Computers & Industrial Engineering 106 (2017) 105–122

Contents lists available at ScienceDirect

Computers & Industrial Engineering

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

Memetic algorithm based on sequential variable neighborhood descent for the minmax multiple traveling salesman problem

Yongzhen Wang *, Yan Chen, Yan Lin

Transportation Management College, Dalian Maritime University, Dalian 116026, China

ARTICLE INFO

Article history:
Received 11 May 2016
Received in revised form 9 December 2016
Accepted 12 December 2016
Available online 18 December 2016

Keywords:
Multiple traveling salesman problem
The minmax objective
Memetic algorithm
Sequential variable neighborhood descent
The total distance

ABSTRACT

In this paper, we consider the multiple traveling salesman problem (MTSP) with the minmax objective, which includes more than one salesman to serve a set of cities while minimizing the maximum distance traveled by any salesman. For this problem, we have proposed a novel memetic algorithm, which integrates with a sequential variable neighborhood descent that is a powerful local search procedure to exhaustively search the areas near the high-quality solutions. However, there are some inefficient neighborhoods in the existing sequential variable neighborhood descent for the minmax MTSP, which could restrict the search performance. Therefore, we have redefined a new neighborhood sequence where only the neighborhoods that move cities from one tour to another unidirectionally are considered. Computational experiments on a wide range of benchmark problems within an acceptable time limit show that compared with six existing algorithms, the proposed algorithm is better than the other algorithms in terms of three aspects, including the precision, the robustness and the convergence speed. Meanwhile, we have also investigated the total distance traveled by all the salesmen when optimizing the minmax objective, and the results show that in comparison with the six existing algorithms, the proposed algorithm has a better or at least competitive capacity to maintain the total distance as short as possible. Furthermore, two kinds of statistical tests are utilized to examine the significance of the presented results, indicating the superiority of the proposed algorithm over the other algorithms on the minmax objective.

1. Introduction

The multiple traveling salesman problem (MTSP) is a variation of the famous traveling salesman problem (TSP), where it aims to determine a group of Hamilton circuits without sub-tours for m ($m > 1$) salesmen to serve a set of n ($n > m$) cities so that each city is visited by exactly one salesman. The MTSP is strongly NP-hard as the TSP is the special case of it. Compared with the TSP, a broader range of real-world problems can be modeled as the MTSP because it is capable of handling more than one salesman, for example, print press scheduling (Gorenstein, 1970), workload balancing (Okonjo-Adigwe, 1988), hot rolling production planning (Tang, Liu, Rong, & Yang, 2000), vehicle routing (Gutierrez, Dieulle, Labadie, & Velasco, 2016), UAVs area covering (Ann, Kim, & Ahn, 2015) and off-grade production minimizing (Király, Christidou, Chován, Karlopoulos, & Abonyi, 2016).

Although a majority of literatures about the MTSP are to minimize the total distance traveled by all the salesmen (minsum), minimizing the maximum distance traveled by any salesman (minmax) is of more practical significance since it is related with balancing the workload among salesmen (Soylu, 2015; Venkatesh & Singh, 2015). Moreover, the minmax objective is also applied to minimizing the makespan of visiting all the cities if the travel time is used instead of the distance. Thus, this objective is especially useful in the circumstances where the cost (distance or time) caused by any salesman is not allowed to exceed a given limit (Bertazzi, Golden, & Wang, 2015; Na, 2007). Nevertheless, there are only a few works in the literatures designed exclusively for the minmax MTSP (França, Gendreau, Laporte, & Müller, 1995; Frederickson, Hecht, & Kim, 1976; Golden, Laporte, & Taillard, 1997; Kivelevitch, Sharma, Ernest, Kumar, & Cohen, 2014; Na, 2007; Somhom, Modares, & Enkawa, 1999; Vallivaara, 2008). Generally speaking, a more common practice is to present a general algorithm for both the objectives, which often lacks a specific mechanism to reduce the maximum distance thoroughly (Brown, Ragsdale, & Carter, 2007; Carter & Ragsdale, 2006; Singh & Baghel, 2009; Venkatesh & Singh, 2015; Yuan, Skinner, Huang, &

* Corresponding author at: Dalian Maritime University, Management Bldg, Room 205, No. 1 Linghai Road, Dalian, Liaoning Province, China.
E-mail address: kuadmu@163.com (Y. Wang).

http://dx.doi.org/10.1016/j.cie.2016.12.017
0360-8352/

Liu, 2013). Recently, *Bertazzi* et al. have indicated that the maximum distance in the minsum MTSP is at most m times that in the minmax one (Bertazzi et al., 2015), which motivates the need to design exclusive algorithms applied to the minmax MTSP. Meanwhile, from the worst-case point of view, they have also shown that the total distance in the minmax MTSP is at most m times that in the minsum one, which implies that both the objectives are conflicting ones.

Compared with the minsum MTSP, the minmax one is much more difficult to resolve. The medium-scale instances of the minsum MTSP can be resolved optimally within an appropriate time limit, but only very small-scale ones of the minmax MTSP can be resolved optimally within the same time limit (Sarin, Sherali, Judd, & Tsai, 2014; Soylu, 2015). Therefore, a number of different heuristic algorithms applied to the minmax MTSP have been developed in the literatures. *Frederickson* et al. proposed some approximation algorithms for that, which included k-near insert, k-near neighbor and k-split tour (Frederickson et al., 1976). *Na* presented a two-stage heuristic algorithm for the minmax MTSP with no depot, where m tours were constructed in the first stage and were improved in the second stage (Na, 2007). One of the earliest applications of the neural network to the minmax MTSP stemmed from Somhom et al. (1999), which introduced a competition method to decide whether a city should be included into a tour. Then Masutti et al. developed a neuro-immune algorithm that used concepts from the self-organizing maps and the artificial immune systems (Masutti & de Castro, 2009). The recent study by *Soylu* proposed a general variable neighborhood search algorithm (Soylu, 2015), which is one of the state-of-the-art algorithms for the minmax MTSP.

Apart from the classical heuristic algorithm, the evolutionary algorithm (EA) is another resolution for the minmax MTSP. They find the optimal or near-optimal solutions via the evolutionary rules that are different from the heuristic search rules. *Carter and Ragsdale* presented a genetic algorithm for the minmax MTSP that used a two-part chromosome encoding and the related operators (Carter & Ragsdale, 2006). Based on that, *Yuan* et al. put forward a new crossover operator named two-part chromosome crossover and outperformed the former (Yuan et al., 2013). In reality though, the minmax MTSP is the grouping problem (Kashan, Akbari, & Ostadi, 2015), which searches for an optimal allocation of cities into different tours subject to some constraints. Therefore, *Brown* et al. and *Singh and Baghel* successively proposed two different grouping genetic algorithms so as to resolve that (Brown et al., 2007; Singh & Baghel, 2009), where the former was by deriving the chromosome encoding and the operators from Falkenauer (1998), but the latter proposed a new chromosome encoding and the corresponding operators. More recently, swarm intelligence algorithms such as the ant colony, the artificial bee colony and the invasive weed optimization, were also applied to the minmax MTSP successfully (Liu, Li, Zhao, & Zheng, 2009; Venkatesh & Singh, 2015).

Notwithstanding it is an acknowledged effective way to combine the EA with a local search procedure (LSP) to improve the search performance (Moscato, 1989), existing LSPs for the minmax MTSP presented by the heuristic algorithms mentioned above are too complicated and time-consuming to be directly used due to their difficulty and complexity. In fact, only four previous works for the minmax MTSP in the literatures attempted to combine an EA with a very simple LSP in order to ameliorate the obtained tours (Liu et al., 2009; Venkatesh & Singh, 2015). Unfortunately, this simple LSP is not specifically responsible for minimizing the maximum distance, but only for improving each tour independently. As a result, the experimental results showed that this technique could only get a bit marginal improvement.

Despite of the important academic and engineering values of the minmax MTSP, the research on it is relatively limited and most of the works are related to its applications. To the best of our knowledge, the largest number of cities on the minmax MTSP by far is 575, while the number of that studied most is still under 200 (Brown et al., 2007; Carter & Ragsdale, 2006; França et al., 1995; Frederickson et al., 1976; Golden et al., 1997; Kivelevitch et al., 2014; Liu et al., 2009; Masutti & de Castro, 2009; Na, 2007; Sarin et al., 2014; Singh & Baghel, 2009; Somhom et al., 1999; Soylu, 2015; Vallivaara, 2008; Venkatesh & Singh, 2015; Yuan et al., 2013). Additionally, according to the results of Sarin et al. (2014), the optimal solution of the minmax MTSP can only be found on the instances with $n \leqslant 10$ within 3600 s. But surprisingly, due to the enormous algorithmic progress, the largest number of cities on the symmetric TSP that can be resolved optimally now is up to 85,900 (Reinhelt, 2014).

In this paper, a memetic algorithm based on sequential variable neighborhood descent (MASVND) is developed for the minmax MTSP. Here, the memetic algorithm (MA) refers to a combination of an EA and a LSP. Our motivations in dealing with this problem are threefold: the first is to put forward a novel MA, which integrates with the sequential variable neighborhood descent (seq_VND) that acts as the mechanism to exhaustively reduce the maximum distance; the second is to employ the proposed algorithm to resolve the large-scale benchmark problems; and the third is to investigate the total distance when optimizing the minmax objective. The performance of the MASVND is compared with four state-of-the-art algorithms and two representative ones, and experimental results clearly demonstrate the superiority of the MASVND over any other compared algorithm. The main contributions of this paper are:

- It proposes a novel MA to resolve the minmax MTSP, and makes the comparisons with the six existing algorithms from three aspects, including the precision, the robustness and the convergence speed.
- It defines a new neighborhood sequence for the seq_VND on the minmax MTSP.
- It increases the largest number of cities on the minmax MTSP from 575 to 1173.
- It investigates the total distance obtained by the various algorithms when optimizing the minmax objective.

The rest of this paper is organized as follows: In Section 2, a brief introduction to the MA and the seq_VND is given, and a detailed analysis of the existing neighborhoods on the minmax MTSP is also presented. Section 3 elaborates the MASVND, and the experimental study is shown in Section 4. Section 5 describes the computational complexity of the MASVND. Finally, Section 6 provides some concluding remarks and directions for the further work.

2. Methodology

2.1. Memetic algorithm

Evolutionary algorithms are a variety of global search methods derived from Darwin theory. Traditional EAs often produce moderate solutions, but meaningful improvements can be obtained by means of hybridization with other techniques (Moscato, 1989). The combination of an EA and a LSP is called the memetic algorithm (Merz, 2001; Moscato & Cotta, 2003) which is inspired by *Dawkin*'s notion of a meme (Dawkins, 2016). The meme is a cultural gene, which can be modified before being included into the next generation. Due to the trade-off between the diversification

of the EA and the intensification of the LSP, the MA has proven to be both more effective and efficient than the conventional EAs in some optimization domains (Lai & Hao, 2016; Sabar, Turky, & Song, 2016; Turky, Sabar, & Song, 2016). As a result, it has gained the wide attention and acceptance in the TSP (Cai, Cheng, Fan, Goodman, & Wang, 2015; Serna & Uran, 2015; Wang, Li, Gao, & Pan, 2011).

However, as far as we know, only four works for the minmax MTSP in the literatures have combined an EA with a 2-opt algorithm (Liu et al., 2009; Venkatesh & Singh, 2015) that is a common LSP for the TSP (Helsgaun, 2009; Karapetyan & Gutin, 2010). Since the minmax MTSP belongs to the grouping problem (Kashan et al., 2015), its optimization process consists of two parts, i.e., the optimal allocation of cities into different tours and the optimal ordering of cities within tours, which leads to the difficulty and complexity to design the corresponding LSP. Therefore, it is too complicated and time-consuming for the EA to directly integrate with the existing LSPs that are for the minmax MTSP. In short, integrating an efficient LSP into the EA to resolve the minmax MTSP still remains challenging.

The pseudo-code of the standard MA is given in Algorithm 1, which is rather similar to the traditional genetic algorithm in the literatures (Holland, 1975). During the first step, the initial population is randomly generated and optimized by a *local_search* procedure. Afterwards all the individuals are evaluated through an *evaluate_ fitness* function. Within the while loop, a *select_individual* operator is employed to determine a subset of individuals for evolutionary operators. The evolutionary operators consist of a *recombine* operator and a *mutate* operator, where the former is responsible for exchanging memes between individuals, while the latter is used for perturbing an individual slightly. Then a *select_new_population* operator acts on the population so as to maintain its size. Finally, the convergence of the population is checked by some criteria such as lack of the diversity of the population. If the criteria are met, a *restart* operator is used and the MA should start from scratch until the stopping criteria are met.

Algorithm 1. The pseudo-code of the standard MA

```
1   iter := 0;
2   P(iter) := initial_random_population();
3   P(iter) := local_search(P(iter));
4   evaluate_fitness(P(iter));
5   while termination condition not satisfied do
6   |   P'(iter) := select_individual(P(iter));
7   |   P'(iter) := recombine(P'(iter));
8   |   P'(iter) := mutate(P'(iter));
9   |   P'(iter) := local_search(P'(iter));
10  |   evaluate_fitness(P'(iter));
11  |   P(iter+1) := select_new_population(P(iter), P'(iter));
12  |   if convergence of P(iter+1) meets convergence criteria then
13  |   |   restart(P(iter+1));
14  |   end
15  |   iter := iter+1;
16  end
```

2.2. Sequential variable neighborhood descent

Sequential variable neighborhood descent is a definitive version of the variable neighborhood search, where the neighborhoods are ordered in a sequence and visited one by one until a local minimum is reached (Hansen, Mladenović, & Pérez, 2010; Mladenović, Urošević, & Ilić, 2012; Sifaleras, Konstantaras, & Mladenović, 2015). We denote N_l ($l = 1, 2, \ldots, l_{max}$) a finite set of neighborhoods, where the l_{max} is considered as the maximum number of different neighborhoods in the seq_VND. Let S be a feasible solution, and $N_l(S)$ be the set of solutions in the l-th neighborhood of S. Each neighborhood N_l actually corresponds to a kind of operator, where each $S' \in N_l(S)$ is several units away from S, i.e., S' is obtained by relocating some units within S. Usually, the seq_VND returns to the first neighborhood in the sequence immediately when an improvement is gained. And there are two kinds of improvement strategies implemented for the seq_VND, where one is called the first improvement (a move is made immediately when a better solution in one neighborhood is found), and another is called the best improvement (a move is made until the best solution in all the neighborhoods is found) (Mladenović et al., 2012). In this paper, we consider the former case, and the pseudo-code of the standard seq_VND is given in Algorithm 2.

Algorithm 2. The pseudo-code of the standard seq_VND

```
1   l := 1;
2   while l ≤ l_max do
3   |   for i := 1 to |N_l| do
4   |   |   S' := neighborhood_move(S, l, i);
5   |   |   if f(S') > f(S) then
6   |   |   |   S := S';
7   |   |   |   l := 1;
8   |   |   |   go to while loop;
9   |   |   end
10  |   end
11  |   l := l+1;
12  end
```

By means of constantly readjusting the neighborhoods, the seq_VND is capable of exhaustively searching the areas near the selected solution. Apparently, the neighborhoods have a direct effect on the final solution so that those inefficient ones would contribute little or none to improving the search performance. And in theory, a greater number of different neighborhoods, i.e., a larger l_{max} would bring about a growth of computation time for sure (Sifaleras et al., 2015). Thus, the key to the seq_VND is to determine the suitable neighborhoods.

Because of the capacity to find the high-quality solutions, the seq_VND has been applied to a lot of combinatorial optimization domains successfully, such as some route planning problems and scheduling problems (de Armas & Melián-Batista, 2015; Fonseca & Santos, 2014; Polat, Kalayci, Kulak, & Günther, 2015). In reality though, it is rarely applied to the MTSP. As can be seen from the literatures, there are two sets of neighborhoods on the MTSP presented successively by Zhao, Chen, and Li (2012) and Soylu (2015), but only the latter has been applied to the minmax MTSP.

2.3. Limitation of the existing neighborhoods on the minmax MTSP

Recently, five kinds of neighborhoods on the minmax MTSP were presented in a general variable neighborhood search algorithm (Soylu, 2015), as shown in Fig. 1. These neighborhoods stem from those which are originally developed for the TSP and the route planning problems (Cordeau, Laporte, Vigo, Savelsbergh, & Vigo, 2007; Groër, Golden, & Wasil, 2010; Kindervater & Savelsbergh, 1997). It is easy to identify from Fig. 1 that these neighborhoods mainly correspond to two types of operators, i.e., the unidirectional ones and the bidirectional ones, where the unidirectional ones move some units (cities) from $tour_1$ to $tour_2$ unidirectionally (Fig. 1(a), (c), (d)), and the bidirectional ones move several cities between two tours simultaneously (Fig. 1 (b) and (e)). When optimizing the minmax MTSP, compared with the unidirectional operators, the bidirectional ones are much more burdensome so as to restrict the search performance, and the corresponding analysis is as follows.

We take the Fig. 1(b) as an example. Let the $tour_1$ (the blue one) be the tour that has the maximum distance, i.e., the longest tour, and the $tour_2$ (the red one) be another tour. Then we assume that

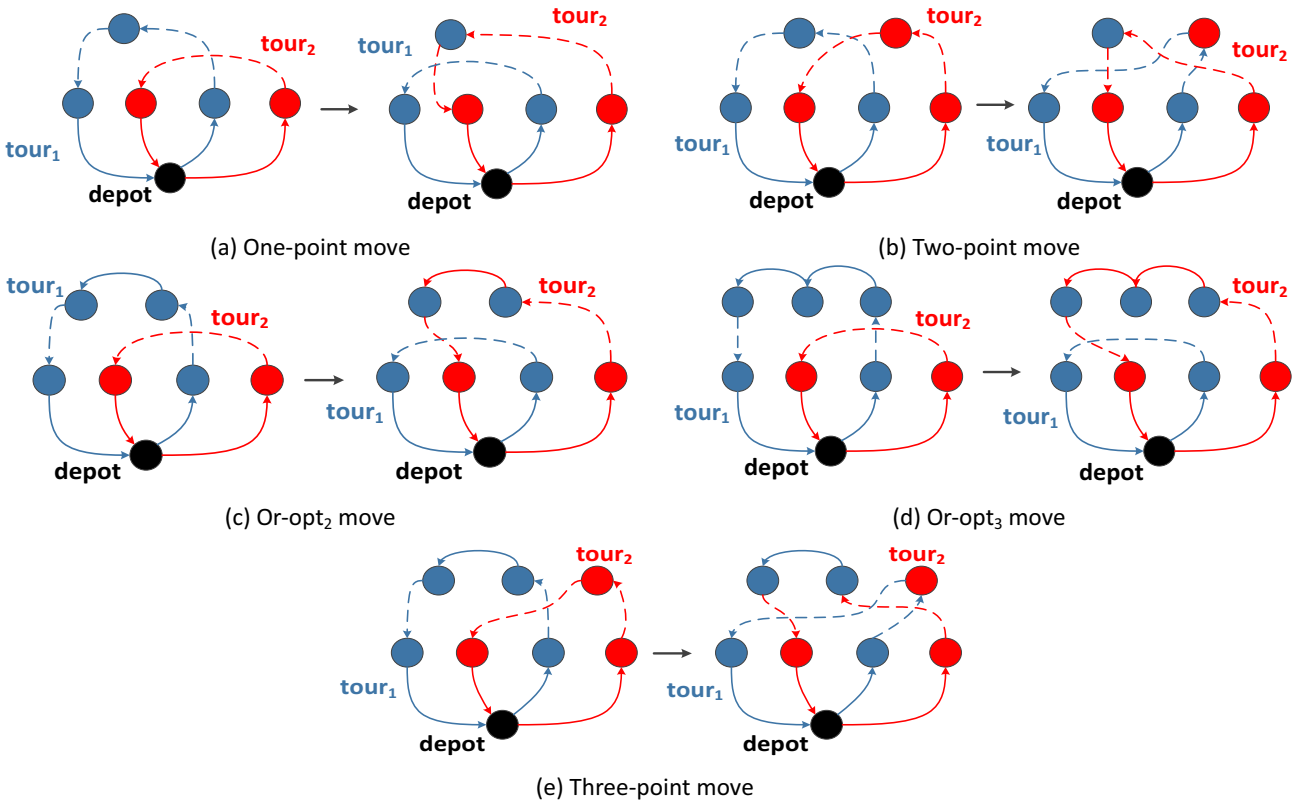


Fig. 1. Five existing neighborhoods on the minmax MTSP (Soylu, 2015).

this operator can be divided into two steps, where the first step is responsible for moving a city from $tour_1$ to $tour_2$ ($step_1$), and the second step is used for moving a different city from $tour_2$ to $tour_1$ ($step_2$). After the $step_1$ is completed, we discuss about the following three cases:

- $tour_1$ is still the longest tour. In this case, it is pointless to continue the $step_2$, which is helpless for optimizing the minmax objective.
- $tour_2$ becomes the longest tour. Let $distance_1$ be the original distance of $tour_1$, $distance_2$ be the final distance of $tour_1$, and $distance_3$ be the final distance of $tour_2$. In this case, it is meaningful to continue the $step_2$ only when that makes $distance_2 < distance_1$ and $distance_3 < distance_1$, otherwise, it is pointless to do even the $step_1$.
- one of the other m-2 tours becomes the longest tour. Similar to the first case, it is pointless to continue the $step_2$ when optimizing the minmax objective.

Through the analysis above, we can find that it is often not efficient for two tours to move cities to each other simultaneously when optimizing the minmax MTSP, because it cannot avoid those pointless steps that would consume much running time. Similar analysis can also be made to the Fig. 1(e). Besides, the effect resulted from using the bidirectional operators can nearly always be equivalent in principle to that brought by executing the unidirectional ones two runs. In fact, the computational results from Soylu (2015) also show that it is not so effective to use any of the two bidirectional operators within a time limit. Therefore, it is more wise and efficient for us to choose the unidirectional operators from the existing ones to resolve the minmax MTSP.

3. Memetic algorithm based on sequential variable neighborhood descent

The proposed algorithm is a combination of an EA (to generate new promising individuals) and a LSP (to locally improve the generated individuals), which provides the search with an ideal trade-off between the exploration and the exploitation. The pseudo-code of the MASVND is given in Algorithm 3.

Algorithm 3. The pseudo-code of the MASVND

```
    Input: parameters for the MASVND and a minmax MTSP benchmark problem
    Output: the best found individual S_temp
1   iter := 0;
2   P(iter) := initial_random_population();
3   P(iter) := 2_opt(P(iter));
4   evaluate_fitness(P(iter));
5   while termination condition not satisfied do
6   |   for i := 1 to |P(iter)| do
7   |   |   r := random();
8   |   |   if r < P_c then
9   |   |   |   P'(iter) := select_individual_recombine(P(iter));
10  |   |   |   P'(iter) := recombine(P'(iter));
11  |   |   end
12  |   |   else
13  |   |   |   P'(iter) := select_individual_mutate(P(iter));
14  |   |   |   P'(iter) := mutate(P'(iter));
15  |   |   end
16  |   |   evaluate_fitness(P'(iter));
17  |   |   subP(iter) := add_individual(subP(iter), P'(iter));
18  |   end
19  |   S_temp := get_individual_fitness_max(subP(iter));
20  |   S_temp := seq_VND(S_temp);
21  |   P(iter+1) := select_new_population(P(iter), subP(iter));
22  |   iter := iter+1;
23  end
24  S_best := get_individual_fitness_max(P(iter));
25  return S_best;
```

In the proposed algorithm, a *recombine* operator alternates with a *mutate* operator in use to generate new individuals (Singh & Gupta, 2007), i.e., each offspring individual is generated via a *recombine* operator or a *mutate* operator but not both. And the *recombine* operator is employed with probability P_c, otherwise the *mutate* operator is used. Then the seq_VND procedure is used to further improve the best offspring individual at each generation, which searches all its neighborhoods according to a sequence and returns the best individual among all the searched neighbors. The main features of the MASVND for the minmax MTSP are described below.

3.1. Population initialization and the fitness function

We have used a set of m tours to represent the individual for the minmax MTSP, as used in Venkatesh and Singh (2015), Singh and Baghel (2009), and there is no ordering among them. Each individual in the initial population is randomly generated, and then in order to speed up the convergence, all these individuals are optimized by a 2-opt algorithm which is a common LSP to ameliorate each tour within each individual independently (Helsgaun, 2009; Karapetyan & Gutin, 2010). This algorithm iteratively deletes two non-adjacent edges from a given tour and adds two new edges while maintaining the tour structure not impaired, as shown in Fig. 2. And our fitness function $f(\cdot)$ is the inverse of the maximum distance, which needs to be maximized.

3.2. Select individuals for the recombine operator and the mutate operator

With respect to the *recombine* operator, it is common to select the participating individuals with probability proportional to their fitness (Al Jadaan, Rajamani, & Rao, 2008). However, since we have employed the seq_VND to improve the best individual at each generation, it is very likely to happen that these dominant individuals rule the rest. In other words, it is easy to produce the premature convergence (Filipović, 2012). To avoid this, we have used the binary tournament selection method, as used in Singh and Baghel (2009), in which two individuals are chosen at random and the better one is selected with probability P_{better} (the worse one with probability 1-P_{better}). Because of adopting the fixed probability, the poor individuals always have the chance (though low) to participate in the *recombine* operator, which is conducive to maintain the diversification of the population.

As for the *mutate* operator, it is usually not a wise choice to employ this operator to a poor individual, which hardly gains a meaningful improvement (Al Jadaan et al., 2008; Filipović, 2012). For the sake of simplicity, here we inherit the trinary tournament method from Singh and Baghel (2009), in which one individual with the maximum fitness is selected from three candidates. It is obvious that this method is parameter-free, and it can also include some randomness so as to prevent from the premature convergence.

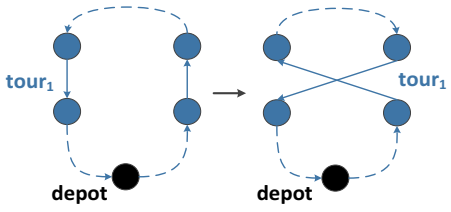


Fig. 2. The sketch of the 2-opt algorithm.

3.3. Recombine operator: exchanging memes between individuals sequentially

The *recombine* operator is one of the two ways to generate new individuals where we try to exchange memes between two parent individuals S_1 and S_2. It works in two steps as follows.

The first step begins by sorting m tours by their distances in an ascending way within S_1 and S_2, and then it constructs the offspring individual S' in an iterative manner. During each iteration, it selects one of the two parent individuals at random and copies the i th-tour (i = 1 to m) to S'. Next it sets $i = i + 1$ and deletes all the cities belonging to this tour from both S_1 and S_2. The process is repeated m times, and obviously at the end of this step some cities still remain unassigned and then it includes them into a list *unassigned*.

Through sorting m tours within S_1 and S_2, this step actually gives the inherited priority to those promising tours. That is, when solving the minmax MTSP, the distance of a tour is inversely proportional to the probability of a tour belonging to the global optimum, where the shorter the distance of a tour, the more likely this tour belongs to the global optimum (Singh & Baghel, 2009; Singh & Gupta, 2007).

The second step reassigns the cities in the *unassigned* one by one to S'. Each time it assigns a city to a particular tour whose distance increases the least by this reassignment while ensuring this tour is not the longest one. A city is assigned to the longest tour only when its distance increases the least and assigning that city to any other tour would make that tour the longest one. At the end of the *recombine* operator, if there is a tour t having only the depot, then we choose a tour t' at random that has more than two cities and move one city from t' to t randomly. We name this operation a *repair* process. The pseudo-code of the *recombine* operator is given in Algorithm 4.

Algorithm 4. The pseudo-code of the *recombine* operator

```
     Input: two parent individuals S1 and S2
     Output: the offspring individual S'
1    S1:= sort_tour_by_distance(S1);
2    S1:= sort_tour_by_distance(S2);
3    for i := 1 to m do
4    |     r := random();
5    |     if r < 0.5 then
6    |     |     t := get_tour(S1, i);
7    |     end
8    |     else
9    |     |     t := get_tour(S2, i);
10   |     end
11   |     S' := add_tour(S', t);
12   |     S1 := delet_city_in_tour(S1, t);
13   |     S2 := delet_city_in_tour(S2, t);
14   end
15   unassigned := save_remaining_city(S1, S2);
16   for i := 1 to |unassigned| do
17   |     c := get_city(unassigned, i);
18   |     S' := reassign_city(S', c);
19   end
20   S' := repair(S');
21   return S';
```

3.4. Mutate operator: perturbing an individual by varied amounts

The *mutate* operator is another way to generate new individuals where we try to make a copy of the selected individual S. This is done by copying each tour within S one after another to the offspring individual S'. However, each city from a tour t within S is copied to the corresponding tour t' within S' with varied probability P_{copy}, otherwise the city is included into the list *unassigned*. The varied probability method here is derived from the spatial

distribution concept in the invasive weed optimization (Venkatesh & Singh, 2015; Zhou, Luo, Chen, He, & Wu, 2015), which is used to perturb the copied individual by varied amounts. It is based on the observation that generally, individuals should be perturbed significantly to maintain their exploration at the early stage of the evolution, while they only need a small amount of perturbation when they are close to the global optimum at the late stage of the evolution. The formula of determining the value of P_{copy} in the *mutate* operator is

$$P_{copy} = \frac{(P_{maxcopy} - P_{min\ copy})}{iter_{max}} \times iter + P_{min\ copy} \quad (1)$$

Here, *iter* is the running time or the number of iterations, $iter_{max}$ is the maximum running time or the maximum number of iterations, $P_{maxcopy}$ and $P_{mincopy}$ respectively represent the maximum value and the minimum value of P_{copy}. When all the cities within S are traversed, those unassigned cities will be reassigned to S' by following the same process as used in the second step of the *recombine* operator. Meanwhile, the *mutate* operator will also carry out a *repair* process at the ending stage. The pseudo-code of the *mutate* operator is given in Algorithm 5.

Algorithm 5. The pseudo-code of the *mutate* operator

```
    Input: the parent individuals S
    Output: the offspring individual S'
1   P_copy := calculate_P_copy (P_maxcopy, P_mincopy, iter);
2   for i := 1 to m do
3   |   t := get_tour(S, i);
4   |   t' := get_tour(S', i);
5   |   for i := 1 to |t| do
6   |   |   r := random();
7   |   |   c := get_city(t, i);
8   |   |   if r < P_copy then
9   |   |   |   t' := add_city(t', c);
10  |   |   end
11  |   |   else
12  |   |   |   unassigned := add_city(unassigned, c);
13  |   |   end
14  |   end
15  end
16  for i := 1 to |unassigned| do
17  |   c := get_city(unassigned, i);
18  |   S' := reassign_city(S', c);
19  end
20  S' := repair(S');
21  return S';
```

3.5. Sequential variable neighborhood descent: locally improving an individual

When all the offspring individuals have been produced at each generation, the seq_VND procedure is used to locally improve the best one among them. Four kinds of neighborhoods on the minmax MTSP are studied in this paper, where three of them are derived from those as used in Soylu (2015), and all of them belongs to the unidirectional operators that move cities form one tour to another unidirectionally, as analyzed in Section 2.3. We have redefined a new neighborhood sequence according to the number of cities moving one time, which consists of One-point move (Fig. 1(a)), Or-opt$_2$ move (Fig. 1(c)), Or-opt$_3$ move (Fig. 1(d)) and Or-opt$_4$ move (Fig. 3, i.e., $l_{max} = 4$). Note that the Or-opt$_4$ move is not included in Soylu (2015), which is an extension of the Or-opt$_3$ move, and we have added this to the neighborhood sequence in order to investigate the impact of the l_{max} on the proposed algorithm (see Section 4.1).

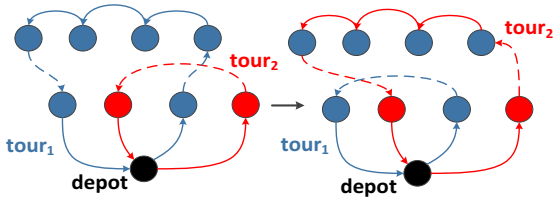


Fig. 3. The sketch of the Or-opt$_4$ move.

Clearly, each operator actually moves a string of cities into a new position in another tour, where the string is defined as l ($l = 1, 2, 3, 4$) adjacent cities. The selected string belongs to the longest tour. Whenever a string movement is realized, a new individual S' is obtained. Each operator is repeated for all the strings of the selected tour and for all the available positions of the other tours until an improvement in the neighborhood is found the first time. In this case, the original individual S_{temp} is replaced by the new individual S', and then the procedure continues to search the next neighborhood until no unsearched neighborhood exists. Note that the procedure does not return to the first neighborhood when an improvement is obtained. When all the operators are performed, the 2-opt algorithm is employed to further improve the resulting tours.

Through constantly readjusting the neighborhoods, the seq_VND is capable of thoroughly searching the areas near the best individual, which acts as the specific mechanism to exhaustively reduce the maximum distance. The pseudo-code of the seq_VND is given in Algorithm 6.

Algorithm 6. The pseudo-code of the seq_VND

```
    Input: the best offspring individual S_temp
    Output: the best individual among all the searched neighbors S_temp
1   l := 1;
2   while l ≤ l_max do
3   |   t := get_longest_tour(S_temp);
4   |   for i := 1 to |m−1| do
5   |   |   t' := get_tour_except_longest(S_temp, i, t);
6   |   |   for j := 1 to |t| do
7   |   |   |   for k := 1 to |t'| do
8   |   |   |   |   S' := neighborhood_ move(S_temp, l, j, k);
9   |   |   |   |   if f(S') > f(S_temp) then
10  |   |   |   |   |   S_temp := S';
11  |   |   |   |   |   l := l+1;
12  |   |   |   |   |   go to while loop;
13  |   |   |   |   end
14  |   |   |   end
15  |   |   end
16  |   end
17  |   l := l+1;
18  end
19  S_temp:= 2_opt(S_temp);
20  return S_temp;
```

4. Experimental results and comparisons

The MASVND was implemented in Java and run on a 3.40 GHz workstation with 4 GB RAM. For analyzing its performance for solving the minmax MTSP, we have tested the proposed algorithm on 8 sets of instances, which are originally the instances of the TSP and are taken from the TSPLIB (Reinhelt, 2014). Table 1 shows the details of these instances, where there are 51–1173 cities and 3–20 salesmen. Among them there are a total of 31 benchmark problems (BPs), including 11 small-scale ones (SBPs) as used in Venkatesh and Singh (2015), Soylu (2015), Carter and Ragsdale (2006), Yuan

Table 1
The definitions of the benchmark problems.

No	Instances	n	m	Scale	The number of benchmark problems
1	eil51	51	3, 5, 10	Small-scale	11
2	rand100	100	3, 5, 10, 20		
3	ch150	150	3, 5, 10, 20		
4	kroA200	200	3, 5, 10, 20	Large-scale	20
5	lin318	318			
6	att532	532			
7	rat783	783			
8	pcb1173	1173			

et al. (2013), Brown et al. (2007), Singh and Baghel (2009), and 20 large-scale ones (LBPs). For convenience, let $N = \{51, 100, 150, 200, 318, 532, 783, 1173\}$ and $M = \{3, 5, 10, 20\}$. Like some previous works for the minmax MTSP, we have considered the single depot case, i.e., all the m salesmen have to start and end at a given single depot and visit at least one city except for the depot, $\forall m \in M$. On all the BPs, the MASVND sets the size of the population P_{size} to 100, and the maximum running time $iter_{max}$ to $n/5$ s so as to analyze the convergence results, $\forall n \in N$.

The MASVND contains four key parameters: P_c, P_{better}, $P_{mincopy}$ and $P_{maxcopy}$. In order to investigate the influence of these parameters on the MASVND, we have applied the Taguchi method for design of experiment (DOE) Ghani, Choudhury, & Hassan, 2004; Wang, Choi, & Lu, 2015 to the BPs. The analytical results indicate that compared with the P_{better}, the MASVND is extremely insensitive to the other three parameters, and we empirically suggest the following values for all these parameters: $P_c = 0.3$, $P_{better} = 0.8$, $P_{mincopy} = 0.15$ and $P_{maxcopy} = 0.9$. The complete content about the DOE can be found in Appendix A.

In order to benchmark, we have compared the proposed algorithm with six existing algorithms available in the literatures. The artificial bee colony and the invasive weed optimization presented by Venkatesh and Singh (2015) will be referred to as ABC(FC), ABC(VC) and IWO respectively, where the ABC(FC) is the version with fixed value of P_{copy}, and the ABC(VC) is the version with varied P_{copy}. The general variable neighborhood search algorithm presented by Soylu (2015) will be referred to as GVNS. These four algorithms are the state-of-the-art ones, and we have also chosen two representative algorithms to compare with. GGASS represents the steady-state grouping genetic algorithm presented by Singh and Baghel (2009), and the ACO represents the ant colony optimization presented by Liu et al. (2009). To be clear, the individual representations of the MASVND, the ABC(FC), the ABC(VC) and the IWO are derived from the GGASS, and the ACO is the only swarm intelligence algorithm in the previous for the minmax MTSP. More importantly, within the same iterations, the GGASS and the ACO are comparable to the state-of-the-art algorithms on the SBPs (Venkatesh & Singh, 2015), and that is why we have also done the comparisons with them. As a footnote, the ABC(FC), the ABC (VC), the IWO and the ACO are the four works that combine an EA with a LSP mentioned in Section 1.

Table 2
The maximum distance obtained by the five test groups on the benchmark problems.

No	n	m	$OptR_{n,m}$	$AveR_{n,m}$				
				$MASVND_0$	$MASVND_1$	$MASVND_2$	$MASVND_3$	$MASVND_4$
1	51	3	159.57	**159.57**	**159.57**	**159.57**	159.73	**159.57**
2	51	5	118.13	120.47	120.06	120.12	120.54	**120.01**
3	51	10	112.07	**112.07**	**112.07**	**112.07**	**112.07**	**112.07**
4	100	3	3034.69	3069.69	3069.07	3065.43	**3063.40**	3071.21
5	100	5	2238.38	**2257.67**	2267.59	2267.30	2273.31	2269.82
6	100	10	2086.38	**2086.38**	**2086.38**	**2086.38**	**2086.38**	**2086.38**
7	100	20	2086.38	**2086.38**	**2086.38**	**2086.38**	**2086.38**	**2086.38**
8	150	3	2407.34	2455.26	**2449.45**	2463.93	2450.13	2450.42
9	150	5	1744.26	1781.67	**1779.02**	1786.30	1796.86	1792.05
10	150	10	1554.64	1561.75	1559.49	1557.60	1557.16	**1557.15**
11	150	20	1554.64	**1554.64**	**1554.64**	**1554.64**	**1554.64**	**1554.64**
12	200	3	10813.13	11062.01	11121.34	11098.93	11045.91	**11043.62**
13	200	5	7415.54	**7540.72**	7585.84	7566.73	7582.08	7595.51
14	200	10	6223.22	6288.36	6284.26	6266.42	6249.17	**6247.80**
15	200	20	6223.22	**6223.22**	**6223.22**	**6223.22**	**6223.22**	**6223.22**
16	318	3	16088.73	**16422.35**	16537.66	16483.92	16477.89	16573.31
17	318	5	11524.29	**11792.52**	11881.84	11864.13	11896.71	11925.25
18	318	10	9731.17	9885.45	9839.01	9822.60	9818.75	**9803.95**
19	318	20	9731.17	**9731.17**	**9731.17**	**9731.17**	**9731.17**	**9731.17**
20	532	3	32403.10	33174.52	33198.15	33175.34	33424.78	**33149.06**
21	532	5	22558.26	23081.06	23117.66	23105.41	23079.34	**23078.53**
22	532	10	18264.42	18544.77	18565.80	**18490.98**	18515.67	18508.82
23	532	20	17641.16	17846.81	17830.72	17759.50	**17662.06**	17664.25
24	783	3	3272.95	3336.83	3332.84	**3331.53**	3336.57	3345.00
25	783	5	2092.77	**2131.65**	2135.88	2137.72	2134.03	2154.01
26	783	10	1432.34	1458.17	1467.03	1456.18	**1452.67**	1463.83
27	783	20	1260.88	1278.07	1281.42	1274.83	1270.31	**1269.40**
28	1173	3	22252.31	**22608.76**	22711.05	22747.22	22781.61	22828.82
29	1173	5	14557.30	14881.66	**14817.89**	14867.40	14861.40	14896.61
30	1173	10	9204.34	9423.47	9406.67	9373.80	9352.28	**9348.61**
31	1173	20	7063.23	7340.78	7338.98	7321.23	7276.69	**7270.61**

4.1. The impact of the l_{max} on the MASVND when optimizing the minmax MTSP

In this section, we have carried out a preliminary experiment to determine the best l_{max} on the MASVND for the minmax MTSP. For this purpose, we have set a number of the MASVND with various l_{max} such as 0 (apply only the 2-opt to the best offspring individual), 1, 2, 3, 4 (5 test groups in total), and run them on all the BPs. The five test groups have been executed on each BP 20 times independently, each time with a different random seed, and we will make the comparisons from the two following aspects: the precision and the robustness.

The results of the 20 replications are summarized in Table 2. For the sake of brevity, on the BP with given n and m, we used the $AveR_{n,m}$ to represent the mean result among the 20 runs (the shortest one is marked in bold), and the $OptR_{n,m}$ to represent the minimum result among the 100 (5×20) runs, $\forall n \in N$, $\forall m \in M$. From this table, we can find that the $MASVND_4$ obtains the shortest $AveR_{n,m}$ on 17 BPs, which is more than the other test groups: 13 ($MASVND_0$), 11($MASVND_1$), 9($MASVND_2$) and 9($MASVND_3$).

In order to study whether there are statistical differences among these results, the one-way ANOVA (95% confidence) has been performed, and Table 3 summarizes the number of the BPs where one test group obtains a statistically longer (<), equivalent (=) or shorter (>) $AveR_{n,m}$ than the others. It must be taken into account that our conclusion is based on the statistical analysis, which contains all the results (not $AveR_{n,m}$). From this table, it can be seen that almost all the test groups show a similar performance on the SBPs, while on the LBPs, the $MASVND_3$ and the $MASVND_4$ obtain a greater number of the shorter $AveR_{n,m}$ than the other three test groups. However, in general, the $MASVND_3$ has a bit better performance, where it obtains a shorter $AveR_{n,m}$ than the others on 6($MASVND_0$), 5($MASVND_1$), 2($MASVND_2$) and 2($MASVND_4$) BPs respectively. In addition, there is only one BP where the $MASVND_3$ is inferior to the $MASVND_1$. Thus, we can draw a primary conclusion that within the same time limit, the $MASVND_3$ has the best precision among the five test groups, which is different from the impression made by the direct observation from Table 2.

Furthermore, we have employed the Friedman test to give an order of the precision among the five test groups. Through sorting the $AveR_{n,m}$ in Table 2 in a descending way row by row, we can get the ranking of each $AveR_{n,m}$ on each row. After that, we can average the rankings for all the test groups on the whole, and these rankings are listed in Table 4. From this table, we can see that among the five test groups, the $MASVND_1$ has the highest ranking on the SBPs (the higher ranking, the better precision), while the $MASVND_3$ has the highest ranking on the LBPs. And in general, the latter has a higher ranking than the other test groups, which comes to a conclusion that is consistent to the one drawn by the one-way ANOVA.

In terms of the robustness, according to the analysis method in Geng, Zhan, and Zhou (2005), we used the $Error_{n,m}$ to represent the percentage deviation of the $AveR_{n,m}$ to the $OptR_{n,m}$, as Eq. (2), which can reflect the relative performance of one test group on one BP. Consequently, on the BP with given n and m, if one test group has a perfect performance then the $Error_{n,m} = 0\%$, otherwise the $Error_{n,m} > 0\%$. Therefore, we can use the mean $Error_{n,m}$ ($AveE$, as Eq. (3)) to subjectively evaluate the robustness of each test group, where the smaller the $AveE$, the better the robustness. Fig. 4 graphically illustrates the box plot of the $Error_{n,m}$ for each test group. According to this figure, it can be observed that for all the test groups their minimum $Error_{n,m}$ are the same (0%), whereas the maximum $Error_{n,m}$ of the $MASVND_3$ is distinctly less than those of the others. Besides, the size of the box plot of the $MASVND_0$ is the smallest among all. To be exact, we have figured out the $AveE$ for the five test groups, which are sequentially 1.42%, 1.51%, 1.42%, 1.39% and 1.44%, and that of the $MASVND_3$ is the least of all.

$$Error_{n,m} = \frac{AveR_{n,m} - OptR_{n,m}}{OptR_{n,m}} \times 100\%, \quad \forall n \in N, \quad \forall m \in M \tag{2}$$

$$AveE = \frac{\sum_n \sum_m Error_{n,m}}{|N| \times |M|} \tag{3}$$

Table 3
The summary of the superiority of one test group over the others when optimizing the minmax MTSP.

Scale	Algorithms	$MASVND_0$			$MASVND_1$			$MASVND_2$			$MASVND_3$			$MASVND_4$		
		<	=	>	<	=	>	<	=	>	<	=	>	<	=	>
Small-Scale	$MASVND_0$	–	–	–	0	11	0	0	11	0	0	11	0	0	11	0
	$MASVND_1$	0	11	0	–	–	–	0	11	0	0	10	1	0	11	0
	$MASVND_2$	0	11	0	0	11	0	–	–	–	0	11	0	0	11	0
	$MASVND_3$	0	11	0	1	10	0	0	11	0	–	–	–	0	11	0
	$MASVND_4$	0	11	0	0	11	0	0	11	0	0	11	0	–	–	–
Large-Scale	$MASVND_0$	–	–	–	0	19	1	2	18	0	6	14	0	6	11	3
	$MASVND_1$	1	19	0	–	–	–	3	17	0	5	15	0	4	15	1
	$MASVND_2$	0	18	2	0	17	3	–	–	–	2	18	0	3	16	1
	$MASVND_3$	0	14	6	0	15	5	0	17	3	–	–	–	0	18	2
	$MASVND_4$	3	11	6	1	15	4	1	16	3	2	18	0	–	–	–
All	$MASVND_0$	–	–	–	0	30	1	2	29	0	6	25	0	6	22	3
	$MASVND_1$	1	30	0	–	–	–	3	28	0	5	25	1	4	26	1
	$MASVND_2$	0	29	2	0	28	3	–	–	–	2	29	0	3	27	1
	$MASVND_3$	0	25	6	1	25	5	0	28	3	–	–	–	0	29	2
	$MASVND_4$	3	22	6	1	26	4	1	27	3	2	29	0	–	–	–

Table 4
The Friedman test rankings for the five test groups when optimizing the minmax MTSP.

Algorithms	$MASVND_0$	$MASVND_1$	$MASVND_2$	$MASVND_3$	$MASVND_4$
Average Ranking (Small-scale)	2.82	**3.36**	3.00	2.86	2.95
Average Ranking (Large-scale)	2.85	2.30	3.23	**3.48**	3.15
Average Ranking (All)	2.84	2.68	3.15	**3.26**	3.08

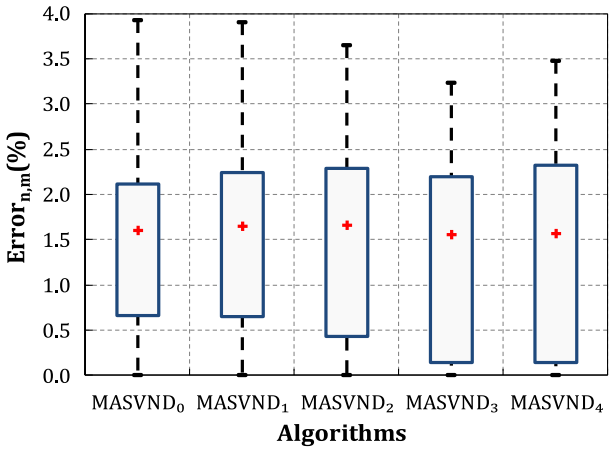


Fig. 4. The box plot of the $Error_{n,m}$ for each test group on the benchmark problems.

Then statistical differences among the $Error_{n,m}$ of the five test groups were indicated according to the one-way ANOVA (95% confidence), and Figs. 5 and 6 illustrate the corresponding multiple comparisons on the SBPs and the LBPs respectively. However, from these two figures, we can find that no one test group has the $Error_{n,m}$ significantly different from the others on both the SBPs and the LBPs, which indicates that all the test groups have a similar robustness.

For the sake of investigating the robustness of each test group to the problem parameters n and m, we denoted the mean $Error_{n,m}$ with given n by $AveE_n$ and that with given m by $AveE_m$, as Eqs. (4) and (5). After that, Figs. 7 and 8 graphically compare the trends

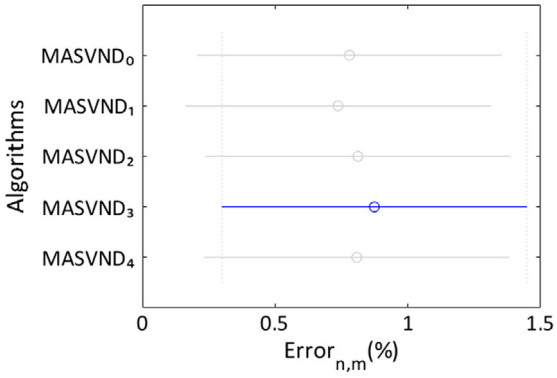


Fig. 5. The multiple comparisons of the $Error_{n,m}$ of the five test groups on the small-scale benchmark problems.

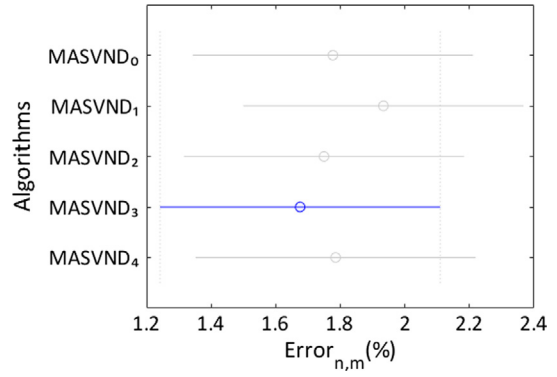


Fig. 6. The multiple comparisons of the $Error_{n,m}$ of the five test groups on the large-scale benchmark problems.

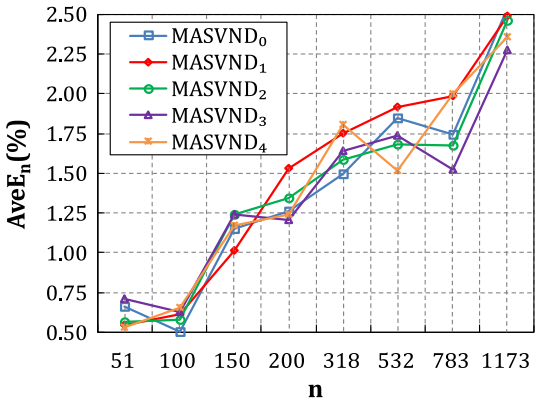


Fig. 7. The trends of $AveE_n$ of the five test groups on the number of the cities.

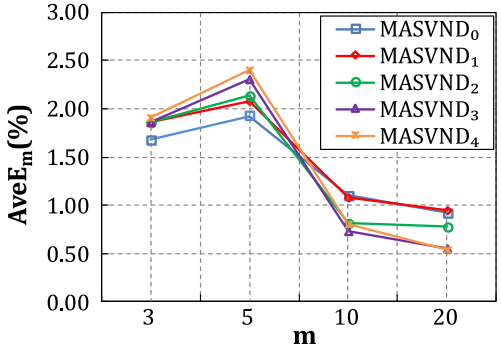


Fig. 8. The trends of $AveE_m$ of the five test groups on the number of the salesmen.

of the $AveE_n$ and the $AveE_m$ of the five test groups. It can be seen from Fig. 7 that as n increases, for each test group the $AveE_n$ experiences an overall upward trend. This makes sense, because although the time limit increases with the enlarging n, the growth of the computational complexity is much sharper (see Section 5). Meanwhile, as can be observed from Fig. 8 that for each test group there exists that $AveE_{m=5} > AveE_{m=3} > AveE_{m=10} > AveE_{m=20}$. It is easy to understand that $AveE_{m=5} > AveE_{m=3}$, but it is worth mentioning that within the same time limit, as m increases up to 10 or 20, the difficulty to determine the optimal allocation of cities into m tours becomes even harder so that each test group gets a poor performance (the gap between the $AveR_{n,m}$ and the $OptR_{n,m}$ is pretty small). In general, no matter when n reaches 783 or 1173, or when m reaches 10 or 20, the $AveE_n$ and the $AveE_m$ of the MASVND$_3$ stands at a lower level than those of the other test groups, which means that the performance of the MASVND$_3$ decreases relatively gently with the enlarging n and m.

$$AveE_n = \frac{\sum_m Error_{n,m}}{|M|}, \quad \forall n \in N \tag{4}$$

$$AveE_m = \frac{\sum_n Error_{n,m}}{|N|}, \quad \forall m \in M \tag{5}$$

In summary, based on a comprehensive consideration of the precision and the robustness, we can conclude that within the same time limit, the MASVND$_3$ has the best performance for optimizing the minmax MTSP among the five test groups, which will be used in the following experiments. Moreover, because the Or-opt4 move (Fig. 3) relocates a string of 4 cities at one time, it is prone to miss the global or local optimum and leads to the waste

of computation time. And this explains why the $MASVND_4$ has a longer neighborhood sequence than the $MASVND_3$, but its performance is inferior to the $MASVND_3$'s.

4.2. Comparison with various algorithms on the maximum distance traveled by any salesman

In this section, the MASVND and the six existing algorithms have been executed on each BP 20 independent times, each time with a different random seed. In addition to the two aspects mentioned in Section 4.1, the convergence speed was included to study the efficiency of the seven algorithms.

The results of the 20 replications are summarized in Table 5. As in Section 4.1, on the BP with given n and m, we used the $AveR_{n,m}$ to represent the mean result among the 20 runs and the $BestR_{n,m}$ the minimum result (the shortest one is marked in bold), $\forall n \in N$, $\forall m \in M$. From this table, we can see that the MASVND obtains the largest number of the shortest $AveR_{n,m}$ and the shortest

Table 5
The maximum distance obtained by the seven algorithms on the benchmark problems.

n	Algorithms	$AveR_{n,m}$	$BestR_{n,m}$	$AveR_{n,m}$	$BestR_{n,m}$	$AveR_{n,m}$	$BestR_{n,m}$	$AveR_{n,m}$	$BestR_{n,m}$
		m = 3		m=5		m=10		m=20	
51	GGASS	164.99	**159.57**	120.56	**118.13**	**112.07**	**112.07**		
	ACO	172.03	167.18	128.68	124.99	112.45	112.45		
	ABC(FC)	**159.57**	**159.57**	**118.19**	**118.13**	**112.07**	**112.07**		
	ABC(VC)	**159.57**	**159.57**	118.33	**118.13**	**112.07**	**112.07**		
	IWO	**159.57**	**159.57**	119.83	**118.13**	**112.07**	**112.07**		
	GVNS	163.62	**159.57**	119.92	**118.13**	**112.07**	**112.07**		
	MASVND	159.73	**159.57**	120.54	**118.13**	**112.07**	**112.07**		
100	GGASS	3203.76	3111.74	2359.25	**2238.38**	2087.55	**2086.38**	**2086.38**	**2086.38**
	ACO	3449.76	3350.45	2718.93	2584.87	2281.03	2229.11	2186.14	2121.76
	ABC(FC)	3069.74	3052.36	2248.33	2241.91	2092.84	**2086.38**	**2086.38**	**2086.38**
	ABC(VC)	**3052.44**	3042.03	**2246.23**	**2238.38**	2088.03	**2086.38**	**2086.38**	**2086.38**
	IWO	3058.62	**3034.69**	2251.59	**2238.38**	**2086.38**	**2086.38**	**2086.38**	**2086.38**
	GVNS	3163.76	3079.36	2332.49	**2238.38**	**2086.38**	**2086.38**	**2086.38**	**2086.38**
	MASVND	3063.40	**3034**.69	2273.31	2241.24	**2086.38**	**2086.38**	**2086.38**	**2086.38**
150	GGASS	2567.24	2497.41	1926.10	1865.01	1565.51	1555.60	**1554.64**	**1554.64**
	ACO	2714.65	2637.79	2115.07	2054.95	1720.06	1688.45	1603.83	1581.72
	ABC(FC)	2477.36	2454.40	1832.00	1797.32	1579.08	1563.39	1554.75	**1554.64**
	ABC(VC)	2450.39	2437.04	1787.22	1764.65	1565.69	1557.94	**1554.64**	**1554.64**
	IWO	**2435.42**	**2413.24**	**1761.32**	**1752.11**	1558.03	**1554.64**	**1554.64**	**1554.64**
	GVNS	2524.27	2427.77	1870.63	1830.50	**1556.78**	**1554.64**	**1554.64**	**1554.64**
	MASVND	2450.13	2429.49	1796.86	1758.08	1557.16	**1554.64**	**1554.64**	**1554.64**
200	GGASS	11684.27	11291.73	8389.96	7895.89	6417.66	6281.19	**6223.22**	**6223.22**
	ACO	12716.42	11891.81	9653.90	9098.85	7451.42	7158.10	7333.37	6922.79
	ABC(FC)	11220.83	10976.60	7911.40	7795.41	6418.85	6291.01	6223.24	**6223.22**
	ABC(VC)	11092.44	10933.11	7741.87	7595.41	6350.11	6294.24	**6223.22**	**6223.22**
	IWO	**10947.79**	**10814.18**	7593.15	7493.24	6278.99	6237.00	**6223.22**	**6223.22**
	GVNS	11429.68	10898.96	8055.72	7836.21	6274.36	**6223.22**	**6223.22**	**6223.22**
	MASVND	11045.91	10831.66	**7582.08**	**7415.54**	**6249.17**	**6223.22**	**6223.22**	**6223.22**
318	GGASS	17641.65	17008.70	12836.25	12506.94	10209.31	10050.36	9743.16	**9731.17**
	ACO	19309.18	18739.18	15272.96	14729.88	12547.00	12159.60	11579.01	11408.53
	ABC(FC)	17293.44	17062.22	12655.57	12449.06	10201.26	10061.30	9732.28	**9731.17**
	ABC(VC)	16824.41	16707.02	12306.18	12088.64	10054.52	9983.23	**9731.17**	**9731.17**
	IWO	**16340.30**	**16200.21**	11908.18	11730.03	9955.42	9845.72	**9731.17**	**9731.17**
	GVNS	17295.02	16861.99	12595.14	12210.40	9911.59	9826.77	**9731.17**	**9731.17**
	MASVND	16477.89	16206.25	**11896.71**	**11752.41**	**9818.75**	**9731.17**	**9731.17**	**9731.17**
532	GGASS	35919.07	34892.15	25798.98	24703.40	20947.36	20284.03	19187.11	18830.89
	ACO	40152.53	39149.35	30438.26	29355.86	24280.77	23349.26	21603.65	21001.04
	ABC(FC)	35590.99	35138.50	25729.10	25033.97	20516.16	19949.41	18542.56	18332.84
	ABC(VC)	34737.20	34401.24	25007.53	24564.33	19916.63	19584.52	18351.24	18156.62
	IWO	33687.30	32988.99	24029.62	23519.68	19439.53	19136.52	18051.02	17850.80
	GVNS	37434.43	36395.54	25561.36	24866.28	19802.01	19278.83	18028.39	17822.23
	MASVND	**33424.78**	**32403.10**	**23079.34**	**22619.66**	**18515.67**	**18390.46**	**17662.06**	**17641.16**
783	GGASS	3567.78	3504.70	2404.86	2324.14	1674.53	1633.50	1439.46	1400.28
	ACO	3825.29	3669.43	2704.11	2627.86	1886.59	1857.15	1538.46	1508.87
	ABC(FC)	3652.01	3622.79	2443.54	2413.85	1668.60	1626.69	1388.76	1375.16
	ABC(VC)	3572.63	3530.31	2371.48	2317.66	1614.42	1587.43	1366.09	1350.96
	IWO	3497.56	3457.97	2303.14	2273.80	1564.70	1542.05	1333.12	1311.30
	GVNS	3603.36	3518.08	2397.27	2325.47	1548.19	1515.03	1591.08	1578.87
	MASVND	**3336.57**	**3279.16**	**2134.03**	**2092.77**	**1452.67**	**1432.34**	**1270.31**	**1260.88**
1173	GGASS	24566.79	23981.28	16879.83	16512.21	11768.64	11407.52	9388.33	9150.17
	ACO	25273.25	23875.35	17424.11	16847.71	11462.13	11057.73	8928.05	8432.10
	ABC(FC)	25032.73	24748.31	16881.71	16590.98	11176.21	10965.66	8593.08	8373.09
	ABC(VC)	24575.92	24384.17	16478.66	16222.91	10901.55	10652.46	8329.41	8228.66
	IWO	24300.25	24008.47	16274.64	16057.19	10667.97	10517.94	8207.88	8063.17
	GVNS	27899.43	24988.00	15985.37	15494.12	10637.17	10386.45	8484.49	8311.38
	MASVND	**22781.61**	**22443.22**	**14861.40**	**14557.30**	**9352.28**	**9222.92**	**7276.69**	**7063.23**

Notes: the detailed results for the optimal longest tours obtained by the MASVND on the benchmark problems can be found in Appendix C.

Table 6
The summary of the superiority of the MASVND over the other algorithms when optimizing the minmax MTSP.

Algorithms	MASVND								
	Small-scale			Large-scale			Total		
	<	=	>	<	=	>	<	=	>
GGASS	6	5	0	18	2	0	24	7	0
ACO	11	0	0	20	0	0	31	0	0
ABC(FC)	2	8	1	17	3	0	19	11	1
ABC(VC)	1	9	1	17	3	0	18	12	1
IWO	0	10	1	12	8	0	12	18	1
GVNS	5	6	0	17	3	0	22	9	0

Table 7
The Friedman test rankings for the seven algorithms when optimizing the minmax MTSP.

No	1	2	3	4	5	6	7
Algorithms	MASVND	IWO	ABC(VC)	GVNS	ABC(FC)	GGASS	ACO
Average Ranking	**6.03**	5.65	4.77	4.16	3.50	2.76	1.13

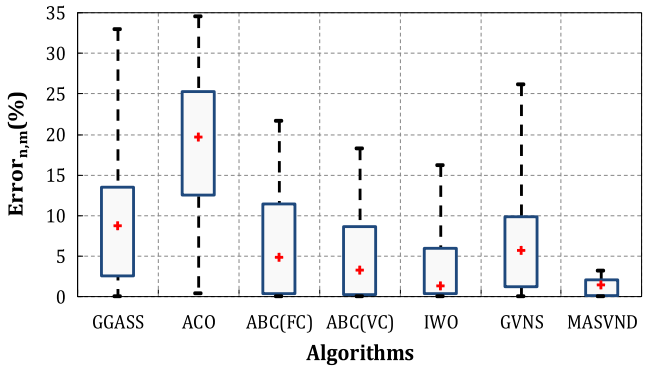


Fig. 9. The box plot of the $Error_{n,m}$ for each algorithm on the benchmark problems.

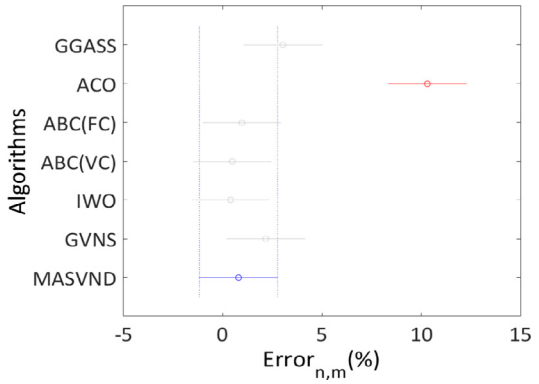


Fig. 10. The multiple comparisons of the $Error_{n,m}$ of the seven algorithms on the small-scale benchmark problems.

$BestR_{n,m}$ among the seven algorithms. To be precise, the MASVND obtains the shortest $AveR_{n,m}$ on 22 BPs, and it also obtains the shortest $BestR_{n,m}$ on 26 BPs. Besides, there are 15 of the 26 BPs where only the MASVND can obtain the shortest $BestR_{n,m}$, as shown in Appendix B.

By means of the one-way ANOVA (95% confidence), Table 6 summarizes the number of the BPs where the MASVND obtains a statistically longer (<), equivalent (=) or shorter (>) $AveR_{n,m}$ than the other algorithms. According to this table, it is obvious to observe that the $AveR_{n,m}$ of the MASVND is always shorter than that of the ACO on all the SBPs, and besides that the MASVND obtains an equivalent $AveR_{n,m}$ to the other five algorithms on 5 (GGASS), 6(GVNS), 8(ABC(FC)), 9(ABC(VC)) and 10(IWO) SBPs respectively. While on the LBPs, the MASVND performs far better than the others, where it obtains a shorter $AveR_{n,m}$ on at least 60% of the LBPs. In general, we can find that the MASVND obtains a shorter $AveR_{n,m}$ than the four state-of-the-art algorithms on 19 (ABC(FC)), 18(ABC(VC)), 12(IWO), and 22(GVNS) BPs respectively. And as for the two representative ones, there are 24(GGASS) and 31(ACO) BPs where the MASVND obtains a shorter $AveR_{n,m}$ than theirs. In addition, there is only one BP where the ABC(FC), the ABC(VC) and the IWO can obtain a shorter $AveR_{n,m}$ than the MASVND. Thus, we can primarily conclude that within the same time limit, the MASVND has the best precision among the seven algorithms, which is consistent to the impression made by the direct observation from Table 5.

In order to give an order of the precision among the seven algorithms, the Friedman test was included to obtain the rankings for them, as shown in Table 7. From this table, it can be seen that among all the algorithms, the MASVND has the highest ranking on the whole, which means the conclusion drawn by the ANOVA is sustained.

When it comes to the robustness, Fig. 9 graphically illustrates the box plot of the $Error_{n,m}$ for each algorithm. Here, the definitions of $Error_{n,m}$, $AveE$, $AveE_n$ and $AveE_m$ are the same as Section 4.1. According to this figure, it can be observed that although for the seven algorithms their minimum $Error_{n,m}$ are the same (0%), the maximum $Error_{n,m}$ of the MASVND is remarkably less than those of the others, and the size of its box plot is also the smallest among all. To be precise, we have figured out the $AveE$ for the seven algorithms, which are 8.69%(GGASS), 19.07%(ACO), 6.74%(ABC(FC)), 5.12%(ABC(VC)), 3.72%(IWO), 7.15%(GVNS) and 1.21%(MASVND). Compared with the other algorithms, the MASVND improves the $AveE$ by 86.08%, 93.65%, 82.05%, 76.37%, 67.47% and 82.66% sequentially.

Then statistical differences among the $Error_{n,m}$ of the seven algorithms were indicated through the one-way ANOVA (95% confidence), and Figs. 10 and 11 illustrate the corresponding multiple comparisons on the SBPs and the LBPs respectively. From these two figures, we can find that in terms of the $Error_{n,m}$, only the ACO has that significantly different from the MASVND on the SBPs, and on the LBPS, five algorithms except for the IWO have that significantly different from the MASVND.

Furthermore, for the sake of investigating the robustness of each algorithm to the problem parameters n and m,

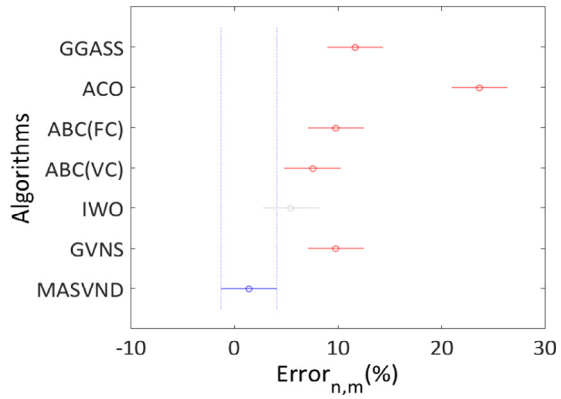


Fig. 11. The multiple comparisons of the $Error_{n,m}$ of the seven algorithms on the large-scale benchmark problems.

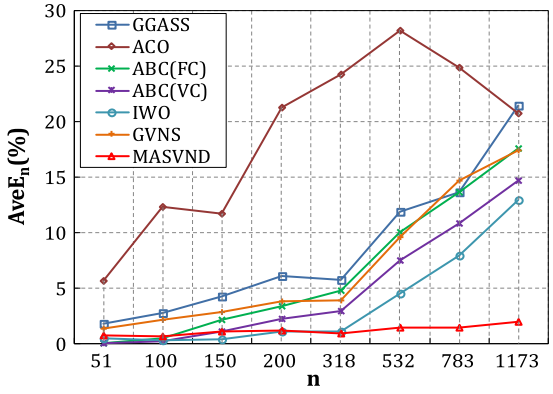


Fig. 12. The trend of $AveE_n$ of the seven algorithms on the number of the cities.

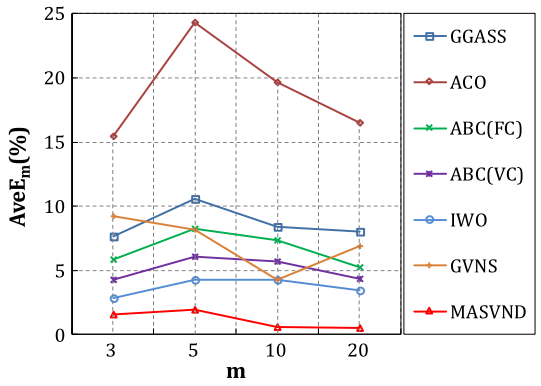


Fig. 13. The trend of $AveE_m$ of the seven algorithms on the number of the salesmen.

Figs. 12 and 13 graphically compare the trends of the $AveE_n$ and the $AveE_m$ of the seven algorithms. It can be seen from Fig. 12 that the $AveE_n$ of the MASVND consistently remains steady at a low level (range from 0.63% to 2.01%), in particular, when n ranges from 532 to 1173, it is far lower than those of the other algorithms. As can be seen from Fig. 13 that the $AveE_m$ of the MASVND stands at the lowest point for each m, and observations similar to Section 4.1 can be found here, in general the $AveE_{m=5}$ is larger than the other cases. Hence, we can come to the conclusion that the MASVND has a better robustness than the other algorithms.

In terms of the convergence speed, Fig. 14 graphically compares the convergence curves of the seven algorithms on the 12 LBPs (n = 532, 783 and 1173). As can be observed from Fig. 12 that, when n reaches 532, 783 and 1173, the gap of the $AveE_n$ between the MASVND and the other algorithms is enormous, for the sake of brevity, only the 12 results are listed. Here, the horizontal axis indicates the running time and the vertical axis shows the obtained maximum distance at each second. According to this figure, the MASVND converges faster than the other algorithms, and no more than one tenth of the allowed maximum time it can converge to a level close to the global optimum. After that, small constant improvements are observed on the MASVND and finally it visibly converges to the lowest level among all. As for the GGASS, ACO and ABC(FC), they often fall into a local optimum and cannot escape when they converge to a relatively low level. And although the ABC (VC) and the IWO can also obtain some small improvements at the late stage of the evolution, the gap of the results between theirs and the MASVND's is still evident. In addition, because of adopting the best improvement strategy, on most of the LBPS, the GVNS consume much more time in the previous generations than any other compared algorithm. Hence, it can be concluded that within the same time limit, the MASVND has a faster convergence speed, i.e., a better efficiency than the other algorithms.

In summary, based on a comprehensive consideration of the precision, the robustness and the convergence speed, we can conclude that within the same time limit, the MASVND has a better performance than the other algorithms for optimizing the minmax MTSP. In particular, compared with the four state-of-the-art algorithms, the specific analysis is as follows. In regard to the three swarm intelligence algorithms, i.e., the ABC(FC), the ABC(VC) and the IWO, in addition to the evolutionary rules, they are short of an efficient mechanism to exhaustively reduce the maximum distance. Thereby, although these algorithms have the similar convergence curves to the MASVND, they cannot gain the further improvements at the late stage of the evolution, especially on the LBPs. As for the GVNS, even though it has a longer neighborhood sequence than the MASVND, it cannot gain a better precision than the MASVND within the same time limit, which can be attributed to the bidirectional neighborhoods it takes. And what is worse, despite that the best improvement strategy it takes is a more thorough way to search the areas near the best individual, this technique consumes a great deal of time but gains no more improvements. As a result, its convergence speed is far behind that of the MASVND.

4.3. Investigation on the total distance traveled by all the salesmen when optimizing the minmax MTSP

Bertazzi et al. have pointed out that it is extremely hard to minimize the maximum distance traveled by any salesman and the total distance traveled by all the salesmen at the same time (Bertazzi et al., 2015). Even from the worst-case point of view, the total distance in the minmax MTSP is at most m times that in the minsum one, so we would certainly hope the total distance as short as possible while minimizing the maximum distance. Fig. 15 introduces an example (n = 51, m = 3) about two groups of tours that have the same maximum distance but different total distance. In this section, we are going to investigate the total distance obtained by the MASVND when optimizing the minmax MTSP, and compare the MASVND with the six existing algorithms to see whether it would deteriorate the total distance.

For analyzing, we have collected the results of the total distance through summing up the distance traveled by each salesman from Section 4.2, as summarized in Table 8. Here, on the BP with given n and m, we used the $AveTR_{n,m}$ to represent the mean result among the 20 runs (the shortest one is marked in bold), and the $OptTR_{n,m}$ to represent the minimum result among the 140 (7×20) runs, $\forall n \in N$, $\forall m \in M$. From this table, we can see that the ABC(VC)

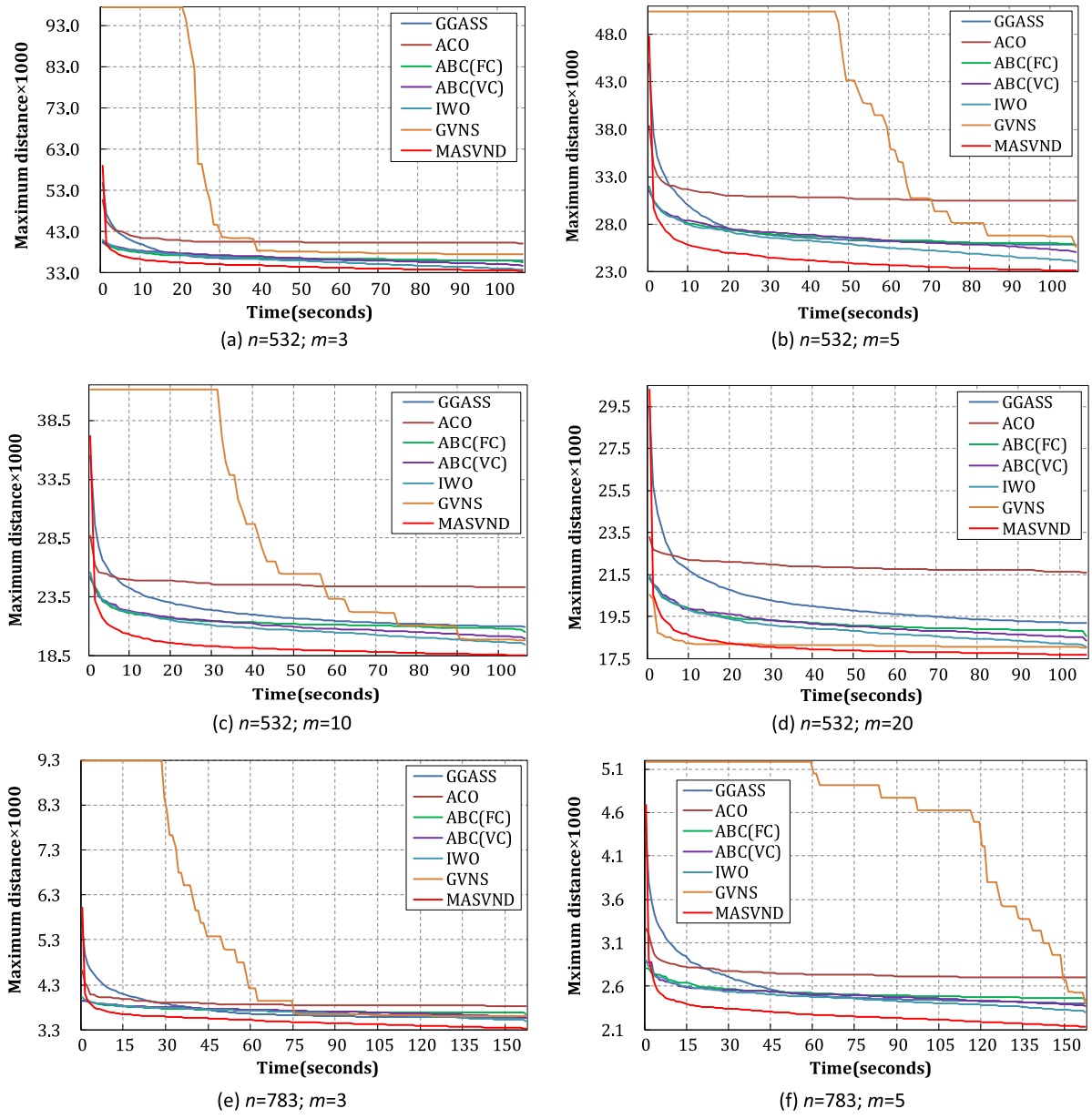


Fig. 14. Convergence curves of the seven algorithms on the 12 large-scale benchmark problems.

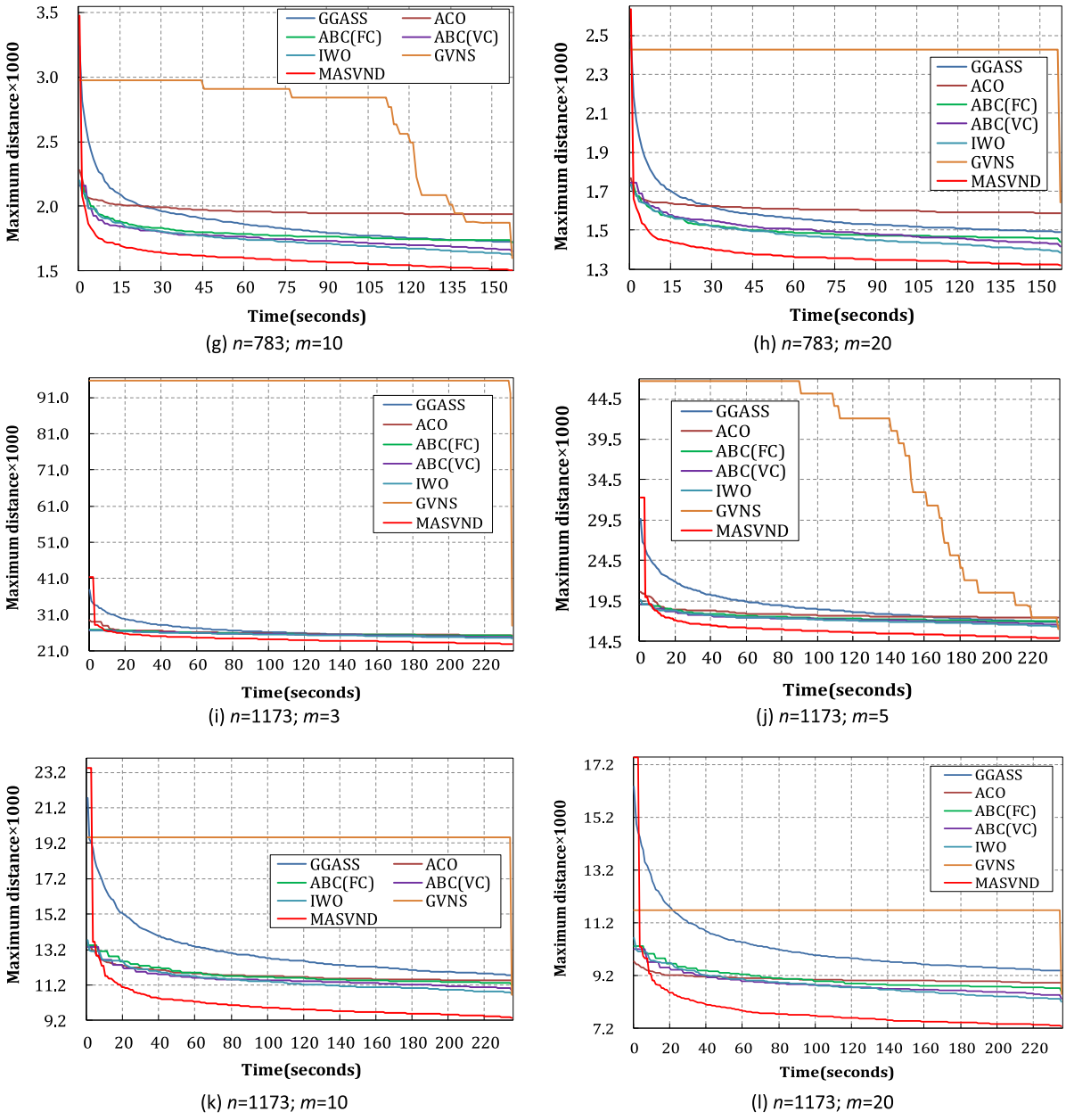
obtains a greater number of the shortest $AveTR_{n,m}$ on the BPs with n ranging from 51 to 100, the IWO with n ranging from 150 to 318, and the MASVND with n ranging from 532 to 1173 respectively.

Through the one-way ANOVA (95% confidence), Table 9 summarizes the number of the BPs where the MASVND obtains a statistically longer (<), equivalent (=) or shorter (>) $AveTR_{n,m}$ than the other algorithms. From this table, in general, the MASVND obtains a greater number of the shorter $AveTR_{n,m}$ than the other algorithms, where in particular, the $AveTR_{n,m}$ of the MASVND is always shorter than that of the ACO. Although, on the SBPs, the $AveTR_{n,m}$ of the ABC(FC), the ABC(VC) and the IWO are slightly shorter than that of the MASVND. Furthermore, Table 10 shows the Friedman test rankings for the seven algorithms. According to this table, it can be seen that among them, the IWO has the highest ranking on the whole, and the MASVND ranks the second. Thus, we can roughly conclude that the MASVND has a good capacity to keep the total distance as short as possible.

In terms of the robustness, Fig. 16 graphically illustrates the box plot of the $ErrorT_{n,m}$ for each algorithm, here the $ErrorT_{n,m}$ represents the percentage deviation of the $AveTR_{n,m}$ to the $OptTR_{n,m}$, as Eq. (6). From this figure, it can be observed that the minimum $ErrorT_{n,m}$ of the MASVND and those of the four state-of-the-art algorithms are very close to each other. Meanwhile, the maximum $ErrorT_{n,m}$ of the GVNS is remarkably less than those of the other algorithms, and its size of the box plot is also the smallest among all. To be exact, we have figured out the $AveET$, as Eq. (7), for the seven algorithms, which are 14.50%(GGASS), 21.08%(ACO), 12.47%(ABC(FC)), 10.01%(ABC(VC)), 8.21%(IWO), 8.83%(GVNS) and 6.98%(MASVND). Compared with the other algorithms, the MASVND improves the $AveET$ by 51.86%, 66.89%, 44.03%, 30.27%, 14.98% and 20.95% sequentially.

$$ErrorT_{n,m} = \frac{AveTR_{n,m} - OptTR_{n,m}}{OptTR_{n,m}} \times 100\%, \quad \forall n \in N, \quad \forall m \in M \tag{6}$$

$$AveET = \frac{\sum_n \sum_m Error_{n,m}}{|N| \times |M|} \times 100\% \tag{7}$$



(g) n=783; m=10 (h) n=783; m=20

(i) n=1173; m=3 (j) n=1173; m=5

(k) n=1173; m=10 (l) n=1173; m=20

Fig. 14 (*continued*)

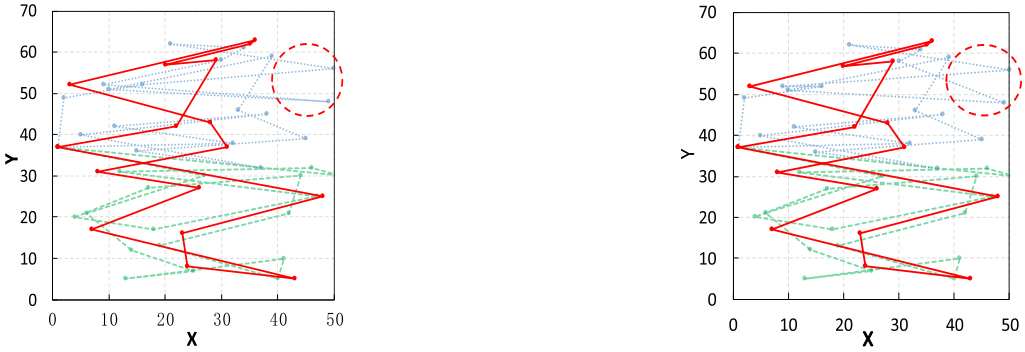
Then statistical differences among the $ErrorT_{n,m}$ of the seven algorithms were indicated according to the one-way ANOVA (95% confidence), and Figs. 17 and 18 illustrate the corresponding multiple comparisons on the SBPs and the LBPs respectively. From these two figures, we can find that in terms of the $ErrorT_{n,m}$, no one algorithm has that significantly different from the MASVND on the SBPs, while on the LBPs, only the ABC(FC) and the two representative algorithms have that significantly different from the MASVND. On the whole, we can come to a rough conclusion that the MASVND's capacity to keep the total distance is robust to the problem parameters n and m.

In summary, we can conclude that when optimizing the minmax MTSP, the MASVND has a better or at least competitive capacity to maintain the total distance than the other algorithms. It is worth mentioning that no extra operator was employed in the MASVND to reduce the total distance. Even for the 2-opt algorithm, it is only able to improve each tour within each individual independently. In Section 4.2, we have demonstrated the superiority of the MASVND over the other algorithms when optimizing the minmax MTSP. And although both the minmax and the minsum objectives are conflicting ones, we can find that the MASVND does not deteriorate the total distance severely. Besides, apart from the MASVND and the IWO, the orders of the rankings for the other five algorithms in Tables 7 and 10 are the same.

5. Computational complexity

In this section, according to the analysis method in Turky and Abdullah (2014), we calculate the computational complexity of the MASVND as follows:

- The time complexity of randomly generating the initial population is $O(P_{size} \times n)$.



(a) The maximum distance is 159.57, and the total distance is 473.64. (b) The maximum distance is 159.57, and the total distance is 476.80.

Fig. 15. An example about two groups of tours that have the same maximum distance but different total distance. Notes: the red tours are the longest ones, and the sub-tours within the red dashed line at the top-right corner are different from each other.

Table 8
The total distance obtained by the seven algorithms on the benchmark problems.

No	n	m	$OptTR_{n,m}$	$AveTR_{n,m}$						
				GGASS	ACO	ABC(FC)	ABC(VC)	IWO	GVNS	MASVND
1	51	3	473.64	492.37	504.29	475.08	474.13	**474.11**	488.41	475.19
2	51	5	571.46	590.51	605.34	582.27	**581.85**	585.96	594.72	591.41
3	51	10	768.23	949.92	967.45	831.70	**831.06**	851.79	936.30	914.68
4	100	3	9062.83	9590.36	10125.37	9170.42	**9129.84**	9158.20	9477.35	9172.16
5	100	5	11040.66	11698.80	12842.65	11163.54	**11144.00**	11176.84	11634.70	11299.17
6	100	10	17016.44	19517.55	20264.75	19516.59	19111.44	**18612.63**	20292.41	19522.95
7	100	20	26136.58	34721.22	36815.07	29893.17	29390.30	29659.91	**29110.60**	31692.22
8	150	3	7238.91	7696.11	7936.09	7411.77	7338.39	**7295.53**	7566.04	7342.78
9	150	5	8727.98	9576.90	10062.69	9078.33	8869.62	**8771.70**	9338.98	8953.17
10	150	10	14231.52	**14893.83**	15677.04	15486.41	15311.72	15037.46	15498.19	15250.75
11	150	20	20705.96	27097.56	27684.10	26155.91	23904.71	**23466.27**	24299.42	26203.55
12	200	3	32434.81	35021.86	36979.68	33552.63	33197.94	**32818.37**	34273.08	33114.92
13	200	5	37015.36	41738.47	45007.39	39212.02	38360.27	37782.75	40214.80	**37694.77**
14	200	10	56651.45	60677.33	65494.16	61418.59	60954.88	**58292.86**	62231.14	60483.63
15	200	20	89555.72	111454.56	116592.35	98026.24	94816.97	**94025.79**	96972.20	97445.22
16	318	3	48386.85	52853.77	56745.38	51736.88	50362.30	**48897.97**	51867.74	49408.34
17	318	5	58519.03	63793.43	72424.81	62929.89	61231.26	**59290.82**	62914.97	59435.81
18	318	10	93806.58	97755.08	112308.84	100734.13	98679.37	97402.92	98804.19	**96655.55**
19	318	20	141608.75	177962.89	194923.08	171389.91	169222.48	163926.91	**149009.77**	168236.25
20	532	3	97180.80	107497.44	116874.55	106668.84	104076.74	100906.02	112291.58	**100256.43**
21	532	5	113032.14	127001.34	142470.00	128303.31	124637.58	119673.03	127613.34	**115252.84**
22	532	10	180174.92	193833.92	222395.66	203309.81	197412.63	192185.93	196447.30	**183060.29**
23	532	20	300030.66	330862.20	395720.34	362132.65	347940.26	342959.95	**313915.79**	334692.05
24	783	3	9835.79	10664.31	11262.43	10936.57	10697.71	10473.37	10809.13	**10007.22**
25	783	5	10448.12	11893.37	12943.15	12182.19	11818.37	11468.61	11947.77	**10656.48**
26	783	10	14212.62	16056.47	17522.81	16554.05	15999.34	15526.55	15396.11	**14423.45**
27	783	20	19207.09	27085.98	28324.37	27321.67	26888.07	26201.60	**19560.56**	24401.92
28	1173	3	67239.06	73304.20	74349.52	75014.66	73639.87	72806.73	80042.86	**68299.78**
29	1173	5	72693.05	82941.06	82877.70	84199.88	82154.02	81092.08	79674.90	**74197.50**
30	1173	10	90447.47	111866.97	104452.56	111049.90	108256.73	105894.88	**92441.33**	92484.53
31	1173	20	121491.88	174006.48	155129.09	169437.13	164308.74	161653.98	**122870.25**	139423.22

Table 9
The summary of the superiority of the MASVND over the other algorithms on the total distance.

Algorithms	MASVND								
	Small-scale			Large-scale			Total		
	<	=	>	<	=	>	<	=	>
GGASS	7	3	1	17	3	0	24	6	1
ACO	11	0	0	20	0	0	31	0	0
ABC(FC)	1	7	3	16	4	0	17	11	3
ABC(VC)	0	7	4	15	5	0	15	12	4
IWO	0	5	6	10	9	1	10	14	7
GVNS	7	2	2	14	2	4	21	4	6

Table 10
The Friedman test rankings for the seven algorithms on the total distance.

No	1	2	3	4	5	6	7
Algorithms	IWO	MASVND	ABC(VC)	GVNS	ABC(FC)	GGASS	ACO
Average Ranking	**5.89**	5.45	4.95	3.90	3.31	3.08	1.42

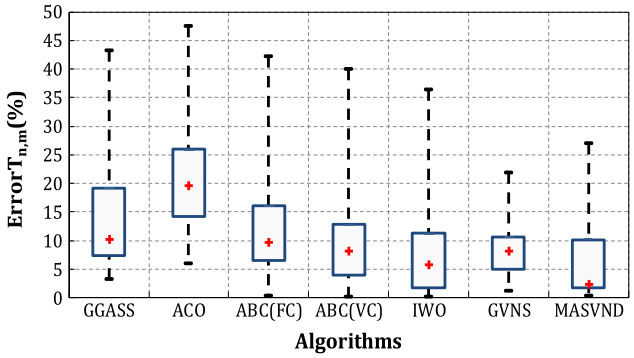


Fig. 16. The box plot of $ErrorT_{n,m}$ for each algorithm on the benchmark problems.

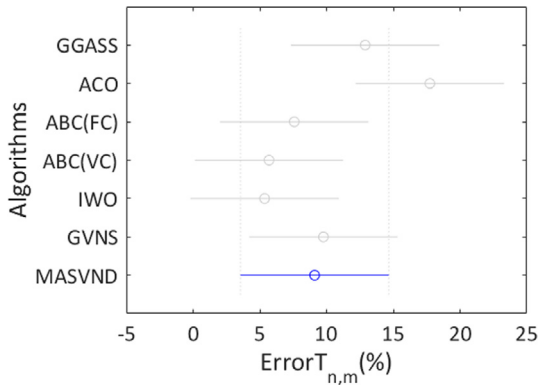


Fig. 17. The multiple comparisons of the $ErrorT_{n,m}$ of the seven algorithms on the small-scale benchmark problems.

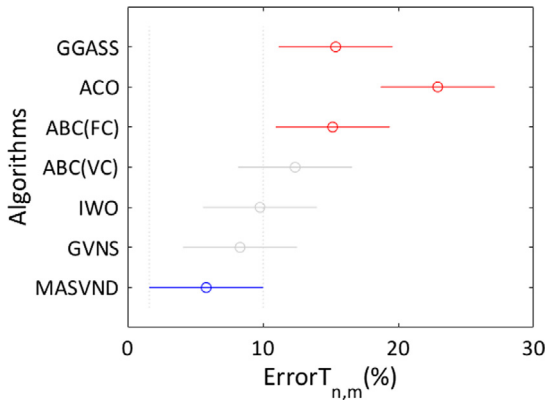


Fig. 18. The multiple comparisons of the $ErrorT_{n,m}$ of the seven algorithms on the large-scale benchmark problems.

- The time complexity of optimizing an individual with the 2-opt algorithm is O($n2\ 1 + n2\ 2 + \ldots + n2\ m$), $n_1 + n_2 + \ldots + n_m = n$, where the n_i is the number of cities in the i-th tour of the individual (Helsgaun, 2009). From the worst-case point of view, there is one tour containing $n - m$ cities, and the other $m - 1$ tours containing only one city except for the depot. In this case, the time complexity of optimizing the population with the 2-opt algorithm is O($P_{size} \times (n - m)^2$).
- The time complexity of evaluating the population is O($P_{size} \times n$).
- Hence, the time complexity of the initialization stage is O($P_{size} \times ((n - m)^2 + 2n)$) = O($P_{size} \times (n - m)^2$).
- The time complexity of generating an offspring individual with the *recombine* operator is O($n + (n - n_{un}) + (n - n_{un} + 1) + \ldots + (n - 1)$), where n_{un} is the number of unassigned cities after the first step of the operator is finished. Let $n_{un} = \alpha \times n$, $0 < \alpha < 1$, and the time complexity can be represented by O($(2\alpha - \alpha^2) \times n^2 + (1 - \alpha) \times n$) = O($n^2$).
- The time complexity of generating an offspring individual with the *mutate* operator is O($n + (n - n \times P_{copy}) + (n - n \times P_{copy} + 1) + \ldots + (n - 1)$), which can be represented by O($(2P_{copy} - P2$ copy$) \times n^2 + (1 - P_{copy}) \times n$) = O($n^2$).
- The time complexity of selecting the individual for the *recombine*/*mutate* operator is O(1).
- The time complexity of finding the best individual in the population which includes evaluating the population, can be represented by O($P_{size}\log(P_{size}) + P_{size} \times n$) = O($P_{size} \times (\log(P_{size}) + n)$).
- The time complexity of each neighborhood in the seq_VND is O(n^2), since in the worst case $0.5n$ cities of the selected tour should be relocated to $0.5n$ available positions (Soylu, 2015). Thus, the time complexity of the seq_VND which includes optimizing the selected individual with the 2-opt algorithm, can be represented by O($l_{max} \times n^2 + (n - m)^2$) = O($n^2$).
- The time complexity of readjusting the size of the population is O(P_{size}).
- Hence, the time complexity of each iteration is O($P_{size} \times (1 + P_c \times n^2 + (1 - P_c) \times n^2 + P_{size} \times (\log(P_{size}) + n) + n^2 + P_{size})$) = O($P_{size} \times n^2$).

6. Conclusions

In this paper, we have proposed a novel memetic algorithm that integrates with the sequential variable neighborhood descent to resolve the minmax MTSP. The framework of the proposed algorithm provides the search with an ideal trade-off between the exploration and the exploitation, where a *recombine* operator alternates with a *mutate* operator in use to generate new individuals, and the seq_VND acts as a specific mechanism to locally improve the best individual at each generation. In the proposed algorithm, the *recombine* operator is responsible for exchanging memes between individuals, where those promising tours are always preferential to be included into the offspring. And the *mutate* operator is used to perturb an individual by varied amounts which are increased from a pre-defined initial value to a pre-defined final value over the iterations. Meanwhile, we have indicated that the existing neighborhoods on the minmax MTSP does limit the search performance and then redefined a new neighborhood sequence where only those moving cities unidirectionally between two tours are considered. We have evaluated and compared our algorithm

with four state-of-the-art algorithms and two representative ones on the 11 SBPs and the 20 LBPs, which increases the largest number of cities on the minmax MTSP from 575 to 1173. Experimental results demonstrate the superiority of our algorithm over any other compared algorithm in terms of the three aspects, including the precision, the robustness and the convergence speed. Furthermore, we have also investigated the total distance obtained by the various algorithms when optimizing the minmax objective, and comparisons show that our algorithm is better than or at least competitive to the other algorithms in maintaining the total distance as short as possible.

Our further work will cover three main areas. Firstly, as we consider the minmax MTSP with a single depot, a possible future direction is to take into account that with more than one depot. It is worth mentioning that in the multiple-depot case, the depot is no longer implicit in the individual representation so that it should be avoided that there is more than one depot in one tour. This means that both the evolutionary operators and the LSP cannot move the depot from one tour to another arbitrarily. Secondly, we would like to develop more efficient neighborhoods on the minmax MTSP, which could improve the search performance within the same time limit. Thirdly, we are going to apply our algorithm to some real-world problems, such as traffic signalization network and scheduling at automated seaport terminals.

Acknowledgement

We are grateful to the anonymous reviewers for their valuable comments and suggestions. This research is supported by National Science Foundation of China under Grants No. 71271034 and Science Foundation of Liaoning Province under Grants No. 2014025015.

Appendix. Supplementary material

Supplementary data associated with this article can be found, in the online version, at http://dx.doi.org/10.1016/j.cie.2016.12.017.

References

Al Jadaan, O., Rajamani, L., & Rao, C. R. (2008). Improved selection operator for GA. *Journal of Theoretical & Applied Information Technology, 4*(4).

Ann, S., Kim, Y., & Ahn, J. (2015). Area allocation algorithm for multiple UAVs area coverage based on clustering and graph method. *IFAC-PapersOnLine, 48*(9), 204–209.

Bertazzi, L., Golden, B., & Wang, X. (2015). Min–max vs. min–sum vehicle routing: A worst-case analysis. *European Journal of Operational Research, 240*(2), 372–381.

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

Cai, X., Cheng, X., Fan, Z., Goodman, E., & Wang, L. (2015). An adaptive memetic framework for multi-objective combinatorial optimization problems: studies on software next release and travelling salesman problems. *Soft Computing*, 1–22.

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

Cordeau, J. F. J., Laporte, G., Vigo, D., Savelsbergh, M. W. P., & Vigo, D. (2007). Vehicle routing. *Methods & Studies Elsevier Science Publishers*, 195–224.

Dawkins, R. (2016). *The selfish gene*. Oxford University Press.

de Armas, J., & Melián-Batista, B. (2015). Variable neighborhood search for a dynamic rich vehicle routing problem with time windows. *Computers & Industrial Engineering, 85*, 120–131.

Falkenauer, E. (1998). *Genetic algorithms and grouping problems*. John Wiley & Sons Inc.

Filipović, V. (2012). Fine-grained tournament selection operator in genetic algorithms. *Computing and Informatics, 22*(2), 143–161.

Fonseca, G. H., & Santos, H. G. (2014). Variable neighborhood search based algorithms for high school timetabling. *Computers & Operations Research, 52*, 203–208.

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

Frederickson, G. N., Hecht, M. S., & Kim, C. E. (1976). Approximation algorithms for some routing problems. In *17th annual symposium on foundations of computer science, 1976* (pp. 216–227). IEEE.

Geng, X., Zhan, D. C., & Zhou, Z. H. (2005). Supervised nonlinear dimensionality reduction for visualization and classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 35*(6), 1098–1107.

Ghani, J. A., Choudhury, I. A., & Hassan, H. H. (2004). Application of Taguchi method in the optimization of end milling parameters. *Journal of Materials Processing Technology, 145*(1), 84–92.

Golden, B. L., Laporte, G., & Taillard, É. D. (1997). An adaptive memory heuristic for a class of vehicle routing problems with minmax objective. *Computers & Operations Research, 24*(5), 445–452.

Gorenstein, S. (1970). Printing press scheduling for multi-edition periodicals. *Management Science, 16*(6), B-373.

Groër, C., Golden, B., & Wasil, E. (2010). A library of local search heuristics for the vehicle routing problem. *Mathematical Programming Computation, 2*(2), 79–101.

Gutierrez, A., Dieulle, L., Labadie, N., & Velasco, N. (2016). A multi population memetic algorithm for the vehicle routing problem with time windows and stochastic travel and service times. *IFAC-PapersOnLine, 49*(12), 1204–1209.

Hansen, P., Mladenović, N., & Pérez, J. A. M. (2010). Variable neighbourhood search: methods and applications. *Annals of Operations Research, 175*(1), 367–407.

Helsgaun, K. (2009). General k-opt submoves for the lin–kernighan tsp heuristic. *Mathematical Programming Computation, 1*(2), 119–163.

Holland, J. H. (1975). *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence*. U Michigan Press.

Karapetyan, D., & Gutin, G. (2010). Lin-Kernighan heuristic adaptations for the generalized traveling salesman problem. *European Journal of Operational Research, 208*(3), 221–232.

Kashan, A. H., Akbari, A. A., & Ostadi, B. (2015). Grouping evolution strategies: An effective approach for grouping problems. *Applied Mathematical Modelling, 39*(9), 2703–2720.

Kindervater, G. A., & Savelsbergh, M. W. (1997). Vehicle routing: handling edge exchanges. *Local Search in Combinatorial Optimization*, 337–360.

Király, A., Christidou, M., Chován, T., Karlopoulos, E., & Abonyi, J. (2016). Minimization of off-grade production in multi-site multi-product plants by solving multiple traveling salesman problem. *Journal of Cleaner Production, 111*, 253–261.

Kivelevitch, E., Sharma, B., Ernest, N., Kumar, M., & Cohen, K. (2014). A hierarchical market solution to the min–max multiple depots vehicle routing problem. *Unmanned Systems, 2*(01), 87–100.

Lai, X., & Hao, J. K. (2016). A tabu search based memetic algorithm for the max-mean dispersion problem. *Computers & Operations Research, 72*, 118–127.

Liu, W., Li, S., Zhao, F., & Zheng, A. (2009). An ant colony optimization algorithm for the multiple traveling salesmen problem. In *2009 4th IEEE conference on industrial electronics and applications* (pp. 1533–1537). IEEE.

Masutti, T. A., & de Castro, L. N. (2009). Neuro-immune approach to solve routing problems. *Neurocomputing, 72*(10), 2189–2197.

Merz, P. (2001). *Memetic algorithms for combinatorial optimization problems: Fitness landscapes and effective search strategies*.

Mladenović, N., Urošević, D., & Ilić, A. (2012). A general variable neighborhood search for the one-commodity pickup-and-delivery travelling salesman problem. *European Journal of Operational Research, 220*(1), 270–285.

Moscato, P. (1989). *On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms*. Caltech concurrent computation program. C3P Report, 826, 1989.

Moscato, P., & Cotta, C. (2003). A gentle introduction to memetic algorithms. In *Handbook of metaheuristics* (pp. 105–144). US: Springer.

Na, B. (2007). *Heurisic approaches for no-depot k-traveling salesmen problem with a minmax objective* Doctoral dissertation. Texas A&M University.

Okonjo-Adigwe, C. (1988). An effective method of balancing the workload amongst salesmen. *Omega, 16*(2), 159–163.

Polat, O., Kalayci, C. B., Kulak, O., & Günther, H. O. (2015). A perturbation based variable neighborhood search heuristic for solving the vehicle routing problem with simultaneous pickup and delivery with time limit. *European Journal of Operational Research, 242*(2), 369–382.

Reinhelt, G. (2014). *{TSPLIB}: A library of sample instances for the TSP (and related problems) from various sources and of various types*. <http://comopt.ifi.uniheidelberg.de/software/TSPLIB95>.

Sabar, N. R., Turky, A., & Song, A. (2016). A multi-memory multi-population memetic algorithm for dynamic shortest path routing in mobile ad-hoc networks. In *Pacific rim international conference on artificial intelligence* (pp. 406–418). Springer International Publishing.

Sarin, S. C., Sherali, H. D., Judd, J. D., & Tsai, P. F. J. (2014). Multiple asymmetric traveling salesmen problem with and without precedence constraints: Performance comparison of alternative formulations. *Computers & Operations Research, 51*, 64–89.

Serna, M. D. A., & Uran, C. A. S. (2015). A memetic algorithm for the traveling salesman problem. *IEEE Latin America Transactions, 13*(8), 2674–2679.

Sifaleras, A., Konstantaras, I., & Mladenović, N. (2015). Variable neighborhood search for the economic lot sizing problem with product returns and recovery. *International Journal of Production Economics, 160*, 133–143.

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

Singh, A., & Gupta, A. K. (2007). Two heuristics for the one-dimensional bin-packing problem. *Operations Research-Spektrum, 29*(4), 765–781.

Somhom, S., Modares, A., & Enkawa, T. (1999). Competition-based neural network for the multiple travelling salesmen problem with minmax objective. *Computers & Operations Research, 26*(4), 395–407.

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

Tang, L., Liu, J., Rong, A., & Yang, Z. (2000). A multiple traveling salesman problem model for hot rolling scheduling in Shanghai Baoshan Iron & Steel Complex. *European Journal of Operational Research, 124*(2), 267–282.

Turky, A. M., & Abdullah, S. (2014). A multi-population harmony search algorithm with external archive for dynamic optimization problems. *Information Sciences, 272*, 84–95.

Turky, A., Sabar, N. R., & Song, A. (2016). A multi-population memetic algorithm for dynamic shortest path routing in mobile ad-hoc networks. In *Cec 2016- IEEE congress on evolutionary computation*. IEEE.

Vallivaara, I. (2008). A team ant colony optimization algorithm for the multiple travelling salesmen problem with minmax objective. In *Proceedings of the 27th IASTED international conference on modelling, identification and control* (pp. 387–392). ACTA Press.

Venkatesh, P., & Singh, A. (2015). Two metaheuristic approaches for the multiple traveling salesperson problem. *Applied Soft Computing, 26*, 74–89.

Wang, K., Choi, S. H., & Lu, H. (2015). A hybrid estimation of distribution algorithm for simulation-based scheduling in a stochastic permutation flowshop. *Computers & Industrial Engineering, 90*, 186–196.

Wang, Y. T., Li, J. Q., Gao, K. Z., & Pan, Q. K. (2011). Memetic algorithm based on improved inver–over operator and lin-kernighan local search for the euclidean traveling salesman problem. *Computers & Mathematics with Applications, 62*(7), 2743–2754.

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

Zhao, W., Chen, H., & Li, H. (2012). A variable neighborhood search approach for multiple traveling salesman problem with deadlines. In *2012 9th IEEE international conference on networking, sensing and control (ICNSC)* (pp. 301–306). IEEE.

Zhou, Y., Luo, Q., Chen, H., He, A., & Wu, J. (2015). A discrete invasive weed optimization algorithm for solving traveling salesman problem. *Neurocomputing, 151*(3), 1227–1236.