Operators of the Two-Part Encoding Genetic Algorithm in Solving the Multiple Traveling Salesmen Problem

Article · November 2011

DOI: 10.1109/TAAI.2011.64

2 authors, including:

Shih-Hsin Chen
Cheng Shiu University
44 PUBLICATIONS **712** CITATIONS

SEE PROFILE

Operators of the Two-Part Encoding Genetic Algorithm in Solving the Multiple Traveling Salesmen Problem

Shih-Hsin Chen
Department of Electronic Commerce Management
Nanhua University
Chiayi 62248, Taiwan (R.O.C.)
Email: shihhsin@mail.nhu.edu.tw

Mu-Chung Chen
Department of Computer Science and Information Technology
Wufeng University
Chiayi 62153, Taiwan (R.O.C.)
Email: muchung.chen@wfu.edu.tw

***Abstract*—The multiple traveling salesmen problem (mTSP) considers the m salesmen to visit n cities. This problem involves the assignment of the salesmen to some locations and we have to optimize the sequence within the route, so it is even harder than the traveling salesman problem (TSP) in nature. As a result, there are some algorithms used to solve the mTSP when the problem size is large. Particularly, genetic algorithm (GA) is quite famous in solving this problem while the problem size is large. When we compare the major existing encoding methods for mTSP, the best approach could be the two-part chromosome encoding due to its solution space is the smallest. The two parts are responsible for the sequence and the number of cities should be visited by each salesman. However, because the two-part chromosome technique is the recently proposed encoding method, the better combination of the crossover operators and mutation operators have not studied for this encoding method. As a result, this paper investigates the genetic operators could be used for this purpose by design-of-experiments (DOE). The appropriate genetic operators are suggested in this paper and it could be used to applied in the GA which employs the two-part chromosome encoding technique.**

Keywords: mTSP, two-part chromosome, crossover operators, nutation operators

I. Introduction

The multiple traveling salesmen problem (mTSP) is of importance to model daily life situations. For example, managers like to assign the m salesmen to visit n locations everyday and each location must be visited by a salesman whereas $n > m$. Salesmen also need to optimize the traveling sequence in the route. It directly leads to the traveling cost and time in the trip. In addition, mTSP could be modeled into different problems [2]. mTSP could be formulated in the area of production scheduling [5], [21], [26], workforce planning [29], [28], transportation planning [1], [10], [27], the design of the global navigation satellite system surveying networks [22], and so on. As a result, the mTSP is important and to need more efforts on it since this problem has not received enough attentions.

From above descriptions of mTSP, this problem is more complicated than the traveling salesman problem (TSP). Even though the mTSP could be solved by the exact algorithms [7], [12], [15], exact algorithms may not tackle with the large size problems efficiently. In order to solve large size mTSP, genetic algorithm (GA) is one of the commonly used algorithms. There are four major variants of the GA, including the one-chromosome [25], two-chromosome [13], [19], two-part chromosome [4], and grouping genetic algorithms [3], [6], [11], [20], [23]. In the four major techniques, the best approach could be the two-part chromosome technique due to it takes less solution space to do the explorations. Thus this study selects this encoding technique to solve the mTSP.

Two-part chromosome encoding is the very recently proposed method. To the best of our knowledge, there is no paper which studies the best genetic operators for the two-part chromosome. In [4], they only suggested the ordered and one-point asexual crossover in the first part and second part of the two-part chromosome. The mutation operators are not mentioned in their paper. Moreover, there is no subsequent papers which examine the genetic operators for two-part chromosome. As a result, this paper investigates those genetic operators that could be used for this purpose by design-of-experiments (DOE). After the appropriate genetic operators are examined, it is beneficial for the researchers employ the two-part chromosome encoding technique when they solve mTSP.

The rest of the paper is organized as follows. Section II presents the two-part chromosome encoding and we introduce the genetic operators for this encoding approach in Section III. Later on, we conduct extensive experiments to test on some benchmark instances from TSPLIB and the parameter configuration in GA. We could obtain which genetic operators could evolve to an optimal (or good) solution in Section IV. Finally, Section V gives the conclusions and the future research.

II. Two-Part Chromosome Encoding Technique

According to Carter and Ragsdale[4], they evaluate the size of the solution space of the three major encodings for mTSP. When we have n cities and m salesmen, the solution space of one chromosome needs $(n + m - 1)!$. Two chromosome approach takes $n!m^n$ and the size of the two-part chromosome is $n!\binom{n-1}{m-1}$. Thus, the solution space of the two-part chromosome is smaller than the other two approaches. In other words, GA would be more efficient when this encoding technique is utilized. It is the reason why this research employs this approach depicted in Fig 1.

Figure 1 illustrates the encoding of n = 15 and m = 3 and there are two distinct parts. The first part of the chromosome represents the permutation of the n cities. The second part of the chromosome shows the number of cities assigned to each m salesmen so that its chromosome length is m. The total sum of the m genes is equal to the number of cities to be visited. Take Fig. 1 for example, the first salesman visits the first 6 cities in the first part chromosome, i.e., the city 7, 8, 10, 15, 12 and 13. Then, the salesman No. 2 goes to 4 cities after the first salesman, which are 5, 2, 14 and 3. Finally, the number of assigned cities for the third salesman is 5 and he/she needs to visit city 11, 1, 6, 9 and 4. So the total sum is 15 when we summarize the value of the three genes in the second part chromosome.

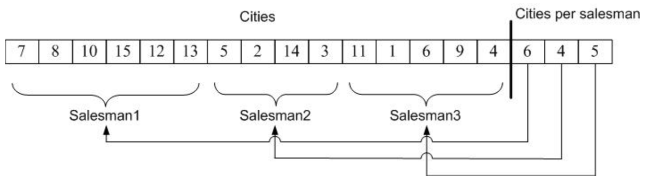

Fig. 1. An example of the two-part chromosome representation for 15 cities and three salesmen

Generally speaking, the two-part chromosome considers the sequence of each salesman and the number of assigned city for each person. When we like to develop genetic operators, we could divide them into this two parts first since they stand for different purpose. Then, the genetic operators are designed separately for the two different parts. It may keep the simplicity when we design new genetic operators. The following section discusses the genetic operator for the two-part chromosome.

III. Genetic Operators for the Two-Part Chromosome Encoding

Because we like to examine the crossover and mutation operators could be used in the two-part chromosome, this section introduces some operators for them in both parts. As a result, there are 4 combinations between the genetic operators (crossover and mutation operator) and the two parts in the chromosome. Each combination is discussed separately in Section III-A to Section III-D.

A. *Crossover Operators for the first part chromosome*

The first part of the two-part chromosome is the permutation of the cities. We could utilize two well-known crossover operators, including the two-point central crossover [17] and cycling crossover [8], [24], [18].

The two-point central crossover is depicted in Fig 2. We select a pair of two cut points randomly, named C_1 and C_2 where where $C_1 < C_2$. The genes located outside the range between C_1 and C_2 are copied from the parent solution into the offspring. The genes within the C_1 and C_2 are copied according to the sequence in the other parent solution.

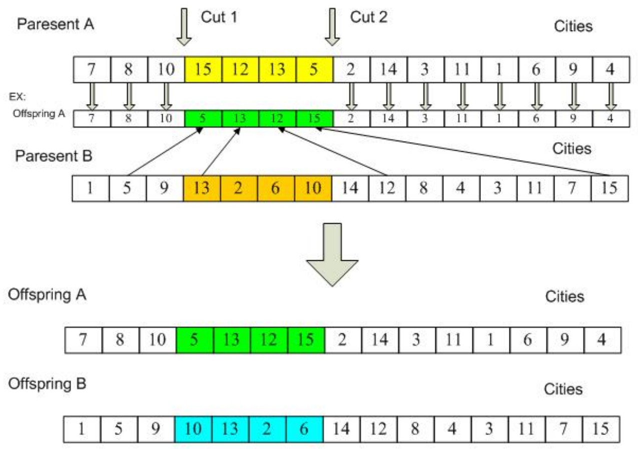

Fig. 2. The two-point central crossover

When it comes to the cycling Crossover, we illustrate this method in Fig 3. The procedure of the cycling crossover is as follows.
(1) Pick a random position in parent A.
(2) Consider the corresponding element in parent B.
(3) Consider the same element in parent A.
(4) Continue this procedure 2-3 until we get an element of parent B that is the first element we considered in parent A. Then, we can get two offsprings after permutation these considering elements for parents A and B.

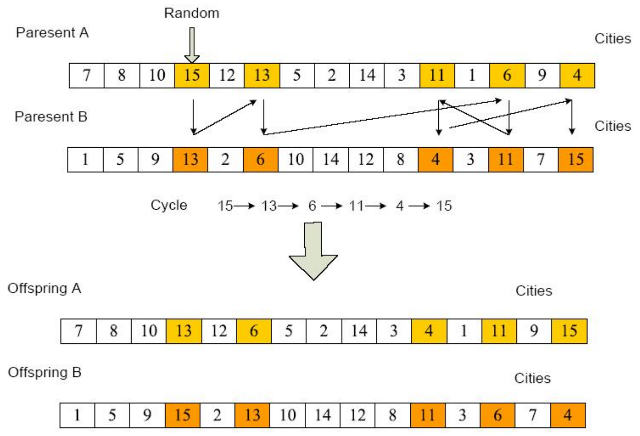

Fig. 3. The cycling crossover

By using the two-point central crossover and cycling Crossover, we could form a new sequence by these crossover operators in the first part of the two-part chromosome.

B. Crossover operators for the second part chromosome

We design two crossover operators which could exchange the genetic information and to maintain the feasibility of the n cities visited by the m salesmen in the second part of the two-part chromosome. When we design the crossover operators for the second part of the two-part chromosome, we should keep in mind about the total sum of the m genes should be equal to n cities. Two kinds of arithmetic crossover operators are considered and we will ensure the feasibility after a new chromosome is produced.

The first method is named arithmetic crossover 1 (ARC1) which is modified from [14]. The original method is used to solve the continuous problem while we transform it into an integer solution space. Suppose there are two chromosomes X_i and X_j selected in the population and they are the second part of the chromosome, a random value $\alpha \in (0,1)$ is generated to decide the weighted-sum value inherited from the two parents. So when we calculate the weighted-sum at the position $[k]$ of chromosome X_i and X_j, the new gene value is computed as Eq. 1 and Eq. 2, respectively.

$$X_{i[k]} = \lfloor \alpha * X_{i[k]} + (1-\alpha) * X_{j[k]} \rfloor \qquad (1)$$

$$X_{j[k]} = \lfloor \alpha * X_{j[k]} + (1-\alpha) * X_{i[k]} \rfloor \qquad (2)$$

where $k = n+1, \ldots, n+m$ and $n > m$. It is noticeable that we take the floor value of the computation results. We could obtain the integer value which represents the assigned cities for the salesmen. Although X_i and X_j may not be feasible, we check them after we introduce the other crossover operator for the second part chromosome.

The second method is named arithmetic crossover 2 (ARC2) which is also mentioned in [14]. The random value α is remain applied to decide the weighted value inherited from the two parents. When we want to mate two chromosomes X_i and X_j at position $[k]$, the calculations are listed in Eq. 3 and Eq. 4.

$$X_{i[k]} = \lfloor X_{i[k]} + \alpha * (X_{i[k]} - X_{j[k]}) \rfloor \qquad (3)$$

$$X_{j[k]} = \lfloor X_{j[k]} + \alpha * (X_{i[k]} - X_{j[k]}) \rfloor \qquad (4)$$

After both X_i and X_j are generated by the ARC1 or ARC2, we need to check the feasibility of the chromosomes by the following Eq. 5 and Eq. 6.

$$\Delta(X_i) = n - \sum_{k=n+1}^{n+m} X_{i[k]} \qquad (5)$$

$$\Delta(X_j) = n - \sum_{k=n+1}^{n+m} X_{j[k]} \qquad (6)$$

If $\Delta(X_i)$ (or $\Delta(X_j)$) is not equal to zero, it means the solution X_i (or X_j) is not feasible. A repair method is used. We randomly distribute the required cities $\Delta(X_i)$ (or $\Delta(X_j)$) to the genes on X_i (or X_j). Thus it ensures that the new solution keeps the feasibility.

C. Mutation Operators of the First Part Chromosome

The mutation operator aims at increasing population diversity in search. For simplicity, we adopt the famous swap and inverse mutation operators in the first part chromosome. In swap mutation (See Fig 4), two random points C_1 and C_2 determine the two genes should be exchanged to the new positions. In our example, the two random points C_1 and C_2 are located at position 4 and position 7. The job 15 is moved to the position 7 and job 5 is changed to the position 4.

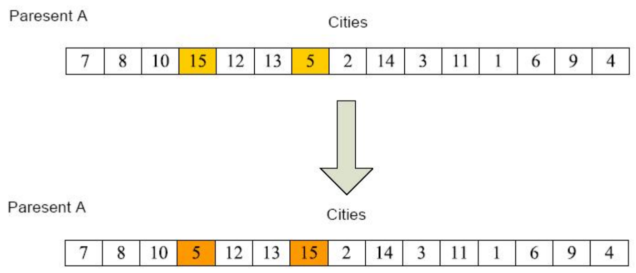

Fig. 4. Swap mutation in a given sequence

In inverse mutation operator (See Fig 5), two random points C_1 and C_2 are generated as the position where $C_2 > C_2$. Then, we inverse the jobs between C_1 and C_2.

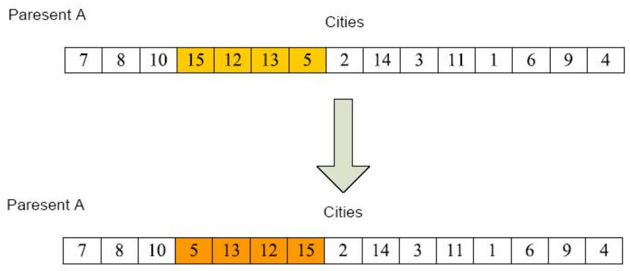

Fig. 5. Inverse mutation in a given sequence.

D. Mutation Operators for the Second Part Chromosome

When we mutate a solution X_i of the second part chromosome, we randomly select two positions P_1 and P_2 where both of them are larger than $n+1$ and less than $n+m+1$. P_1 and P_2 are definitely not equal and it is not necessary that $P_2 > P_1$. We will move some cities from the salesman at position P_2 to the other salesman at position P_1. The random value α is drawn from the uniform distribution to decide the percentage of how many cities should be moved. Eq. 7 and Eq. 8 represent the computation of the mutation operator used for the second part chromosome. Eq. 8 automatically satisfies the constraint of each salesman would visit at least one city.

$$X_{i[P_1]} = X_{i[P_1]} + \lfloor (X_{i[P_2]} * \alpha * 0.5) \rfloor \tag{7}$$

$$X_{i[P_2]} = X_{i[P_2]} - \lfloor (X_{i[P_2]} * \alpha * 0.5) \rfloor \tag{8}$$

Above sections demonstrate the genetic operators could be used in the two-part chromosome when we do the crossover and mutation procedures. The next section shows the experimental results of the best combination of these genetic operators.

IV. Experimental Results

A. Settings

To validate the genetic operators used in the two-part chromosome, this paper conducts extensive experiments on benchmark instances on TSPLIB[1]. Across all the experiments, each instance was replicates 30 times on each combination of methods and problems. The problem combinations are the number of salesmen and the number of cities, which are m = 3, 15, and 20, and n = 51 and 150 [2]. We set the final point in these instances to be the home city. In addition, the objective function is to minimize the total traveling distances of all salesmen. So all salesmen start and end the traveling in the home city.

The termination criterion is $500 \times m \times n$ function evaluations. The population size is set as 100 in the experiments. We code the algorithms by Java on a Windows 2003 server (Intel Xeon 3.2 GHZ).

Apart from the methods should be examined, GA has some parameters should be tuned, including the crossover rate and mutation rate. These results are analyzed by ANOVA which provides the powerful analysis when we need to to find a stable and better parameter configuration. The parameter settings of the GA are shown in Table I.

TABLE I
Parameter settings.

Factor	Levels
1st part crossover (CX1)	Two-point central crossover and cycling crossover and order crossover
2nd part crossover (CX2)	ARC1 and ARC2 and Asexual
1st part mutation (Mutation1)	Swap and inverse mutation
2nd part mutation (Mutation2)	Arithmetic Mutation and Swap-Val
Crossover rate (P_c)	0.5 and 0.9
Mutation rate (P_m)	0.1 and 0.5
instance	eli51 and kroa150
No. of Salesmen (m)	3, 15, 20

[1]http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/index.html

[2]The corresponding instances are eli51 and kroa150 on TSPLIB.

B. Comparisons

The ANOVA results are presented in Table II. In the ANVOA table, the source indicates factors and combinations of factors. DF represents the degree of freedom and SS is the sum of squares. The mean square is equal to SS divided by DF. If the $Pr-value$ of a factor source is less than 0.05, it means that there is a significant difference in this factor [16]. Due to Pr-values of the some factors are less than 0.0001 in Table II, there exists a significant difference between the two methods. We further conduct the Duncan grouping test to differentiate the performance of these algorithms (See Table III to Table VI).

In Duncan grouping test, $Mean$ is the average value and N is the number of the observations. If the levels share the same alphabet (i.e., they are in the same group), there is no significant difference between them. Otherwise they are significantly different [16]. It is an evident from the Duncan grouping test from Table III to Table VI that the levels are significantly different in their performance.

For understanding interaction between two factors, interaction plots of the two factors are shown in Fig 6. In Fig 6, we already set the P_c is as the 0.5.

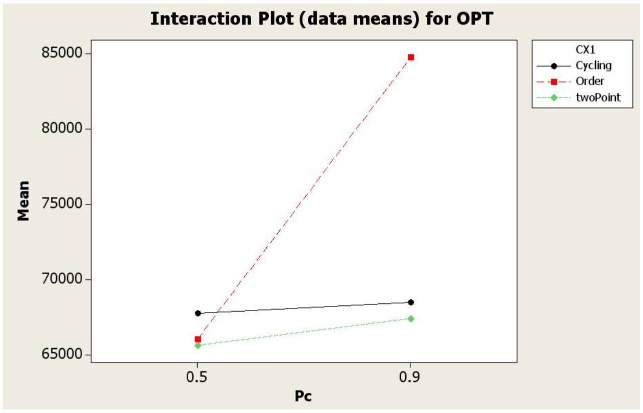

Fig. 6. Interaction plot of CX1 and P_c.

To summarize the experiment results of the operators selection and parameter configurations, we list them in Table VII. We may suggest the researchers to apply these settings when they employ the two-part chromosome technique to solve the mTSP.

V. Conclusions and Future Research

mTSP is an important problem no matter in academic research or in practice because mTSP could be transformed into different kinds of problems. Since this problem is much more complex than TSP, we have to select a better and efficient approach to solve this problem. This research studies the two-part chromosome encoding technique that could solve the mTSP efficiently. Most important of all, this study fills the gap of the genetic operators in the two-part chromosome. We may suggest the two-point central crossover to process the

TABLE II
ANOVA RESULTS ON THE OBJECTIVE VALUES OF THE MTSP PROBLEM WITH MINIMIZATION OF TOTAL DISTANCE

Source	DF	SS	Mean Square	F Value	P Value
Instances	1	1.2330908E14	1.2330908E14	474684	<**.0001**
CX1	2	385498734274	192749367137	742.00	<**.0001**
CX2	2	55463486.618	27731743.309	0.11	0.8987
CX1*CX2	4	31294915.636	7823728.9091	0.03	0.9983
Mutation1	1	57094925200	57094925200	219.79	<**.0001**
CX1*Mutation1	2	35713617284	17856808642	68.74	<**.0001**
CX2*Mutation1	2	85818972.864	42909486.432	0.17	0.8477
CX1*CX2*Mutation1	4	100140773.42	25035193.355	0.10	0.9836
Mutation2	1	81356956.087	81356956.087	0.31	0.5757
CX1*Mutation2	2	103585212.39	51792606.193	0.20	0.8192
CX2*Mutation2	2	28110123.566	14055061.783	0.05	0.9473
CX1*CX2*Mutation2	4	22179408.791	5544852.1978	0.02	0.9991
Mutation1*Mutation2	1	98014607.719	98014607.719	0.38	0.5391
CX1*Mutatio*Mutation	2	6967609.5067	3483804.7534	0.01	0.9867
CX2*Mutatio*Mutation	2	51391272.511	25695636.255	0.10	0.9058
CX1*CX2*Mutat*Mutati	4	54977002.111	13744250.528	0.05	0.9948
Pc	1	323502401739	323502401739	1245.34	<**.0001**
CX1*Pc	2	438194970602	219097485301	843.43	<**.0001**
CX2*Pc	2	9301718.9419	4650859.4709	0.02	0.9823
CX1*CX2*Pc	4	46385259.753	11596314.938	0.04	0.9962
Mutation1*Pc	1	25209377276	25209377276	97.04	<**.0001**
CX1*Mutation1*Pc	2	7297063308.6	3648531654.3	14.05	<**.0001**
CX2*Mutation1*Pc	2	15492572.664	7746286.332	0.03	0.9706
CX1*CX2*Mutation1*Pc	4	224549031.27	56137257.817	0.22	0.9296
Mutation2*Pc	1	9324997.3279	9324997.3279	0.04	0.8497
CX1*Mutation2*Pc	2	32571065.624	16285532.812	0.06	0.9392
CX2*Mutation2*Pc	2	41599723.555	20799861.777	0.08	0.9231
CX1*CX2*Mutation2*Pc	4	152832731.65	38208182.913	0.15	0.9643
Mutation*Mutation*Pc	1	5035793.7761	5035793.7761	0.02	0.8893
CX1*Mutati*Mutati*Pc	2	25554361.964	12777180.982	0.05	0.9520
CX2*Mutati*Mutati*Pc	2	24516197.26	12258098.63	0.05	0.9539
CX1*CX2*Muta*Muta*Pc	4	65411510.188	16352877.547	0.06	0.9927
Pm	1	110198952455	110198952455	424.22	<**.0001**
CX1*Pm	2	84318620761	42159310381	162.29	<**.0001**
CX2*Pm	2	74471777.999	37235889	0.14	0.8665
CX1*CX2*Pm	4	244197428.57	61049357.142	0.24	0.9187
Mutation1*Pm	1	23953877133	23953877133	92.21	<**.0001**
CX1*Mutation1*Pm	2	22226682309	11113341154	42.78	<**.0001**
CX2*Mutation1*Pm	2	42630687.234	21315343.617	0.08	0.9212
CX1*CX2*Mutation1*Pm	4	61933882.891	15483470.723	0.06	0.9934
Mutation2*Pm	1	14373122.419	14373122.419	0.06	0.8140
CX1*Mutation2*Pm	2	6988185.6177	3494092.8089	0.01	0.9866
CX2*Mutation2*Pm	2	290139891.91	145069945.95	0.56	0.5721
CX1*CX2*Mutation2*Pm	4	35470902.23	8867725.5574	0.03	0.9978
Mutation*Mutation*Pm	1	11043842.122	11043842.122	0.04	0.8366
CX1*Mutati*Mutati*Pm	2	8169893.0432	4084946.5216	0.02	0.9844
CX2*Mutati*Mutati*Pm	2	1252398.3467	626199.17333	0.00	0.9976
CX1*CX2*Muta*Muta*Pm	4	114509975.51	28627493.879	0.11	0.9790
Pc*Pm	1	329468426.07	329468426.07	1.27	0.2601
CX1*Pc*Pm	2	14658873991	7329436995.5	28.22	<**.0001**
CX2*Pc*Pm	2	10391912.775	5195956.3876	0.02	0.9802
CX1*CX2*Pc*Pm	4	88789278.047	22197319.512	0.09	0.9870
Mutation1*Pc*Pm	1	737303256.26	737303256.26	2.84	0.0921
CX1*Mutation1*Pc*Pm	2	11791952017	5895976008.6	22.70	<**.0001**
CX2*Mutation1*Pc*Pm	2	194703562.06	97351781.032	0.37	0.6875
CX1*CX2*Mutati*Pc*Pm	4	26762371.31	6690592.8276	0.03	0.9987
Mutation2*Pc*Pm	1	8144812.2402	8144812.2402	0.03	0.8595
CX1*Mutation2*Pc*Pm	2	77286940.519	38643470.26	0.15	0.8618
CX2*Mutation2*Pc*Pm	2	10729415.86	5364707.93	0.02	0.9796
CX1*CX2*Mutati*Pc*Pm	4	44541323.096	11135330.774	0.04	0.9965
Mutati*Mutatio*Pc*Pm	1	4743023.641	4743023.641	0.02	0.8925
CX1*Muta*Mutat*Pc*Pm	2	27887154.685	13943577.343	0.05	0.9477
CX2*Muta*Mutat*Pc*Pm	2	8489311.0968	4244655.5484	0.02	0.9838
CX*CX2*Mut*Mut*Pc*Pm	4	83654332.027	20913583.007	0.08	0.9883
m	2	4.62E12	2.31E12	8892.5	<**.0001**
Error	25487	6.62E12	259770805.52		
Total	25919	1.36E14			

TABLE III
DUNCAN GROUPING TEST ON THE FIRST PART CROSSOVER OPERATOR

Grouping	Mean	N	CX1
A	75387.5	8640	Order Crossover
B	68119.3	8640	Cycling Crossover
C	66527.8	8640	Two-Point Central Crossover

TABLE IV
DUNCAN GROUPING TEST ON THE FIRST PART MUTATION OPERATOR

Grouping	Mean	N	Mutation1
A	71495.7	12960	Swap
B	68527.4	12960	Inverse

TABLE V
DUNCAN GROUPING TEST ON THE CROSSOVER RATE

Duncan Grouping	Mean	N	P_c
A	73544.3	12960	0.9
B	66478.7	12960	0.5

TABLE VI
DUNCAN GROUPING TEST ON THE MUTATION RATE

Duncan Grouping	Mean	N	P_m
A	72073.4	12960	0.5
B	67949.6	12960	0.1

TABLE VII
PARAMETER SETTINGS.

Factor	Levels
CX1	Two-point central crossover
CX2	ARC1
Mutation1	Inverse Mutation
Mutation2	Arithmetic Mutation
P_c	0.5
P_m	0.1

permutation in the first part chromosome. Then, the ARC2 could be applied when we do the crossover for the second part chromosome. In addition, a feasibility check is implemented after we do the crossover on the second part chromosome. When it comes to the mutation for the first part and the second part chromosome, inverse mutation and arithmetic mutation are recommended. We also test the crossover rate (P_c) and (P_m) in GA. The better parameter configuration of them could be the 0.5 and 0.1. By using above suggested genetic operators and parameter settings, this work provides an important reference for the researchers or practitioners when they want to design a new two-part chromosome genetic algorithm.

Finally, because the mTSP is a complex problem, the solution quality could be enhanced by using some algorithms which may improve the solution quality within the route. For example, the best heuristic in solving the TSP is the enhanced version of Lin-Kernighan [9] which is used to solve more than 10,000 cities. When the Lin-Kernighan heuristic is involved in the GA, the performance would be greatly improved. Because prior research has not used the Lin-Kernighan heuristic, it might be a good direction for the future research.

REFERENCES

[1] R. Angel, W. Caudle, R. Noonan, and A. Whinston, "Computer-assisted school bus scheduling," *Management Science*, vol. 18, no. 6, pp. 279–288, 1972.

[2] T. Bektas, "The multiple traveling salesman problem: an overview of formulations and solution procedures," *Omega*, vol. 34, no. 3, pp. 209–219, 2006.

[3] E. Brown, C. Ragsdale, and A. Carter, “A grouping genetic algorithm for the multiple traveling salesperson problem,” *International Journal of Information Technology and Decision Making*, vol. 6, no. 2, pp. 333–347, 2007.

[4] A. Carter and C. Ragsdale, “A new approach to solving the multiple traveling salesperson problem using genetic algorithms,” *European journal of operational research*, vol. 175, no. 1, pp. 246–257, 2006.

[5] A. E. Carter. and C. T. Ragsdale, “Scheduling pre-printed newspaper advertising inserts using genetic algorithms,” *Omega*, vol. 30, no. 6, pp. 415–421, 2002.

[6] E. Falkenauer, “A hybrid grouping genetic algorithm for bin packing,” *Journal of heuristics*, vol. 2, no. 1, pp. 5–30, 1996.

[7] B. Gavish and K. Srikanth, “An optimal solution method for large-scale multiple traveling salesmen problems,” *Operations Research*, pp. 698–717, 1986.

[8] D. Goldberg, *Genetic algorithms in search, optimization, and machine learning*. Addison-Wesley Professional, 1989.

[9] K. Helsgaun, “An effective implementation of the lin-kernighan traveling salesman heuristic,” *European Journal of Operational Research*, vol. 126, no. 1, pp. 106–130, 2000.

[10] K. Kim and Y. Park, “A crane scheduling method for port container terminals,” *European Journal of Operational Research*, vol. 156, no. 3, pp. 752–768, 2004.

[11] A. Király and J. Abonyi, “A novel approach to solve multiple traveling salesmen problem by genetic algorithm,” *Computational Intelligence in Engineering*, pp. 141–151, 2010.

[12] G. Laporte and Y. Nobert, “A cutting planes algorithm for the m-salesmen problem,” *Journal of the Operational Research Society*, vol. 31, no. 11, pp. 1017–1023, 1980.

[13] C. Malmborg, “A genetic algorithm for service level based vehicle scheduling,” *European Journal of Operational Research*, vol. 93, no. 1, pp. 121–134, 1996.

[14] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-verlag Berlin, 1992.

[15] D. Miller and J. Pekny, “Exact solution of large asymmetric traveling salesman problems,” *Science*, vol. 251, no. 4995, p. 754, 1991.

[16] D. Montgomery, *Design and analysis of experiments*. John Wiley & Sons Inc, 2008.

[17] T. Muruta and H. Ishibuchi, “Performance evolution of genetic algorithms for flowshop scheduling problems,” 1994.

[18] I. Oliver, D. Smith, and J. Holland, “A study of permutation crossover operators on the traveling salesman problem,” in *Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application*. L. Erlbaum Associates Inc., 1987, pp. 224–230.

[19] Y. Park, “A hybrid genetic algorithm for the vehicle scheduling problem with due times and time deadlines,” *International Journal of Production Economics*, vol. 73, no. 2, pp. 175–188, 2001.

[20] B. Rekiek, A. Delchambre, and H. Saleh, “Handicapped person transportation: An application of the grouping genetic algorithm,” *Engineering Applications of Artificial Intelligence*, vol. 19, no. 5, pp. 511–520, 2006.

[21] G. S., “Printing press scheduling for multi-edition periodicals,” *Management Science*, vol. 16, no. 6, pp. 373–383, 1970.

[22] H. Saleh and R. Chelouah, “The design of the global navigation satellite system surveying networks using genetic algorithms,” *Engineering Applications of Artificial Intelligence*, vol. 17, no. 1, pp. 111–122, 2004.

[23] A. Singh and A. Baghel, “A new grouping genetic algorithm approach to the multiple traveling salesperson problem,” *Soft Computing-A Fusion of Foundations, Methodologies and Applications*, vol. 13, no. 1, pp. 95–101, 2009.

[24] S. Starkweather and C. Whitley, “A comparison of genetic sequencing operators,” in *Proceedings of the Fourth International Conference on Genetic Algorithms: University of California, San Diego, July 13-16, 1991*. Morgan Kaufmann Publishers, 1991, p. 69.

[25] L. Tang, J. Liu, A. Rong, and Z. Yang, “A multiple traveling salesman problem model for hot rolling scheduling in shanghai baoshan iron & steel complex,” *European Journal of Operational Research*, vol. 124, no. 2, pp. 267–282, 2000.

[26] ——, “A multiple traveling salesman problem model for hot rolling scheduling in shanghai baoshan iron & steel complex* 1,” *European Journal of Operational Research*, vol. 124, no. 2, pp. 267–282, 2000.

[27] X. Wang and A. Regan, “Local truckload pickup and delivery with hard time window constraints,” *Transportation Research Part B: Methodological*, vol. 36, no. 2, pp. 97–112, 2002.

[28] R. Wolfler Calvo and R. Cordone, “A heuristic approach to the overnight security service problem,” *Computers & Operations Research*, vol. 30, no. 9, pp. 1269–1287, 2003.

[29] T. Zhang, W. Gruver, and M. Smith, “Team scheduling by genetic search,” vol. 2, pp. 839–844, 1999.