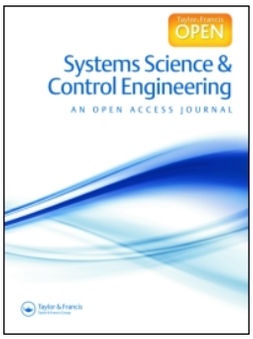

Systems Science & Control Engineering

An Open Access Journal

ISSN: 2164-2583 (Online) Journal homepage: www.tandfonline.com/journals/tssc20

An effective method for solving multiple travelling salesman problem based on NSGA-II

Yang Shuai, Shao Yunfeng & Zhang Kai

To cite this article: Yang Shuai, Shao Yunfeng & Zhang Kai (2019) An effective method for solving multiple travelling salesman problem based on NSGA-II, Systems Science & Control Engineering, 7:2, 108-116, DOI: 10.1080/21642583.2019.1674220

To link to this article: https://doi.org/10.1080/21642583.2019.1674220

SYSTEMS SCIENCE & CONTROL ENGINEERING: AN OPEN ACCESS JOURNAL
2019, VOL. 7, NO. 2, 108–116
https://doi.org/10.1080/21642583.2019.1674220

An effective method for solving multiple travelling salesman problem based on NSGA-II

Yang Shuai[a], Shao Yunfeng[a] and Zhang Kai[a,b]

[a]School of Computer Science and Technology, Wuhan University of Science and Technology, Wuhan, People's Republic of China; [b]Hubei Province Key Laboratory of Intelligent Information Processing and Real-time Industrial System, Wuhan, People's Republic of China

ABSTRACT

In this paper, an effective multi-objective evolutionary algorithm is proposed to solve the multiple travelling salesman problem. In order to obtain minimum total visited distance and minimum range between all salesmen, some novel representation, crossover and mutation operators are designed to enhance the local and global search behaviours, then NSGA-II framework is applied to find well-convergent and well-diversity non-dominated solutions. The proposed algorithm is compared with several state-of-the-art approaches, and the comparison results show the proposed algorithm is effective and efficient to solve the multiple travelling salesman problems.

ARTICLE HISTORY
Received 15 March 2019
Accepted 26 September 2019

KEYWORDS
Multi-objective evolutionary algorithm; multiple travelling salesman problem; NSGA-II

1. Introduction

Multiple Travelling Salesman Problem (MTSP) is an extension of the famous Travelling Salesman Problem (TSP) that visiting each city exactly once with no sub-tours (Gerhard, 1994). MTSP involves assigning *m* salesmen to *n* cities, and each city must be visited by a salesman while requiring a minimum total cost. Lots of real-life problems can be modelled as MTSP, such as printing press scheduling problem (Gorenstein, 1970), distribution of emergence materials problem (Liu & Zhang, 2014), vehicle routing problem (Angel, Caudle, Noonan, & Whinston, 1972), UAVs planning problem (Ann, Kim, & Ahn, 2015) and hot rolling scheduling problem (Tang, Liu, Rong, & Yang, 2000).

As an NP-hard problem, MTSP has always been valued by researchers. Currently, evolutionary computation is the main method for solving the MTSP problem. In the application process of genetic algorithm, researchers have made a series of improvements to the representation of chromosomes to reduce redundant solutions and improve search efficiency. The one chromosome technique is proposed and applied to hot rolling scheduling problems (Tang et al., 2000). The two-chromosome technique is proposed and used to solve Vehicle Scheduling Problem (Malmborg, 1996; Park, 2001). The two-part chromosome technology is proposed which can reduce the algorithm search space effectively (Carter & Ragsdale, 2005). An estimation of distribution algorithm (EDA) with a gradient search is used to solve MOmTSP which objective function is set as the weighted sum of the total travelling costs of all salesmen and the highest travelling cost of any single salesman (Shim, Tan, & Tan, 2012), in which the author considers minimizing the longest cost to balance the workload between salesmen. An effective evolutionary algorithm, reinforced by a post-optimization procedure based on path-relinking (PR), is used to deal with a bi-objective multiple travelling salesman problem with profits (Labadie, Melechovsky, & Prins, 2014).

Meanwhile, other evolutionary algorithms based on swarm intelligence are used to solve the MTSP problem. To minimize the number of mobile nodes in desired wireless sensor network, the routes plan for all mobile nodes is modelled with multiple travel salesman problem and solved with PSO algorithm (Zhang, Chen, Cheng, & Fang, 2011). A new acceleration particle Swarm optimization is constructed to solve the MTSP problem (Qing, Kang, & Marine, 2015), which can effectively overcome the premature convergence. The ant colony optimization (ACO) is applied to the MTSP with ability constraint and the results are encouraging (Pan & Wang, 2006). A multi-objective EA which combine ACO and the multi-objective evolutionary algorithm (EA) based on decomposition (MOEA/D) is proposed to solve MTSP (Ke, Zhang, & Battiti, 2013). Several multi-objective ACSs are proposed to tackle MTSP from a bi-criteria perspective which need to minimize the total cost of travelled subtours while achieving balanced subtours (Necula, Breaban, & Raschip, 2015).

CONTACT Zhang Kai zhangkai@wust.edu.cn
This article has been republished with minor changes. These changes do not impact the academic content of the article.

But in the research on MTSP, most of the problems are treated as a single-objective problem, or considered from two separate perspectives. For the latter, there are usually two objective functions that are used: minimizing the total distance travelled and minimizing the travel distance of the longest traveller. The second objective usually as a condition for balancing the subtours, which involves some problems such as the workload among salesmen and service time of each customer in practical situation. However, considering that reducing only the total distance will result in a height imbalance between each subtour. The result is that one traveller has gone through most of the journey, and other travellers only pass a city near the deport. If we only consider the balance between subtours, this will often make salesman walk unnecessary distances. Therefore, optimizing the total distance and the balance between subtours are two conflicting goals which determines that we cannot consider it separately. In this paper, the difference between the longest route and the shortest route is used to balance the workload between salesman, and the total distance of the salesman is taken as the total cost. Then we use the NSGA-II framework to optimize both targets simultaneously, and improve the crossover operator and mutation operator to get a better non-dominated frontier. Compared with the results of the existing literature, the feasibility of the algorithm is shown. The effect of our algorithm is encouraging.

2. Mathematical model

Based on the number of depots, the MTSP problem can be divided into single-depot multiple TSP (SD-MTSP) and multi-depot multiple TSP (MD-MTSP), the former means that the salesmen start from the same starting depot and the latter means that the salesmen start from different starting depot. This paper mainly explores the two objective SD-MTSP, where the two objective is conflicting. Usually, the problem can be described as follows: there is a set of n number of cities and m number of salesmen, the set is expressed as $C = \{i\}, i = 0, 1, 2, \ldots, n$, and $V = \{k\}, k = 1, 2, \ldots, m$. Each salesman departs from the same depot, takes a tour route and returns to the original starting city. Here we use c_{ij} for the distance between cities i and j, and x_{ijk} for salesman k from cities i to j. Each city will be visited exactly once (except the starting point). Ideally, the total travel distance is minimized while the travel distance between salesmen is as close as possible. The mathematical model is as follows:

$$\text{Mininize}: \ F = (f_1, f_2) \tag{1}$$

$$f_1 = \sum_{k=1}^{m}\sum_{i=0}^{n}\sum_{j=0}^{n} c_{ij}x_{ijk} \tag{2}$$

$$f_2 = \max_{1\le k<\le m}\sum_{i=0}^{n}\sum_{j=0}^{n} c_{ij}x_{ijk} - \min_{1\le k\le m}\sum_{i=0}^{n}\sum_{j=0}^{n} c_{ij}x_{ijk} \tag{3}$$

where

$$x_{ijk} = \begin{cases} 1, & \text{salesman } k \text{ passes from city } i \text{ to city } j \\ 0, & \text{else} \end{cases} \tag{4}$$

$$\text{s.t.} \begin{cases} \sum_{k=1}^{m}\sum_{i=1}^{n} x_{ijk} = 1; \quad \forall j = 1, \ldots, n \\ \sum_{k=1}^{m}\sum_{j=1}^{n} x_{ijk} = 1; \quad \forall i = 1, \ldots, n \\ \sum_{k=1}^{m}\sum_{i=1}^{n} x_{i0k} = m \\ \sum_{k=1}^{m}\sum_{j=1}^{n} x_{ojk} = m \\ \sum_{i\in S}\sum_{j\notin S} x_{ijk} \ge 1; \quad \forall k \in V, \ \forall S \subseteq C. \end{cases} \tag{5}$$

Equations (2) and (3) respectively represent two objective functions: the total distance of the salesman and the difference between the longest route and the shortest route. What we need to do is to minimize both f_1 and f_2. Equation (5) represents the constraint: all salesmen start from the same starting city 1. It is required that, except for the starting city, there is one and only one salesman in each city passing through, and all salesmen return to their starting city. The final solution doesn't generate subtours.

3. Proposed algorithm

3.1. Chromosome representations

In order to reduce the redundancy of solution space, we use two-part chromosome technology to encode the problem. For the sake of convenience, we have made some modifications to the chromosomes. Figure 1 shows a chromosome representing of the MTSP solution (where $n = 10$ and $m = 3$), in which n cities are represented by natural numbers between 0 and $n-1$, and 0 represents the central city. Among them, the chromosome consists of two parts: the first part is the arrangement of $n-1$ natural numbers; the second part is $m-1$ break points, which divides the first part into m groups, each group is represented by a salesperson.

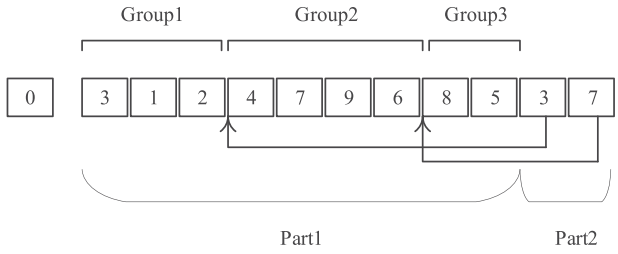

Figure 1. One chromosome for 10 city MTSP with three salespersons.

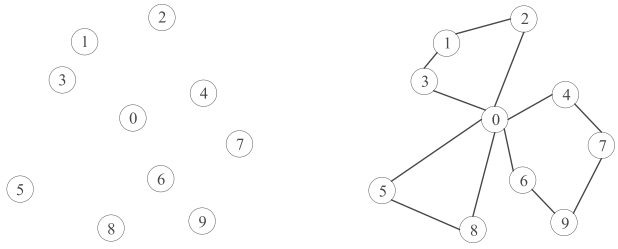

Figure 2. The relative position of the city and one of its solutions.

In order to more intuitively display the route of each traveller represented by the chromosome code, we assume that the relative positions of the 10 cities are as shown in the left of Figure 2, and the solution corresponding to the chromosome in Figure 1 is as shown in the right of Figure 2. From this, the tours of three travellers can be obtained: $0 \rightarrow 3 \rightarrow 1 \rightarrow 2 \rightarrow 0$; $0 \rightarrow 4 \rightarrow 7 \rightarrow 9 \rightarrow 6 \rightarrow 0$; $0 \rightarrow 8 \rightarrow 5 \rightarrow 0$. We use P1 for 31247968537 and P2 for 031204796085, and the process of chromosomes from P1 to P2 is called decoding.

3.2. Crossover operator

The crossover operator in genetic algorithm is designed to simulate the intersection of chromosomes in nature. A good crossover operator is critical to the algorithm's local search capabilities. For the natural number coding form of TSP problem, Partial-Mapped Crossover, Cycle Crossover and Order Crossover are proposed and widely used (Goldberg & Lingle, 1985; Oliver, Smith, & Holland, 1987), but in practical applications, the convergence degree is slow and the effect is not ideal. A crossover operator named HAG for real-coded traveling salesman problem is proposed, which has a good effect in practical applications (Tang, 1999).

In the two-objective problem, which minimizes two functions simultaneously, once the crossover operator has strong local search ability on a target, we control the evolution direction of the population through the objective function, then the Pareto front can be as close as possible to the coordinate origin. This is the main idea of the crossover operator design in this paper. The crossover operators used in this paper are improved on HGA to make it suitable for MTSP, and the main process of the crossover is shown in Algorithm 1.

Algorithm 1: The core method of crossover.

```
Input: PA, PB, MARK //PA and PB are arrays, MARK
       is search direction
Output: RESULT
len = PA.length;
k = Random(1, len);
RESULT=[k];
while len > 1 do
    if MARK == latter then
        x = PA.latterCity(k);
        y = PB.latterCity(k);
    else
        x = PA.formerCity(k);
        y = PB.formerCity(k);
    end
    delete k in PA and PB;
    dx = distanceOfTwoCity(k, x);
    dy = distanceOfTwoCity(k, y);
    if dx<dy then
        k = x;
    else
        k = y;
    end
    RESULT = RESULT.push(k);
    len = PA.length;
end
```

In Algorithm 1, PA and PB represent two parents, and the children are generated in different search directions according to the input *MARK* value. If we take the input PA as 123456789, and k is set as 5. When the value of *MARK* is *latter*, the search direction is from front to back in the PA. At this time, the next city adjacent to the k in the PA is found by the *latterCity*() function, i.e. 6. When the value of *MARK* is *former*, the search direction is from back to front in the PA. In this case, the previous city in the PA adjacent to k is found by the *formerCity*() function, i.e. 3. The same applies to PB. The role of *distanceOfTwoCity*() is to find the Euclidean distance between the cities represented by the two parameters.

Through experiments, we found that the derived children have different performances on the two objective functions for different inputs of PA and PB. Under the chromosome representation of this paper, if we take the part1 of the parent chromosome as input, most of the results obtained will have a better balance between

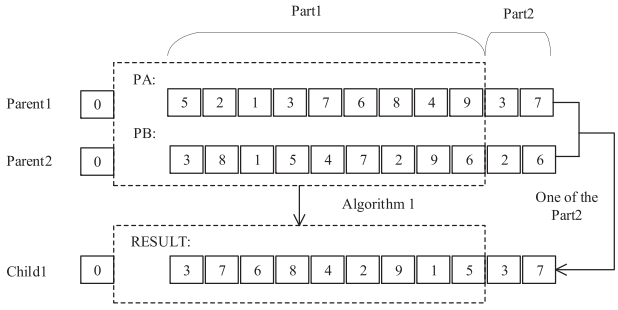

Figure 3. The formation process of child1.

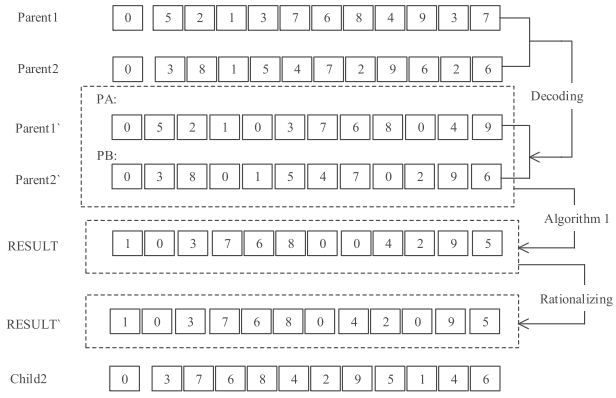

Figure 4. The formation process of child2.

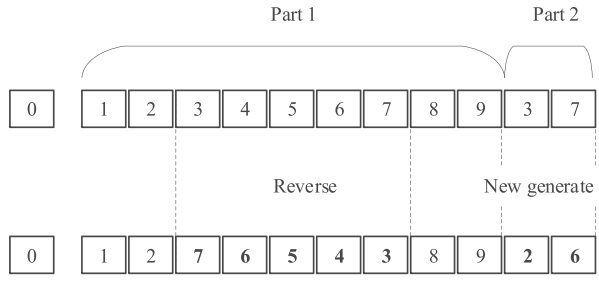

Figure 5. Reversal mutation.

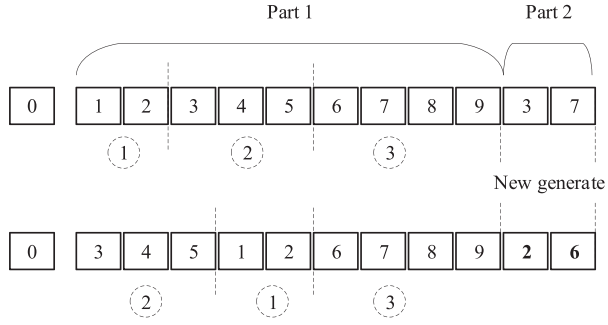

Figure 6. Fragment transposition variation.

salesmen, but it is not so good in total distance. If we take the decoded parents as input and finally rationalize the result, most of the results obtained will have a better performance in total distance, but it is not ideal in balance. The reason for this phenomenon is that HGA-based crossover operators have a certain bias in the goal of minimizing the total distance. This bias is stronger when using decoded chromosomes as input. So, in order to obtain a well-diversity set of solutions, we combine the above two methods as the final crossover operation.

The crossover operation used in our paper is called combined HGA, which generates descendants in two different ways. As is shown in Figure 3, the part1 of the parents is used as the input to algorithm 1, and the output is taken as part1 of child1, then part2 of one of the parents is taken as part2 of child1. For child2, we use the decoded parents as the input of algorithm 1, and then the output is subjected to the removal of adjacent zeros and the random generation of breakpoints, so that the result is rationalized. The whole process is shown in Figure 4.

3.3. Mutation operator

The main function of the mutation operator is to prevent the search process from falling into local optimum. Two mutations are designed for the coding method of chromosomes. During the algorithm, different mutations are used with a specific probability. Two mutation operators are designed to obtain more chromosomal variant forms and combinations, so that the search process can have a greater probability of jumping out of the local optimum.

Method 1: In the part 1, two numbers i and $j(i < j)$ are randomly generated, and the order of the genes between the two points i and $j+1$ is inverted. In the part 2, randomly generate new $m-1$ split points, as shown in Figure 5.

Method 2: In the part 1, two numbers i and $j(i < j)$ are randomly generated, and these two points divide the chromosome into three parts: 1, 2 and 3, then the new chromosomes are combined in the order of 2, 1 and 3. In the part 2, a new $m-1$ points are randomly generated, as shown in Figure 6.

3.4. Overall algorithm

In the framework of NSGA-II (Deb, Pratap, Agarwal, & Meyarivan, 2002), the process of producing new populations is shown in Figure 7. There are two parts in this process that involve selection operators.

The first part is the generation of Qt through genetic algorithm. Here the binary tournament selection method is adopted: a population arrangement is randomly generated, and the better individual are selected from

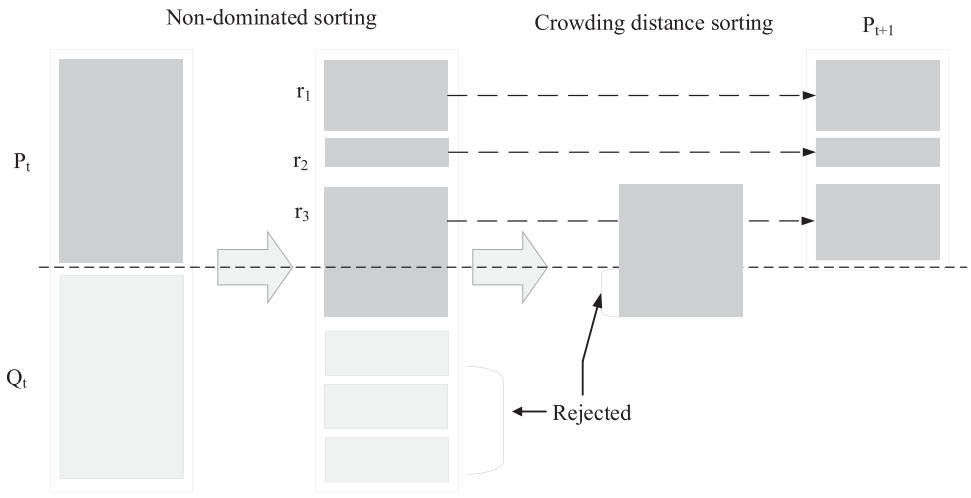

Figure 7. Composition of new population.

two adjacent individuals, in which the standard for this comparison is the non-dominated level and the crowding distance. Executing twice in the above method, and then a descendant population of the same size as the original population will be obtained.

The second part is the generation of P_{t+1}: combining P_t and Q_t, and performing non-dominated sorting to generate boundary set $R = \{r_1, r_2, r_3, \ldots\}$; the boundary set is sequentially merged into P_{t+1} until $|P_{t+1} \cup r_{i+1}|$ is greater than $|P_t|$; calculating the crowding distance for individuals in r_{i+1} and sort by this, and then adding individuals to P_{t+1} until $|P_{t+1}|$ is equal to $|P_t|$.

To show the specific steps of our algorithm, we present detailed pseudo-code. In Algorithm 2, the P_0 is a random initial parent population, N is the number of iterations and MP is the probability of mutation. The population P_i in the ith iteration generates a new population Q_i through *binaryTournamentSelection*(), *crossover*() and mutation operation, in which the three operations of the genetic algorithm have been described in detail above. The *fastNonDominatedSort*() and the *crowdingDistanceAssignment*() are two important operations in the NSGA-II. At the first operation, a combined population R_i that is included all previous and current population members is sorted by nondomination, which is used to ensure the convergence of the algorithm. For the second operation, the crowding-distance is used to measure the fitness of the solution on the same non-dominated set, which is used to maintain the diversity of the algorithm.

Algorithm 2: The whole process of our algorithm.

Input: P_0, N, MP
Output: P_N
$i = 0$;
while $i < N$ **do**
  $CROSSOVER\_SET = binaryTournamentSelection(P_i)$;
  $Q_i = crossover(CROSSOVER\_SET)$;
  for $k = 0$ *to* $Q_i.length$ **do**
    p_k is the kth chromosome in Q_i;
    if *Random(0,1)* $\leq MP$ **then**
      if *Random(0,1)* ≤ 0.5 **then**
        $mutation1(p_k)$;
      else
        $mutation2(p_k)$;
      end
    end
  end
  $R_i = P_i \bigcup Q_i$;
  $F = fastNonDominatedSort(R_i) // F = (F_1, F_2, \cdots, F_j, \cdots)$;
  $P_{i+1} = \phi$ and $j = 0$;
  while $P_{i+1}.length + F_j.length \leq P_0.length$ **do**
    $P_{i+1} = P_{i+1} \bigcup F_j$;
    $j = j + 1$;
  end
  $crowdingDistanceAssignment(F_j)$;
  $sort(F_j)$ //sort by crowding_distance;
  $P_{i+1} = P_{i+1} \bigcup F_j[0 : (P_0.length - P_{i+1}.length)]$;
  $i = i + 1$;
end

4. Results and discussion

Regarding the multi-objective MTSP problem, there are currently no benchmark examples like TSPLIB that can be used. Most of the existing work is done on the TSP benchmark which the first city is set as depot, where the number of salesman is predetermined. In Necula et al. (2015),

Table 1. Computational test conditions.

Instance	Number of cities	Number of salesmen
eil51	51	7
berlin52	52	5
berlin52	52	7
eil76	76	3
eil76	76	7
rat99	99	7

the author selected four examples based on the location and distance of the central city relative to other cities, and simultaneously optimizes the total distance and the "amplitude" of the subtours. To verify the feasibility of our algorithm, this paper selects the same example to conduct experiments, as shown in Table 1.

The proposed algorithm has been implemented in JavaScript on a CPU Intel Core i5-7500 with 3.40 GHz and 8 GB of RAM. The scale of the population is set as 100 and the probability of mutation is set as 0.05. Because our algorithm is based on the random arrangement of the population, and the binary tournament selection is used, here is no need to set the crossover rate. In order to compare with the existing experimental results, we run 10 times on each instance, and then perform non-dominated sorting on the obtained 10 solution sets

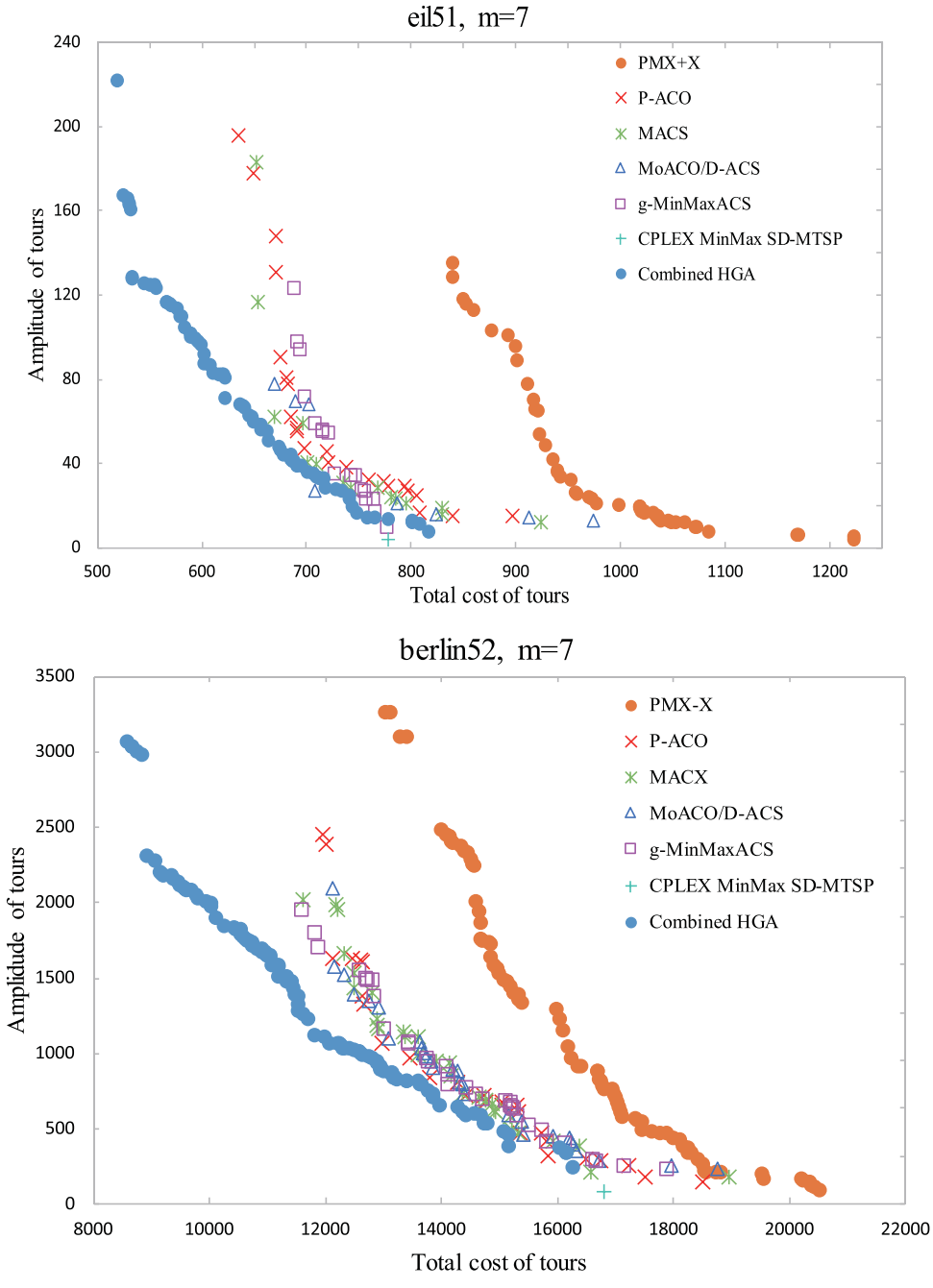

Figure 8. Non-dominated fronts for eil51-m7 and berlin52-m7 instances.

and take the non-dominated front as the final solution. For 51 cities and 52 cities, we set the number of iterations to 1400. For the instances of 76 cities and 99 cities, we set the number of iterations to 1800 and 2200, respectively.

In order to show the improvement effect of the algorithm, we also present another set of experimental data, in which the crossover operator we make the following changes: the part1 of the parents uses PMX operation to generate two part1 of the child, and part2 of child1 retains the part2 of the one of the parents, the part2 of child2 is randomly generated, and the rest of the algorithm remains unchanged. We will call it PMX+X later.

In the literature (Necula et al., 2015), they have investigated three multi-objective ACO-based algorithms and a single-objective ACO algorithm with the objective of minimizing the longest tour. Where P-ACO belongs to an algorithm with multiple pheromone trail matrices and one single heuristic matrix. Conversely, MACS belongs to the algorithms which use a single pheromone trail matrix and several heuristic matrices. The third is multi-objective ant colony optimization based on decomposition in

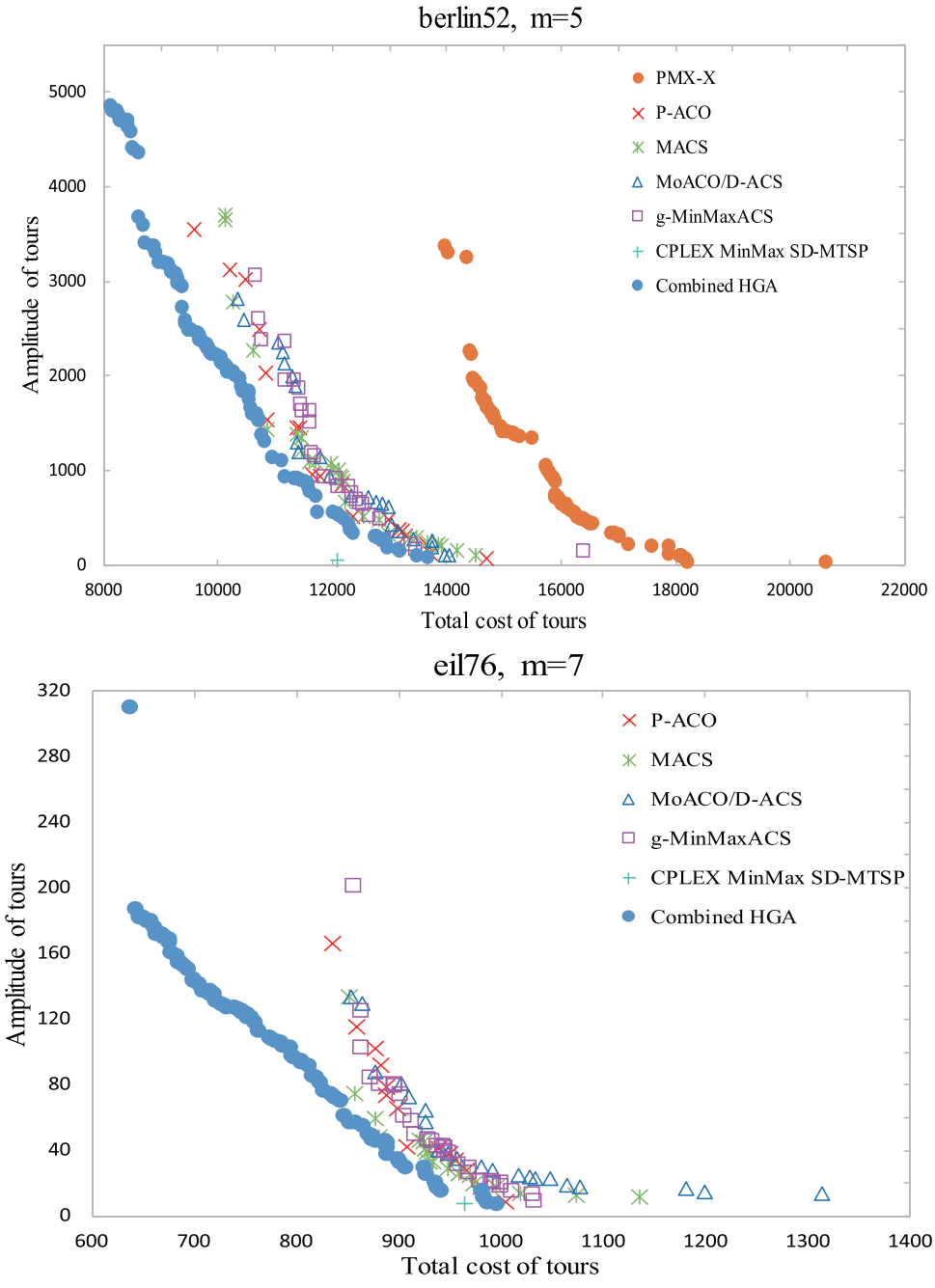

Figure 9. Non-dominated fronts for berlin52-m5 and eil76-m7 instances.

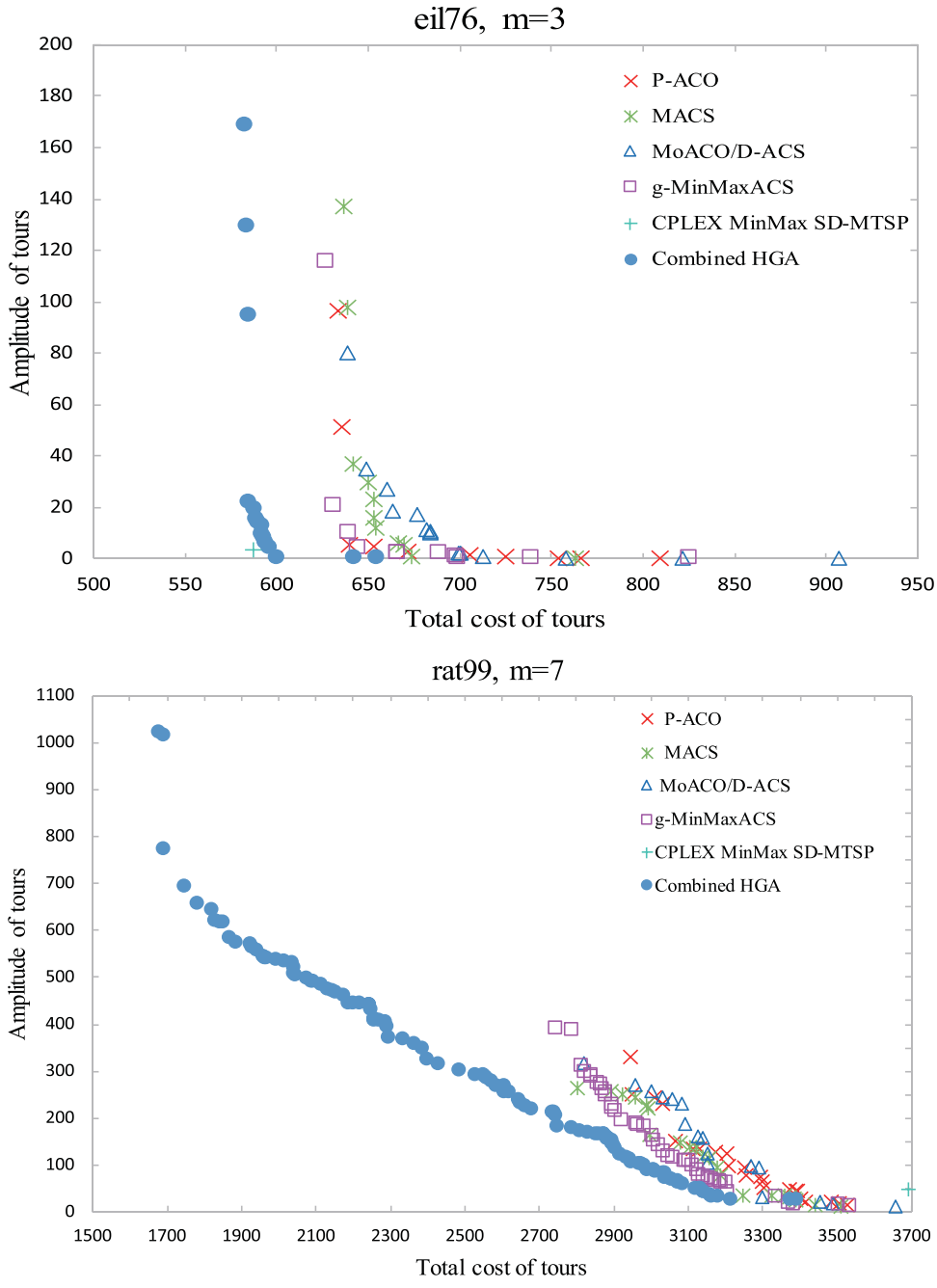

Figure 10. Non-dominated fronts for eil76-m3 and rat99-m7 instances.

combination with ACS which is called MoACO/D-ACS. As for g-MinMaxACS, it treats the two-objective problem with a compacted method, through resorting to the Min-Max approach. At last, CPLEX MinMax SD-MTSP are also given in the literature, which indicates that the CPLEX optimizer is used to optimize the longest subtour. Here we put our own experimental results on the same axis as the non-dominated solution set obtained by the above method.

In all the following figures, the horizontal axis represents the total distance of tours and the vertical axis represents the difference between the longest subtour and the shortest subtour. In Figure 8, it is obvious that the non-dominated front obtained by combined HGA is better than the non-dominated front edge obtained by PMX-X. The former solution tends to have less total distance and lower difference. Compared with the other four methods, the solution from our algorithm can dominate most of the solution obtained by other methods, especially the instance eil51-m7. In berlin52-m7, our result is to show a clear advantage in the objective of minimizing the total cost, but not very good in minimizing the balance.

As shown in berlin52-m5, combined HGA still performs better than PMX-X at the same number of iterations. In

eil76-m7, the result of the algorithm with PMX-X has a relatively poor performance, so we don't show it on the chart. From both two instances in Figure 9, it is obviously that our algorithm can get better non-dominated front.

In Figure 10, our algorithm also gets a better non-dominated front. Especially in the rat99-m7 example, our non-dominated solution set reflects a good diversity. Since our crossover operator is based on the shortest distance, our results do not perform well in the extreme case of minimizing the balance in some instances.

In general, the non-dominated front derived from our algorithm is closer to the origin than the non-dominated solution set obtained by any of the other four methods, the results we obtain simultaneously are of good diversity and can be provided to decision makers more choices.

5. Conclusion

This paper proposed an effective method based on NSGA-II to solve the multiple traveling salesman problem. The total travel distance and the difference between the longest subtour and the shortest one are the two conflict objectives. A novel crossover operator is designed to generate new offspring, and two mutation operators are proposed so that the search process can jump out of the local optimum. Our algorithm is tested with several MTSP benchmark instances and compared with several state-of-the-art approaches. The comparison result shows the effectively and efficiency of our algorithm.

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

This work was supported by the National Natural Science Foundation of China (Grant No. 61472293).

References

Angel, R. D., Caudle, W. L., Noonan, R., & Whinston, A (1972). Computer-assisted school bus scheduling. *Management Science*, *18*(6), B-279–B-288.

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.

Carter, A. E., & Ragsdale, C. T (2005). A new approach to solving the multiple traveling salesperson problem using genetic algorithms. *European Journal of Operational Research*, *175*(1), 246–257.

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. A. M. T (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, *6*(2), 182–197.

Gerhard, R (1994). The traveling salesman: Computational solutions for TSP applications. *Lecture Notes in Computer Science*, *840*, 1–223.

Goldberg, D. E., & Lingle, R. (1985). Alleles, loci, and the traveling salesman problem. *Proceedings of an international conference on genetic algorithms and their applications* (Vol. 154, pp. 154–159). Hillsdale, NJ: Lawrence Erlbaum.

Gorenstein, S (1970). Printing press scheduling for multi-edition periodicals. *Management Science*, *16*(6), B-373–B-383.

Ke, L., Zhang, Q., & Battiti, R (2013). Moea/d-aco: A multiobjective evolutionary algorithm using decomposition and antcolony. *IEEE Transactions on Cybernetics*, *43*(6), 1845–1859.

Labadie, N., Melechovsky, J., & Prins, C. (2014). An evolutionary algorithm with path relinking for a bi-objective multiple traveling salesman problem with profits. In *Applications of Multi-Criteria and Game Theory Approaches* (pp. 195–223). London: Springer.

Liu, M., & Zhang, P (2014). New hybrid genetic algorithm for solving the multiple traveling salesman problem: An example of distribution of emergence materials. *Journal of Systems & Management*, *23*(2), 247–254.

Malmborg, C. J (1996). A genetic algorithm for service level based vehicle scheduling. *European Journal of Operational Research*, *93*(1), 121–134.

Necula, R., Breaban, M., & Raschip, M. (2015). Tackling the bi-criteria facet of multiple traveling salesman problem with ant colony systems. *2015 IEEE 27th International Conference on Tools with Artificial Intelligence (ICTAI)* (pp. 873–880). Vietri sul Mare, Italy: IEEE.

Oliver, I. M., Smith, D. J., & Holland, J. R. C. (1987). A study of permutation crossover operators on the traveling salesman problem. *Proceedings of the second international conference on genetic algorithms on genetic algorithms and their application* (pp.224–230). Hillsdale, NJ, USA: L. Erlbaum Associates Inc.

Pan, J., & Wang, D. (2006). An ant colony optimization algorithm for multiple travelling salesman problem. *International conference on innovative computing* (Vol. 1, 210–213). Washington, DC, USA: IEEE Computer Society.

Park, Y. B. (2001). A hybrid genetic algorithm for the vehicle scheduling problem with due times and time deadlines. *International Journal of Production Economics*, *73*(2), 175–188.

Qing, N., Kang, F., & Marine, S. O. (2015). Application of a new acceleration particle swarm optimization for solving multiple traveling salesman problems. *Journal of Shaanxi Normal University*, *43*(6), 7.

Shim, V. A., Tan, K. C., & Tan, K. K. (2012). A hybrid estimation of distribution algorithm for solving the multi-objective multiple traveling salesman problem. *IEEE congress on evolutionary computation* (pp. 1–8). Brisbane, QLD, Australia: IEEE.

Tang, L (1999). Improved genetic algorithms for TSP. *Journal of Northeastern University*, *1*, 40–43.

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.

Zhang, Z., Chen, Y., Cheng, H., & Fang, Q. (2011). MTSP based solution for minimum mobile node number problem in sweep converge of wireless sensor network. *Proceedings of 2011 international conference on computer science and network technology* (Vol. 3(8), pp. 1827–1830). Harbin, China: IEEE.