APEM journal
Advances in Production Engineering & Management
Volume 16 | Number 2 | June 2021 | pp 173–184
https://doi.org/10.14743/apem2021.2.392
ISSN 1854-6250
Journal home: apem-journal.org
Original scientific paper

Improved Genetic Algorithm (VNS-GA) using polar coordinate classification for workload balanced multiple Traveling Salesman Problem (mTSP)

Wang, Y.D.[a], **Lu, X.C.**[b,*], **Shen, J.R.**[c]

[a]Beijing Jiaotong University, Shangyuan Village, Haidian District, Beijing, P.R. China
[b]Beijing Jiaotong University, Shangyuan Village, Haidian District, Beijing, P.R. China
[c]Beijing Capital Agribusiness & Food Group Co., Ltd., Xicheng District, Beijing, P.R. China

ABSTRACT

The multiple traveling salesman problem (mTSP) is an extension of the traveling salesman problem (TSP), which has wider applications in real life than the traveling salesman problem such as transportation and delivery, task allocation, etc. In this paper, an improved genetic algorithm (VNS-GA) that uses polar coordinate classification to generate the initial solutions is proposed. It integrates the variable neighbourhood algorithm to solve the multiple objective optimization of the mTSP with workload balance. Aiming to workload balance, the first design of this paper is about generating initial solutions based on the polar coordinate classification. Then a distance comparison insertion operator is designed as a neighbourhood action for allocating paths in a targeted manner. Finally, the neighbourhood descent process in the variable neighbourhood algorithm is fused into the genetic algorithm for the expansion of search space. The improved algorithm is tested on the TSPLIB standard data set and compared with other genetic algorithms. The results show that the improved genetic algorithm can increase computational efficiency and obtain a better solution for workload balance and this algorithm has wild applications in real life such as multiple robots task allocation, school bus routing problem and other optimization problems.

ARTICLE INFO

Keywords:
Multiple traveling salesman problem (mTSP);
Workload balance;
Variable neighbourhood search algorithm (VNS);
Genetic algorithm (GA);
Polar coordinates;
Classification

**Corresponding author:*
xclu@bjtu.edu.cn
(Lu, X.C.)

Article history:
Received 4 June 2021
Revised 15 June 2021
Accepted 17 June 2021

1. Introduction

Multiple Traveling Salesmen Problem (mTSP) is an extension of the classic Traveling Salesman Problem (TSP)[1]. The mTSP with workload balance is different from the basic TSP. In some scenarios, the balanced distribution of workload is an important consideration. For example, school bus routing problem, route arrangement problem, postman distribution problem, express delivery area division, etc., can all be abstracted as mTSP with balanced workload. Omar *et al*. [2] proposed that mTSP can be used to solve vehicles, robots, and UAVs routing problems. In single-objective optimization of mTSP, the total cost is the most commonly used optimization objective. It causes the problem of an unbalanced distribution of distance among traveling salesmen, which is directly reflected in the path as the distance traveled by one traveler is longer than other travelers. This situation is hoped to be improved in some industries in reality because the

unbalanced workload distribution results in operational difficulties. Moreover, avoiding crossing paths with one another is the most significant problem in practical work. For instance, in school bus routing problem, if a certain route is too long during planning school bus route, it may cause the first arrival point to arrive too early, which will affect students' study and other performance. In postmen routing problem, avoiding crossing paths can improve the efficiency of work. Therefore, our main focus is on the mTSP with workload balance. To solve this problem, we build a multiple objective optimization model and propose an improved genetic algorithm that aims to minimize the total cost and balancing the workload in this paper.

2. Literature review

Research on TSP and mTSP is of long historical making. The original solution to this type of problem mainly used accurate algorithms. Ali *et al*. [3] used the branch and bound method to solve the symmetrical mTSP with 59 cities. Gavish *et al*. [4] described that limiting the lower bound of the branch improved the branch-and-bound method, which can solve the mTSP with 10 traveling salesmen and 100 cities. However, as the scale of the problem expands, the scale of the solution space also increases exponentially, making it difficult to use accurate algorithms. Therefore, heuristic algorithms have become the choice of more and more researchers.

Russell [5] was the first one to apply the heuristic algorithm to solve the mTSP, which transformed mTSP into TSP and then solved it with the Lin-Kernighan algorithm. It also obtained a more accurate approximate solution. For solving the mTSP with the shortest total distance and the shortest sub-path, Carter *et al*. [6] designed a two-stage chromosome coding scheme and designed a genetic operator to improve the solution speed. Singh *et al*. [7] used the invading weed algorithm to obtain a comparative optimal solution. Zhou and Li [8] used a mutation operator combined with a 2-opt search operator to improve the genetic algorithm, avoiding the premature phenomenon of genetic algorithm, and used simulation methods to verify the effectiveness of the algorithm. Xu *et al*. [9] also combined the 2-opt operator with a genetic algorithm to solve the mTSP. Hu *et al*. [10] proposed an improved genetic algorithm that integrates the reproduction mechanism of the weed algorithm and the locally optimized mutation operator to solve the mTSP with balanced workload. Guo [11] used a genetic algorithm to analyze and verify the design of chromosome coding scheme for the genetic algorithm to solve mTSP. Lu *et al*. [12] adopted a two-stage algorithm to solve mTSP while avoiding the path-crossing problem between traveling salesmen. Bostanci *et al*. [13] used the integer linear programming method to solve the problem of finding the shortest route on networks with GIS. This study provided a new method to solve the optimization problems, however, only TSP was solved by this method, problems more complicated such as mTSP are difficult to find the optimal solution.

Recently, with the rise of various heuristic algorithms and machine learning, in addition to genetic algorithms, ant colony optimization algorithm (ACO) [14], artificial bee colony algorithm (ABC)[15], tabu search algorithm (TS)[16] and other heuristic algorithms have also been tried to solve mTSP. Song [17] used the simulated annealing algorithm (SA) to solve the problem of three traveling salesmen in 400 cities, but the algorithm took a long time. Hu [18] constructed an architecture composed of a shared graph neural network and a distributed strategy network to generate an approximate optimal solution for mTSP and used reinforcement learning to train the model, and this method shows good results in large-scale examples. Justus [19] abstracted the problems in the path of staff guiding tourists during the Mecca pilgrimage as multiple objective mTSP with time window and proposed an interactive method for providing a solution. Aiming the mTSP under different goals, Liu *et al*. [20] proposed the ForestTraversal algorithm and the Retrace algorithm to solve the maximum-minimum mTSP and the mTSP that minimizes travel costs, respectively.

Genetic algorithm (GA) is a heuristic algorithm that is often used when solving mTSP. This method mainly focuses on coding design [5] and how to avoid genetic algorithms prematurely falling into local optimality. The variable neighbourhood search algorithm (VNS) has the ability to expand the search range by systematically changing the neighbourhood structure. It guarantees the local search capability while ensuring solutions for having good diversity [21]. There-

fore, this paper designs a genetic algorithm based on polar coordinates to quickly generate high-quality initial solutions, and combines VNS to retain the characteristics of neighbourhood diversity, designs different neighbourhood actions to improve the algorithm's search ability to solve mTSP with workload balance.

3. Mathematical model and problem solving

This paper proposes an improved genetic algorithm, which is improved by fusing the variable neighbourhood algorithm (Variable neighbourhood search genetic algorithm), and VNS-GA is referred too, as an abbreviation in the text.

3.1 Problem description and mathematical model

The mTSP studied in this paper can be described as given n cities, m traveling salesmen and the distance matrix between nodes $D_{ij} = (d_{ij})_{n\cdot n}$, travelers start from the same node (source node), visit a certain number of nodes and then return to the source node. All nodes are required to be accessed, and the rest of the nodes except the source node can only be accessed once. The aim is to find a Hamiltonian circuit that makes total distance shortest and workload balance. Assuming the problem is symmetrical mTSP, that is, $d_{ij} = d_{ji}$, the multiple objective optimization model can be described as follows:

$$Min \sum_{i=0}^{n} \sum_{j=0}^{n} d_{ij} x_{ijk} \quad (k = 1,2, \dots, m) \tag{1}$$

$$Min\, J = \max\left(d_{ij} \times x_{ijk}\right) - \min\left(d_{ij} \times x_{ijk}\right), i, j \in A, k \in T \tag{2}$$

$$\sum_{i=0}^{n} x_{ijk} = y_{kj} \ \forall j = 0,1, \dots, n; k = 1,2, \dots, m \tag{3}$$

$$\sum_{j=0}^{n} x_{ijk} = y_{ki} \ \forall i = 0,1, \dots, n\,; k = 1,2, \dots, m \tag{4}$$

$$\sum_{k=1}^{m} y_{ki} = \begin{cases} 1, i = 1,2, \dots, n \\ m\,, \qquad i = 0 \end{cases} \tag{5}$$

$$X = (x_{ijk}) \in S \tag{6}$$

$$S = \{x_{ijk} | \sum_{i \in Q} \sum_{j \notin Q} x_{ijk} \geq 1, Q \subset \{0,1,2, \dots, n\}, \forall k = 1,2, \dots, m\} \tag{7}$$

$$x_{ijk} = \begin{cases} 1 & \text{traveler } k \text{ went through arc } ij \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

$$y_{ki} = \begin{cases} 1 & \text{traveler } k \text{ visited node } i \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

Where $V = \{0,1,2,...,n\}$ is the node set, n is the number of nodes; $T = \{1,2,...,m\}$ is the travelers set, m is the traveling salesman number; $A = \{(i, j, k) | i, j = 0,1,2,...,m, i \neq j, \forall k = 1,2,...,m\}$ is the arc set, which represents the set of routes that the travelers may pass; $D = \{d_{ij} | i, j \in A\}$ is the distance matrix, representing the distance from node i to node j.

Eqs. 1 and 2 are the objectives of these functions. Eq. 1 minimizes the total path distance; Eq. 2 minimizes the difference between the longest path and the shortest path in m paths. Eq. 3 means that for any downstream node, only one upstream node is allowed for its connection, and Eq. 4 means that in each upstream node, only one downstream node is allowed to be connected; Eq. 5 means that except for the source node, all other nodes are only visited once, and all travelers start from the source node. In Eq. 6, S is the branch elimination constraint, and Eq. 7 is an expression of S [22], which means that for any node in the travelers' paths, any of its subsets must be connected to the other subsets in the solution.

3.2 Fitness function

In multiple objective optimization problems, the goals often conflict with each other. The most common method is the generation method (including weighting method, constraint method, etc.), interactive method, and hybrid method to solve multiple objectives. The weighting method assigns different weights to multiple goals which can convert the multiple objective optimization problem into a single-objective optimization problem and makes the simple and easy implementation of the problem. Therefore, the weighting method is used in this paper to transform the two objectives optimization problem into a single-objective optimization problem.

$$Min\ Z = w_1 \times Z_1 + w_2 \times J \tag{10}$$

Where $z_1 = \sum_{i=0}^{n} \sum_{j=0}^{n} d_{ij} x_{ijk}$, $k = 1,2, \ldots, m$. w_1, w_2 are the weight coefficients, which belong to hyperparameters and the values of coefficients vary with the data set. It can be determined by expert evaluation method or comparison method, etc.
Therefore, the fitness function in the genetic algorithm is:

$$F = 1/Z \tag{11}$$

It can be seen from Eq. 11 that the greater the fitness, the stronger the individual adaptability, and the better the represented feasible solution.

3.3 Chromosome coding

In the genetic algorithm, the design of the chromosome encoding needs to reflect the genetic characteristics of the individual and should be easy to operate. To express the path more effectively, a one-piece decimal coding scheme is adopted in this paper. A chromosome should include at least one node except the source node, as shown below: Take ten nodes and three travelers as an example, use 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 as it respectively represents ten nodes, of which 0 is the starting node of the three travelers. Then the path of a traveler can be expressed as: 0-1-2-3-4-5-6-7-8-9.

To represent multiple travelers, virtual nodes 10 and 11 are inserted to represent the starting node of the route, which is also the source node. Then, the new chromosome can be formed as: 0-1-2-3-4-5-10-6-7-11-8-9.

Taking virtual nodes 10 and 11 as the path dividing nodes, then the three paths can be expressed as: (1) 0-1-2-3-4-5-0, (2) 0-6-7-0, (3) 0-8-9-0.

3.4 Polar coordinate classification method initialize population

The quality of the initial population plays an important role in the intelligent group optimization algorithm. For mTSP with a single starting node and closed cycle, it is a common optimization method to firstly use the clustering method to optimize grouping, and then use a heuristic algorithm to optimize the order within the group. In 2019, Min *et al*. [23] proposed a method using MRISA (multi-restart-iteration sweeping) and tabu search to solve vehicle routing problem. This two-stage algorithm also can be described as clustering by distance in polar coordinates and optimization by tabu search. Then in the same year, they improved this algorithm by adding an adjust operation named put in & put out. This improved algorithm has three stages: cluster the customer points, use put in & put out operations to adjust the load demand and use tabu search to solve the TSP [24]. Traditional clustering algorithms such as K-means clustering or fuzzy C-means clustering and other clustering methods based on centroids (although they can quickly cluster the set of nodes that need to be visited) can't well represent the starting point as the origin where each traveler's sub-path is distributed around the same starting point. Therefore, a classification method based on polar coordinates is proposed in this paper.

The specific operation process of the polar coordinate classification algorithm to generate the initial population is as follows. First, establish a polar coordinate system with the starting node as the origin and map node-set (*n* nodes) into the polar coordinate system then sort each node by the angle in the polar coordinate system. Then calculate the two nodes with the furthest angular distances and select one of them to connect with the origin to establish a new polar coor-

dinate system axis. After that use the new axis as the starting position to rotate clockwise, and use the traveler number (m travelers) to equally divide the number of nodes that the travelers need to visit. The nodes in the area swept during the rotation are the node-set to be visited by a traveler, and m initial paths (that is, every traveler roughly needs to visit n/m nodes) constitute an initial feasible solution. Finally, using chromosome fragment flipping as neighborhood shaking to generate enough chromosomes as the initial population. The process of the polar coordinate classification algorithm is as shown in Algorithm1:

Algorithm1: Polar Coordinate initialize population

Input: The number of population p, the number of salesmen n, cartesian coordinates of the cities (x_i,y_i) and the number of cities m
Output: Initialized population
$k = 1$
(θ_i, d_i): change cartesian coordinates (x_i, y_i) to polar coordinates and sort the cities by the angle θ_i
$max(\theta, d)$: calculate the maximum angle difference among the cities and build new polar coordinate by joining the origin city and one of maximum angle difference cities
s_0: generate a solution by spinning clockwise and each salesman visits n/m cities
repeat
  s: neighborhood shaking s_0
  $k = k + 1$
until
  $k > p$
end

3.5 Neighbourhood actions in VNS

Cross-recombination and mutation operators are often used in genetic algorithm to generate new populations. To enable the algorithm in finding a more efficient combination that makes the travelers' workload roughly balanced, the distance comparison insertion operator cooperates with cross-recombination (neighbourhood action 2) and mutation operater (neighbourhood action 3) is designed in this paper to produce the next generation more efficiently.

Neighbourhood action 1: Distance comparison operator to find a better solution for making the workload balance faster. The neighbourhood action adopted in this paper is to randomly select a node from the sub-path with the longest distance and insert it into the sub-path with the shortest distance. As shown in Fig. 1: Assume that path 0-1-2-3-4-5-0 has the longest distance among all sub-paths, and 0-6-7-0 is the path with the shortest distance among all sub-paths; In the domain action, randomly select a node in the longest path (select node 3) and insert it into the shortest path to form a new chromosome. If the fitness of the new chromosome increases after inserting the selected node, the new chromosome will then be retained. Otherwise, the paternal chromosome will be retained and this inferior solution will be recorded to prevent duplication.

If the fitness of the new chromosome increases after inserting the selected node, suggests that the new offspring chromosomes generated after the neighbourhood action are: 0-1-2-4-5-10-6-3-7-11-8-9.

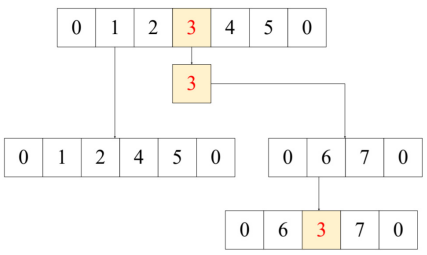

Fig. 1 Schematic of gene insertion in neighbourhood action 1

To avoid repeated calculations, only the difference between the distance of the sequence is calculated each time in the calculation process. Assuming that the selected node is the node at the *i*-th position in the *n*-th path, and the insertion position is the node at the *j*-th position in the *m*-th path, then actual distance to be calculated is $d = d_n - (d_{i-1,i} + d_{i,i+1}) + d_{i-1,i+1} - (d_m + d_{j-1,j} + d_{j,j+1} - d_{j-1,j+1})$. Where d_n, d_m are the path length before path *n* and *m* to perform the neighbourhood action, $i - 1$ and $i + 1$ are the previous and next nodes of *i* in path *n*, the same as $j - 1$, and $j + 1$ is the previous and next nodes of *j* in path *m*, respectively. $d \geq 0$ means that the fitness is not increased after performing the neighbourhood action, otherwise, it also means that the fitness is increased, and the current solution is better than the parent solution.

Neighbourhood action 2: local crossover operator. Neighbourhood action 1 can make the offspring develop in the direction of workload balance, but it is easy to fall into the local optimum. Therefore, an improved crossover operator to avoid the offspring from falling into the local optimum is proposed in this paper.

To avoid repeated fragments or missing fragments in the offspring that are generated after the crossover and for quickly completing the crossover, we first randomly select a sample gene fragment from the parent 2 and inserts it into the offspring at the original position. Then traverses parent 1 to find the genes that are different from the sample gene fragment and inserted them into the offspring in sequence, the genes that are the same as the sample gene fragment is skipped. The crossover process is shown in Fig. 2.

Neighbourhood action 3: mutation operator. Judging from the characteristics of the traveling salesman problem, there should be no cross edges in the route of the optimal solution. Based on this feature, as shown in Fig. 3 (*a* is before mutation, *b* is after mutation), the destruction & reconstruction method is used to mutate individuals to reduce crossover between paths. Suppose the path set is $S = \{s_1, s_2,...,s_i,...,s_k,...,s_n\}$, then sort the distance matrix by distance to get sorted node pairs and randomly selecting a node *p* in a path s_i and interchange it with the nearest neighbour node *m* in the nearest neighbour route s_k to reduce the intersection between paths. If the fitness increases, update the current solution; otherwise, it will not update and record the node pair to node pair distance matrix for avoiding repeated exchanges.

G1	0	6	3	8	10	4	5	11	7	9	1	2
G2	0	1	2	3	4	10	5	6	7	11	8	9
New	0	3	8	11	4	10	5	6	7	9	1	2

Fig. 2 Cross process diagram

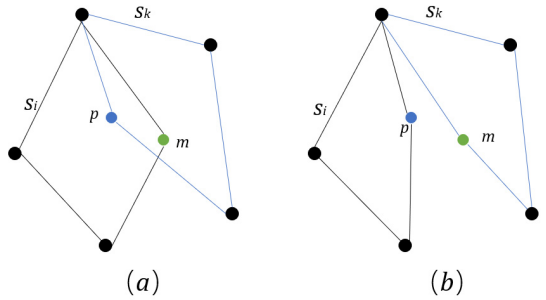

Fig. 3 Mutation operator diagram

3.6 Generate offspring

We introduced the idea of searching the neighbourhood structure alternately in VNS to expand the search space of GA, along with the elite retention strategy which is used for retaining the dominant individuals to improve the convergence speed.

The specific process of generating the next population is firstly for calculating the fitness of each chromosome in the current population and record the optimal individual. Then, use the rou-

lette to select separate chromosomes and search in the neighbourhood structure N_k (k = 1,2,...,n) to find a better solution. The better solution will be added to the progeny population and k will be set to 1. If any better solution cannot be found, skip to the next neighbourhood, and repeat the process of selection and neighbourhood descent until the size of the population reaches maximum. At this time, individuals with greater fitness can be selected more in the selection process and the individual's neighbourhood space can also be fully searched. The pseudo-code of the generating next population algorithm is shown in Algorithm2:

Algorithm2: Variable neighbourhood descent to generate next generation

```
Input: origin population S0
Output: new population S
s0: optimal solution of origin population
k =1
add s0 to S
Sn = 1
repeat
      s: roulette a solution from S0
      repeat
            s': find best neighbour s'
      if f(s') > f(s) then
         add s' to S
         k = 1
      else
         k = k + 1
       until
      k = kmax
until
   Sn = Smax
end
```

3.7 Algorithm flow

The VNS-GA algorithm process is as below:

1) Use polar coordinate classification algorithm to generate initial population S_0; defines n neighbourhoods and denoted as N_k (k = 1,2,...,n).
2) Verify the feasibility of the initial population, and use neighbourhood shaking to eliminate infeasible solutions.
3) Calculate the fitness of individuals in the population, and select the current optimal solution s.
4) The variable neighbourhood descent process: search optimal solutions in neighbourhood structure N_1, if it finds better solution s' than s, then let $s = s'$.
5) If no better solution can be found in the neighbourhood structure N_1, then let $k + 1$ and go to step 4.
6) Add the current optimal solution to the next-generation population;
7) Repeat steps 4-6 until the number of individuals in the population reaches the maximum value.
8) Repeat step 7 until the maximum number of iterations is reached, then output the current optimal solution.

4. Results and discussion

To verify the performance of VNS-GA, first, we quoted the data of the Chinese Traveling Salesman Problem (CTSP) in literature [25]. Use Python code to test it on a computer with the operating system Windows10, I5-8250, 8G RAM. The algorithm runs ten times independently on each case set. For the sake of fairness, the same genetic algorithm parameters as in literature [25] are

used, and the population size is set to 50, the heredity is 1000 generations, the crossover probability is 0.8, and the mutation probability is 0.15. Because the results obtained using SA in literature [25] are wrong. According to the detailed path diagram obtained by the simulated annealing algorithm in the literature, the results are revised as shown in Table 1.

In the comparison of the three algorithms, the traditional genetic algorithm performs the worst among the three algorithms because of the problem easy falling into the local optimal solution. After using the variable neighbourhood algorithm to improve the genetic algorithm, the effect is promoted. Compared with the detailed path diagram given in [25] (Fig. 4), it can be stated that the improved genetic algorithm is consistent with the nodes set assigned by the simulated annealing algorithm in the literature. But the improved genetic algorithm has better performance than the genetic algorithm on a single path, so, the total obtained path is shorter.

In addition, this paper selects three data sets: eil51, kroA100, and kroB150 in TSPLIB to test the performance of the algorithm. The performance of the algorithm under the conditions of 3, 5, 10, and 20 travelers is tested against the scale of three data sets, and the parameter settings used in each data set are shown in Table 2 through experiments.

We compared the measured data set results with the test results of other algorithms (data from literature [14] and [26]). The results are shown in Table 3, where *n* is the number of nodes and *m* is the number of travelers. GA1C is single chromosome coding genetic algorithm, GA2C is the two-chromosome coding genetic algorithm, GA2PC is two-segment chromosome coding genetic algorithm, GGA-SS is steady-state grouping genetic algorithm, TCX is improved two-segment chromosome coding genetic algorithm, and RGA is belt Genetic algorithm with reproduction mechanism, RLGA is a genetic algorithm that integrates the reproduction mechanism of the weed algorithm and local optimization, IWO is the invasive weed algorithm, and VNS-GA is the improved genetic algorithm proposed in this paper.

Table 1 Comparison results between VNS-GA and other algorithms (CTSP data set)

Algorithm	Total distance(km)		Sub-path distance(km)		Variance
GA	20225		6106 6477 7643		643261
	Origin data	Modified data	Origin data	Modified data	Modified variance
SA	17731	17404	5527 5616 6578	5527 4909 6968	1116314
VNS-GA	17153		6968 4658 5527		1361290

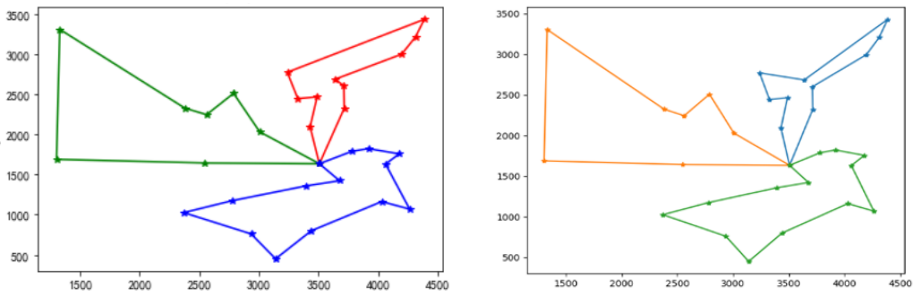

Fig. 4 The specific solution path of VNS-GA and SA

Table 2 Parameter setting of different examples

Standard data set	Number of travelers	Population size	Cross rate	Mutation rate
eil51	3, 5, 10	60	0.8	0.2
kroA100	3, 5, 10, 20	80	0.8	0.2
kroB150	3, 5, 10, 20	80	0.8	0.2

Table 3 Comparison of the longest path (VNS-GA with other genetic algorithms)

Data	*n*	*m*	GA1C	GA2C	GA2PC	GGA-SS	TCX	RGA	RLGA	IWO	VNS-GA	*D* (%)
51	51	3	234	275	203	161	203	174	167	160	165	−3.13
	51	5	173	220	164	119	154	125	118	118	121	−2.54
	51	10	140	165	123	112	113	122	112	112	112	0.00
100	100	3	14722	16229	13556	8542	12726	10231	10115	8509	8613	−1.22
	100	5	11193	11606	10589	6852	10086	7895	7812	6767	6445	4.76
	100	10	9960	10200	9463	6370	7064	6234	6225	6358	5764	7.41
	100	20	9235	9470	8388	6359	6402	6233	6211	6358	5395	13.14
150	150	3	19875	21067	19687	13268	18019	14886	14629	13168	10878	17.39
	150	5	15229	15450	14748	8660	12619	8998	8927	8479	7711	9.06
	150	10	12154	12382	11158	5875	8054	5723	5613	5594	5937	−6.13
	150	20	10206	10338	10044	5252	5673	5372	5251	5246	5750	−9.61

The data given in Table 3 are the values of the longest sub-path. Under the three data sets and different traveler numbers circumstances, the results of VNS-GA are far superior to the GA1C, GA2C, and GA2PC algorithms. The comparison with the GGA-SS algorithm shows that only in the case of 3 travelers in the kroA100 data set, the results are slightly worse and the remaining results are better than the results of GGA-SS. When compared with the TCX algorithm, the results are slightly worse only in the case of 20 travelers in the kroB150 data set, and the other results are better than the TCX algorithm. In the case of the kroB150 data set of 10 and 20 travelers, the results are slightly worse than RGA and RLGA algorithms. We use D to visually demonstrate the improvement or lack of results where D = (best_result_among_other_genetic_algorithms – VNS-GA_result)/best_result_among_other_genetic_algorithms × 100 %. Thus, $D > 0$ means the result is improved and is the best result among all genetic algorithms while $D < 0$ means the result is worse than the best result. To visualize how the performance of the algorithm changes with the number of travelers increase, we introduce the parameter E_i. E_i is the average of different numbers of travelers (i is the number of travelers and i = 3, 5, 10, 20). As can be seen from Fig. 5, with the increase of travelers' number, the performance of the algorithm decrease, but in general, the performance of the algorithm is improved compared with other genetic algorithms.

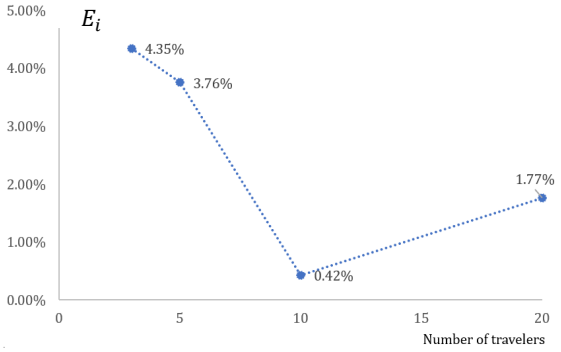

Fig. 5 VNS-GA performance variation with the number of travelers

Table 4 Comparison of total path (VNS-GA with other genetic algorithms)

Data	*n*	*m*	GA1C	GA2C	GA2PC	GGA-SS	TCX	IWO	VNS-GA
51	51	3	529	570	543	449	492	448	495
	51	5	564	627	586	479	519	478	552
	51	10	801	879	723	584	670	583	1100
100	100	3	27036	30972	26653	22051	26130	21941	25524
	100	5	29753	44062	30408	23678	28612	23319	31222
	100	10	36890	65116	31227	28488	30988	27072	49606
	100	20	62471	95568	54700	40892	44686	38357	101235
150	150	3	46111	48108	47418	38434	44674	38055	31960
	150	5	49443	51101	49947	39962	47811	38881	35754
	150	10	59341	64893	54958	44274	51326	42462	55505
	150	20	94291	100037	73934	56412	62400	53612	106917

The data in Table 4 are the values of the total path. The balance degree needs to be compared with the longest sub-path value and the total path value. Although on some data sets, the value of the longest path in the solution obtained by the VNS-GA algorithm is slightly higher than other algorithms, literature [14]is not multiple objective optimization and the objective of RGA and RLGA algorithms in the literature [26] is to make the longest sub-path the shortest, neither provide the total path length and the balance degree. Therefore, we propose the balance ratio (R) to measure the workload balance. The calculation formula is *(longest sub-path length - shortest sub-path length)/average length of each sub-path × 100%*, given the total path length of each standard data set in the case of different traveler numbers and the calculation results of balance degree are shown in Table 5.

It can be seen from Tables 3 to 5 that in comparison with other algorithms, the VNS-GA can ensure that the value of the total path is within an acceptable range while ensuring that the value of the longest sub-path is shortest. Thus, it can be seen as VNS-GA can find solutions with better balance and shorter total path length.

Fig. 6 is a detailed path diagram of the optimal solution of the VNS-GA algorithm after running on each data set. It can be intuitively seen from the figure that the number of nodes contained in each path is roughly equivalent, and the travel distance required for each traveling salesman is roughly the same.

In the case of different numbers of travelers, the VNS-GA has achieved good results when experimenting on several standard data sets. It indicates that the VNS can better solve the situation where GA is easy to fall into local optimality. VNS has a deeper search of the solution space, and the insertion operator designed in the neighbourhood action can optimize the workload balance problem so that the solution can be developed in the direction of equilibrium and the total path is shorter.

Table 5 Balance of VNS-GA on standard data set

Data	n	m	Total path	Longest path	R
51	51	3	495	165	0 %
	51	5	552	121	29.04 %
	51	10	1100	112	4.5 %
100	100	3	25524	8613	2.60 %
	100	5	31222	6445	9.40 %
	100	10	49606	5764	22.48 %
	100	20	101235	5395	26.90 %
150	150	3	31960	10878	4.76 %
	150	5	35754	7711	17.98 %
	150	10	55505	5937	11.03 %
	150	20	102650	5754	14.50 %

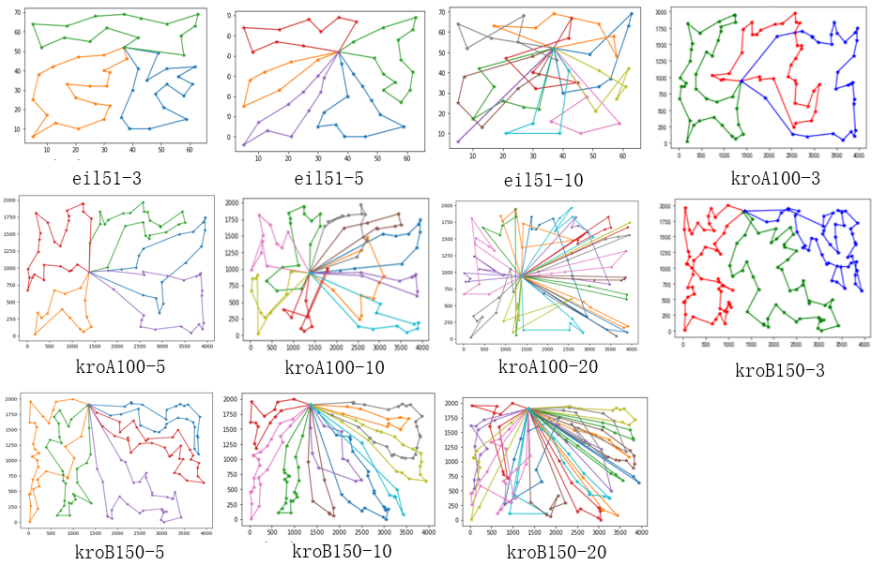

Fig. 6 Path diagram of VNS-GA on standard data set

5. Conclusion

In current times, the research on TSP has been relatively mature, but the mTSP of workload balance involves more constraints and the problem is more complicated. Thus, there are relatively few studies. This paper proposes an improved genetic algorithm (VNS-GA) to solve mTSP with workload balance. Aiming at the goal of workload balance in the mTSP, firstly, the polar coordinate classification algorithm is designed to reduce the intersection between paths and quickly obtain better initial solutions. Then, a distance comparison insertion operator is designed, which specifically takes out the node in the longest path and inserts it into the shortest path to achieve the goal of workload balance faster and to improve the efficiency of the algorithm. To avoid the genetic algorithm falling into the local optimal solution prematurely, the variable neighbourhood descent process is introduced to generate offspring, and different neighbourhood actions are used to search alternately the neighbourhood solution space adequately. Finally, the algorithm is tested on the standard data set of TSPLIB. The experimental results showed that the improved genetic algorithm (VNS-GA) is very competitive. VNS-GA is superior to other improved genetic algorithms on small and medium-sized example sets especially when the number of travelers is small.

At the same time, there are still a lot of crossed paths in the detailed path graph, which indicates that the algorithm still has room for improvement. In the future, other neighbourhood actions with better performance can be introduced to increase the space for the search of the algorithm and to improve the search efficiency. Secondly, how to find a better solution when the number of travelers is large remains to be studied further. In addition, the performance of the algorithm on very large data sets still needs further study.

Acknowledgement

We thank the Editor and two anonymous referees for their many helpful comments on an earlier version of our paper. This work was supported in part by the National Natural Science Foundation of China under grant numbers 72171016; and Beijing Social Science Foundation under grant numbers 20JCC005; and the Beijing Logistics Informatics Research Base.

References

[1] Bektas, T. (2006). The multiple traveling salesman problem: An overview of formulations and solution procedures, *Omega*, Vol. 34, No. 3, 209-219, doi: 10.1016/j.omega.2004.10.004.

[2] Bostanci, B., Karaağaç, A. (2019). Investigating the shortest survey route in a GNSS traverse network, *Tehnički Vjesnik – Technical Gazette,* Vol. 26, No. 2, 355-362, doi: 10.17559/TV-20170924174221.

[3] Iqbal Ali, A., Kennington, J.L. (1986). The asymmetric M-traveling salesmen problem: A duality based branch-and-bound algorithm, *Discrete Applied Mathematics,* Vol. 13, No. 2-3, 259-276, doi: 10.1016/0166-218X(86)90087-9.

[4] Gavish, B., Srikanth, K. (1986). An optimal solution method for large-scale multiple traveling salesman problems, *Operations Research*, Vol. 34, No. 5, 698-717, doi: 10.1287/opre.34.5.698.

[5] Yu, Q.S., Lin, D.M., Wang, D. (2012). An overview of multiple traveling salesman problem, *Value Engineering*, Vol. 31, No. 2, 166-168, doi: 10.14018/j.cnki.cn13-1085/n.2012.02.143.

[6] 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*, Vol. 175, No. 1, 246-257, doi: 10.1016/j.ejor.2005.04.027.

[7] Singh, A., Baghel, A.S. (2009). A new grouping genetic algorithm approach to the multiple traveling salesperson problem, *Soft Computing*, Vol. 13, 95-101, doi: 10.1007/s00500-008-0312-1.

[8] Zhou, W., Li, Y. (2010). An improved genetic algorithm for multiple traveling salesman problem, In: *Proceedings of 2nd International Asia Conference on Informatics in Control, Automation and Robotics (CAR 2010)*, Wuhan, China, 493-495, doi: 10.1109/CAR.2010.5456787.

[9] Koh, S.P., bin Aris, I., Ho, C.K., Bashi, S.M. (2006). Design and performance optimization of a multi-TSP (Traveling Salesman Problem) algorithm, *Artificial Intelligence and Machine Learning AIML*, Vol. 6, No. 3, 29-33.

[10] Hu, S.J., Lu, H.Y., Huang, Y., Xu, K.B. (2019). Improved genetic algorithm for solving multiple traveling salesman problem with balanced workload, *Computer Engineering and Applications*, Vol. 55, No. 17, 150-155.

[11] Guo, S. (2019). *Solutions space analysis of MTSP and application in VRP optimization,* Beijing University of Posts and Telecommunications, Beijing, China, from *https://kns.cnki.net/KCMS/detail/detail.aspx?dbname=CMFD201902&filename=1019113259.nh*, accessed April 11, 2021.

[12] Lu, Z., Zhang, K., He, J., Niu, Y. (2016), Applying K-means clustering and genetic algorithm for Solving MTSP, In: Gong, M., Pan, L., Song, T., Zhang, G. (eds.), *Bio-inspired Computing – Theories and Applications*, Springer Singapore, 278-284, doi: 10.1007/978-981-10-3614-9_34.
[13] Cheikhrouhou, O., Khoufi, I. (2021). A comprehensive survey on the multiple travelling salesman problem: Applications, approaches and taxonom, *Computer Science Review*, Vol. 40, doi: /10.1016/j.cosrev.2021.100369.
[14] Pan, J., Wang, D. (2006). An ant colony optimization algorithm for multiple travelling salesman problem, In: *Proceedings of the First International Conference on Innovative Computing, Information and Control - Volume I (ICICIC'06),* Beijing, China, 210-213, doi: 10.1109/icicic.2006.40.
[15] Venkatesh, P., Singh, A. (2015). Two metaheuristic approaches for the multiple traveling salesperson problem, *Applied Soft Computing*, Vol. 26, 74-89, doi: 10.1016/j.asoc.2014.09.029.
[16] Ryan, J.L., Bailey, T.G., Moore, J.T., Carlton, W.B. (1998), Reactive tabu search in unmanned aerial reconnaissance simulations, In: *Proceedings of the 30th 1998 Winter Simulation Conference. Proceedings (Cat. No.98CH36274),* Washington, USA, Vol. 1, 873-879, doi: 10.1109/wsc.1998.745084.
[17] Song, C.H., Lee, K., Lee, W.D. (2003). Extended simulated annealing for augmented TSP and multi-salesmen TSP, In: *Proceedings of the International Joint Conference on Neural Networks 2003,* Oregon, USA, Vol. 3, 2340-2343, doi: 10.1109/IJCNN.2003.1223777.
[18] Hu, Y., Yao, Y., Lee, W.S. (2020). A reinforcement learning approach for optimizing multiple traveling salesman problems over graphs, *Knowledge-Based Systems*, Vol. 204, Article No. 106244, doi: 10.1016/j.knosys.2020.106244.
[19] Bonz, J. (2021). Application of a multi-objective multi traveling salesperson problem with time windows, *Public Transport,* Vol. 13, 35-57, doi: 10.1007/s12469-020-00258-6.
[20] Liu, H., Zhang, H., Xu, Y. (2021). The m-Steiner traveling salesman problem with online edge blockages, *Journal of Combinatorial Optimization,* Vol. 41, 844-860, doi: 10.1007/s10878-021-00720-6.
[21] Dong, H.Y., Huang, M., Wang, X.W., Zheng, B.L. (2009). Review of variable neighborhood search algorithm, *Control Engineering of China*, Vol. 16, No. 2, 1-5.
[22] Li, J., Guo, Y.H. (2001). *Theory and method of optimal scheduling of logistics distribution vehicles*, China Fortune Press, Beijing, China.
[23] Min, J.N., Jin, C., Lu, L.J. (2019). Split-delivery vehicle routing problems based on a multi-restart improved sweep approach, *International Journal of Simulation Modelling*, Vol. 18, No. 4, 708-719, doi: 10.2507/IJSIMM18(4)CO19.
[24] Min, J.N., Jin, C., Lu, L.J. (2019). Maximum-minimum distance clustering method for split-delivery vehicle-routing problem: Case studies and performance comparisons, *Advances in Production Engineering & Management*, Vol. 14, No. 1, 125-135, doi: 10.14743/apem2019.1.316.
[25] Xiong, C., Wu, H.P., Li, B. (2010). Improved genetic algorithm for solving MTSP, In: *Proceedings of the 4th China Intelligent Computing Conference*, Beijing, China, 143-149.
[26] Hu, S.J. (2019). *Research on multiple traveling salesman problem based on improved genetic algorithm,* Master Thesis, Jiangnan University, Jiangnan, China.