An Improved Adaptive Genetic Algorithm for Mobile Robot Path Planning Analogous to TSP with Constraints on City Priorities

Junjie Jiang
School of Mechanical and Automobile Engineering South China University of Technology
Guangzhou, China
304796768@qq.com

Xifan Yao
School of Mechanical and Automobile Engineering South China University of Technology
Guangzhou, China
mexfyao@scut.edu.cn

Erfu Yang
Department of Design, Manufacturing and Engineering Management University of Strathclyde
Glasgow, U.K
erfu.yang@strath.ac.uk

Jorn Mehnen
Department of Design, Manufacturing and Engineering Management University of Strathclyde
Glasgow, U.K
jorn.mehnen@strath.ac.uk

***Abstract*—The material transportation planning with a mobile robot can be regarded as the classic Traveling Salesman Problem (TSP). To solve such problems with different priorities at stations, an improved adaptive genetic simulated annealing algorithm is proposed. Firstly, the priority matrix is defined according to station priorities. Based on standard genetic algorithm, the generating strategy of the initial population is improved to prevent the emergence of non-feasible solutions, and an improved adaptive operator is introduced to improve the population ability for escaping local optimal solutions and avoid premature phenomena. Moreover, to speed up the convergence of the proposed algorithm, the simulated annealing strategy is utilized in mutation operations. The experimental results indicate that the proposed algorithm has the characteristics of strong ability to avoid local optima and the fasterr convergence speed.**

Keywords—Traveling Salesman Problem (TSP), genetic algorithm, simulated annealing, crossover and mutation, path planning, mobile robots

I. INTRODUCTION

With the rapid development of modern industrial technologies, mobile robost such as Automated Guided Vehicles (AGVs) are increasingly playing the role of material transportation in large factories [1]. Since AGVs were introduced in the 1950s, over the past decades, today' AGV guidance technology has evolved from electromagnetic guidance into laser and visual guidance. Such breakthrough makes AGV no longer be limited by magnetic track, and therefore have more freedom and flexibility to move in the factory floors. As a result, AGV path planning problems have arised and become a research hotspot, which attracts many researchers' interest in scientific research, and.

AGV/Mobile robot path planning aims at optimizing one or more indicators (e.g., path length, elapsed time, cost, etc.) to quickly find an optimal route among many alternative non-collision paths [2]. Path planning can be divided into global and local path planning according to the knowledge of the surrounding environment [3]. The former belongs to static planning while the latter is dynamic planning. In mobile robot path planning, a common problem is planning the shortest route, which connects each station, so that AGVs can supply material for each station along the road during the movement. The prototype of such problems is Traveling Salesman Problem (TSP) which is a classic NP-Hard problem [4]. The specific description is that a salesman wants to promote goods in different cities. He can start from any city and traverse all the cities until he gets back to the original one. The constraint is that each city must be visited only once, and the journey is required to be the shortest, which minimizes the objective function. We assume that the number of cities is n and the travel path is $S = (s_1, s_2, s_3 \ldots s_n)$. So, the objective function can be described as follows:

$$f(S) = \sum_{i=1}^{n-1} d(s_i, s_{i+1}) + d(s_n, s_1) \qquad (1)$$

where s_i is the *i*th city and $d(s_i, s_j)$ depicts the distance between city s_i and s_j.

To solve such TSPs, there exist many optimization algorithms such as ant colony algorithm (ACO) [5], shuffled frog leading algorithm (SFLA) [6], genetic algorithm (GA) [7], simulated annealing (SA) [8], firefly algorithm (FA) [9], and bat algorithm (BA) [10]. GA is a relatively simple and practical algorithm, which imitates the mechanisms of genetics including selection, crossover and mutation operators. It initially operates on individuals from a randomly generated population to gradually improve the fitness of individuals, and eventually gets the best individual and find the optimal solution to the problem. However, GA has the slow convergence speed and easily fallsinto local optimal solutions in practical use. The principle of SA [11] is randomly searching in the search space, iterating for several times, and gradually converging to the optimal solution by setting the initial temperature, final temperature, annealing temperature function, Markov chain length, etc. Its greatest advantage is that the global optimal solution is more likely to be obtained, and its convergence speed is faster than GA. Moreover, SA has strong robustness. Since GA is easy to integrate with other algorithms, the combination of GA and SA can help to obtain the advantages of the both algorithms, and there are some applications for solving TSP. Luo, et al [12] proposed a Heuristic Simulated Annealing Genetic Algorithm (HSAGA) in which GA functions as global search strategy while the designed Heuristic Simulated Annealing (HSA) algorithm acts as a local search strategy. HSA can enhance the search effectiveness and avoid getting stuck into a local optimal trap. Yao, et al [13] improved the selection of the initial solution, the generation of the new solution and the improvement of the current solution in the simulated annealing and genetic algorithm. The method of intergenerational crossing is adopted to improve the search speed of the optimal solution. He, et al [8] proposed an improved genetic simulated annealing algorithm, which improves the ability of the algorithm to jump out fromlocal

optimal solutions. The algorithm has an obvious optimization effect and can find the optimal solution for many cases.

In traditional TSP, the order of visiting cities isn't restricted. Therefore, the initial population generated in GA can be completely random. While in a factory, the order of each workstation's demand for materials strictly depends on the process corresponding to the station, so it is more reasonable for AGVs to transport materials according to the process. Thus, workstations should be assigned different priorities and AGV should traverse each station according to its priority. Such TSPs with constraints on city priorities are essentially different from the classic Traveling Salesman Problem with Precedence Constraint (TSPPC), as studied in [14] and [15]. New improvements must be made in order to solve such special TSPs by regarding workstations as cities for our study. To this end, we propose an improved adaptive genetic simulated annealing algorithm (IAGSAA), which is based on standardGA, and improved by generating initial population to avoid non-feasible solutions, and in which the improved adaptive operator is introduced to improve the ability of the algorithm to jump out from local optimal solutions and resistance power to the destruction of excellent individuals in each generation at the same time. Moreover, the simulated annealing mutation strategy is introduced into the GA mutation operation to improve the convergence speed of the proposed algorithm.

II. Mathematical Model Construction

We assume that the total number of stations in the factory is n , the number of priorities is k, and the corresponding number of stations in each priority is $n_1, n_2, \dots, n_k$, so the equation can be described as follows:

$$n = \sum_{i=1}^{k} n_i \tag{2}$$

It is assumed that the smaller the station priority value is, the higher the station priority is. So, AGV must firstly traverse all stations in priority 1, and then stations in prioritis 2,3,... ,k, respectively. Therefore, the travel path is $P = (p_{1,1}, p_{1,2}, \dots, p_{1,n_1}, p_{2,1}, p_{2,2}, \dots, p_{2,n_2}, \dots, p_{k,1}, p_{k,2}, \dots, p_{k,n_k})$. Then, the path length function of AGV traversing all the stations is defined as follows:

$$f(P) = \sum_{i=1}^{k} \sum_{j=1}^{n_i-1} d(p_{i,j}, p_{i,j+1}) + \sum_{i=1}^{k-1} d(p_{i,n_i}, p_{i+1,1}) + d(p_{k,n_k}, p_{1,1}) \tag{3}$$

where $p_{i,j}$ is the station number, representing the jth station in priority i; and $d(p_{i,j}, p_{i,j+1})$ is the distance between station $p_{i,j}$ and station $p_{i,j+1}$. The optimal solution should make (3) obtain the minimum value.

III. Design of the Proposed Algorithm

A. Defining Priority Matrix

We define the priority matrix as follows:

$$G = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n-1} & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n-1} & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{k-1,1} & a_{k-1,2} & \cdots & a_{k-1,n-1} & a_{k-1,n} \\ a_{k,1} & a_{k,2} & \cdots & a_{k,n-1} & a_{k,n} \end{bmatrix} \tag{4}$$

where G is a $k \times n$ matrix, representing totally n stations divided into k priorities. The value of each element in matrix G is 0 or 1, whose element $a_{i,j}$ is defined as follows:

$$a_{i,j} = \begin{cases} 0 & \text{if station } j \notin \text{priority } i \\ 1 & \text{if station } j \in \text{priority } i \end{cases} \tag{5}$$

so, in each column of matrix G, only one element is 1, and the rest are 0s.

B. Preliminary Sorting

Before sorting, the travel path is the same as the station number sequence. Suppose that the n-dimensional vector corresponding to the travel is $t = [1 \quad 2 \quad \cdots \quad n]$ and priority i contains n_i stations. We can preliminarily sort stations numbered 1 to n according to the priority matrix, sorting steps are described as follows:

Step 1: define k zero vectors $t_i = [0 \quad 0 \quad \cdots \quad 0] (i = 1,2, \dots, k)$, so the dimension of t_i is n_i.

Step 2: successively examine elements of jth column of matrix G, if $a_{i,j} = 1$, then assign the jth element of vector t to vector t_i.

Step 3: define a new n-dimensional vector $\mathrm{t_{new}}$ ($\mathrm{t_{new}} = [\mathrm{t_1\ t_2\ \dots\ t_k}]$), so the order of elements of vector t_{new} is preliminarily sorted by the station priorities.

C. Fitness Function

We normalize the path length corresponding to the individuals in each generation of population and the normalized value is used to represent the fitness value of the individual. Then, the fitness function can be defined as follows:

$$f_i = (maxl - l_i)/(maxl - minl + a) \tag{6}$$

where l_i represents the path length corresponding to the ith individual; $maxl$ and $minl$ respectively represent the longest and shortest paths in this generation of population; and the parameter a is an extremely small positive number, which prevents the denominator of (6) from being 0 later in the iteration. We set $a = 0.0001$ in the proposed algorithm. The shorter the individual's corresponding path is, the larger the fitness value is and the higher the survival probability is. The fitness value calculated from (6) is ranged in [0-1].

D. Generating Piecewise Initial Populations

In GA, the initial population is randomly generated. While, the generation criteria of the initial population will be adjusted in an TSP with constraints on city priorities. As we have divided vector t_{new} into k subvectors, the priorities of stations corresponding to the elements in different subvectors are different. The elements in each subvector are respectively and randomly sorted to generate the initial population for ensuring that the generated initial population still satisfies the priority requirement. Thus, the generated initial population is piecewise and the generating process is shown in Fig. 1, in which there are 10 stations that are divided into three priorities.

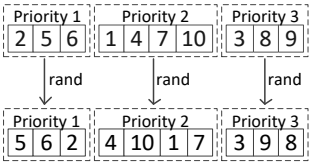


Fig. 1. Generating piecewise initial population.

E. Genetic Operator Design

a) Selection Operator

The individuals of larger fitness values will be retained, while those of smaller fitness values will be discarded in each generation. The larger the individual fitness value is, the higher the probability of being retained is. Suppose that the fitness values of 1-10 individuals in a generation are 0.4, 0.27, 0.53, 0.82, 0.68, 1, 0.45, 0.73, 0.96, and 0.14, respectively, individuals 4, 5, 6, 8, 9 will be retained and survived while the rest will be eliminated if the generated random number is 0.6.

b) Crossover and Mutation Operators

The crossover operation, which can improve the searching ability of the population, is to match chromosomes randomly and exchange some genes with a certain crossover probability p_c. In this paper, we adopt piecewise Partial Mapped Crossover (PMX) [16], to ensure that the crossed gene positions belong to the same priority. The operation process is described as follows:

Step1: randomly select two paternal chromosomes A and B.

Step2: generate a random number k' from 1 to k and randomly find two adjacent gene positions in the k'th subvector, cross chromosomes A and B at the selected gene positions.

Step3: modify the gene values outside the crossed gene positions according to the mapping relation of the crossed gene values in order that the same gene value does not appear on one chromosome. Therefore, generate two chromosomes A1 and B1.

As shown in Fig. 2, we use the chromosome segmentation in the previous section to illustrate the crossover process. Suppose $k' = 2$, the crossed gene positions are 2 and 3.

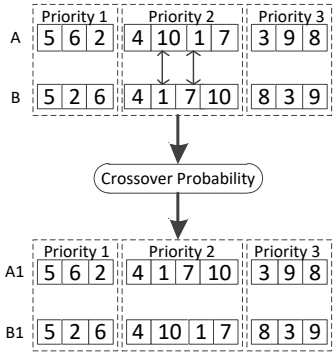


Fig. 2. Piecewise partially mapped crossover.

The mutation operation, which can improve population diversity and avoid premature phenomena in some degree [17], is to change one or more gene values of individuals in a population with a certain mutation probability p_m. As mentioned above, SA is better than GA in the convergence speed. In this paper, we take the treatment strategy of the deteriorating solution in SA to deal with the new individual generated after mutation operation in GA. Therefore, the strategy of combining piecewise two point exchange mutation and Simulated Annealing Mutation (SAM) is utilized. The operation process is described as follows:

Step1: perform piecewise two point exchange mutation operation based on the crossover operation, i.e., generate two different random numbers $i_{k'1}$ and $i_{k'2}$ ($i_{k'1}, i_{k'2} \le j_{k'}$, where $j_{k'}$ is the total number of genes on the k'th segment of chromosome) on the k'thsegment of chromosome A1 and B1, then, exchange two genes in gene positions $i_{k'1}$ and $i_{k'2}$. The mutation process illustration is shown in Fig. 3 (suppose $i_{k'1} = 1$ and $i_{k'2} = 4$).

Step2: calculate the fitness values of the two individuals before and after the mutation respectively. The mutation is accepted if the fitness value becomes larger. Otherwise, whether the mutation accepted or not is determined by a certain annealing probability p_t.

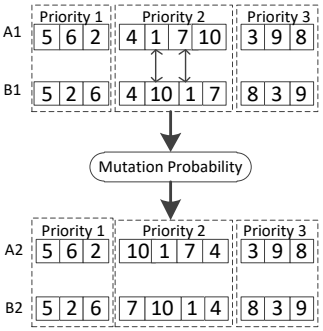


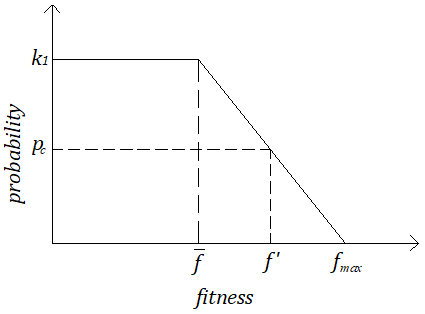
Fig. 3. Piecewise two point exchange mutation.

The crossover probability p_c and mutation probability p_m will directly affect the performance of the algorithm. In standard GA(SGA), the values of p_c and p_m are fixed. The average fitness value of the population can be rapidly increased at the initial stage of the algorithm. However the better individuals will be destroyed at the later stage of the algorithm, leading to premature phenomena. In this paper, the values of crossover and mutation probabilities will change adaptively. Adapted genetic algorithm (AGA) originally was proposed by Srinivas and Patnaik with the aim of increasing the crossover and mutation probabilities later in the iteration, so that the population can jump out of the local optimal solutions. The formulas for calculating the adaptive crossover and mutation probabilities given in [18] are as follows:

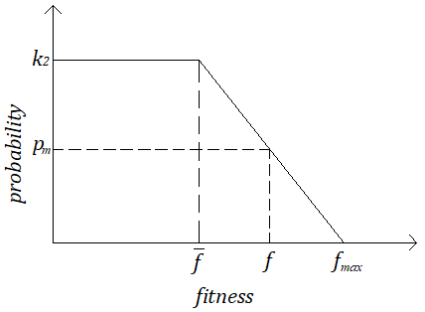
$$p_c = \begin{cases} \dfrac{k_1(f_{max} - f')}{f_{max} - \bar{f}}, f' \ge \bar{f} \\ k_3 \qquad\qquad , f' \le \bar{f} \end{cases} \tag{7}$$

$$p_m = \begin{cases} \dfrac{k_2(f_{max} - f)}{f_{max} - \bar{f}}, f \ge \bar{f} \\ k_4 \qquad\qquad , f \le \bar{f} \end{cases} \tag{8}$$

where f_{max} represents the maximum fitness value of the population; $\bar{f}$ represents the average fitness value of the population; f' is the larger of the fitness values of the individuals to be crossed; f is the fitness values of the individuals to be mutated; $k_1, k_2, k_3$and k_4 are parameters. The adjustment curves of the crossover and mutation probability corresponding to (7) and (8) are shown in Fig. 4.



(a) Adjustment curve of crossover probability.



(b) Adjustment curve of mutation probability.

Fig. 4. Adjustment curves of crossover and mutation probabilites in AGA.

According to Fig. 4, if the fitness value of the individual is smaller than the average value, higher crossover and mutation probabilities are used; and if the fitness value of the individual is larger than the average value, lower crossover and mutation probabilities are used. The probabilities vary with the change of the fitness values of individuals. However, the crossover and mutation probabilities of the optimal individual in the population calculated from (7) and (8) are 0. At the initial stage of the algorithm, even the individual with the largest fitness value is generally not the global optimal solution. Therefore, if the individual genes cannot be changed, inversely, they are retained so many that the algorithm is still likely to get stuck at the local optimum. To overcome this shortcoming, many improved formulas [19] [20] [21] have been used to calculate the adaptive crossover and mutation probabilities. In addition, when there are more individuals, whose fitness values are near the average fitness values, in the population, they have advantages in the population evolution because the individual genes are such similar that poor effect of the subsequent evolution is resulted in. The crossover and mutation probabilities of the individuals, whose fitness values are near the maximum fitness value, are such different that some better individuals are more likely to be destroyed because of the relatively high crossover and mutation probabilities [21]. To solve the problem, the adaptive adjustment curve in $\bar{f}$ and f_{max} should be flattened out. We adopt nonlinear adjustment as follows:

$$pc = \begin{cases} \sqrt{2}(pc_1 - pc_2) + pc_2 - (pc_1 - pc_2) \times sin\left(\dfrac{(f' - \bar{f}) \times \pi}{(f_{max} - \bar{f}) \times 2}\right), f' \geq \dfrac{\bar{f} + f_{max}}{2} \\ pc_2 + (pc_1 - pc_2) \times cos\left(\dfrac{(f' - \bar{f}) \times \pi}{(f_{max} - \bar{f}) \times 2}\right) \quad , \bar{f} \leq f' \leq \dfrac{\bar{f} + f_{max}}{2} \\ pc_1 \quad , f' \leq \bar{f} \end{cases} \tag{9}$$

$$pm = \begin{cases} \sqrt{2}(pm_1 - pm_2) + pm_2 - (pm_1 - pm_2) \times sin\left(\dfrac{(f - \bar{f}) \times \pi}{(f_{max} - \bar{f}) \times 2}\right), f \geq \dfrac{\bar{f} + f_{max}}{2} \\ pm_2 + (pm_1 - pm_2) \times cos\left(\dfrac{(f - \bar{f}) \times \pi}{(f_{max} - \bar{f}) \times 2}\right) \quad , \bar{f} \leq f \leq \dfrac{\bar{f} + f_{max}}{2} \\ pm_1 \quad , f \leq \bar{f} \end{cases} \tag{10}$$

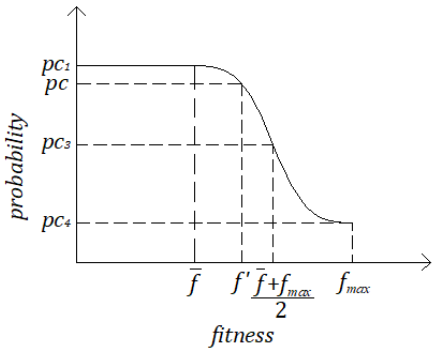
where f_{max}, $\bar{f}$ and f' have the same meanings as in (7) and (8); f is the smaller of the mutated individual fitness values; and pc_1, pc_2, pm_1 and pm_2 are parameters. Respectively, pc_1 and pm_1 determine the maximum crossover and mutation probabilities; pc_1 and pc_2 co-determine the minimum crossover probability; pm_1 and pm_2 co-determine the minimum mutation probability. The mutation probability calculated from (10) can further protect the better individuals generated by the crossover operation. The improved adjustment curves of the crossover and mutation probabilities are shown in Fig. 5, where:

$$pc_3 = \frac{\sqrt{2}}{2} pc_1 + \frac{2 - \sqrt{2}}{2} pc_2 \tag{11}$$

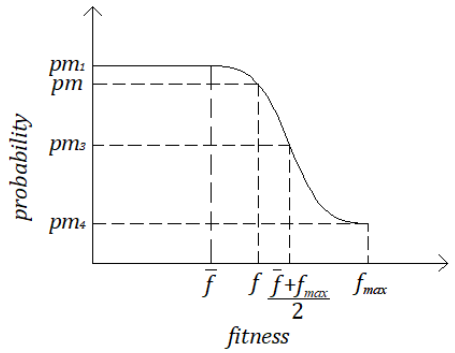
$$pc_4 = (\sqrt{2} - 1) pc_1 + (2 - \sqrt{2}) pc_2 \tag{12}$$

$$pm_3 = \frac{\sqrt{2}}{2} pm_1 + \frac{2 - \sqrt{2}}{2} pm_2 \tag{13}$$

$$pm_4 = (\sqrt{2} - 1) pm_1 + (2 - \sqrt{2}) pm_2 \tag{14}$$



(a) Improved adjustment curve of crossover probability.



(b) Improved adjustment curve of mutation probability.

Fig. 5. Adjustment curves of crossover and mutation probabilities for this study.

The calculation formula of annealing probability pt and the annealing temperature function are as follows:

$$p_t = exp((f_{new} - f_{old})/T) \quad (15)$$

$$T(n+1) = K \times T(n) \quad (16)$$

where f_{old} and f_{new} are fitness values before and after individual mutation; T is a temperature parameter that varies with the number of iterations n, so $T(0)$ is the initial temperature; and K is the temperature attenuation parameter.

F. Terminal Condition

We set T_{max} as the maximum number of iterations. If $n \geq T_{max}$, then the iteration will be terminated and the optimal result will be output.

G. Improved algorithm flowchart

The whole algorithm flowchart mainly includes inputting the priority matrix, setting parameters, preliminary sorting, as well as selection, crossover and mutation operations in the improved genetic algorithm, as shown in Fig. 6.

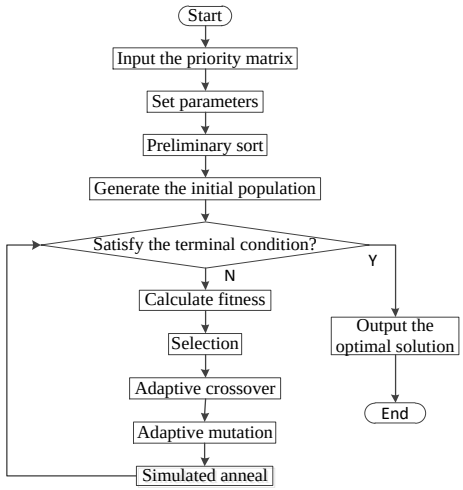


Fig. 6. The flowchart of the proposed algorithm.

IV. Simulation Experiments and Result Analysis

There are no known optimal solutions because any study case of TSP with constraints on city priorities cannot be found in the existing literature. In order to verify the effectiveness of proposed IAGSAA in this paper, the chn31 data sets in the TSPLIB test library is utilized for simulation experiments for various cases. Without the loss of generality, set three stations priorities in case I, four in case II and five in case III. Four algorithms (IAGSAA, SGA, AGA and Genetic Simulated Annealing Algorithm (GSAA)) are compared from five aspects, the optimal solution, the frequency of the optimal solution, stable iteration, the operation time and the algorithm stability. Each algorithm is run 20 times in MATLAB R2014a. Set 200 as the population quantity and 1000 as the maximum number of iterations. The initial temperature $T(0)$ is 0.31 and the temperature attenuation parameter K is 0.995 in GSAA and IAGSAA. The values of other parameters in each algorithm are shown in TABLE I.

TABLE I. Parameter Setting

Algorithm	Parameter	Value
SGA	p_c	0.8
	p_m	0.1
AGA	k_1	0.85
	k_2	0.15
	k_3	0.85
	k_4	0.15
GSAA	p_c	0.8
	p_m	0.1
IAGSAA	pc_1	0.8
	pc_2	0.4
	pm_1	0.1
	pm_2	0.001

A. Case I

As shown in TABLE II, the priority of each station in case I is given.

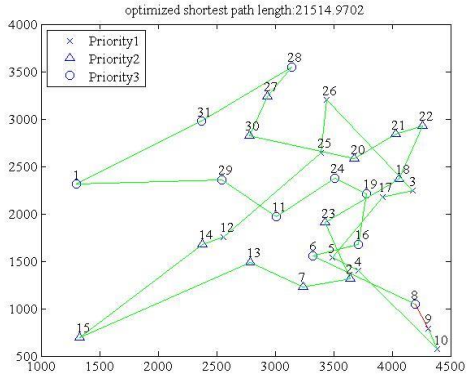
TABLE II. The Priority of Each Station in Case I

Priority	Number of Stations	Station Number
1	9	3/4/5/9/10/12/17/25/26
2	12	2/7/13/14/15/18/20/21/22/23/27/30
3	10	1/6/8/11/16/19/24/28/29/31

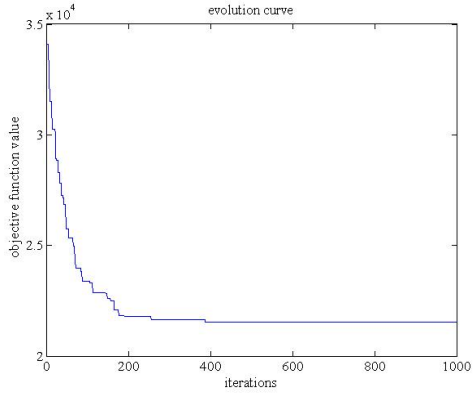
Hence, the priority matrix is defined as follows:

$$G = \begin{bmatrix} 0&0&1&1&1&0&0&0&1&1&0&1&0&0&0&0&1&0&0&0&0&0&0&0&1&1&0&0&0&0&0 \\ 0&1&0&0&0&0&1&0&0&0&0&0&1&1&1&0&0&1&0&1&1&1&1&0&0&0&1&0&0&1&0 \\ 1&0&0&0&0&1&0&1&0&0&1&0&0&0&0&1&0&0&1&0&0&0&0&1&0&0&0&1&1&0&1 \end{bmatrix} \quad (17)$$

The result of preliminary sorting is 3-4-5-9-10-12-17-25-26-2-7-13-14-15-18-20-21-22-23-27-30-1-6-8-11-16-19-24-28-29-31. Results of IAGSAA operation is shown in Fig. 7. The optimized shortest path length is 21514.9702 and the travel path is 9-10-4-5-17-3-26-25-12-14-15-13-7-2-23-18-22-21-20-30-27-28-31-1-29-11-24-19-16-6-8.



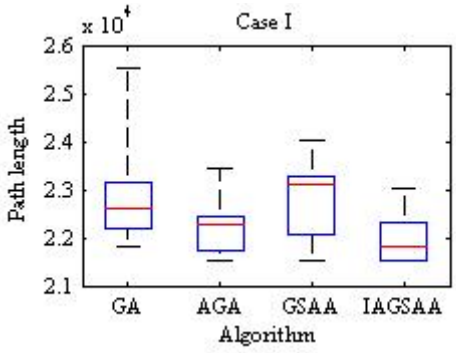
(a) The shortest path length



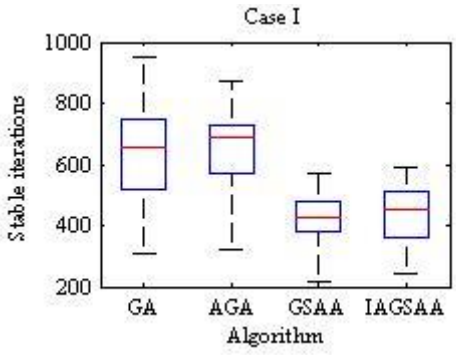
(b) The evolution curve

Fig. 7. Results of IAGSAA operation in case I.

Results of running 20 times of four algorithms are shown in Fig. 8, and specific comparisons of the results are shown in TABLE III.



(a) Path length in case I



(b) Stable iterations in case I

Fig. 8. Results of running four algorithms 20 times in case I.

TABLE III. SPECIFIC COMPARISON OF RESULTS IN CASE Ⅰ

Algorithm	Shortest Path Length	Frequency of The Optimal Solution	Average Path Length	Standard Deviation of The Path Length	Average Stable Iteration	Average Elapsed Time (s)
SGA	21802.3659	0/20	22856.3086	977.6633	631.75	15.38
AGA	21514.9702	3/20	22202.4037	539.4112	639.05	15.18
GSAA	21514.9702	1/20	22832.9546	811.2824	419.70	15.91
IAGSAA	21514.9702	7/20	22002.4758	513.6103	435.90	16.36

B. *Case II*

The priority of each station in case II is given in TABLE IV.

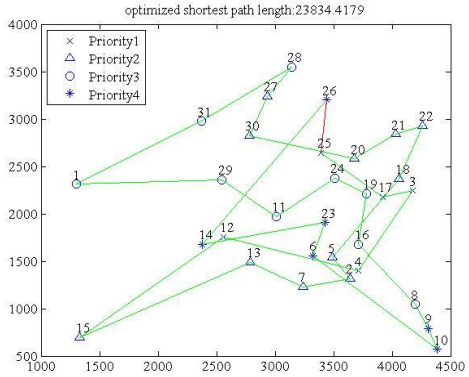
TABLE IV. THE PRIORITY OF EACH STATION IN CASE Ⅱ

Priority	Number of Stations	Station Number
1	5	3/4/12/17/25
2	11	2/5/7/13/15/18/20/21/22/27/30
3	9	1/8/11/16/19/24/28/29/31
4	6	6/9/10/14/23/26

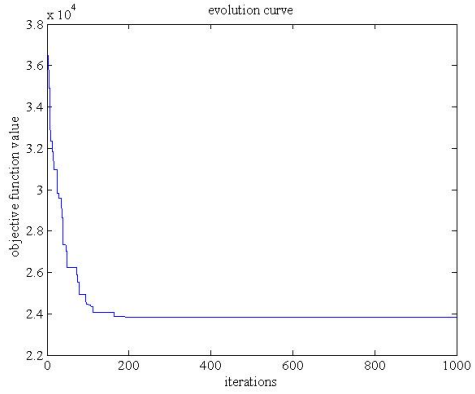
Hence, the priority matrix is defined as follows:

$$G=\begin{bmatrix}0&0&1&1&0&0&0&0&0&0&0&1&0&0&0&0&1&0&0&0&0&0&0&0&1&0&0&0&0&0&0\\0&1&0&0&1&0&1&0&0&0&0&0&1&0&1&0&0&1&0&1&1&1&0&0&0&0&1&0&0&1&0\\1&0&0&0&0&0&0&1&0&0&1&0&0&0&0&1&0&0&1&0&0&0&0&1&0&0&0&1&1&0&1\\0&0&0&0&0&1&0&0&1&1&0&0&0&1&0&0&0&0&0&0&0&0&1&0&0&1&0&0&0&0&0\end{bmatrix} \tag{18}$$

The result of preliminary sorting is 3-4-12-17-25-2-5-7-13-15-18-20-21-22-27-30-1-8-11-16-19-24-28-29-31-6-9-10-14-23-26. Results of IAGSAA operation is shown in Fig. 9. The optimized shortest path length is 23834.4179 and the travel path is 25-17-3-4-12-15-13-7-2-5-18-22-21-20-30-27-28-31-1-29-11-24-19-16-8-9-10-6-23-14-26.



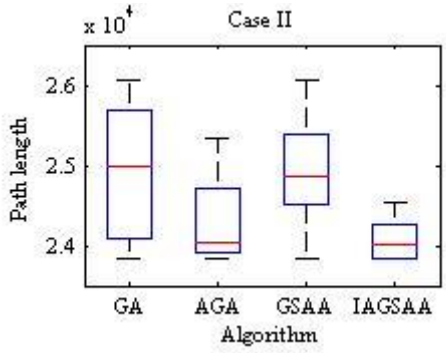
(a) The shortest path length



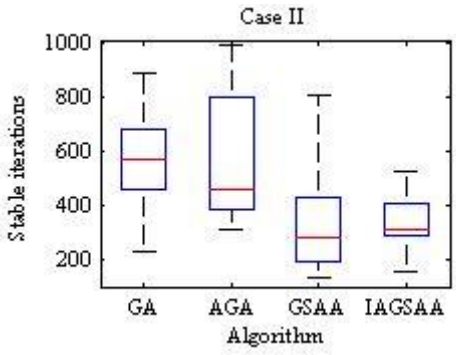
(b) The evolution curve

Fig. 9. Results of IAGSAA operation in case II.

Results of running 20 times of four algorithms are shown in Fig. 10, and specific comparisons of the results are shown in TABLE V.



(a) Path length in case II



(b) Stable iterations in case II

Fig. 10. Results of running four algorithms 20 times in case II.

TABLE V. SPECIFIC COMPARISON OF RESULTS IN CASE II

Algorithm	Shortest Path Length	Frequency of The Optimal Solution	Average Path Length	Standard Deviation of The Path Length	Average Stable Iteration	Average Elapsed Time (s)
SGA	23834.4179	3/20	24942.4687	804.1380	551.45	15.19
AGA	23834.4179	5/20	24266.3097	494.7937	560.50	15.86
GSAA	23834.4179	2/20	24927.9354	655.0082	322.25	15.67
IAGSAA	23834.4179	9/20	24058.1916	258.5429	335.35	16.45

C. *Case III*

The priority of each station in case III is given in TABLE VI.

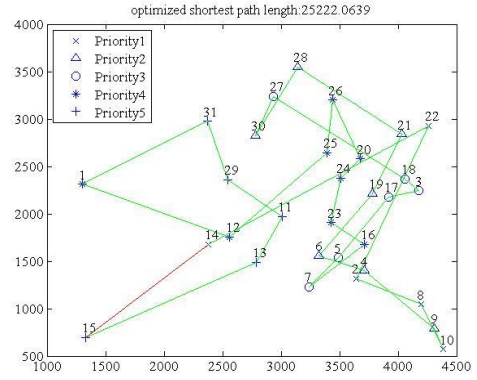
TABLE VI. THE PRIORITY OF EACH STATION IN CASE III

Priority	Number of Stations	Station Number
1	5	2/810/14/22
2	7	4/6/9/19/21/28/30
3	6	3/5/7/17/18/27
4	8	1/12/16/20/23/24/25/26
5	5	11/13/15/29/31

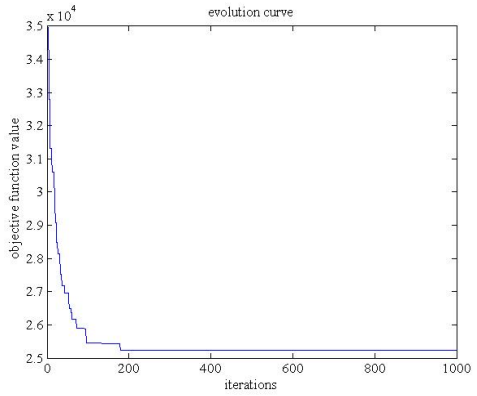
Hence, the priority matrix is defined as follows:

$$G=\begin{bmatrix}0&1&0&1&0&0&0&1&0&1&0&0&0&1&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0\\0&0&0&1&0&1&0&0&1&0&0&0&0&0&0&0&0&0&1&0&1&0&0&0&0&0&1&0&1&0\\0&0&1&0&1&0&1&0&0&0&0&0&0&0&0&0&1&1&0&0&0&0&0&0&0&1&0&0&0&0\\1&0&0&0&0&0&0&0&0&0&0&1&0&0&0&1&0&0&0&1&0&0&1&1&1&1&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0&1&0&1&0&1&0&0&0&0&0&0&0&0&0&0&0&0&1&0&1\end{bmatrix} \quad (19)$$

The result of preliminary sorting is 2-8-10-14-22-4-6-9-19-21-28-30-3-5-7-17-18-27-1-12-16-20-23-24-25-26-11-13-15-29-31. Results of IAGSAA operation is shown in Fig. 11. The optimized shortest path length is 25222.0639 and the travel path is 14-22-2-8-10-9-4-6-19-21-28-30-27-18-3-17-5-7-16-23-24-20-26-25-12-1-31-29-11-13-15.



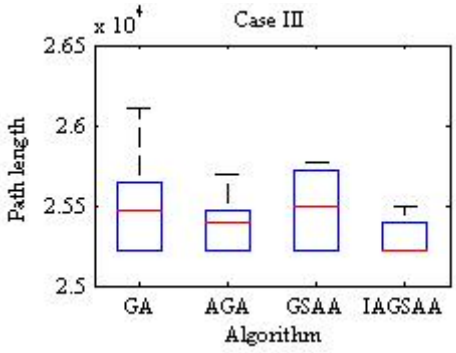
(a) The shortest path length



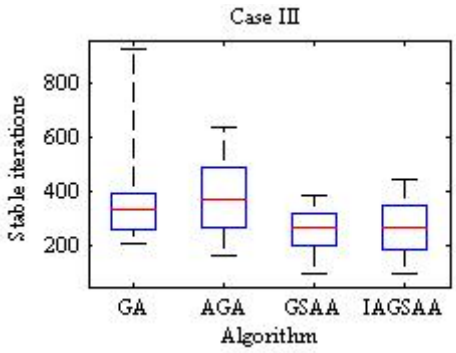
(b) The evolution curve

Fig. 11. Results of IAGSAA operation in case III.

Results of running 20 times of four algorithms are shown in Fig. 12, and specific comparisons of the results are shown in TABLE VII.



(a) Path length in case III



(b) Stable iterations in case III

Fig. 12. Results of running four algorithms 20 times in case III.

TABLE VII. SPECIFIC COMPARISON OF RESULTS IN CASE III

Algorithm	Shortest Path Length	Frequency of The Optimal Solution	Average Path Length	Standard Deviation of The Path Length	Average Stable Iteration	Average Elapsed Time (*s*)
SGA	25222.0639	6/20	25488.2436	246.3877	363.65	15.91
AGA	25222.0639	8/20	25371.8903	149.2661	372.20	15.38
GSAA	25222.0639	6/20	25466.7864	222.6532	258.10	16.95
IAGSAA	25222.0639	13/20	25296.1873	108.3068	265.55	17.38

According to the data in TABLE III, TABLE V and TABLE VII, the advantage of AGA is that it can jump out of the local optimal solution, but it has a slow convergence speed; GSAA can accelerate the convergence of the calculation process, but premature phenomena occur; although the IAGSAA proposed in this paper is slightly longer than the other three algorithms in the elapsed time, its average path length and average stable iterations improve significantly at the same time. In addition, the frequency of the optimal solutions is greatly increased, and the standard deviation of the path length is also greatly reduced, indicating the stability of the algorithm. If there are more stations in each priority, more combinations are obtained, and the improvement of IAGSAA is more significant. The number of stations in each priority is relatively small in case III. Compared with SGA, the average path length obtained by IAGSAA is reduced by about 200, the standard deviation of the path length is reduced by about 140, average stable iteration is reduced by about 100, and the frequency of the optimal solutions is approximately doubled. The number of stations in each priority is relatively big in case I. Compared with SGA, the average path length obtained by IAGSAA is reduced by about 850, the standard deviation of the path length is reduced by about 460, average stable iteration is reduced by about 200, and the frequency of the optimal solutions could be thought to have increased approximately sixfold. Therefore, the algorithm proposed in this paper has good effectiveness, which is able to avoid premature phenomena, as well as accelerate the convergence speed.

V. CONCLUSION

This study divides stations in a factory into several priorities based on the process, which is represented by the priority matrix, and analogous to a TSP with constraints on city priorities. Since solving the problem by SGA shows a slow convergence speed and premature phenomena, we have proposed the IAGSAA, in which the generation strategy of the initial population of SGA is improved, and improved adaptive crossover and mutation, as well as SAM are introduced. Later

in the iteration of the IAGSAA, the adaptive crossover and mutation can enrich the diversity of the population to find new search directions and jump out of local optimal solutions. SAM is very useful to accelerate the convergence speed of the algorithm. Simulation results have indicated that the IAGSAA can integrate the advantages of AGA and GSAA, and has good performance.

Since the environment in the factory is complex and dynamic, dynamic characteristics will be taken into account in the further research, and the algorithm will be optimized to shorten the operation time and make AGV respond quickly. …

ACKNOWLEDGMENT

This work supported by the National Natural Science Foundation of China (51675186), the National Natural Science Foundation of China and the Royal Society of Edinburgh (51911530245), and the Science and Technology Project of Guangdong Province (2018A030321002).

REFERENCES

[1] E. Liu, X. Yao, M. Liu and H. Jin, "AGV Path Planning Based on Improved Grey Wolf Optimization Algorithm and its Implementation Prototype Platform," Commputer Integrated Manufacturing Systems, vol. 24, no. 11, pp. 131-143, 2018.

[2] C. He, Y. Song, Q. Le, X. Lv, R. Liu and J. Chen, "Integrated Scheduling of Multiple AGVs and Machines in Flexible Job Shops," China Mechanical Engineering, vol. 30, no. 04, pp. 64-73, 2019.

[3] E. Shi, Y. Huang, C. Zhu and Y. Zhang, "A Novel Method of Planing path for DDWMR," China Mechanical Engineering, vol. 23, no. 23, pp. 2805-2809, 2012.

[4] Y. Chen and C. Han, "An Evolutionary Multiobjective Optimization Method for Traveling Salemans Problems," Control and Decision, vol. 34, no. 04, pp. 775-780, 2019.

[5] S. Chowdhury, M. Marufuzzaman, H. Tunc, L. Bian and W. Bullington, "A Modified Ant Colony Optimization Algorithm to Solve A Dynamic Traveling Salesman Problem: A Case Study with Drones for Wildlife Surveillance," Journal of Computational Design and Engineering, vol. 6, no. 3, pp. 368-386, 2018.

[6] J. Zhang, L. Ma and Y. Li, "Improved Shuffled Frog-leaping Algorithm for Traveling Salesman Problem," Computer Engineering and Applications, vol. 48, no.11, pp. 47-50, 2012.

[7] D. Wang, "Improved Genetic Algorithm in Application of the TSP Problem," Journal of Liaoning University of Technology (Natural Science Edition), vol.39, no. 4, pp.235-239, 2019.

[8] Q. He, Y. Wu and T. Xu, "Application of Improved Genetic Simulated Annealing Algorithm in TSP Optimization," Control and Decision, vol. 33, no. 02, pp. 219-225, 2018.

[9] L. Zhang, Y. Gao and T. Fei, "Firefly Genetic Algorithm for Traveling Salesman Problem," Computer Engineering and Design, vol.40, no. 7, pp. 1939-1944, 2019.

[10] E. Osaba, X. Yang, F. Diaz, L. G. Pedro and C. Roberto, "An Improved Discrete Bat Algorithm for Symmetric and Asymmetric Traveling Salesman Problems," Engineering Applications of Artificial Intelligence, vol. 48, no. C, pp. 59-71, 2016.

[11] W. Yuan, X. You, S. Liu and Y. Zhu, "Adaptive Simulated Annealing Ant Colony Algorithm for Solving TSP Problem," Computer Applications and Software, vol. 35, no. 2, pp. 261-266, 2018.

[12] D. Luo, L. Zhang and Z. Xu, "Heuristic Simulated Annealing Genetic Algorithm for Traveling Salesman Problem," 2011 6th International Conference on Computer Science & Education (ICCSE), pp. 260-264, 2011.

[13] M. Yao, N. Wang and L. Zhao, "Improved Simulated Annealing Algorithm and Genetic Algorithm for TSP," Computer Engineering and Applications, vol. 49, no. 14, pp. 60-65, 2013.

[14] L. Zhu, Q. Liao and L. Zou, "Resolution of the Traveling Salesman Problem with Precedence Constraint Applying Genetic Algorithm," Journal of South China University of Technology (Natural Science Edition), vol. 32, no. 4, pp. 99-102, 2004.

[15] W. Zhang, Y. Li and X. Zhou, "A State Transition Algorithm for Traveling Salesman Problem with Constraints on the Order of Visiting Cities," 2018 China automation conference (CAC), pp. 286-291, 2018.

[16] S. Li, X. Sun, D. Sun and W. Bian, "Summary of Crossover Operator of Genetic Algorithm," Computer Engineering and Applications, vol. 48, no. 1, pp. 36-39, 2012.

[17] E. Liu, X. Yao, H. Lan and H. Jin, "AGV Dynamic Path Planning Based on Improved Genetic Algorithm and its Implementation," Commputer Integrated Manufacturing Systems, vol. 24, no. 6, pp. 133-145, 2018.

[18] M. Srinivas and L. M. Patnaik, "Adaptive Probabilities of Crossover and Mutation in Genetic Algorithms," IEEE Transactions on Systems, Man and Cybernetics, vol. 24, no. 4, pp. 656-667, 1994.

[19] L. Wang and M. Li, "Application of Improved Adaptive Genetic Algorithm in Mobile Robot Path Planning," Journal of Nanjing University of Science and Technology (Natural Science Edition), vol. 41, no. 5, pp. 627-633, 2017.

[20] C. Yang, Q. Qian, F. Wang and M. Sun, "An Improved Adaptive Genetic Algorithm for Function Optimization," 2016 IEEE International Conference on Information and Automation (ICIA), pp, 675-680, 2016.

[21] J. Jin and Y. Su, "An Improved Adaptive Genetic Algorithm," Computer Engineering and Applications, vol. 41, no. 18, pp. 64-69, 2005.