2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

A Route Planning Strategy for the Automatic Garment Cutter Based on Genetic Algorithm

Wenchao Yu
Dept. Automation of Shanghai Jiao Tong University
Key Lab of System Control and Information Processing
Shanghai, P. R. China
ywc689@sjtu.edu.cn

Linji Lu
Dept. Automation of Shanghai Jiao Tong University
Key Lab of System Control and Information Processing
Shanghai, P. R. China
lulinji@sjtu.edu.cn

***Abstract*—This paper proposes a route planning algorithm for the automatic garment cutter, a machine extensively used in the clothing industry, aiming at reducing the length and improving the smoothness of quick moving route for the cutter. With proper constraints for the cloth segments and knife-down points, the route planning problem is resolved into a generalized travelling salesman problem (GTSP) of the first category, for which an enhanced genetic algorithm is proposed. In this paper, we firstly outline the procedure of the algorithm and discuss some important details, including individual fitness calculation based on the multistage graph problem, a local search algorithm with 2-opt method, etc. Then a position-reservation crossover operator based on dual-relevancy, and an adaptive mutation operator based on population dispersion are proposed, which can accelerate convergence of the algorithm as well as prevent locking into local minima as much as possible. Finally, experimental tests are performed on the GTSP Instances Library and the data of garment CAD files, which demonstrates the effectiveness of our route planning strategy in terms of both solution quality and running time.**

I. INTRODUCTION

With the increasing market competition and growing personalized customer needs, the industries of clothing, shoemaking and bags are confronted with new challenges on the quality and efficiency of production. The cutting equipment, as an important tool for these industries, relates directly to the efficiency of production and the quality of products. Therefore, development of high-performance automatic garment cutter has become the urgent requirement for the development of these industries.

Cutting route optimization is one of the key issues needed to be solved in the research of high-performance automatic garment cutter. Generally speaking, in numerical control machining, cutting route can be divided into two kinds: one is the actual feed path of the knife to cut cloth, leather or other materials; another is the quick moving route which has no cutting effect on the materials [1]. The length of the former is determined by the contours of cloth segments, which cannot be changed after the confirmation of the garment CAD file, while the length of the latter can be shortened with proper optimization methods to accelerate speed of processing and improve productivity. If the garment CAD file is cut according to the order of segment numbers, the performance of the quick moving route is rather poor in terms of both length and smoothness of the route, as shown in Fig. 1. Therefore, the purpose of the paper is to show how to determine the actual knife-down points for each cloth segment from hundreds of feasible knife-down points with an enhanced genetic algorithm, so that the quick moving route connected sequentially by actual knife-down points is shortest.

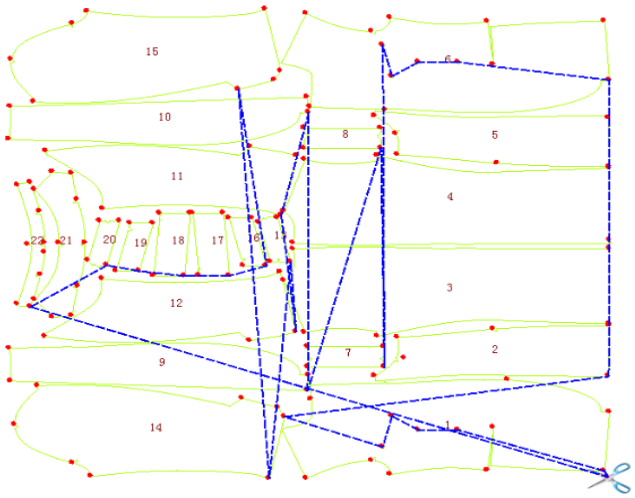

Fig. 1. Route planning problem for automatic garment cutter. The figure shows cloth segment contours (solid green lines) and feasible knife-down points (red dots on segment contours). The dash blue lines give a quick moving route using segment number based strategy. The segment number is marked at the center of each segment. The data comes from the CAD file SAKKO.ISO of Topcut–bullmer Co. Ltd.

A. Cutter Route Planning and Generalized Travelling Salesman Problem (GTSP)

GTSP is an extension of the classical traveling salesman problem, and belongs in the category of NP-hard combinatorial optimization problems. In the 1960s, Henry-Labordere [2], Srivastava [3] and Saksena [4] put forward the GTSP problem almost at the same time. The problem requires that m cities are selected sequentially from the n cities which have been divided into m clusters to form a Hamilton loop, so that the cost of the loop is lowest. According to different requirements for traversing the

clusters, there are two categories of GTSP problems: one is only one city can be visited for each cluster, and another is more than one city can be visited for each cluster. By imposing some constraints on cloth segments and knife-down points, the cutter route planning problem can be converted to the GTSP problem of the first category.

As mentioned above, the cutter route is made up of feasible knife-down points on the cloth segment contour. However, not every interpolation point is suitable as feasible knife-down points, and there are isolated drilling points in some CAD files, such as the segment 1 at the bottom right and the segment 6 at the upper right in Fig. 1. Moreover, we should specify the origin position for the cutter. In order to convert the cutter route planning problem to the GTSP problem, we extend the concept of feasible knife-down points and cloth segments. The generalized knife-down points (GKFP) consist of the following three parts:

a) The start point for cutting. The start point can be either the knife-down point on the first segment contour or the origin point of the cutter.

b) Feasible knife-down points on segment contours. These points are given by the CAD file parsing program.

c) Isolated drilling points. Every drilling point is regarded as a feasible knife-down point.

The general cloth segments (GCS) also consist of three parts: the real cloth segments given by the CAD file, the isolated knife-down points, and the start point for cutting which is not on the first segment contour. With the definition of GKFP and GCS, the cutter route planning problem is converted to the GTSP problem: m GCS correspond to m city clusters, and n GKFP correspond to n cities, and the best cutter route corresponds to the lowest cost Hamilton loop.

B. State of Art for GTSP

The current solution methods for GTSP problems can be roughly divided into accurate solution methods, structural approximation methods, and heuristic optimization methods. Laporte and Norbert resolved the GTSP problem to the integer programming problem and proposed an accurate solution approach for the problem through branch and bound method [5]. Latter Fischetti improved the branch and bound method, providing us with the accurate solution to the GTSP problems in the GTSP instances library with up to 89 clusters and 442 cities [6]. Because of the combinatorial explosion problem of the solution space for the search of best solution, these accurate solution methods based on branch and bound are time-consuming and only applicable to small scale problems. Structural approximation methods, such as $3\rho/2$ algorithm proposed by Slavik [7], usually can solve the problem quickly, but the quality of solution is often unsatisfying. Current research mainly focuses on heuristic optimization methods. Many effective heuristic search methods adopted in the classic travelling salesman problem, such as genetic algorithm [8], particle swarm optimization algorithm [9], simulated annealing algorithm [10], and ant colony optimization algorithm [11], have been adapted and applied to the GTSP problem, thus forming some efficient algorithms for the GTSP problem, typical representatives among which are GI3 algorithm [12], Synder algorithm [13], mrOX algorithm [14] and LNS algorithm [15].

II. Cutter Route Planning Algorithm

A. Profile of Cutter Route Planning Algorithm

As a probability optimization techniques based on biological genetic and evolutionary mechanisms, genetic algorithm is simple in operation, flexible and efficient in search, and with the characteristic of explicit parallelism in execution if compared to other heuristic algorithms. Therefore, the genetic algorithm is suitable for optimization computation of large complex systems, especially the NP-hard combinatorial problems with multi-variable, multiple targets, and poor connectivity in many areas [16]-[18]. The basic idea of genetic algorithm mainly stems from the proposition of "survival of the fittest" in biological evolution. Fig. 2 shows the basic process of genetic algorithm.

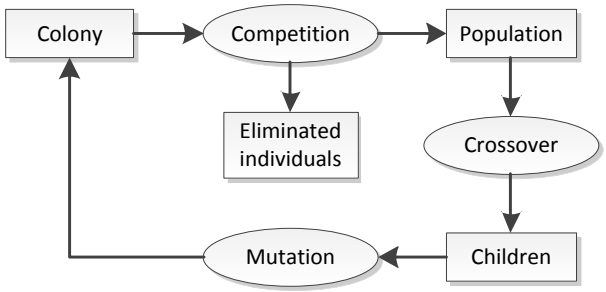

Fig. 2. Basic process of genetic algorithm.

Input information of knife-down points and segments
Compute distance-matrices of cities and clusters
Initiate the colony with N individuals randomly
Calculate fitness for individuals of the initial colony
do:
  Population selection through competition
  Couple the individuals randomly for crossover
  Apply crossover operator for each couple
  for *k:=1 to N*
    Apply mutation operator
    Renew fitness through shortest-path algorithm
    Apply 2-opt local search algorithm
  until: *Termination rules are satisfied*
  Renew best fitness, best route of current colony
  Calculate statistic characteristics of current colony
end

Algo. 1. Profile of cutter route planning algorithm. Note that the knife-down points and segments correspond to cities and clusters in GTSP problem, respectively.

We solve the GTSP problem that stems from the cutter route planning problem using the above mentioned genetic algorithm. On the basis of the existing algorithm, the crossover operator and mutation operator are improved in order to balance the contradiction between accelerating convergence and avoiding premature of the algorithm. Alog. 1 shows the profile of the whole algorithm with pseudo-code.

B. Chromosome coding and Shortest-path algorithm

Every gene corresponds to a generalized cloth segment in the cutter route planning problem and a city cluster in the GTSP problem, and genes with a particular arrangement constitute a chromosome. Therefore, individuals can be represented with chromosome coding. In this paper, we adopt the irregular coding method for chromosome coding, i.e., coding the chromosome with the indices of ordered sequence of generalized cloth segments. Take the virtual simplified CAD file in Fig. 3 as example. It consists of 4 generalized cloth segments and 12 generalized knife-down

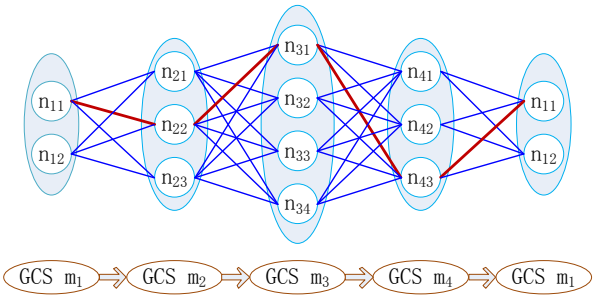

Fig. 3. Multistage graph for individual representation and shortest-path algorithm. GCS is the abbreviation for generalized cloth segment defined in the preceding part and is represented by "m" while generalized knife-down point is represented by "n". The bold line in the figure gives a closed Hamilton loop. The GCS sequence in the figure below shows the chromosome coding for an individual.

points, and the ordered sequence "m_1-m_2-m_3-m_4-m_1" forms the chromosome of an individual, where $m_i \neq m_j (i \neq j)$ and $i \in \{1,2,3,4\}$. Thus every individual defines a cluster of cutting routes, which are of the same segment order but different knife-down points. As a matter of fact, every individual corresponds to a directed acyclic multistage graph shown in Fig. 3 above, on which the individual fitness can be obtained through a shortest-path algorithm. The main idea of the algorithm is backtracking dynamic programming

$$f_k(m_k) = \min_{d_{ij} \in dis\tan ce(m_k, m_{k+1})} \{f_{k+1}(m_{k+1}) + d_{ij}\}, \quad (1)$$

where $f_k(m_k)$ is the shortest distance set between the knife-down points in stage k and the generalized cloth segment of final stage, and $dis\tan ce(m_k, m_{k+1})$ is the distance set of two knife-down points between stage k and stage k+1. Obviously,

$$|f(m_k)| = n_k \quad (2)$$

and

$$|dis\tan ce(m_k, m_{k+1})| = n_k n_{k+1}, \quad (3)$$

where n_k is the number of knife-down points on segment m_k . It is necessary to note that when the start position is set to be the knife-down point on the first cloth segment, the dimension of the first (or last) shortest distance set may be greater than 1. To ensure a closed Hamilton loop, we should deal with the first GCS separately: apply the shortest-path algorithm for each knife-down point in the first (or last) GCS. Therefore, we obtain $f(m_1)$, and the length of the shortest path $f_{\min}$ for an individual is

$$f_{\min} = \min_{f_k \in f_1(m_1)} \{f_k\}. \quad (4)$$

C. 2-opt Local Search

The cutter route planning algorithm adopt the efficient 2-opt local search to fasten convergence and improve solution quality of the algorithm. 2-opt local search algorithm can find the best solution within time complexity of $O(m^2)$ from the 2-opt space, where m is the length of chromosome coding, and 2-opt space is the solution space that made up of all the feasible routes by substituting two new edges in the original route. Fig. 4 and Algo. 2 show the basic idea and pseudo-code of 2-opt local search algorithm, respectively.

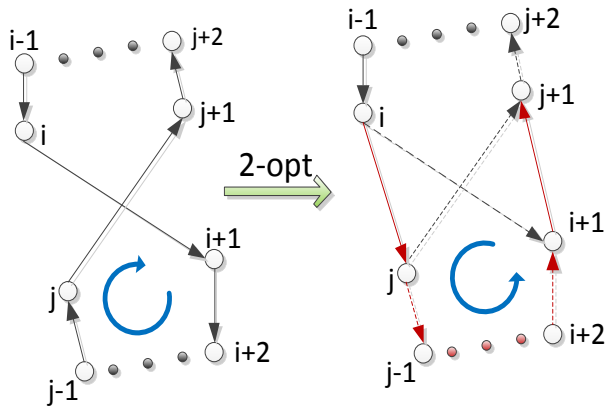

Fig. 4. Basic principle of 2-opt local search. The dash lines in the right figure are the original edges, E(i, i+1) and E(j, j+1), which are replaced by the new ones E(i, j) and E(i+1,j+1), and the clockwise and anticlockwise arrows show that the order of genes between two 2-opt points is reversed.

```
for i:=1 to m-3
    for j:=i+2 to m
        if (d_{i,i+1}+d_{j,j+1}>d_{i,j}+d_{i+1,j+1})
            for k:=0 to (j-i)/2
                swap(x_{j-k}, x_{i+k+1})
                f_new=f_old - ( d_{i,i+1}+d_{j,j+1}-d_{i,j}-d_{i+1,j+1} )
```

Algo. 2. 2-opt local search algorithm

D. Population Selection

Population selection strategy for the cutter route planning algorithm is based on fitness-sorting function and roulette selection. Specifically, sort the colony in ascending order of individual fitness, and then select the individual using roulette wheel according to the following probability:

$$p(i) = \frac{2i}{m(m+1)} \quad ,1 \le i \le m \quad (5)$$

where i is the individual index after sorting, m is the length of chromosome coding. Compared to the traditional method that directly uses individual fitness for population selection, fitness-sorting method is better to prevent the algorithm locking into local minima by always selecting the best

individual with the probability of $\frac{2}{m(m+1)}$.

E. Termination Rules

The termination strategy for the cutter route planning algorithm consists of the following 4 rules: (1) The algorithm has reached the maximum generations; (2) The global optimum has not changed for specified generations; (3) The colony has not evolved towards favorable direction for specified generations; (4) The variety of colony has been diminished and not improved for specified generations. The criterion for rule (3) is the trends of average fitness of the colony. The criterion for rule (4) is the logarithm ratio of the colony fitness dispersion (discussed in detail in part III). Four rules can be used alone or by random combination.

III. Enhanced Genetic Algorithm Operators

A. Enhanced Crossover Operator

The position-reservation method is one of the most effective crossover techniques used in genetic algorithm. Its principal idea is to generate a reservation vector of the same dimension with chromosome coding, and the genes corresponding to bit "1" in the reservation vector are passed on to the offspring directly while the genes corresponding to bit "0" are rearranged with the different genes according to the order of location in the spouse. Fig. 5 shows the basic approach for position-reservation crossover.

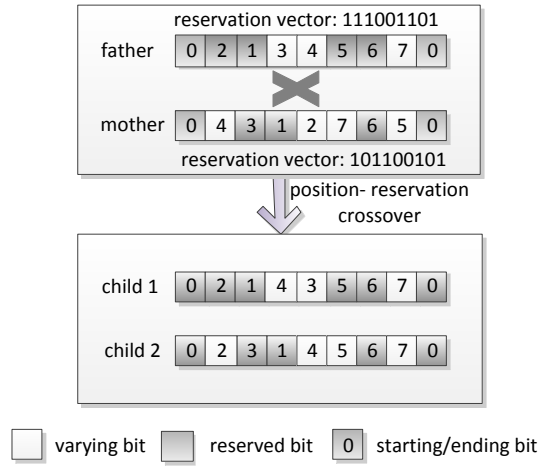

Fig. 5. Basic principle of position-reservation crossover

Thus, it is of critical importance for the selection of the position-reservation vectors. Generally, position-reservation vectors are selected randomly, which cannot guarantee that the superior genes of parents will be able to pass on to their children. So it is likely to result in poor global search ability and slow convergence of the algorithm. Therefore, we improved the crossover operator to avoid unnecessary search in the poor solution spaces. To be more specific, we take the cluster neighborhoods into consideration when selecting position-reservation vectors: neighboring genes of high correlation are selected as reserved bits while neighboring genes of poor correlation are selected as varying bits. In this way, the superior genes are more likely to be inherited by children, and the overall performance of the algorithm can be improved because of the reduction of unnecessary search in the poor solution spaces.

The distance between genes is introduced in order to quantitatively characterize the "correlation" of genes. As we know, every gene corresponds to a generalized cloth segment, and every generalized cloth segment may contain more than one generalized knife-down points. Thus, the *gene-to-gene distance* is categorized as the distance of two point sets. Two typical representatives of definition for distance of point sets are the average distance and the minimum distance, as defined in the formula (6) and formula (7), respectively.

$$d_{ij} = \frac{\sum_{s=1}^{|m_i|}\sum_{t=1}^{|m_j|} dis\tan ce(m_{i_s}, m_{j_t})}{|m_i||m_j|} \tag{6}$$

$$d_{ij} = \min_{\substack{s\in\{1,2,\ldots|m_i|\}\\ t\in\{1,2,\ldots|m_j|\}}} \{dis\tan ce(m_{i_s}, m_{j_t}\} \tag{7}$$

$|m_i|$ is the number of knife-down points on gene i in the above formulae. Compared to average distance, which is the measurement of average distance of two different point sets, minimum distance is more feasible for the gene-to-gene distance because we always seek to find the optimal individual fitness value in the cutter route planning problem. With the gene-to-gene distance, the definition of *partial correlation* is given in the following formula (8)

$$Rm_{ij} = \frac{1}{m-1}(1-\frac{d_{ij}}{\sum_{k=1}^{m} d_{ik}}), \tag{8}$$

where, m is the number of genes in an individual, i.e., the number of generalized cloth segments. Obviously,

$$\sum_{j=1}^{m} Rm_{ij} = 1, \tag{9}$$

$$Rm_{ij} \neq Rm_{ji}. \tag{10}$$

That is, partial correlation has the property of summation normalization and asymmetry. To overcome the negative influence caused by the asymmetry property of partial correlation, through the combination and normalization of partial correlation, we further define the *dual correlation*

$$Rb_{ij} = \frac{[(a_i Rm_{ij} + b_i)(a_j Rm_{ji} + b_j)]^2}{[\sum_{k=1}^{m}(a_i Rm_{ik} + b_i)(a_k Rm_{ki} + b_k)][\sum_{k=1}^{m}(a_j Rm_{jk} + b_j)(a_k Rm_{kj} + b_k)]}, \tag{11}$$

where, a_i and b_i are the coefficients for standardized linear transformation of partial correlation, i.e.,

$$a_i = \frac{1}{\max_{j\in\{1,2,\ldots|m_j|\}}\{Rm_{ij}\} - \min_{j\in\{1,2,\ldots|m_j|\}}\{Rm_{ij}\}} \tag{12}$$

$$b_i = -\frac{\min\limits_{j\in\{1,2,\ldots,|m_j|\}}\{Rm_{ij}\}}{\max\limits_{j\in\{1,2,\ldots,|m_j|\}}\{Rm_{ij}\} - \min\limits_{j\in\{1,2,\ldots,|m_j|\}}\{Rm_{ij}\}} \quad (13)$$

The pseudo-code of Algo. 3 clarifies the procedure for the position-reservation crossover algorithm based on dual correlation. It is noted that gene-to-gene distance and partial correlation are calculated off-line during the initiation stage of the genetic algorithm and the results are saved for future iteration. Even though dual correlation must be recalculated on-line for each generation due to relatively higher memory cost of off-line method, the calculation can be achieved in O(1) of time complexity by using the normalized partial correlation. So the time cost for correlation calculation is very low compared to the whole iteration of generations. Moreover, the number of reserved bits is chosen randomly from the interval [m/3, 2m/3], where m is the length of chromosome coding.

Calculate gene-to-gene distances, partial correlations
Normalize and save partial correlations
Couple all the N individuals randomly
for *i:=1 to N/2*
  Calculate the lists of dual correlations of parents
  Sort the lists of dual correlations in descending order
  Generate the number of reserved bits for parents
  Generate the reserved bits for parents respectively
  Crossover using position-reservation method
end
Renew the colony by substituting children for parents

Algo. 3. Enhanced position-reservation crossover algorithm

B. Enhanced Mutation Operator

The mutation probability is a key parameter in mutation operator. Normally, fixed mutation probability is used in genetic algorithm, which does not take into account the evolution process of a colony. Therefore, we propose an adaptive mutation strategy based on population dispersion and individual fitness: increase mutation probability to escape the trap of local minima when the individual fitness of the colony tends to be the same, and decrease mutation probability to protect the superior individuals when the distribution of individual fitness is very dispersed; apply smaller mutation probability to the individuals of better fitness than average to protect superior genes, and apply larger mutation probability to individuals of worse fitness than average to accelerate the evolution of inferior individuals. Through this adaptive adjustment strategy in mutation probability, the superior individuals are well protected while the evolution speed of the inferior individuals is accelerated, which nicely balances the inherent contradictions of genetic algorithms between fastening convergence and preventing locking into local minima.

The adaptive mutation probability is determined by

$$p = \begin{cases} p_{\min} & f \le f_{ave} \\ p_{\max} - \dfrac{(p_{\max} - p_{\min})(f_{\max} - f)}{f_{\max} - f_{ave}} & f > f_{ave} \end{cases}, \quad (14)$$

where $p_{\min}$ and $p_{\max}$ denote the lower bound and the upper bound of mutation probability respectively, f denotes individual fitness, $f_{\min}$, $f_{\max}$ and f_{ave} denote the minimum, the maximum, and the average value of the colony fitness respectively. It is noted that the fitness value is negatively correlated to the superiority of the individual, so $f_{\min}$ corresponds to the most superior individual.

The adaptive mutation formula (14) takes into account of the relation of fitness value between the isolated individual and the entire colony. However, it fails to reflect the dynamic change during the evolution of the colony, in which process the variety of the colony tends to decline. In order to prevent the diversity of the colony from declining quickly and enlarge search space of the algorithm as much as possible, we propose a strategy to dynamically adjust the lower and upper bounds of mutation probability according to the average fitness and standard deviation of the colony fitness. In order to avoid the negative influence by the order of magnitude or the unit of individual fitness for different colonies, we use the logarithm ratio between standard deviation and average value of colony fitness

$$\gamma = \log_{10}\frac{\sigma}{f_{ave}} \quad (15)$$

to evaluate the degree of dispersion for the colony, where σ and f_{ave} are the standard deviation and average value of the colony fitness respectively. The detailed implement method is shown in table I, in which we divide the scope of fitness dispersion into 6 fuzzy sets -- EG, G, LG, LS, S, and ES -- and the membership functions for these fuzzy sets are given in the second column of the table.

TABLE I
BOUNDS FOR ADAPTIVE MUTATION PROBABILITY

Colony Fitness Dispersion γ	Membership Function μ	Lower Bound	Upper Bound
extremely great (EG)	$[1+e^{5(2-\lambda)}]^{-1}$	0.00	0.00
great (G)	$[1+e^{12.5(\gamma-1.5)^2}]^{-1}$	0.01	0.05
a little great (LG)	$[1+e^{12.5(\gamma-0.5)^2}]^{-1}$	0.05	0.10
a little small (LS)	$[1+e^{12.5(\gamma+0.5)^2}]^{-1}$	0.10	0.20
small (S)	$[1+e^{12.5(\gamma+1.5)^2}]^{-1}$	0.20	0.50
extremely small (ES)	$[1+e^{5(\gamma+2)}]^{-1}$	0.50	1.00

Renew average fitness and colony standard deviation
Calculate membership function value for current colony
Decide lower and upper bounds for mutation probability
for i:=0 to N
  Determine repeated times of 4-opt insertion operations
  Calculate mutation probability p_i for individual i
  Apply 4-opt insertion operations with probability p_i
end

Algo. 4. Enhanced mutation algorithm

Basic operations of the enhanced mutation operator adopted in this paper are based on 4-opt insertion. Specifically, we choose a gene and a position for insertion

randomly, and then insert the chosen gene after the chosen position. This process is performed several times for each mutation individual to improve the randomness of mutation. The specific repeated times of the 4-opt insertion operation are determined randomly according to the length of chromosome coding. Algo. 4 shows the pseudo-code for the basic procedure of enhanced mutation operator.

IV. SIMULATIONS AND APPLICATIONS

The whole algorithm is implemented with C++ on a personal computer with 2.2GHz Intel T6600 CPU, 2GB RAM, and Windows 7 operating system. There are two types of simulation data: one is from the standard GTSP instances library, another is from the parsing data of garment CAD files. The computational results of the former provide us with a way to compare performances of different GTSP algorithms while the computational results of the latter show the successful application of the algorithm in cutter route planning problem.

A. Simulation of the Enhanced Crossover Operator

For the test of crossover operators, we choose 37 instances among 11berlin52 ~ 84fl417 and 115u674 ~ 217vm1084 of the standard GTSP instances library. As shown in Fig. 6, we compare three crossover operators: the basic crossover operator using randomly chosen reserved bits, the enhanced crossover operator using average gene-to-gene distance and the enhanced crossover operator using minimum gene-to-gene distance. We record the generation of the first appearance of the best global solution for each crossover operator applied to Algo. 1, and normalize the results with respect to every instance, hence the Fig. 6 above.

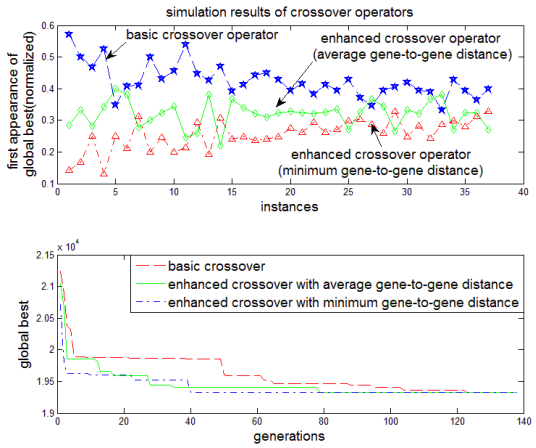

Fig. 6. Simulation results of crossover operators. The figure above shows the first appearance of global best for different instances. The results are normalized corresponding to each instance. The figure below shows the convergence process of the algorithm. The dot line corresponds to the basic crossover operator with randomly chosen reserved bits; the solid line corresponds to enhanced crossover operator using average gene-to-gene distance; the dash line corresponds to another enhanced crossover operator using minimum gene-to-gene distance.

The Fig. 6 below shows the convergence process of the algorithm for the three crossover operators using parsing data of a garment CAD file named "SAKKO.ISO" from Topcut–bullmer Co. Ltd. We can know from Fig. 6 that the enhanced crossover operators tend to find the global best more quickly than the basic crossover operator, and the minimum gene-to-gene distance is more effective than the average gene-to-gene distance for GTSP problems.

B. Simulation of the Enhanced Mutation Operator

To test the performance of mutation operators, we select 7 larger instances among 115u574 ~ 217vm1084 of the standard GTSP instances library. Fig. 7 shows the simulation results of the basic mutation operator using fixed mutation probability of 0.1, and the enhanced mutation operator using the adaptive mutation probability proposed in the paper. The figure above gives the percentage error of different instances while the figure below shows the running time of corresponding instances. Note that the running time is normalized with respect to each instance. According to the simulation results, the adaptive mutation operator can help find better solutions for large scale problems while the running time increases justly slightly compared to the basic mutation operator.

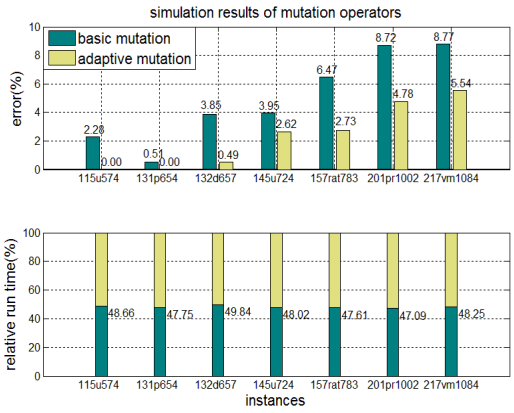

Fig. 7. Simulation results of mutation operators. The figure above shows the percentage error for different instances. The figure below shows the relative running time of the algorithm. Note that the running time is normalized with respect to each instance. The bar in dark color corresponds to the basic mutation with fixed mutation probability; the bar in light color corresponds to the adaptive mutation based on dispersion of colony fitness.

C. Comparison with Other Algorithms

Because the cutter route planning algorithm (CRPA) is based on the GTSP problem, we compare our algorithm with some representative GTSP algorithms to show the effectiveness of our method. For convenience, we select 18instances widely used for comparison in papers on GTSP algorithms, and compare the percentage error and running time of different algorithms. Table II shows the comparison results. The best solutions of these instances are provided by BC algorithm [6]. Though the running time in Table II cannot reflects the performance of different algorithms faithfully because of the hardware/software environment differences of test platforms. Nevertheless, we can use it as references for different instances of a specific algorithm. Seen from Table II, the cutter route planning algorithm gains obvious advantages over GI^3 algorithm (GI^3) [12], ant colony algorithm (ACO) [11], and simulated annealing algorithm (SA) [10] in both solution quality and running time.

TABLE II
COMPARISON OF DIFFERENT ALGORITHMS

GTSPLIB INTSTANCES	BC		GI^3		MROX		LNS		SA		ACO		CRPA	
	best	CPU	error	CPU	error	CPU	error	CPU	error	CPU	error	CPU	error	CPU
30kroA150	11018	100.3	0	17.8	0	0.98	0	5.95	0.16	152	5.99	104	0	0.725
30kroB150	12196	60.6	0	14.2	0	0.98	0	5.02	0.02	78	6.02	67	0	0.595
31pr152	51576	94.8	0.47	17.6	0	0.97	0	5.24	1.12	79	1.6	69	0	0.58
32u159	22664	146.4	2.6	18.5	0	0.98	0	5.58	1.9	89	8.68	75	0	0.819
39rat195	854	245.9	0	37.2	0	1.37	0	11.01	1.09	198	5.86	145	0	2.072
40d198	10557	763.1	0.6	60.4	0	1.63	0	10.15	0.53	112	10.77	99	0	2.061
40kroA200	13406	187.4	0	29.7	0	1.66	0	10.41	6.02	107	10.77	99	0	1.652
40kroB200	13111	268.5	0	35.8	0.05	1.63	0	10.81	0.38	108	8.34	99	0	1.717
45ts225	68340	37875.9	0.61	89	0.14	1.71	0.04	31.45	1.57	325	5.38	223	0	3.161
46pr226	64007	106.9	0	25.5	0	1.54	0	8.25	2.7	130	7.51	124	0	2.492
53gil262	1013	6624.1	5.03	115.4	0.45	3.64	0.14	24.34	5.24	142	15.92	148	0.12	4.408
53pr264	29549	337	0.36	64.4	0	2.36	0	18.27	1.87	146	12.06	150	0	4.387
60pr299	22615	812.8	2.23	90.3	0.05	4.59	0	21.25	7	165	15.97	184	0	6.493
64lin318	20765	1671.9	4.59	206.8	0	8.08	0	26.33	5.73	166	13.57	199	0.06	8.699
80rd400	6361	7021.4	1.23	403.5	0.58	14.58	0.42	32.21	10.4	225	21.67	299	0.46	21.746
84fl417	9651	16719.4	0.48	427.1	0.04	8.15	0	31.63	9.95	282	10.14	345	0	5.91
88pr439	60099	5422.8	3.52	611	0	19.06	0	42.55	13.1	276	16.14	368	0	31.398
89pcb442	21657	58770.5	5.91	567.7	0.01	23.43	0.19	42.53	11.2	253	19.48	376	0.04	24.048

BC algorithm gives the best solution of each instance, and "error" is the percentage error value relative to the best solution. The "CPU" represents the running time of the algorithm with unit of second (s) . The algorithms shown in the table from left to right are: branch and bound algorithm, GI^3 algorithm, mrOX algorithm, large neighborhood crossover algorithm, simulated annealing algorithm, ant colony algorithm, and cutter route planning algorithm. The results presented above are the mean results of 5 attempts for each instance. Because of the hardware/software environment differences of test platforms, the above listed CPU time is for reference only.

Moreover, the performance of our algorithm can also attain or surpass the performance of some other complex algorithms base on the genetic algorithm, such as mrOX algorithm (MROX) [14] and large neighborhood crossover algorithm (LNS) [15].

D. Applications in Automatic Cutter Route Planning

We apply the cutter route planning algorithm proposed in this paper to the practical problem encountered during our endeavors for the development of an automatic garment cutter for Topcut–bullmer Co. Ltd. Currently, two strategies for cutter quick moving route are widely adopted in this industry: segment number based strategy, and "Z" order based strategy, as shown in Fig. 1 and Fig. 8, respectively. The "Z" order based strategy tends to choose the nearest feasible knife-down point from current position as the next cutter position. Fig. 9 shows GA based strategy of the quick moving route for the cutter proposed in this paper. The figure is taken from the user graphic interface (GUI) program of the automatic garment cutter implemented with

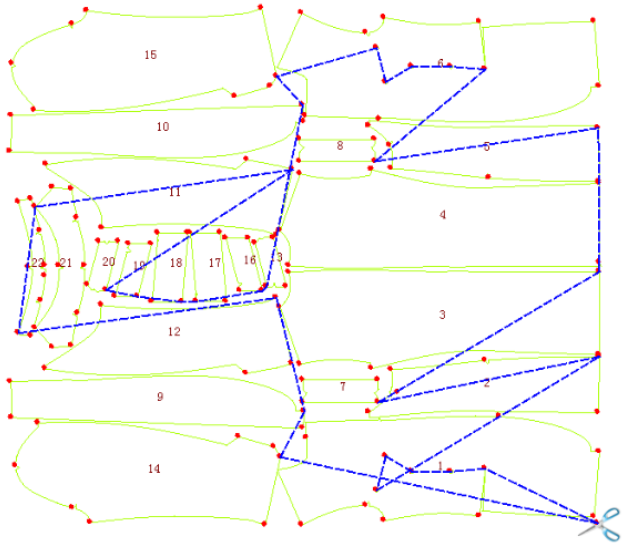

Fig. 8 . Quick moving route using the "Z" order based strategy

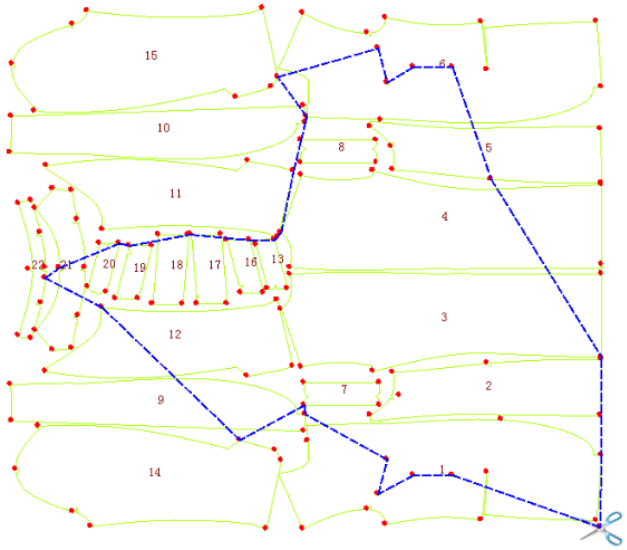

Fig. 9 . Quick moving route using the GA based strategy of this paper

PyQt. Table III lists the comparison results of GA base strategy and conventional strategies. The CAD file used in these tests is "SAKKO.ISO", which contains 22 cloth segments, 119 feasible knife-down points, 8 drilling points, and a starting point outside of the contours of cloth segments. Therefore, the number of GCS is 31, and the number of GKDP is 128. The running time of the cutter route planning algorithm is 0.461 seconds. From Fig. 1 and Fig. 8, we know that the route planning strategy based on genetic algorithm can shorten the length of quick moving route for the cutter by reducing redundant reciprocating motions to maximum extent with relatively low time costs, thus improving the working efficiency of the automatic cutter.

TABLE III
COMPARISON WITH CONVENTIONAL STRATEGIES

	segment number based strategy	"Z" order based strategy	GA based strategy
Route Length	45347	38291	19203
Length Reduction	57.7%	49.8%	0
Machining Time	580.7	559.9	497.4
Time Reduction	14.3%	11.1%	0

The "Route Length" is represented in unit of millimeter (mm), and the "Machining Time" is represented in unit of second (s). The "Length/Time Reduction" is the ratio of the value between the reduced costs and the original costs. Default parameters for configuration of the cutter are adopted. The machining time is the mean value of 3 trials for the CAD file SAKKO.ISO of Topcut–bullmer Co. Ltd.

V. CONCLUSIONS

We propose an improved genetic algorithm for the route planning problem of the automatic garment cutter in this paper. The route planning problem is transformed into the GTSP problem by applying some constraints on knife-down points and cloth segments. Then an adaptive genetic algorithm for the transformed GTSP problem is introduced. The algorithm resolves the chromosome coding into the coding of cloth segment sequence through a variant method of dynamic planning, and evolves by generations through population selection, crossover, mutation and other basic procedures of genetic algorithm. The termination of the algorithm is controlled by four different conditions to ensure the solution quality, and a 2-opt local search strategy is applied to strengthen the search ability. The main efforts of this paper lie in the enhanced operators for crossover and mutation.

For the crossover operator, the partial-correlation and dual-correlation of genes are defined, and the reservation vector is generated based on the dual-correlation so that the superior genes can be passed on to children, thus fastening the convergence of the algorithm. For the mutation operator, we propose a strategy to adaptively change the mutation probability based on the colony dispersion and individual fitness, which can protect the superior individuals and prevent locking into local minima for the algorithm at the same time.

Finally, simulations are performed on both the standard GTSP instances library and the garment CAD files. The simulation results demonstrate the effectiveness of our algorithm and its successful application to the garment cutter route planning problem.

REFERENCES

[1] J. C. Chen, Research on strategy of motion optimization and real-time complex trajectory control in multi-axis coordinated CNC machining," Ph.D. dissertation, Dept. Mech. Eng., Shanghai JiaoTong Univ., Shanghai, 1991.

[2] A. L. Henry-Labordere, "The record balancing problem: A dynamic programming solution of a generalized traveling salesman problem," *Revue Francaise D Informatique DeRecherche Operationnelle*, vol. 3, no. 2, pp. 43-49, 1969.

[3] S. S. Srivastava, S. Kumar, and R. C. Garg, "Generalized traveling salesman problem through n sets of nodes," *CORS Journal*, vol. 7, pp. 97-101, July 1969.

[4] J. P. Saksena, "Mathomatical model of scheduling clients through welfare eagencies," *CORS Journal*, vol. 8, pp. 185-200, Aug 1970.

[5] G. Laporte, Y. Gilbert, and P. Pelletier, Generalized travelling salesman problem through n sets of nodes: An integer programming approach. Montréal: Université de Montréal, 1980.

[6] M. Fischetti, J. J. S. Matteo, and P. Toth, "A branch-and-cut algorithm for the symmetric generalized traveling salesman problem," *Operations Research*, vol. 45, no. 3, pp. 378-394, 1997.

[7] P. Slavik, "On the approximation of the generalized traveling salesman problem," Department of Computer Science, SUNY-Buffalo, Tech. Rep. 1997.

[8] G. Gutin and D. Karapetyan, "A memetic algorithm for the generalized traveling salesman problem," *Natural Computing*, vol. 9, no. 1, pp. 47-60, Sept. 2010.

[9] M. F. Tasgetiren and P. N. Suganthan, "A discrete particle swarm optimization algorithm for the generalized traveling salesman problem," in *9th Annual Conference on Genetic and Evolutionary Computation*, 2007, pp. 158–167.

[10] C. C. Skiscim and B. L. Golden, " Optimization by simulated annealing: a preliminary computational study for the TSP," in *15th Conference on Winter Simulation*, 1983, pp. 523–535.

[11] X. Song, B. Li, and H. Yang, "Improved ant colony algorithm and its applications in TSP," in *6th International Conference on Intelligent Systems Design and Applica-tions*, 2006, pp. 1145–1148.

[12] J. Renaud and F. F. Boctor, "An efficient composite heuristic for the symmetric generalized traveling salesman problem," *European Journal of Operational Research*, vol. 108, no. 3, pp. 571–584, 1998.

[13] L. V. Snyder and M. S. Daskin, "A random-key genetic algorithm for the generalized traveling salesman problem," *European Journal of Operational Research*, vol. 174, no. 1, pp. 38–53, 2006.

[14] J. Silberholz and B. Golden, "The generalized traveling salesman problem: a new genetic algorithm approach," in *Extending the Horizons: Advances in Computing, Optimization and Decision Technologies* , New York: Springer, 2007, pp. 165–181.

[15] B. Bontoux, C. Artigues, and D. Feillet, "A memetic alogrithm with a large neighbourhood crossover operator for the generalized travelling salesman problem," *Computers & Operations Research*, vol. 37, no. 11, pp. 1844-1852, Nov. 2010.

[16] M. D. Vose, The Simple Genetic Algorithm: Foundations and Theory. Cambridge, MA: MIT Press, 1998.

[17] K. K. F. Man, K. S. Tang, and S. Kwong, *Genetic Algorithms: Concepts and Designs*. Berlin: Springer, 1999.

[18] G. Winte, J. Periaux, M. Galan, and P. Cuesta, *Genetic Algorithms in Engineering and Computer science*. New York: Wiley, 1995.

[19] G. Reinelt, "TSPLIB --a traveling salesman problem library," *ORSA Journal on Computing*, vol. 3, no. 4, pp. 376-384, 1991