algorithm. Manage. Sci. 19 (7), 790–799.

Taillard, Éric, Badeau, Philippe, Gendreau, Michel, Guertin, François, Potvin, Jean-Yves, 1997. A tabu search heuristic for the vehicle routing problem with soft time windows. Transp. Sci. 31 (2), 170–186.

Tang, Lixin, Liu, Jiyin, Rong, Aiying, Yang, Zihou, 2000. A multiple traveling salesman problem model for hot rolling scheduling in Shanghai Baoshan Iron & steel complex. Eur. J. Oper. Res. 124 (2), 267–282.

Toth, Paolo, Vigo, Daniele, 2003. The granular tabu search and its application to the vehicle-routing problem. INFORMS J. Comput. 15 (4), 333–346.

Uchoa, Eduardo, Pecin, Diego, Pessoa, Artur, Poggi, Marcus, Vidal, Thibaut, Subramanian, Anand, 2017. New benchmark instances for the capacitated vehicle routing problem. Eur. J. Oper. Res. 257 (3), 845–858.

Vidal, Thibaut, Crainic, Teodor Gabriel, Gendreau, Michel, Prins, Christian, 2013. Heuristics for multi-attribute vehicle routing problems: A survey and synthesis. Eur. J. Oper. Res. 231 (1), 1–21.

Wang, Yongzhen, Chen, Yan, Lin, Yan, 2017. Memetic algorithm based on sequential variable neighborhood descent for the minmax multiple traveling salesman problem. Comput. Ind. Eng. 106, 105–122.

Whizzkids'96, 1996. https://www.win.tue.nl/whizzkids/1996/.

Yuan, Shuai, Skinner, Bradley, Huang, Shoudong, Liu, Dikai, 2013. A new crossover approach for solving the multiple travelling salesmen problem using genetic algorithms. Eur. J. Oper. Res. 228 (1), 72–82.

Zhan, Cheng, Zeng, Yong, 2019. Completion time minimization for multi-UAV-enabled data collection. IEEE Trans. Wirel. Commun. 18 (10), 4859–4872.

Zhang, Ruiyou, Zhao, Haishu, Moon, Ilkyeong, 2018. Range-based truck-state transition modeling method for foldable container drayage services. Transp. Res. Part E: Logist. Transp. Rev. 118, 225–239.