



**HAL**  
open science

# Hybrid search with neighborhood reduction for the multiple traveling salesman problem

Pengfei He, Jin-Kao Hao

► **To cite this version:**

Pengfei He, Jin-Kao Hao. Hybrid search with neighborhood reduction for the multiple traveling salesman problem. *Computers and Operations Research*, 2022, 142, pp.105726. <10.1016/j.cor.2022.105726>. <hal-03735784>

**HAL Id: hal-03735784**

**<https://hal.science/hal-03735784v1>**

Submitted on 22 Jul 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Distributed under a Creative Commons CC BY-NC 4.0 - Attribution - Non-commercial use - International License

# Hybrid search with neighborhood reduction for the multiple traveling salesman problem

Pengfei He and Jin-Kao Hao \*

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

Minor revision Jan 2022; 1st rev. Oct 2021; Init. sub. June 2021

---

## 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.

*Keywords:* Traveling salesman; Multiple traveling salesman; Hybrid heuristic; Neighborhood reduction.

---

## 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  $V = \{0, 1, \dots, n\}$  and a set of arcs  $A$ , where 0 of  $V$  is the depot and the remaining vertices  $N = \{1, \dots, n\}$  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

---

\* Corresponding author.

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

9 matrix  $C$  is said to be symmetric when  $c_{ij} = c_{ji}, (i, j) \in A$  and asymmetric  
10 otherwise. A *feasible* solution is a partition of the set of cities  $N$  into  $m$  distinct  
11 Hamiltonian tours  $\{r_1, r_2, \dots, r_m\}$ , such that each tour  $r_k$  ( $k \in \{1, \dots, m\}$ )  
12 starts and ends at the depot, and includes at least one city. The minsum  
13 mTSP, first proposed in [39], is to minimize the total traveling tour-length of  
14 a given mTSP instance and can be described by the following mathematical  
15 model [10].

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

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

20 By minimizing the total tour-length of all the salesmen, the minsum mTSP  
21 aims to optimize the total efficiency of a solution. In some contexts, it is useful  
22 to consider the equity criterion by avoiding excessive tour-length differences  
23 among the salesmen. To this end, the minmax mTSP was introduced in [13],  
24 which minimizes the longest tour and can be formulated by the mathematical  
25 model as follows [10].

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

26 From an application perspective, these mTSP models are useful for a number  
27 of real problems that cannot be formulated conveniently with the classical  
28 TSP model [10]. Representative examples include news paper delivery [46],  
29 hot rolling scheduling [41], 3D path planning [12], multi-unit service schedul-  
30 ing [9], path planning for robot and UAV [48,23], container drayage services  
31 [49,36], and harvesters scheduling [19,18]. Additional practical problems can  
32 be formulated by extended mTSP variants [5,29,33].

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

35 Due to its theoretical and practical interest, the mTSP has received much  
36 attention from various fields including engineering, operations research and

37 computer science. There are exact algorithms for the minsum mTSP, includ-  
38 ing a branch-and-bound algorithm [16] and a cutting plane algorithm [24].  
39 Optimal results were reported on instances with up to 500 vertices and 10  
40 salesmen. There are also exact algorithms for variants of the minmax mTSP.  
41 For example, a branch-and-cut algorithm [1] was presented to solve a minmax  
42 vehicle routing problem on instances up to 120 cities and 4 vehicles. Benders  
43 decomposition algorithms [5] were proposed to optimally solve the mTSP with  
44 load balancing on instances with up to 171 cities and 10 salesmen. Given the  
45 NP-hard nature of the problem, a number of heuristic and metaheuristic al-  
46 gorithms have been developed to find suboptimal solutions for large instances  
47 that cannot be optimally solved, as reviewed in Section 2.

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

54 In this work, we aim to advance the state-of-the-art of solving large-scale  
55 instances of the mTSP for both objectives. For this purpose, we introduce  
56 an effective hybrid search algorithm that performs well especially on large  
57 mTSP instances. The proposed algorithm benefits from the symbiosis of inter-  
58 tour optimization and intra-tour optimization. The inter-tour optimization  
59 uses neighborhood search to improve the solution by exchanging information  
60 between two tours (via the *insert* and *cross-exchange* operators). The intra-  
61 tour optimization applies a TSP method (the EAX heuristic [30]) to keep  
62 each individual tour as short as possible. We carry out extensive experiments  
63 to show the competitiveness of the proposed algorithm. We perform additional  
64 experiments to assess the usefulness of its key ingredients. Finally, we present  
65 for the first time computational experiments of applying the TSP heuristic  
66 EAX to the minsum mTSP, and draw conclusions regarding the effectiveness  
67 of this approach.

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

## 73 2 Literature review

74 In this section, we provide a literature review of the most representative  
75 heuristic algorithms for the mTSP. These algorithms are divided into three  
76 categories: population-based evolutionary algorithms, swarm intelligence algo-

77 rithms and neighborhood-based local optimization. The reviewed algorithms  
78 are summarized in Table 1, where “both” means the corresponding algorithm  
79 solves both the minsum and minmax mTSP. For a comprehensive survey of  
80 exact and heuristic methods, the reader is referred to [4] and [10].

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 [8]	✓			both
Brown et al. [7]	✓			both
Singh and Baghel [37]	✓			both
Yuan et al. [47]	✓			both
Wang et al. [45]	✓			minmax
Karabulut et al. [22]	✓			both
Pan and Wang [31]		✓		both
Liu et al. [26]		✓		both
Pandiri and Singh [32]		✓		both
Lu and Yue [28]		✓		minmax
Soylu [38]			✓	both
<a href="#">Penna et al. [34]</a>			✓	minsum
<a href="#">Uchoa et al. [43]</a>			✓	minsum

81 Various population-based evolutionary algorithms have been proposed for  
82 solving the mTSP. In 2006, Carter and Ragsdale [8] presented a grouping  
83 genetic algorithm for the mTSP using a two-part chromosome to represent a  
84 solution. Compared to two previous chromosome representations, the two-part  
85 chromosome representation avoids redundant solutions and thus reduces the  
86 solution space. This work also introduced a set of benchmark instances with  
87 50-150 cities and 3-30 salesmen, and showed comparisons with genetic algo-  
88 rithms using other representations. Similarly, in 2007, Brown et al. [7] showed  
89 a follow-up study [8] of using another two-part chromosome representation  
90 where both real-valued genes and integer-valued genes are used. Another group  
91 of benchmark instances was proposed for their computational studies. Subse-  
92 quently, in 2009, Singh and Baghel [37] presented another grouping genetic  
93 algorithm with the so-called m-tour chromosome representation, where each  
94 tour is represented by an array and no ordering is imposed among tours. This  
95 algorithm employed a steady-state population replacement method, and out-  
96 performed the genetic algorithms of [8,7] in terms of the minsum mTSP and  
97 the minmax mTSP. In 2013, Yuan et al. [47] investigated a specific crossover  
98 operator (called TCX) based on the two-part chromosome of [8]. The proposed  
99 crossover aims to better preserve building block information during solution  
100 recombination while ensuring a good diversity. They showed a superior per-  
101 formance of their TCX-based genetic algorithm over genetic algorithms using  
102 three other crossover operators including the algorithm of [8]. In 2017, Wang et  
103 al. [45] designed a memetic algorithm (MASVND) for the minmax mTSP. The

104 algorithm employs recombination and mutation operators based on spatial dis-  
105 tribution [32] and incorporates four neighborhood search operators (one-point  
106 move, Or-opt<sub>2</sub> move, Or-opt<sub>3</sub> move and Or-opt<sub>4</sub> move) for the variable neigh-  
107 borhood descent. They introduced a new set of (large) benchmark instances  
108 and assessed MASVND for the minmax mTSP compared to ABC [32], IWO  
109 [32] and GVNS [38]. The results indicated that MASVND outperforms its  
110 competitors on large instances (with 532–1173 cities), but performs worse  
111 than IWO on small instances (with 51–318 cities). In 2021, Karabulut et al.  
112 [22] proposed an evolution strategy (ES) approach for solving the mTSP and  
113 multi-depots mTSP with non-predetermined depots. This approach adopts a  
114 self-adaptive Ruin and Recreate heuristic to generate offspring solutions, and  
115 a local search, including 3-opt, to further enhance the solution quality. The  
116 computational experiments showed the competitiveness of this approach on  
117 the minsum and minmax mTSP instances.

118 Another popular approach for solving the mTSP concerns swarm intelligence  
119 methods. In 2006, Pan and Wang [31] presented a basic ant colony optimiza-  
120 tion (ACO) algorithm and showed a limited comparison with a genetic algo-  
121 rithm. In 2009, Liu et al. [26] exposed another ACO algorithm which inte-  
122 grates local search for search intensification. They showed competitive results  
123 for the minsum mTSP and the minmax mTSP compared to a genetic algo-  
124 rithm on some benchmark instances. In 2019, Lu and Yue [28] introduced a  
125 mission-oriented ant-team ACO algorithm and reported comparative studies  
126 with previous algorithms on the instances of [8]. In 2015, Pandiri and Singh  
127 [32] presented several algorithms based on artificial bee colony (ABC) and  
128 invasive weed optimization (IWO) for the minsum mTSP and the minmax  
129 mTSP, which use local search for the post-optimization. There are two ver-  
130 sions of the ABC algorithm, where neighboring solutions are generated from  
131 the original solution based on different distance strategies. IWO can be con-  
132 sidered as a reinforced ABC algorithm because it generalizes ABC, by visiting  
133 more neighboring solutions at each generation. These algorithms showed ex-  
134 cellent performances and updated a majority of the best results of previous  
135 algorithms for the benchmark instances of [8,7,37].

136 Compared to the aforementioned approaches, [there are relatively few studies](#)  
137 [using neighborhood-based local optimization to solve the mTSP, among which](#)  
138 [the general variable neighborhood search heuristic \(GVNS\) presented by Soylu](#)  
139 [\[38\] is a representative example](#). Based on the m-tour solution representation,  
140 this algorithm applies six neighborhood search operators (one-point move,  
141 two types of Or-opt move, two-point move and three-point move, as well as  
142 2-opt) to find local optima and uses a random shaking method to escape local  
143 optimum traps. Experimental results indicated that the algorithm globally  
144 competes well with previous methods, except IWO [32] which showed superior  
145 results on the instances of [8].

146 One notices that iterated local search (ILS) algorithms were designed for the  
147 related capacitated vehicle routing problem (CVRP), that becomes the min-  
148 sum mTSP when the capacity is set to 1. In particular, Penna et al. [34]  
149 proposed an ILS algorithm which uses a variable neighborhood descent proce-  
150 dure, with a random neighborhood ordering, in the local search phase. Uchoa  
151 et al. [43] tested an ILS-based matheuristic algorithm on a set of new CVRP  
152 benchmark instances and reported several good results for the CVRP with ca-  
153 pacity of 1, which is equivalent to the minsum mTSP. Local search algorithms  
154 were also proposed for the balanced mTSP [14] and balanced dynamic mTSP  
155 [15].

156 Among the reviewed studies, the following algorithms hold the best-known re-  
157 sults on the commonly used mTSP benchmark instances introduced in [8,7,45]:  
158 ABC(VC), IWO [32], GVNS [38], MASVND [45] (for the minmax mTSP only)  
159 and ES [22]. Thus they can be considered to be the state-of-the-art methods  
160 for solving the mTSP, and are used as the main reference algorithms for the  
161 computational studies in this work. Nevertheless, none of the existing mTSP  
162 algorithms can be considered as the most effective for all benchmark instances  
163 for both the minsum and minmax objectives of the mTSP.

164 According to the reviewed studies, we observe that most existing mTSP al-  
165 gorithms are based on population-based and swarm intelligence approaches.  
166 These algorithms have fast convergences, and typically performed well on  
167 small instances. However, they showed inferior performances on large instances  
168 [45,22]. To advance the state-of-the-art of solving the mTSP, especially on large  
169 instances, this work introduces a hybrid algorithm that combines an efficient  
170 neighborhood search (for inter-tour optimization) and a traveling salesman  
171 heuristic (for intra-tour optimization).

172 Finally, it is known that the minsum mTSP can be conveniently transformed  
173 to the conventional TSP [21,35]. For a minsum mTSP instance  $G$  with  $n$   
174 vertices and  $m$  tours, this transformation leads to an equivalent TSP instance  
175  $G^T$  with  $n + m - 1$  vertices.  $G^T$  is an extension of  $G$  with  $m - 1$  additional  
176 vertices such that each new vertex is a duplicate of the depot in  $G$  and each  
177 pair of depots have a large enough (e.g., infinite) distance between them. Then  
178 a mTSP solution of  $G$  with  $m$  tours ( $m > 1$ ) can be obtained from a TSP  
179 solution of  $G^T$  (one single tour) by splitting the TSP solution of  $G^T$  with each  
180 depot as the delimiter. As the result, the minsum mTSP can be solved by any  
181 TSP algorithm in principle. However, this approach has not been investigated  
182 experimentally in the literature. We fill the gap in this study by reporting the  
183 first computational results obtained by a TSP heuristic algorithm. We also use  
184 these results as additional references to assess our algorithm on the minsum  
185 mTSP instances.

### 186 3 Hybrid Search with Neighborhood Reduction

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

#### 191 3.1 General procedure

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

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

#### 215 3.2 Initial solution

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

---

**Algorithm 1:** Main framework of HSNR for the mTSP

---

**Input:** Instance  $I$ , number of initial solutions  $\mu$ , parameter  $\tau$ , depth of tabu search  $\gamma$ , tabu tenure parameter  $\beta$ ;

**Output:** The best solution  $\varphi^*$  found so far;

```
1 begin
2    $\varphi \leftarrow Initialization(I, \mu)$ ; /* Generate an initial solution,
      Section 3.2 */
3    $\varphi^* \leftarrow \varphi$ ; /*  $\varphi^*$  records the best solution found so far */
4   while Stopping condition is not met do
5      $\langle \varphi, \varphi^*, R \rangle \leftarrow Insert\_based\_TS(\varphi, \varphi^*, \gamma, \beta)$ ; /* Inter-tour
      optimization by tabu search with the insert operator,
      Sections 3.3.1 & 3.3.4 */
6      $\langle \varphi, \varphi^*, R \rangle \leftarrow CrossExchange\_based\_TS(\varphi, \varphi^*, \gamma, \beta, \tau)$ ;
      /* Inter-tour optimization by tabu search with the
      cross-exchange operator, Sections 3.3.2 & 3.3.4 */
7      $\varphi \leftarrow EAX(\varphi, R)$ ; /* Intra-tour optimization with the TSP
      heuristic EAX, Section 3.4 */
8   end
9   return  $\varphi^*$ ;
10 end
```

---

224 minmax mTSP).

225 Specifically, in the case of the minsum mTSP, a random tour  $r_k$  is picked first  
226 among the  $m$  initial tours including only the depot and another city. Then the  
227 unassigned cities in  $N^-$  are randomly considered one after the other and each  
228 selected city is greedily inserted into the tour  $r_k$  at the position that leads  
229 to the smallest increase of the minsum objective. For the minmax mTSP, the  
230 unassigned cities are also randomly considered one by one. However, given  
231 that its objective is to minimize the longest tour, each selected city is inserted  
232 into the *current shortest* tour  $r_{cs}$  at the position with the least increase of  
233 this shortest tour  $r_{cs}$ . It is worth noting that for the minsum mTSP, the same  
234 tour  $r_k$  is used to host all the unassigned cities in  $N^-$ , while for the minmax  
235 mTSP, the shortest tour  $r_{cs}$  used for each city insertion could change between  
236 two successive iterations.

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

242 The inter-tour optimization component of HSNR relies on the insert and cross-  
 243 exchange operators, which are popular for solving a variety of vehicle routing  
 244 problems (e.g., [2,40,44]). For the mTSP, the insert operator was previously  
 245 used in the GVNS algorithm [38] as one of its six move operators and the  
 246 MASVND algorithm [45] one of the four move operators. In this work, in  
 247 addition to the basic insert operator, we adopt for the first time the cross-  
 248 exchange operator for solving the mTSP. Compared to insert, cross-exchange  
 249 is a large neighborhood operator, which may help the algorithm to attain  
 250 solutions that cannot be accessed with the insert operator.

## 251 3.3.1 Insert

252 Let  $\varphi = \{r_1, r_2, \dots, r_m\}$  be a candidate solution composed of  $m$  tours where  
 253  $r_k$  ( $k \in \{1, \dots, m\}$ ) represents the  $k$ th tour including the cities visited by the  
 254  $k$ th salesman. For each city, the insert operator looks for the best alternative  
 255 position for the city with the minimal move gain (i.e., objective variation).  
 256 When all cities are examined, the best move involving a pair of cities  $a$  and  $\pi_b$   
 257 is identified. Then the insert operator removes city  $a$  from tour  $r_a$  and reinserts  
 258  $a$  after city  $\pi_b$  in  $r_b$  ( $r_a \neq r_b$ ). After that, tour  $r_a$  is reconnected by linking  
 259 the city preceding  $a$  and the city succeeding  $a$ , while tour  $r_b$  is updated by  
 260 removing the link between the city preceding  $b$  and  $b$ . Fig. 1 illustrates one  
 261 insert operation with the reconnection of the two impacted tours  $r_a$  and  $r_b$ .

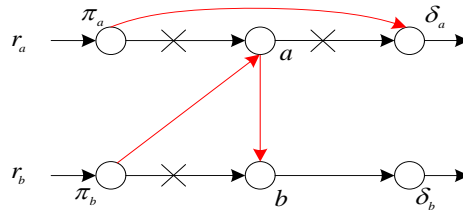


Fig. 1. Illustrative example of the insert operator. Removed links are marked with a cross and new links are marked in red.

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

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

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

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

276 where  $\pi_a$  and  $\delta_a$  are the city preceding and succeeding  $a$  in tour  $r_a$ , respectively,  
 277 while  $\pi_b$  and  $\delta_b$  are the city preceding and succeeding  $b$  in tour  $r_b$ , respectively.

278 For the minmax mTSP,  $\Delta$  is also obtained in constant time by Eq. (4).

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

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

### 280 3.3.2 Cross-exchange

281 Given a solution  $\varphi = \{r_1, \dots, r_m\}$ , the cross-exchange operator modifies two  
 282 tours (say  $r_a$  and  $r_b$ ) to generate a neighboring solution by removing four arcs  
 283 in  $r_a$  and  $r_b$ , and then adding four other arcs (see Fig. 2). Equivalently, a cross-  
 284 exchange operation can be viewed as exchanging a substring  $\hat{r}_a = (a, \dots, \sigma_a)$   
 285 from  $r_a$  and a substring  $\hat{r}_b = (b, \dots, \sigma_b)$  from another tour  $r_b$ . Besides, one  
 286 of the two substrings is reversible when they are exchanged, as shown in Fig.  
 287 2 (right) where the substring  $\hat{r}_a = (a, \dots, \sigma_a)$  is reversed. Clearly, without  
 288 any additional condition, this operator can lead to an extremely large neigh-  
 289 borhood (denoted by  $N_{CE}$ ) due to the size of the two exchanged substrings,  
 290 making its exploration highly time-consuming.

291 To reduce the cross-exchange neighborhood to a reasonable size, we follow the  
 292 idea of [40] developed for the vehicle routing problem (VRP) and limit the  
 293 number of cities (the substring size) of the two candidate substrings  $\hat{r}_a$  and  $\hat{r}_b$   
 294 to  $\tau$  cities at most (i.e.,  $|\hat{r}_a| \leq \tau$  and  $|\hat{r}_b| \leq \tau$ ) where  $\tau$  is a parameter. With  
 295 this constraint, the cardinality of  $N_{CE}(\varphi)$  is bounded by  $O(n^2 \times \tau^2)$  in the  
 296 general case.

297 Specifically, as shown in Fig. 2 (left), given a city  $a$ , a new neighbor in another  
 298 tour needs to be found. Let  $\pi_b$  be such a neighbor. Suppose that  $(\pi_b, a)$  is

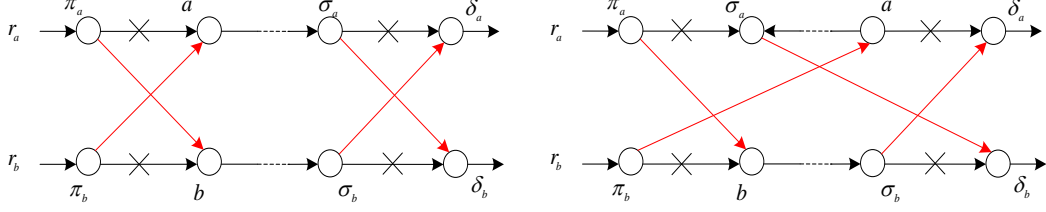


Fig. 2. Illustrative example of the cross-exchange operator. The removed arcs are marked with a cross and the added arcs are marked in red.

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

313 For the minsum mTSP, the move gain  $\Delta$  is computed by Eq. (5).

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

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

$$\begin{aligned} \Delta &= \max\{F'(r_a), F'(r_b), F(r_s)\} - F(r_a), \text{ if } r_b \neq r_s \\ \Delta &= \max\{F'(r_a), F'(r_b), F(r_t)\} - F(r_a), \text{ if } r_b = r_s \end{aligned} \quad (6)$$

$$\begin{aligned} F'(r_a) &= F(r_a) + c_{\pi_a b} + F(\hat{r}_b) + c_{\sigma_b \delta_a} - c_{\pi_a a} - F(\hat{r}_a) - c_{\sigma_a \delta_a} \\ F'(r_b) &= F(r_b) + c_{\pi_b a} + F(\hat{r}_a) + c_{\sigma_a \delta_b} - c_{\pi_b b} - F(\hat{r}_b) - c_{\sigma_b \delta_b} \end{aligned}$$

320 It is obvious that the move gain  $\Delta$  can be calculated in  $O(1)$  time for both

321 the minsum and minmax objectives.

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

### 330 3.3.3 Neighborhood reduction

331 The difficulty of exploring the large cross-exchange neighborhood has been  
332 recognized in the VRP communities for a long time. To cope with the diffi-  
333 culty related to large neighborhoods, neighborhood pruning techniques have  
334 been introduced for the VRP, such as  $\delta$ -nearest neighbors [3] and granular  
335 neighborhoods [42]. Rather than examining the entire neighborhood, pruning  
336 techniques limit the considered neighboring solutions to specifically identified  
337 (promising) solutions. Similar neighborhood reduction techniques have been  
338 proposed to accelerate TSP algorithms for solving large instances. One popu-  
339 lar technique is the  $\alpha$ -nearness strategy [20] that was designed to improve the  
340 computational efficiency of the well known Lin-Kernighan (LK) heuristic for  
341 the TSP [25] and was also applied to the VRP [2].

342 The  $\alpha$ -nearness strategy is developed by Helsgaun [20] based on sensitivity  
343 analysis using minimum spanning 1-trees and showing a high similarity be-  
344 tween a minimum 1-tree and an optimal TSP solution (they typically have  
345 70% to 80% of edges in common). In other words, edges that belong to a min-  
346 imum 1-tree stand a good chance of also belonging to an optimal tour and vice  
347 versa. Based on this, the  $\alpha$ -nearness strategy uses minimum 1-trees to identify  
348 a set of promising edges  $S$  that are more likely involved in the optimal TSP  
349 solution. Given that the mTSP is an extension of the TSP, it is reasonable to  
350 use minimum 1-trees as a nearness measure for the mTSP as well. As such,  
351 the edges belonging to minimum 1-trees will be considered as promising in the  
352 sense that they are highly probably part of the optimal solution of the mTSP.  
353 Therefore, the set of promising edges  $S$  identified by the  $\alpha$ -nearness strategy  
354 [20] can be beneficially adopted for solving the mTSP.

355 In this work, we explore for the first time the idea of using the  $\alpha$ -nearness to  
356 accelerate the insert and cross-exchange operations for the mTSP and show  
357 its practical effectiveness especially for handling large instances. The basic  
358 rationale is that one can ignore many neighboring solutions of low quality in-  
359 duced by the insert and cross-exchange operators and focus only on promising  
360 neighboring solutions. Consider the insert operator shown in Fig. 1 and let  $S$

361 be the set of promising edges identified by the  $\alpha$ -nearness as explained next.  
362 If an edge (say  $(\pi_b, a)$ ) belongs to  $S$ , then the corresponding move gain  $\Delta$  is  
363 evaluated; otherwise, the corresponding neighboring solution is ignored. When  
364 all the edges of  $S$  are considered and the corresponding move gains are eval-  
365 uated, the best neighboring solution is selected. Because the time complexity  
366 of evaluating a move gain is  $O(1)$  and  $|S|$  neighboring solutions are evalu-  
367 ated, the time complexity of evaluating the insert neighborhood is reduced to  
368  $O(|S|)$ . Similarly, for the cross-exchange operator shown in Fig. 2, if an arc  
369 (say  $(\pi_b, a)$ ) belongs to  $S$ , then the corresponding  $\tau^2$  move gains need to be  
370 evaluated. When all the edges of the set  $S$  are considered, the best neighbor-  
371 ing solution is acquired. Therefore, the time complexity of exploring the  $N_{CE}$   
372 neighborhood is reduced to  $O(|S| \times \tau^2)$ .

---

**Algorithm 2:** Generation of the set of promising edges  $S$  by the  $\alpha$ -nearness technique

---

**Input:** Input graph  $G = (V, A)$ , parameter  $\alpha$ ;

**Output:** The set of promising edges  $S$ ;

```

1 begin
2    $S \leftarrow \emptyset$ ;
3   Generate a minimum spanning tree ( $T^-$ ) for the cities of  $N$ ;
4     /* Prim's algorithm */
5   Generate a minimum 1-tree ( $T$ ); /* By adding to  $T^-$  two
6     shortest edges of  $A$  incident to the depot 0 */
7   for  $i = 0$  to  $n$  do
8     for  $j = 0$  to  $n$  do
9       Add edge  $(i, j)$  to  $T$ ;
10      Generate a new 1-tree ( $T^+$ ) /* By deleting the longest
11        edge from the new cycle containing edge  $(i, j)$  in the
12        tree ( $T$ ) */
13      Calculate the length of  $T^+$ ;
14    end
15    Get the  $\alpha$  shortest 1-trees from  $n$  1-trees;
16    Get the  $\alpha$  edges ( $E$ ) corresponding to the  $\alpha$  shortest 1-trees;
17     $S \leftarrow S \cup E$ ;
18  end
19  return  $S$ ;
20 end

```

---

373 We now explain how the set of promising edges  $S$  is identified with the  $\alpha$ -  
374 nearness technique based on the notion of 1-tree. As shown in Algorithm  
375 2, the minimum 1-tree ( $T$ ) for a graph  $G = (V, A)$  is a minimum spanning  
376 tree covering the cities of  $N$  together with two edges of  $A$  incident to the  
377 depot 0 (lines 3-4). By inserting a new edge  $(i, j)$  to  $T$ , a cycle containing  
378 edge  $(i, j)$  in the spanning tree is generated (line 7). Then, a new 1-tree is  
379 obtained by removing the longest edge on the cycle (line 8). When all edges

380 from  $V$  incident to vertex  $i$  are considered, the  $\alpha$  edges ( $\alpha$  is a parameter)  
381 corresponding to the  $\alpha$  shortest 1-trees ( $T^+$ ) are saved in the set  $S$  (lines  
382 11-12). This process continues until all the vertices in  $V$  are considered, and  
383 then the set of promising edges  $S$  is obtained. Based on the implementation  
384 techniques in [20], building the set  $S$  with the  $\alpha$ -nearness technique requires  
385  $O(n^2)$  time.

386 It is worth mentioning that no neighborhood reduction technique was em-  
387 ployed in the existing mTSP algorithms including the neighborhood search  
388 algorithm GVNS [38]. As we show in Section 5.1, the  $\alpha$ -nearness technique  
389 contributes positively to the performance of the HSNR algorithm.

### 390 3.3.4 Neighborhood exploration with tabu search

391 To examine candidate solutions of a mTSP instance, HSNR employs the well-  
392 known tabu search (TS) metaheuristic [17]. One notices that TS is a popular  
393 method for solving routing problems (e.g., [40,42]), that are more general  
394 models than the mTSP. In our case, we design the first tabu search procedure  
395 to explore the insert neighborhood  $N_I$  and the cross-exchange neighborhood  
396  $N_{CE}$  that are reduced by the  $\alpha$ -nearness technique of Section 3.3.3.

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

413 During the tabu search, if its best solution found ( $\varphi^*$ ) is not updated during  
414  $\gamma$  consecutive iterations, the search is judged to be exhausted and terminates  
415 while returning the best solution found, the current solution ( $\varphi$ ) and the set  
416 of modified tours ( $R$ ).

---

**Algorithm 3:** General tabu search

---

**Input:** Input solution  $\varphi$ , best solution  $\varphi^*$ , neighborhood  $N$ , depth of tabu search  $\gamma$ , tabu tenure parameter  $\beta$ ;

**Output:** Updated best solution  $\varphi^*$ , ending solution  $\varphi$ , set of modified tours  $R$ ;

```
1 begin
2    $i \leftarrow 0$ ;
3    $R \leftarrow \emptyset$ ;
4   Initialize tabu list  $L$ ;
5   while  $i \leq \gamma$  do
6     Choose the best eligible neighboring solution  $\varphi' \in N(\varphi)$ ;
7      $\varphi \leftarrow \varphi'$ ;
8     Update  $L$  and  $R$ ; /* Update the tabu list and set of
9       modified tours */
10    if  $F(\varphi) < F(\varphi^*)$  then
11       $\varphi^* \leftarrow \varphi$ ; /* Update the best solution  $\varphi^*$  */
12       $i \leftarrow 0$ ;
13    else
14       $i \leftarrow c + 1$ ;
15    end
16  end
17 return  $\langle \varphi, \varphi^*, R \rangle$ ;
18 end
```

---

#### 417 3.4 Intra-tour optimization with the TSP heuristic EAX

418 Given a candidate solution  $\varphi = \{r_1, \dots, r_m\}$ , it is easy to observe that each  
419 individual tour  $r_k$  can be considered as a TSP tour. As the result, existing TSP  
420 algorithms (e.g., 2-opt and LK) can directly be used to optimize the mTSP  
421 objectives by minimizing an individual tour without the need for designing  
422 new optimization methods. Indeed, this idea proved to be quite effective for  
423 several VRPs [2,3] and has been used in the GVNS algorithm for the mTSP  
424 (with the 2-opt heuristic) [38] as well. In this work, the EAX heuristic [30]<sup>1</sup>,  
425 which is among the best TSP heuristics, is adopted for intra-tour optimization.

426 Specifically, for each tour  $r_k$  in the set  $R$  (It records the tours modified by the  
427 insert and cross-exchange operators during tabu search), EAX is applied to  
428 minimize the tour as follows. First, the tour  $r_k$  is mapped to a standard TSP  
429 tour, by renaming the cities of the tour with consecutive numbers. Second,  
430 EAX is run to optimize the TSP tour. Given that the number of cities in  
431 a tour is relatively small (typically from several tens to several hundreds of  
432 cities for the mTSP benchmark instances), EAX needs a short time to make

---

<sup>1</sup> <https://github.com/sugia/GA-for-TSP>

433 the TSP tour optimal or close-to-optimal. Third, we map the optimized TSP  
434 tour back to the corresponding mTSP tour. Experiments showed that the  
435 intra-tour optimization using EAX contributes favorably to the performance  
436 of the HSNR algorithm.

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

- 443 (1) Define the undirected multigraph  $G_{AB} = (V, E_A \cup E_B)$  from edge sets  $E_A$   
444 and  $E_B$ ;
- 445 (2) Partition the edges of  $E_A \cup E_B$  into *AB-cycles*, where an *AB-cycle* is a  
446 cycle in  $G_{AB}$ , such that edges of  $E_A$  and edges of  $E_B$  are alternatively  
447 linked;
- 448 (3) Build an  $E_{set}$  by selecting some *AB-cycles* according to a selection crite-  
449 rion;
- 450 (4) Build an intermediate solution  $E_C$  from  $p_A$  by removing the edges of  $E_A$   
451 that appear in  $E_{set}$  and adding the edges of  $E_B$  that appear in  $E_{set}$ , i.e.,  
452  $E_C := (E_A \setminus (E_{set} \cap E_A)) \cup (E_{set} \cap E_B)$ ;
- 453 (5) Generate an offspring solution by connecting all subtours of  $E_C$  to obtain  
454 a single tour.

455 As we show in Section 5.1, the EAX heuristic is quite beneficial for the pro-  
456 posed algorithm. This is the first application of this TSP heuristic within a  
457 mTSP algorithm.

## 458 4 Computational Results and Comparisons

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

### 462 4.1 Benchmark instances

463 Our experiments are based on two sets of 77 instances covering small, medium  
464 and large instances (available from the link of footnote 3).

465 **Set I (41 instances):** These instances were introduced in [8,7,45]. Carter  
466 and Ragsdale [8] presented 12 instances using 3 TSP graphs (with 51, 100,  
467 150 cities and 3, 5, 10, 20 and 30 tours), while Brown et al. [7] also defined  
468 12 instances using 3 TSP graphs (from 51 cities and 3 tours up to 150 cities  
469 and 30 tours). Note that among these 3 graphs adopted in [7], only one graph

470 (*gtsp150*) is not used in [8]. Therefore, most of the instances in [8] and [7]  
471 share the same features. We thus exclude the redundant instances and keep  
472 17 distinct instances out of these 24 instances. For these 17 instances, the  
473 best-known objective values are available in the literature for both mTSP  
474 objectives. Wang et al. [45] defined 31 instances using 8 graphs (with 51-1173  
475 cities and 3-20 tours) and tested them only for the minmax mTSP. Among  
476 the 8 used graphs, one is a graph used in [8] and one is a graph used in [7].  
477 By eliminating these redundant instances, we retain 24 instances out of the 31  
478 instances. For these instances, the best-known objective values are available  
479 only for the minmax mTSP. The instances of Set I are limited to 1173 cities  
480 and 30 tours and their optimal values are still unknown in the literature.

481 **Set II (36 instances):** This is a new set of large instances with 1379-  
482 5915 cities and 3-20 tours introduced in this study. Like previous benchmark  
483 instances, these instances were generated from 9 TSP graphs in TSPLib<sup>2</sup>  
484 (*nrrw1379*, *fl1400*, *d1655*, *u2152*, *pr2392*, *pcb3038*, *fl3795*, *fnl4461*, *rl5915*),  
485 which come from different practical problems. The optimal values for these  
486 instances are unknown.

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

#### 490 4.2 *Experimental protocol and reference algorithms*

491 **Parameter setting.** HSNR has 5 parameters: number of candidate solutions  
492 for initialization  $\mu$ , neighborhood reduction parameter  $\alpha$ , substring size  $\tau$ ,  
493 depth of tabu search  $\gamma$  and tabu tenure parameter  $\beta$ . In order to calibrate these  
494 parameters, the "IRACE" package [27] was used to automatically identify a  
495 set of suitable values. The tuning was performed on 8 representative instances  
496 (with 150-1173 cities). For the experiment, the tuning budget was set to 1080  
497 runs, with a cutoff time of  $n/100$  minutes. The candidate values of these  
498 parameters and their final values given by IRACE are shown in Table 2.

499 **Reference algorithms.** According to the literature, five algorithms (IWO &  
500 ABC(VC) [32], GVNS [38], MASVND [45] and ES [22]) represent the state-  
501 of-the-art for solving the mTSP (MASVND for the minmax mTSP only).  
502 Thus these algorithms are adopted as the main references for our compar-  
503 ative studies. Given that only one code is available (an executable code of  
504 ES kindly provided by its authors), we faithfully re-implemented ABC(VC),  
505 IWO, GVNS and MASVND (denoted by re-ABC(VC), re-IWO, re-GVNS and  
506 re-MASVND) and verified that our implementations were able to match the  
507 results reported in [32,38,45].

---

<sup>2</sup> <http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/index.html>

508 Finally, as indicated in Section 2, the minsum mTSP can be transformed to  
 509 the standard TSP. We provide the results obtained by the TSP heuristic EAX  
 510 [30] in Appendix A.2.

511 **Experimental setting.** HSNR and the re-implemented reference algorithms  
 512 were programmed in C++<sup>3</sup> and compiled with the g++ compiler with the  
 513 -O3 option. All the experiments were conducted on a computer with an Intel  
 514 Xeon E5-2670 processor of 2.5 GHz CPU and 6 GB RAM running Linux.  
 515 Given the stochastic nature of the compared algorithms, each algorithm was  
 516 run 20 times on each instance with different random seeds. We used the default  
 517 parameter setting of Table 2 to run HSNR, while for the reference algorithms,  
 518 we adopted their default parameter settings given in [32,38,45].

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

Table 2  
 Parameters tuning results

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

### 523 4.3 Computational results and comparison

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

531 For each instance, we show the best-known objective value BKS ever reported  
 532 in the literature (when it is available), the best objective value obtained by an  
 533 algorithm *Best* and the average objective value *Avg.*. For our HSNR algorithm,  
 534 we additionally report the gap of its best objective value to the previous best  
 535 objective value calculated as  $Gap(\%) = 100(Best - AllBest)/AllBest$  with

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

536 *Best* and *AllBest* being respectively the best objective value of HSNR and the  
537 best objective value from all reference algorithms (including those published  
538 in the literature). Given that the mTSP is a minimization problem, a negative  
539 gap indicates an improved best result. The background of the top results for  
540 each instance is highlighted in dark gray; the second best results in medium  
541 gray; and the worst results in the lightest gray. Note that in the literature, the  
542 results are rounded to the nearest integers, and we report our results in more  
543 precise real values.

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

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

#### 554 4.3.1 Results for the minsum mTSP

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

557 From Table 3, we can make the following comments about the instances of  
558 Set I. First, for the 17 instances for which the best-known results (BKS) are  
559 available, HSNR finds 6 improved results (with an improvement gap up to  
560 -0.24%), 7 equal results and 4 worse results. Second, for the remaining 24  
561 instances of Set I, HSNR clearly outperforms the reference algorithms both in  
562 terms of the best and average results, with more important improvements for  
563 the largest instances with at least 200 cities (improvement gap up to 10.39%  
564 for the largest instance). Also, even the average results of HSNR are better  
565 than the best results of the reference algorithms. Third, the small *p-values*  
566 from the Wilcoxon signed-rank tests confirm the statistical difference between  
567 the HSNR algorithm and the reference algorithms in terms of the best and  
568 average results.

569 From Table 4 on the large instances of Set II, we observe that the dominance  
570 of the HSNR algorithm over the reference algorithms is even more significant.  
571 Indeed, HSNR systematically reports better results in terms of the best and  
572 average values, with improvement gaps from 2.37% to 19.45% compared to the  
573 best results of the reference algorithms. Once again, even the average results  
574 of HSNR are far better than the best results of the compared algorithms.

Table 3. The minimum 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	21920.60	21920.60	21797.60	21807.54	21797.60	21952.94	21797.60	21797.60	0.00
mtsp100-5	23175	23256.10	23334.94	23282.45	23174.90	23282.45	23203.09	23137.60	27235.18	23174.90	23174.90	0.00
mtsp100-10	26927	27474.70	27775.28	27065.57	27074.80	27227.84	27227.84	27137.60	38603.80	26926.60	26983.51	0.00
mtsp100-20	38245	39581.30	40876.66	39165.78	38844.90	39105.67	38603.80	38603.80	8186.56	38245.10	38259.98	0.00
rand100-3	-	8012.13	8033.16	8018.91	8012.13	8046.32	8012.13	8012.13	8186.56	8012.13	8012.13	0.00
rand100-5	-	8257.84	8337.79	8252.37	8223.91	8252.37	8331.35	8308.88	8450.00	8223.91	8223.91	0.00
rand100-10	-	9635.39	9762.62	9485.79	9366.80	9485.79	9377.28	9366.80	9366.80	9366.80	9366.80	0.00
rand100-20	-	14175.00	14450.33	13627.89	13529.20	13627.89	13587.70	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	70281.97	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	6574.20	-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	8964.90	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	64553.50	64739.49	61923.50	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	68816.90	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	-	-	-

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)		
	<i>Best</i>	<i>Avg.</i>	<i>Best</i>	<i>Avg.</i>	<i>Best</i>	<i>Avg.</i>	<i>Best</i>	<i>Avg.</i>	<i>Gap(%)</i>
nrw1379-3	62099.80	62413.66	62211.90	62384.95	62449.60	63614.69	56775.70	56775.70	<b>-8.57</b>
nrw1379-5	62853.40	63036.26	62788.40	63011.51	63593.80	65998.39	56992.60	56999.16	<b>-9.23</b>
nrw1379-10	64985.10	65396.08	65147.40	65392.47	65011.90	69268.91	57636.20	57795.15	<b>-11.31</b>
nrw1379-20	72415.90	73267.10	71915.30	73075.37	69900.30	74382.44	59618.40	60278.03	<b>-14.71</b>
fl1400-3	21733.90	21819.77	21682.60	21771.70	24456.90	25566.53	21169.40	21169.47	<b>-2.37</b>
fl1400-5	23051.40	23179.70	22841.20	23068.25	24030.00	26993.65	22066.20	22238.10	<b>-3.39</b>
fl1400-10	27960.10	28563.58	27556.10	27933.99	28276.70	30150.92	24373.90	25069.65	<b>-11.55</b>
fl1400-20	44588.20	47458.31	44715.00	45981.11	32713.30	34886.35	29579.20	31966.86	<b>-9.58</b>
d1655-3	76672.20	77095.10	76471.40	76887.31	78155.30	79462.89	68364.40	68370.50	<b>-10.60</b>
d1655-5	83908.00	84208.31	83221.80	83962.59	86806.30	89456.39	74273.50	74292.65	<b>-10.75</b>
d1655-10	102457.00	103865.80	102268.00	103386.30	100732.00	105478.45	89262.50	89856.83	<b>-11.39</b>
d1655-20	146870.00	149739.75	147454.00	149130.20	134860.00	143426.30	121373.00	124263.45	<b>-10.00</b>
u2152-3	75107.40	75322.56	74957.90	75399.52	73757.10	75777.34	65064.90	65072.31	<b>-11.78</b>
u2152-5	75533.50	76109.51	75686.10	76083.68	74271.40	78510.40	65201.70	65219.93	<b>-12.21</b>
u2152-10	78836.20	79676.56	78726.40	79471.17	75482.90	83485.66	65762.50	66291.71	<b>-12.88</b>
u2152-20	89564.50	91776.90	89331.80	91322.73	80486.60	85760.90	67993.10	71115.74	<b>-15.52</b>
pr2392-3	428886.00	430482.05	428802.00	429994.15	423607.00	433789.50	378661.00	378661.00	<b>-10.61</b>
pr2392-5	433633.00	437696.40	435449.00	438130.75	426073.00	444213.90	380061.00	380069.40	<b>-10.80</b>
pr2392-10	462078.00	465864.35	458177.00	465361.70	441436.00	476382.30	387498.00	389012.85	<b>-12.22</b>
pr2392-20	539219.00	549174.10	542251.00	549066.05	459442.00	502937.95	417424.00	421532.30	<b>-9.15</b>
pcb3038-3	156742.00	157141.25	156844.00	157227.80	153338.00	155312.45	137916.00	137925.00	<b>-10.06</b>
pcb3038-5	158160.00	158614.05	157607.00	158559.90	156678.00	159923.10	138121.00	138123.20	<b>-11.84</b>
pcb3038-10	162709.00	164019.75	163743.00	164470.35	156525.00	162016.80	139142.00	139379.85	<b>-11.11</b>
pcb3038-20	181677.00	183532.75	181894.00	183531.15	153084.00	170283.40	144295.00	146491.65	<b>-5.74</b>
fl3795-3	32749.00	32983.87	32678.10	32817.07	34634.30	37772.26	29589.90	29823.75	<b>-9.45</b>
fl3795-5	33924.60	34497.01	33833.20	34198.05	37162.40	40342.25	30480.80	31048.26	<b>-9.91</b>
fl3795-10	39470.20	40288.27	38864.50	39779.70	36823.70	41088.57	32729.60	35467.72	<b>-11.12</b>
fl3795-20	53852.70	55606.56	53723.40	55121.13	41337.00	45838.94	39083.80	45437.27	<b>-5.45</b>
fln4461-3	204334.00	204844.15	204490.00	204833.45	203756.00	206706.75	182888.00	182890.85	<b>-10.24</b>
fln4461-5	205639.00	206196.00	205745.00	206132.15	207600.00	212214.50	183074.00	183076.50	<b>-10.97</b>
fln4461-10	210341.00	211064.95	210158.00	210906.80	215447.00	224158.65	183808.00	184811.75	<b>-12.54</b>
fln4461-20	224749.00	225855.50	223448.00	225219.15	221402.00	236283.55	191025.00	193356.10	<b>-13.72</b>
rl5915-3	676316.00	678576.60	676268.00	679179.35	666852.00	707708.75	565949.00	566066.70	<b>-15.13</b>
rl5915-5	678177.00	680809.90	673768.00	680248.85	703003.00	746016.20	566626.00	566780.55	<b>-15.90</b>
rl5915-10	692109.00	694947.55	689402.00	694087.15	783210.00	811408.35	569619.00	573689.20	<b>-17.37</b>
rl5915-20	744400.00	752084.65	742284.00	750748.75	777638.00	861515.54	597878.00	609385.79	<b>-19.45</b>
AVG	206327.84	207978.02	206011.24	207718.79	204834.24	216892.61	173371.56	174716.80	-
BKS#	0	0	0	0	0	0	<b>36</b>	<b>36</b>	-
<i>p-value</i>	1.68E-07	1.68E-07	1.68E-07	1.68E-07	1.68E-07	1.68E-07	-	-	-

575 Finally, the Wilcoxon signed-rank tests confirm the statistical difference of  
576 these comparisons.

577 To further assess the compared algorithms, we also present the performance  
578 profiles [11] to visually illustrate the performance of each algorithm. Perform-  
579 ance profiles rely on a specific performance metric (in our case, we use  $f_{best}$   
580 and  $f_{avg}$ ). To compare a set of algorithms  $S$  over a set of problems  $Q$ , the  
581 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  
582 report result for a problem  $q$ ,  $r_{s,q} = +\infty$ . The performance function of an  
583 algorithm  $s$  is computed by  $Q_s(\tau) = \frac{|q \in Q | r_{s,q} \leq \tau|}{|Q|}$ . The value  $Q_s(\tau)$  computes  
584 the fraction of problems that algorithm  $s$  can solve with at most  $\tau$  many times  
585 the cost of the best algorithm. For example,  $Q_s(1)$  equals the number of prob-  
586 lems that algorithm  $s$  solved better than, or as good as the other algorithms  
587 in  $Q$ . Similarly, the value  $Q_s(r_f)$  is the maximum number of problems that  
588 algorithm  $s$  solved. Therefore,  $Q_s(1)$  and  $Q_s(r_f)$  represent the efficiency and  
589 robustness of algorithm  $s$ . Fig. 3 visually illustrates the competitiveness of  
590 HSNR in terms of the best and average values on the benchmark 77 instances.

591 Indeed, HSNR has a much higher  $Q_s(1)$  value compared to the reference algo-  
 592 rithms, by finding better or equal results for nearly all instances. Furthermore,  
 593 HSNR also reaches  $Q_s(r_f)$  first, indicating a high robustness of our approach.  
 594 In brief, compared with the reference algorithms, HSNR is the best solution  
 595 approach for the minsum mTSP on both small and large scale instances.

596 Finally, since the minsum mTSP can be transformed to the TSP, we show in  
 597 Appendix A.2 the results obtained by the effective TSP heuristic EAX [30].

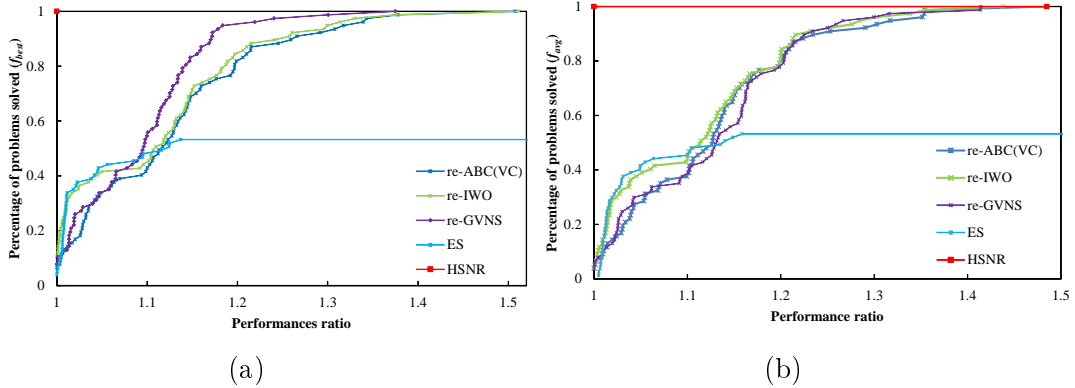


Fig. 3. The minsum mTSP: performance profiles of HSNR and four reference algorithms on all the 77 benchmark instances. The left figure corresponds to the best results while the right figure is for the average results.

#### 598 4.3.2 Results for the minmax mTSP

599 We now assess the performance of the HSNR algorithm for the minmax mTSP.  
 600 For this problem, ABC(VC) [32], IWO [32], GVNS [38], MASVND [45] and  
 601 ES [22] are the state-of-the-art algorithms, which are used for our comparative  
 602 study. Note that for three graphs *kroA200*, *lin318*, *att532*, the initial solutions  
 603 of HSNR are generated in such a way that each city is greedily inserted in an  
 604 arbitrary random tour, not limited to the shortest tour.

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

617 best and average results.

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

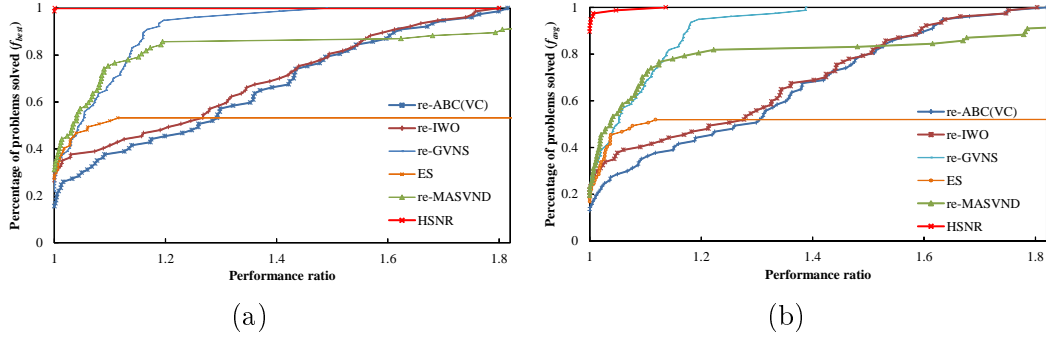


Fig. 4. The minmax mTSP: performance profiles of HSNR and five reference algorithms on all the 77 benchmark instances. The left figure corresponds to the best results while the right figure is for the average results.

626 Finally, Table 7 summarizes the comparative results of each pair of compared  
 627 algorithms on the 77 benchmark instances, by providing the number of in-  
 628 stances for which HSNR obtained a better ( $\#Wins$ ), equal ( $\#Ties$ ) or worse  
 629 ( $\#Losses$ ) result compared to each reference algorithm and the BKS value.

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

## 633 5 Analysis

634 The computational results and comparisons with the state-of-the-art algo-  
 635 rithms presented in Section 4 showed high effectiveness of the HSNR algo-  
 636 rithm. This section aims to investigate the contributions of two important  
 637 ingredients of HSNR: the neighborhood reduction strategy (Section 3.3.3) for  
 638 efficient neighborhood examination and the EAX heuristic (Section 3.4) for  
 639 effective intra-tour optimization. For this purpose, we performed additional  
 640 experiments to compare HSNR with several HSNR variants where the studied  
 641 component (i.e., neighborhood reduction and EAX) was disabled and replaced  
 642 by another alternative method. These experiments were based on 20 represen-  
 643 tative instances with different sizes ( $n$  from 150 to 2392,  $m$  from 3 to 20) and

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	re-ABC(VC) (2015)		re-IWO (2015)		re-GVNS (2015)		ES (2021)		re-MASVND (2017)		HSNR (this work)			
	BKS	Avg.	Best	Avg.	Best	Avg.	Best	Avg.	Best	Avg.	Best	Avg.	Gap(%)	
mtsp51-3	159.57	159.57	159.57	159.85	159.57	159.57	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	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	0.00	
mtsp100-3	8509.00	8544.69	8590.07	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	6833.45	6767.82	6776.61	6767.02	6832.74	6767.02	6801.75	6767.02	6770.67	0.00	
mtsp100-10	6358.00	6358.49	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	0.00	
rand100-3	-	3032.58	3044.01	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	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	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	0.00	
mtsp150-3	13151.00	13497.20	13645.64	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	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	5751.31	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	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	0.00	
gtsp150-3	2407.34	2445.92	2486.43	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	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	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	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	0.00	
kroA200-3	10768.10	11223.30	11511.69	10801.30	10965.59	11061.60	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	7536.91	7770.43	7484.17	7634.61	7418.87	7494.44	0.04	
kroA200-10	6223.22	6299.77	6456.58	6223.22	6223.22	6223.22	6223.22	6223.22	6223.22	6223.22	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	0.00	
lin318-3	16088.73	17339.30	18119.72	16133.40	17006.92	16362.30	16532.08	16349.60	16797.80	16551.60	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	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	9731.17	9731.17	0.00	
lin318-20	9731.17	9750.68	9787.11	9731.17	9731.18	9731.17	9731.17	9731.17	9731.17	9731.17	9731.17	9731.17	0.00	
att532-3	-	1201.00	12169.15	11258.00	11525.50	10656.00	10762.95	10585.00	10953.90	10566.00	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	7067.00	7334.00	-2.91	
att532-10	-	6696.00	6829.10	6427.00	6562.90	6449.00	6650.00	5761.00	5806.75	5745.00	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	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	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	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	1334.76	1345.88	-1.92	
rat783-20	1231.69	1386.96	1416.13	1371.32	1390.09	1231.69	1240.96	1231.69	1231.69	1231.69	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	20813.80	21144.92	-4.44	
pcb1173-5	14099.50	18692.20	19292.07	18703.50	19226.82	14566.20	14853.24	14333.30	14601.30	14088.40	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	8052.28	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	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	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-MAVSND (2017)		HSNR (this work)		
	Best.	Avg.	Best.	Avg.	Best.	Avg.	Best.	Avg.	Best.	Avg.	Gap(%)
nw1379-3	25566.10	26173.22	24401.20	25204.30	21746.00	21946.34	22296.40	23349.12	20495.90	20765.70	-5.75
nw1379-5	17765.00	18097.04	17636.70	18019.88	14105.50	14382.16	13368.80	13847.14	12416.50	12652.56	-7.12
nw1379-10	10185.60	10595.08	10145.40	10404.29	8026.78	8240.14	7588.59	7748.69	7114.71	7212.24	-6.18
nw1379-20	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
p2392-3	192348.00	195388.25	186013.00	190584.70	151300.00	155742.30	254034.00	256052.65	141627.00	143703.00	-6.39
p2392-5	134496.00	135676.40	133780.00	135073.30	102087.00	104603.90	104977.00	132626.10	88083.20	89582.83	-13.72
p2392-10	82834.80	84149.45	80135.10	83131.04	58955.70	60860.42	55337.60	56650.63	51085.30	52100.80	-7.68
p2392-20	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	58795.40	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
fl14461-5	59246.10	60047.22	59532.70	60170.68	48352.60	49343.13	83650.40	84430.87	66903.70	67971.34	-12.25
fl14461-10	34671.80	34942.84	34068.20	34741.06	26182.10	26871.35	25385.20	43581.63	22041.50	22891.45	-13.17
fl14461-20	22113.00	22798.53	22142.80	22852.80	15810.10	16341.60	14611.30	15262.97	12630.10	13046.38	-13.56
r15915-3	329296.00	334606.95	328020.00	332327.15	284176.00	314531.05	443748.00	445979.00	213864.00	226819.75	-24.74
r15915-5	224206.00	226396.90	221495.00	225566.65	198641.00	201423.65	362776.00	364717.65	133457.00	145173.07	-32.81
r15915-10	135096.00	137649.80	133266.00	137737.55	89353.00	98436.47	267295.00	270354.45	76585.20	84459.02	-14.29
r15915-20	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

644 followed the experimental protocol of Section 4.2.

### 645 5.1 Importance of the the $\alpha$ -nearness technique for neighborhood reduction

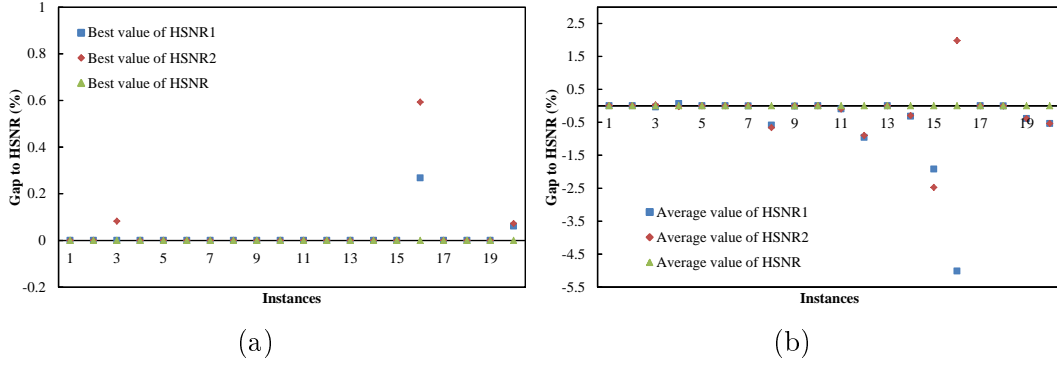


Fig. 5. Minsum mTSP: comparative results of HSNR with HSNR1 (using  $\delta$ -nearest neighbors) and HSNR2 (without pruning).

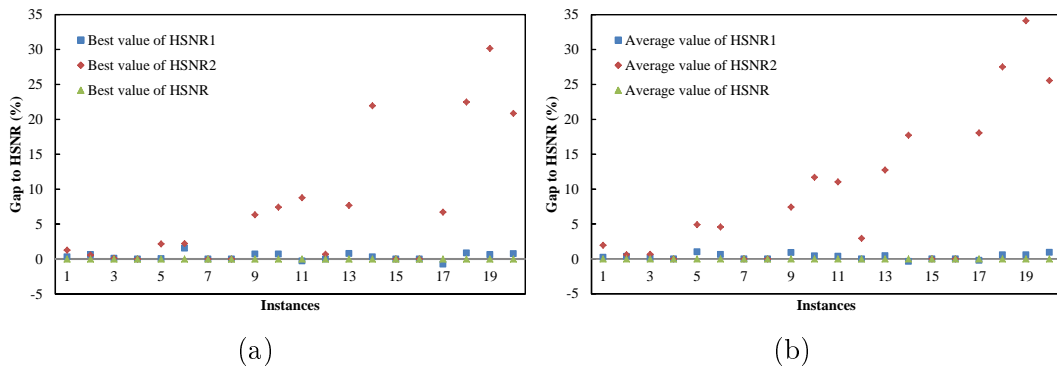


Fig. 6. Minmax mTSP: comparative results of HSNR with HSNR1 (using  $\delta$ -nearest neighbors) and HSNR2 (without pruning).

646 To study the benefit of the  $\alpha$ -nearness pruning technique (Section 3.3.3), we  
 647 compared HSNR with two alternative versions: HSNR1 where the  $\alpha$ -nearness

648 pruning technique was replaced by the method of  $\delta$ -nearest neighbors [2,6], and  
649 HSNR2 where no pruning technique was used. As such, at each neighborhood  
650 search iteration of HSNR1, city  $a$  must be one of the  $\delta$ -nearest cities of city  $\pi_b$   
651 ( $\delta$  was set to 40), as shown in the illustrative example of Fig. 1. For HSNR2,  
652 there is no any restriction between city  $a$  and  $\pi_b$ .

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

657 For the minsum mTSP, compared to HSNR2 which doesn't use any neigh-  
658 borhood pruning technique, both reductions ( $\alpha$ -nearness pruning for HSNR  
659 and  $\delta$ -nearest pruning for HSNR1) led to slightly better results in terms of  
660 the best objectives values, while the average quality was slightly scarified in  
661 several cases. The Wilcoxon signed-rank tests in Table 8, however, don't con-  
662 firm statistically significant differences between the compared algorithms. For  
663 the minmax mTSP, both HSNR and HSNR1 significantly outperformed the  
664 HSNR2 variant in terms of the best and average values (confirmed by the  
665 Wilcoxon signed-rank tests). The importance of the pruning techniques is  
666 even more amplified on large instances. One also observes that HSNR using  
667 the  $\alpha$ -nearness pruning technique systematically showed better performances  
668 than HSNR1 using the  $\delta$ -nearest neighbors technique. As an example, the con-  
669 vergence charts shown in Fig. 7 also illustrate the usefulness of the  $\alpha$ -nearness  
670 pruning technique on a representative instance.

671 This experiment confirms the interest of heuristic pruning techniques, espe-  
672 cially the  $\alpha$ -nearness technique adopted in the HSNR algorithm. By avoiding  
673 useless examinations of non-promising neighboring solutions, the neighbor-  
674 hood reduction strategy is particularly useful for solving large instances of the  
675 minmax mTSP, even if its contribution to the minsum mTSP is less significant.

## 676 5.2 Importance of the EAX heuristic for intra-optimization

677 To evaluate the benefits of the EAX heuristic for intra-tour optimization (Sec-  
678 tion 3.4), we compare HSNR with two alternative algorithms: HSNR3 where  
679 EAX is replaced by the popular 2-opt heuristic, and HSNR4 where EAX is  
680 replaced by the LK algorithm [25]. The comparative results are shown in Figs.  
681 8 and 9 as well as Table 8.

682 For the minsum mTSP, HSNR with EAX significantly dominates its variants  
683 with the 2-opt and LK heuristics in terms of the best and average results (con-  
684 firmed by the Wilcoxon signed-rank tests). For the minmax mTSP, HSNR also  
685 performs better than its competitors except for a small number of instances.  
686 This experiment demonstrates clearly the usefulness of the TSP heuristic EAX

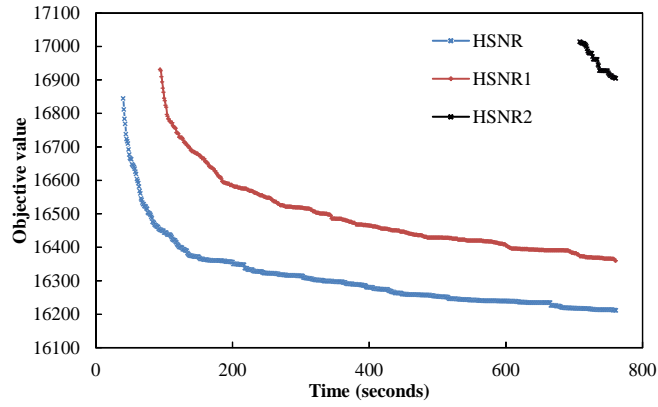


Fig. 7. Convergence chart (running profiles) of HSNR and two HSNR variants for solving instance *lin318-3* with the minmax mTSP. The results were obtained from 20 independent executions of each algorithm.

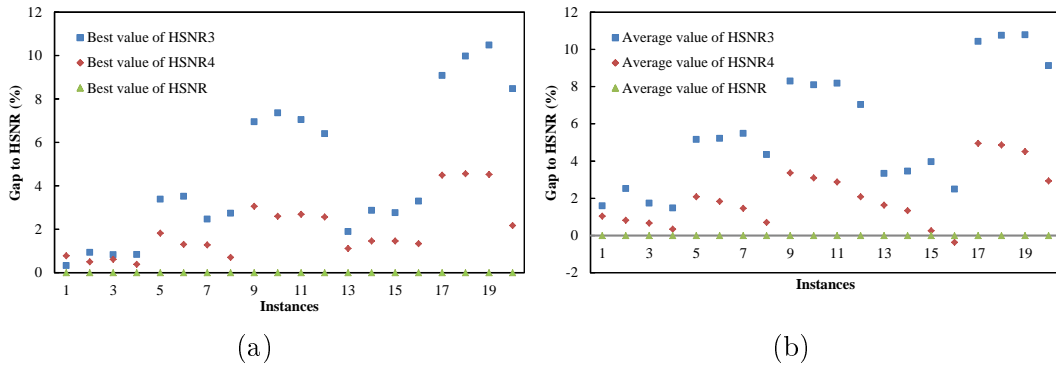


Fig. 8. Minsum mTSP: comparative results of HSNR (using EAX) with HSNR3 (using the 2-opt heuristic) and HSNR4 (using the LK algorithm).

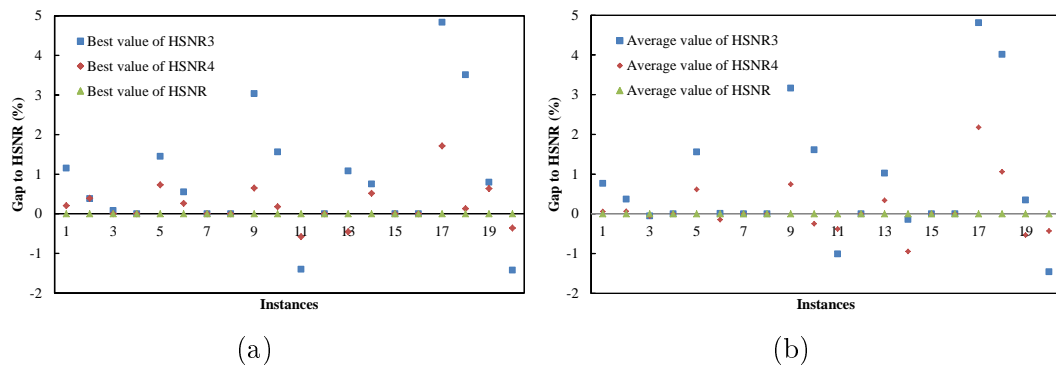


Fig. 9. Minmax mTSP: comparative results of HSNR (using EAX) with HSNR3 (using the 2-opt heuristic) and HSNR4 (using the LK algorithm).

687 as a critical intra-tour optimization tool for the mTSP.

Table 8

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

Pair	#Instances	Best				Avg.			
		#Wins	#Ties	#Losses	<i>p-value</i>	#Wins	#Ties	#Losses	<i>p-value</i>
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

## 688 6 Conclusions

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

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

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

711 For future work, there are several perspectives. First, it would be interesting  
712 to adopt the main idea of this study (i.e., neighborhood reduction, TSP tool)  
713 to design effective heuristics for other TSP variants and routing problems,  
714 including practical problems faced in real-life applications. Second, even if  
715 the minsum mTSP can be effectively solved by popular TSP algorithms, this

716 is not the case for the minmax mTSP. As such, more efforts are needed to  
717 design effective algorithms for the minmax mTSP. In this regard, it is worth  
718 investigating other search framework such as memetic algorithms integrating  
719 dedicated crossover operators. Also, few exact algorithms exist for the minmax  
720 mTSP, there is much room for making progressive in this area.

## 721 **Declaration of competing interest**

722 The authors declare that they have no known competing interests that could  
723 have appeared to influence the work reported in this paper.

## 724 **Acknowledgments**

725 We are grateful to the reviewers for their useful comments and suggestions  
726 which helped us to significantly improve the paper. We would like to thank  
727 authors of [22,32,45,47]: Prof. K. Karabulut and Prof. M. F. Tasgetiren for  
728 sharing their executable code; Prof. A. Singh, Dr. Y. Wang, and Dr. S. Yuan  
729 for providing their test problems and answering our questions.

## 730 **References**

- 731 [1] D. Applegate, W. Cook, S. Dash, A. Rohe, Solution of a min-max vehicle routing  
732 problem, *INFORMS Journal on Computing* 14 (2) (2002) 132–143.
- 733 [2] F. Arnold, M. Gendreau, K. Sörensen, Efficiently solving very large-scale routing  
734 problems, *Computers & Operations Research* 107 (2019) 32–42.
- 735 [3] F. Arnold, K. Sörensen, Knowledge-guided local search for the vehicle routing  
736 problem, *Computers & Operations Research* 105 (2019) 32–46.
- 737 [4] T. Bektas, The multiple traveling salesman problem: an overview of formulations  
738 and solution procedures, *Omega* 34 (3) (2006) 209–219.
- 739 [5] T. Bektaş, Formulations and benders decomposition algorithms for multidepot  
740 salesmen problems with load balancing, *European Journal of Operational  
741 Research* 216 (1) (2012) 83–93.
- 742 [6] J. Brandão, A memory-based iterated local search algorithm for the multi-depot  
743 open vehicle routing problem, *European Journal of Operational Research* 284 (2)  
744 (2020) 559–571.
- 745 [7] E. C. Brown, C. T. Ragsdale, A. E. Carter, A grouping genetic algorithm for  
746 the multiple traveling salesperson problem, *International Journal of Information  
747 Technology & Decision Making* 6 (02) (2007) 333–347.

- 748 [8] A. E. Carter, C. T. Ragsdale, A new approach to solving the multiple traveling  
749 salesperson problem using genetic algorithms, *European Journal of Operational*  
750 *Research* 175 (1) (2006) 246–257.
- 751 [9] A. E. Carter, C. T. Ragsdale, Quality inspection scheduling for multi-unit service  
752 enterprises, *European Journal of Operational Research* 194 (1) (2009) 114–126.
- 753 [10] O. Cheikhrouhou, I. Khoufi, A comprehensive survey on the multiple traveling  
754 salesman problem: Applications, approaches and taxonomy, *Computer Science*  
755 *Review* 40 (2021) 100369.
- 756 [11] E. D. Dolan, J. J. Moré, Benchmarking optimization software with performance  
757 profiles, *Mathematical Programming* 91 (2) (2002) 201–213.
- 758 [12] H. Ergezer, K. Leblebicioğlu, 3d path planning for multiple UAVs for maximum  
759 information collection, *Journal of Intelligent & Robotic Systems* 73 (1-4) (2014)  
760 737–762.
- 761 [13] P. M. França, M. Gendreau, G. Laporte, F. M. Müller, The m-traveling salesman  
762 problem with minmax objective, *Transportation Science* 29 (3) (1995) 267–275.
- 763 [14] W. Garn, Closed form distance formula for the balanced multiple travelling  
764 salesmen, arXiv preprint arXiv:2001.07749.
- 765 [15] W. Garn, Balanced dynamic multiple travelling salesmen: Algorithms and  
766 continuous approximations, *Computers & Operations Research* 136 (2021)  
767 105509.
- 768 [16] B. Gavish, K. Srikanth, An optimal solution method for large-scale multiple  
769 traveling salesmen problems, *Operations Research* 34 (5) (1986) 698–717.
- 770 [17] F. W. Glover, M. Laguna, *Tabu Search*, Kluwer, 1997.
- 771 [18] P. He, J. Li, H. Qin, Y. He, G. Cao, Using hybrid algorithm to reduce  
772 non-working distance in intra-and inter-field logistics simultaneously for  
773 heterogeneous harvesters, *Computers & Electronics in Agriculture* 167 (2019)  
774 105065.
- 775 [19] P. He, J. Li, X. Wang, Wheat harvest schedule model for agricultural machinery  
776 cooperatives considering fragmental farmlands, *Computers & Electronics in*  
777 *Agriculture* 145 (2018) 226–234.
- 778 [20] K. Helsgaun, An effective implementation of the Lin–Kernighan traveling  
779 salesman heuristic, *European Journal of Operational Research* 126 (1) (2000)  
780 106–130.
- 781 [21] S. Hong, M. W. Padberg, A note on the symmetric multiple traveling salesman  
782 problem with fixed charges, *Operations Research* 25 (5) (1977) 871–874.
- 783 [22] K. Karabulut, H. Öztop, L. Kandiller, M. F. Tasgetiren, Modeling and  
784 optimization of multiple traveling salesmen problems: An evolution strategy  
785 approach, *Computers & Operations Research* 129 (2021) 105192.

- 786 [23] A. Koubâa, O. Cheikhrouhou, H. Bennaceur, M.-F. Sriti, Y. Javed, A. Ammar,  
787 Move and improve: a market-based mechanism for the multiple depot multiple  
788 travelling salesmen problem, *Journal of Intelligent & Robotic Systems* 85 (2)  
789 (2017) 307–330.
- 790 [24] G. Laporte, Y. Nobert, A cutting planes algorithm for the m-salesmen problem,  
791 *Journal of the Operational Research Society* 31 (11) (1980) 1017–1023.
- 792 [25] S. Lin, B. W. Kernighan, An effective heuristic algorithm for the traveling-  
793 salesman problem, *Operations Research* 21 (2) (1973) 498–516.
- 794 [26] W. Liu, S. Li, F. Zhao, A. Zheng, An ant colony optimization algorithm for the  
795 multiple traveling salesmen problem, in: *2009 4th IEEE Conference on Industrial  
796 Electronics and Applications*, pages 1533–1537, IEEE, 2009.
- 797 [27] M. López-Ibáñez, J. Dubois-Lacoste, L. P. Cáceres, M. Birattari, T. Stützle, The  
798 irace package: Iterated racing for automatic algorithm configuration, *Operations  
799 Research Perspectives* 3 (2016) 43–58.
- 800 [28] L.-C. Lu, T.-W. Yue, Mission-oriented ant-team ACO for min–max MTSP,  
801 *Applied Soft Computing* 76 (2019) 436–444.
- 802 [29] Y. Lu, U. Benlic, Q. Wu, A population algorithm based on randomized tabu  
803 thresholding for the multi-commodity pickup-and-delivery traveling salesman  
804 problem, *Computers & Operations Research* 101 (2019) 285–297.
- 805 [30] Y. Nagata, S. Kobayashi, A powerful genetic algorithm using edge assembly  
806 crossover for the traveling salesman problem, *INFORMS Journal on Computing*  
807 25 (2) (2013) 346–363.
- 808 [31] J. Pan, D. Wang, An ant colony optimization algorithm for multiple travelling  
809 salesman problem, in: *First International Conference on Innovative Computing,  
810 Information and Control*, pages 210–213, IEEE Computer Society, 2006.
- 811 [32] V. Pandiri, A. Singh, Two metaheuristic approaches for the multiple traveling  
812 salesperson problem, *Applied Soft Computing* 26 (2015) 74–89.
- 813 [33] M. M. Paydar, I. Mahdavi, I. Sharafuddin, M. Solimanpur, Applying simulated  
814 annealing for designing cellular manufacturing systems using MDmTSP,  
815 *Computers & Industrial Engineering* 59 (4) (2010) 929–936.
- 816 [34] P. H. V. Penna, A. Subramanian, L. S. Ochi, An iterated local search heuristic  
817 for the heterogeneous fleet vehicle routing problem, *Journal of Heuristics* 19 (2)  
818 (2013) 201–232.
- 819 [35] M. Rao, A note on the multiple traveling salesmen problem, *Operations Research*  
820 28 (3-part-i) (1980) 628–632.
- 821 [36] S. Shiri, N. Huynh, Optimization of drayage operations with time-window  
822 constraints, *International Journal of Production Economics* 176 (2016) 7–20.
- 823 [37] A. Singh, A. S. Baghel, A new grouping genetic algorithm approach to the  
824 multiple traveling salesperson problem, *Soft Computing* 13 (1) (2009) 95–101.

- 825 [38] B. Soylu, A general variable neighborhood search heuristic for multiple traveling  
826 salesmen problem, *Computers & Industrial Engineering* 90 (2015) 390–401.
- 827 [39] J. A. Svestka, V. E. Huckfeldt, Computational experience with an m-salesman  
828 traveling salesman algorithm, *Management Science* 19 (7) (1973) 790–799.
- 829 [40] É. Taillard, P. Badeau, M. Gendreau, F. Guertin, J.-Y. Potvin, A tabu search  
830 heuristic for the vehicle routing problem with soft time windows, *Transportation*  
831 *Science* 31 (2) (1997) 170–186.
- 832 [41] L. Tang, J. Liu, A. Rong, Z. Yang, A multiple traveling salesman problem model  
833 for hot rolling scheduling in Shanghai Baoshan Iron & Steel Complex, *European*  
834 *Journal of Operational Research* 124 (2) (2000) 267–282.
- 835 [42] P. Toth, D. Vigo, The granular tabu search and its application to the vehicle-  
836 routing problem, *INFORMS Journal on Computing* 15 (4) (2003) 333–346.
- 837 [43] E. Uchoa, D. Pecin, A. Pessoa, M. Poggi, T. Vidal, A. Subramanian, New  
838 benchmark instances for the capacitated vehicle routing problem, *European*  
839 *Journal of Operational Research* 257 (3) (2017) 845–858.
- 840 [44] T. Vidal, T. G. Crainic, M. Gendreau, C. Prins, Heuristics for multi-  
841 attribute vehicle routing problems: A survey and synthesis, *European Journal*  
842 *of Operational Research* 231 (1) (2013) 1–21.
- 843 [45] Y. Wang, Y. Chen, Y. Lin, Memetic algorithm based on sequential variable  
844 neighborhood descent for the minmax multiple traveling salesman problem,  
845 *Computers & Industrial Engineering* 106 (2017) 105–122.
- 846 [46] Whizzkids’96, <https://www.win.tue.nl/whizzkids/1996/> (1996).
- 847 [47] S. Yuan, B. Skinner, S. Huang, D. Liu, A new crossover approach for solving  
848 the multiple travelling salesmen problem using genetic algorithms, *European*  
849 *Journal of Operational Research* 228 (1) (2013) 72–82.
- 850 [48] C. Zhan, Y. Zeng, Completion time minimization for multi-UAV-enabled data  
851 collection, *IEEE Transactions on Wireless Communications* 18 (10) (2019) 4859–  
852 4872.
- 853 [49] R. Zhang, H. Zhao, I. Moon, Range-based truck-state transition modeling  
854 method for foldable container drayage services, *Transportation Research Part*  
855 *E: Logistics and Transportation Review* 118 (2018) 225–239.

## 856 **A Appendix**

857 This appendix includes computational results of two additional experiments.  
858 The first experiment concerns a comparison between the proposed HSNR al-  
859 gorithm and the reference algorithms under a short cutoff time for the minsum  
860 mTSP and the minmax mTSP. The second experiment is about solving the  
861 minsum mTSP by running a TSP solver, given that the minsum mTSP can  
862 be transformed to the TSP [21,35]. Even if this transformation is known for

863 a long time, to our knowledge, this is the first study reporting extensive com-  
864 putational results using this approach.

### 865 *A.1 Additional computational results and comparisons*

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

- 872 - IWO [32], which reports results on 17 instances of Set I for the minsum  
873 mTSP and the minmax mTSP. The algorithm was written in C and run on  
874 a computer with a 2.83 GHz CPU and the stopping condition is a maximum  
875 of 1000 iteration steps.
- 876 - ABC(VC) [32], which reports results on 17 instances of Set I for the minsum  
877 mTSP and the minmax mTSP. The algorithm was written in C and run on  
878 the same computer under the same stopping condition as IWO.
- 879 - GVNS [38], which reports results on 12 instances of Set I for the minsum  
880 mTSP and the minmax mTSP. The algorithm was written in C++ and  
881 run on a computer with a 2.4 GHz CPU, and the stopping condition is a  
882 maximum running time of  $n$  seconds.
- 883 - MASVND [45], which is designed for the minmax mTSP only and reports  
884 results on 31 out of the 41 instances of Set I. The algorithm was written  
885 in Java and run on a computer with a 3.4 GHz CPU, and the stopping  
886 condition is a maximum running time of  $n/5$  seconds.
- 887 - ES [22], which reports results on 12 instances of Set I for the minsum mTSP  
888 and 31 out of the 41 instances of Set I for the minmax mTSP. The algorithm  
889 was written in C++ and run on a computer with a 2.66 GHz CPU, and the  
890 stopping condition is a maximum time of  $n$  and  $n/5$  seconds for the minsum  
891 mTSP and the minmax mTSP, respectively.

892 To make the comparison as meaningful as possible, we adopted as our stop-  
893 ping condition the shortest cutoff time among those used by the reference  
894 algorithms, i.e.,  $n/5$  seconds used in [45]. We used the CPU frequency to con-  
895 vert this cutoff time to our computer, leading to a cutoff time of  $(1.36 \times n)/5$   
896 seconds for our HSNR algorithm on our computer. Note that MASVND re-  
897 ports results for the minmax mTSP only, while the other reference algorithms  
898 report results for both the minsum mTSP and the minmax mTSP.

899 *A.1.1 Comparative results for the minsum mTSP*

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

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

909 *A.1.2 Comparative results for the minmax mTSP*

910 We show in Table A.2 the computational results of the compared algorithms  
911 for the the minmax mTSP with the same information as in Section 4. In this  
912 table, we included the results of IWO-Wang [45], which is a re-implementation  
913 of the IWO algorithm of [32].

914 Table A.2 indicates that HSNR performs competitively compared to the main  
915 reference algorithms, that is MASVND [45] and ES [22]. In terms of the best  
916 objective value, HSNR updates the best upper bounds (BKS) for 9 out of 33  
917 instances and reaches the BKS values for 17 instances. Given that the BKS  
918 values are compiled from the best results ever reported by all existing algo-  
919 rithms in the literature, the performance of HSNR for the minmax mTSP can  
920 be considered as remarkable. In summary, these results confirm the competi-  
921 tiveness of HSNR over the state-of-the-art algorithms for the minmax mTSP  
922 also under this short cutoff limit.

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

924 We report computational results of running the EAX heuristic [30] on the TSP  
925 instances transformed from the minsum mTSP instances. Given that most of  
926 the 77 instances involve distance matrices of real numbers, we updated the  
927 data type of EAX from integer numbers to real numbers. For this experi-  
928 ment, we ran the EAX code with its default parameter setting under the same  
929 stopping condition as HSNR (i.e.,  $(n/100) \times 4$  minutes, see Section 4). Each  
930 instance was solved 20 times by EAX with difference random seeds. Note that  
931 EAX may also terminate if the gap between the average tour length and the  
932 shortest tour length in the population becomes less than 0.0001.

933 Tables A.3 and A.4 show the comparative results of EAX and HSNR with the  
934 same information as in Section 4.3.1. The background of the top results for  
935 each instance is highlighted in dark gray; the second best results in medium

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.

Instance	BKS	ABC(VC) [32]		IWO [32]		GVNS [38]		ES [22]		HSNR (this work)		
		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	21798	21839	21797.6	21852.42	0.00
mtsp100-5	23175	23182	23222	23294	23319	23175	23383	23175	23252	23174.9	23195.31	0.00
mtsp100-10	26927	26961	27004	26961	27072	27008	27368	26927	26927	27026.4	27081.47	0.37
mtsp100-20	38245	38333	38397	38245	38357	38326	38867	38245	38257	38297.1	38882.61	0.14
mtsp150-3	37957	38066	38263	37957	38055	38430	38827	38072	38241.1	37910.7	37912.88	-0.12
mtsp150-5	38714	38979	39202	38714	38881	39171	39566	38907	39132.5	38714.4	38768.54	0.00
mtsp150-10	42203	42441	42712	42234	42462	42730	42922	42203	42428.1	42268.4	42393.11	0.15
mtsp150-20	53343	53603	53877	53475	53612	53576	53854	53343	53516.4	53608.3	54142.57	0.50
mtsp150-30	68541	68865	69046	68541	68751	68558	68804	68606	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	23282.71	23376.53	-	-	-	-	23308.62	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	-	-	-	-	-	-	-

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.

Instance	ABC(VC) [32]		IWO [32]		GVNS [38]		IWO-Wang [45]		MASVND [45]		ES [22]		HSNR (this work)		
	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	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	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	112.07	112.07	112.07	112.07	112.07	112.07	112.07	112.07	0.00
mtsp100-3	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	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	6358.49	6358.49	0.00
mtsp100-20	6358.00	6358.00	6358.00	6358.00	6358.00	6358.00	-	-	-	-	6358.00	6358.00	6358.49	6358.49	0.00
mtsp150-3	13151.00	13761.00	13168.00	13313.00	13376.00	13628.00	-	-	-	-	13151.00	13272.20	13174.30	13352.37	0.18
mtsp150-5	8466.00	8795.00	8467.00	8567.00	8467.00	8601.00	-	-	-	-	8466.00	8572.50	8479.60	8602.15	0.16
mtsp150-10	5557.00	5834.00	5594.00	5654.00	5674.00	5736.00	-	-	-	-	5557.00	5609.60	5616.71	5668.59	1.07
mtsp150-20	5246.00	5281.00	5246.00	5246.00	5246.00	5246.00	-	-	-	-	5246.00	5246.00	5246.49	5246.49	0.00
mtsp150-30	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	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	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	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	-	-	1554.64	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	-	-	1554.64	1554.64	1554.64	1554.64	1554.64	1554.64	1554.64	1554.64	0.00
kroA200-3	10768.10	-	-	-	-	-	10814.18	10947.79	10831.66	11045.91	10768.10	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	-	-	-	-	-	16200.21	16340.3	16206.25	16477.89	16273.80	16753.24	16094.90	16482.00	0.04
lin318-5	11524.29	-	-	-	-	-	11730.03	11908.18	11752.41	11896.71	11604.20	11876.42	11458.20	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	22443.22	22781.61	22601.70	23095.02	21430.10	21928.16	-3.96
pcb1173-5	14099.50	-	-	-	-	-	16057.19	16274.64	14557.3	14861.4	14099.50	14346.76	13402.30	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	3	0	0	0	3	1	0	5	5	8	12	-
p-value	7.18E-01	-	-	-	-	-	-	-	-	-	-	-	-	-	-

936 gray. The results of Tables A.3 and A.4 clearly indicate that EAX signifi-  
937 cantly dominates HSNR in terms of the best and average results for both sets  
938 of instances. Only on three large instances of Set II, HSNR reported better  
939 results. Given that HSNR performs better than the existing minsum mTSP al-  
940 gorithms in the literature, we can safely say that EAX dominates all existing  
941 minsum mTSP algorithms. Finally, even if we did not show detailed run-time  
942 information, we mention that EAX converges much faster than the existing  
943 algorithms (by at least one order of magnitude). EAX requires no more than  
944 30 seconds for Set I and no more than 400 seconds for Set II.

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

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 [30]			HSNR (this work)		
	<i>Best</i>	<i>Avg.</i>	$\sigma$	<i>Best</i>	<i>Avg.</i>	$\sigma$
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#	<b>7</b>	<b>23</b>	-	0	0	-
<i>p-value</i>	1.95E-02	3.25E-05	-	-	-	-

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 [30]			HSNR (this work)		
	<i>Best</i>	<i>Avg.</i>	$\sigma$	<i>Best</i>	<i>Avg.</i>	$\sigma$
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#	<b>15</b>	<b>33</b>	-	3	1	-
<i>p-value</i>	6.50E-03	5.39E-07	-	-	-	-