2011 International Conference on Mechatronic Science, Electric Engineering and Computer
August 19-22, 2011, Jilin, China

A novel evolutionary algorithm for the traveling salesman problem

Chengjun Li
School of Computer
China University of Geosciences
Wuhan, China
cuglicj@gmail.com

Si Xu
School of Computer
China University of Geosciences
Wuhan, China
823253541@qq.com

Yong Xia
School of Computer
China University of Geosciences
Wuhan, China
455399810@qq.com

Wei Zhan
College of Computer Science
Yangtze University
Jingzhou, China
zhanwei814@gmail.com
(Corresponding author)

***Abstract* — The traveling salesman problem (TSP) is a famous NP-hard problem. The established evolutionary algorithms (EAs) cannot get satisfactory solutions of large or even medium scale TSP instances. To change this situation, a effective EA based on inver-over operator is introduced. In this algorithm, two novel crossover operators, the segment replacing crossover and the order adjusting crossover, are proposed. Moreover, a mechanism for changing the crossover and the mutation rate of the inver-over operator according to generations and a parameter, the critical value, is employed. The results of the experiment show that this new algorithm outperforms the basic EA based on the inver-over operator in all the 4 instances.**

Keywords- traveling salesman problem; evolutionary algorithm; inver-over operator; segment replacing crossover; order adjusting crossover

I. INTRODUCTION

The traveling salesman problem (TSP) is conceptually very simple. The traveling salesman must visit every city in his territory exactly once and then return home to his starting point. Given the cost of traveling between each pair of cities, the task is to arrange a complete tour that achieves minimum total cost. The size of its possible search space is n! [1]. As a famous NP-hard problem, it has attracted many researches in different fields for several hundred years. Till now, there are still lots of papers focus on it.

The researches on using evolutionary algorithms (EAs) to solving the TSP have existed for some decades. However, in general, EAs for the TSP can not get satisfactory solutions of large or even medium scale TSP instances in [2]. This is the main motivations of this paper.

Based on the inver-over operator [3], a novel EA for the TSP is introduced in this paper. The segment replacing crossover and the order adjusting crossover those are new crossover operators are proposed in this algorithm. Additionally, a mechanism for changing the crossover and the mutation rate of the inver-over operator according to generations and the critical value that is a parameter to control operators' action is applied in it. Owning to the low executed probability of these two operators, the EA based on them is called EA based on low probability crossover operators, or, in short, LPCOEA.

This paper is set out as follows: In Section 2, we describe related researches including the algorithm in [3]. In Section 3, we present the LPCOEA. Experiments and the analysis are in Section 4. Finally, a conclusion and a prospect are dealt with in Section 5.

II. RELATED RESEARCHES

A. EAs for the TSP

To complete an EA for the TSP, these things should be taken into consideration.

• Representation

In existing EAs for the TSP, there are a variety of representations including adjacency, ordinal and path representation. Specially, there have been attempts to evolve solutions for the TSP where solutions were represented in a matrix [1]. The path one is the most popular representation in established EAs for TSP.

• Crossover and mutation operator

Based on different representation, the different crossover and mutation operators have been used in many EAs for the TSP. For instance, if adjacency representation is used in an EA, alternating-edges, subtour-chunks and heuristic crossover can be selected in this algorithm. There are many crossover and mutation operators based on the path representation. The best-known crossover operators defined for the path representation are: partially-mapped, order, and cycle crossover operators [1]. Moreover, subtour preservation crossover is introduced in [4].

B. Algorithms analysis

In recent years, algorithms analysis is becoming a new focus. For instance, [5] provides a methodology to determine if the meta-data is sufficient, and demonstrates the critical role played by instance generation methods. Moreover, [6] conducts a runtime analysis of a simple

978-1-61284-722-1/11/$26.00 ©2011 IEEE

Evolutionary Algorithm called (1+1) EA on a TSP instance.

C. The inver-over operator for the TSP

Based on the path representation, an inver-over operator was proposed in [3]. One can view this operator as a mixture of mutation and crossover: on one hand, the inversion is applied to a part of a single individual; however, the selection of a segment to be inverted depends on other individuals in the population. The current operation of this operator is determined by probability p. In an operation, two markers in an individual are made to decide which segment should be inversed. If a random pure decimal is more than p, a randomly selected individual provides a clue for the second marker for inversion. Then, it is a crossover operation. Otherwise, this operation is a mutation operation.

It is reported in [3] that the inver-over operator outperforms many other operators. Moreover, the EA with this operator is quite fast and the quality of the results is very high. In this paper, the EA in [3] is called IOEA in short.

III. THE EA BASED ON LOW PROBABILITY CROSSOVER OPERATORS (LPCOEA)

A. The segment replacing crossover

Two chromosomes S_x and S_y are selected from the population randomly. Let the fitness of S_y is better than that of S_x. Then, a segment $\triangle S_x$ is selected form S_x randomly. If there exists a segment $\triangle S_y$, which have the same numbers of cities as $\triangle S_x$ and the first city is the same, in S_y, the $\triangle S_x$ in S_x should be replaced by $\triangle S_y$. The remaining of the S_x should be reorganized according to partially mapped crossover.

B. The order adjusting crossover

Two chromosomes S_x and S_y are selected from the population randomly. Let the fitness of S_y is better than that of S_x. Then, a segment $\triangle S_x$ is selected form S_x randomly. If there exists a segment $\triangle S_y$, which is constituted by the same cities as $\triangle S_x$ with a different sequence, in S_y and the fitness of $\triangle S_y$ is better than that of $\triangle S_x$, the $\triangle S_x$ in S_x should be replaced by $\triangle S_y$.

C. The mechanism for changing the crossover and the mutation rate of the inver-over operator according to generations

In the inver-over operator, the sum of the crossover and mutation rate is 1. Let the current generations be g, the maximal generations be mg, the initial mutation rate be p0, the current mutation rate be p. The mutation rate p can be measured by formula below:

$$p = p_0 \times (1 - g \times 0.01 / mg)$$

D. The critical value of the evolution velocity

The evolution velocity denotes that the best solution in current generations makes how great a progress than that in the previous generations. The critical value of the evolution velocity is a parameter which is proposed to decide some actions of the algorithm.

When the evolution velocity exceeds the critical value, in the crossover operation of the inver-over operator, whether the parent should be replaced will be decided after the all inversions. Otherwise, in such an operation, after every inversion, the parent may be replaced as soon as the current chromosome is better than it.

Moreover, when the evolution velocity exceeds the critical value, the segment replacing crossover is forbidden to be executed.

E. The pseudo-code

The pseudo-code of this EA is as below (the words in bold are the things proposed in this paper):

Begin
Initialization for the population P={S_0, S_1,……, S_{n-1}}and each parameter
Do
i=0
Chromosome S'=S_i
A city C is chosen in S' randomly
Do
If a random pure decimal is more than p
Another chromosome is S'' chosen.
In S'', the next city to C is regarded as C'
If C' is a neighbor of C in S'
Break
The segment between C and C' and including C' is inversed
The change in the length of the path d is computed
If d<0 and evolution velocity is less than critical value
Break
C=C'
While there are still some cities in S' which have not ever become C
The fitness of S' is computed
If S' is better than S_i
S' replace S_i
i=i+1
The evolution velocity is computed
The mutation rate is refreshed
If evolution velocity is less than critical value and a random pure decimal is less than the probability of the segment replace crossover p_r
Two chromosomes S_x and S_y are chosen randomly (Let S_y is better than S_x)
A segment $\triangle S_x$ whose starting point and end point is randomly decided is chosen from S_x
A segment $\triangle S_y$ which have the same numbers of cities as $\triangle S_x$ and the first city is the same is searched in S_y
If $\triangle S_y$ exists
$\triangle S_x$ in S_x is replaced by it.
Other gene locus should be reorganized according to partially mapped crossover.
While stop condition is not satisfied
The best chromosome of the finial population S_b is divided into ten segments.
For every segment, another segment which has the same cities with the different sequence is searched in all the other chromosomes.

If such a segment exist
The segment in the S_b is replaced by it.
Print the best chromosome
End

IV. EXPERIMENTS AND THE ANALYSIS

In order to test the performance of the proposed algorithm, it and the IOEA are executed 20 times to solve four instances of the TSP form [2], respectively. These four instances are form rat195 to pcb442, which are difficult to EAs for their scales.

The experiments run on single core of Dawning TC5000A high-performance computing platform using Open MPI 1.4 under UNIX.

The parameters of the two algorithms are listed in Table 1. Table 2 is the comparison on general results of the two algorithms.

In Table 2, the results achieved by both algorithms for the four instances are presented. It can be seen that the LPCOEA can keep its convergence ability for much more generations than the IOEA. It is observed that the LPCOEA get better and more stable solutions than the IOEA in the experiment. However, it is found in this experiment that the LPCOEA needs more time to run the same generations than IOEA. The main cause of this phenomenon is the time consuming of the two low probability crossover operators.

V. CONCLUDING REMARKS

In this paper, we have presented the LPCOEA to enhance the solving performance of IOEA. The results of the experiment show that the LPCOEA outperforms IOEA. In the LPCOEA, the two low probability crossover operators are proposed. In fact, the two crossover operators can be employed in any EA for the TSP based on the path representation. Hence, they have strong practice value.

Further research will concentrate on how to improve these two crossover operator by setting parameters more reasonable.

TABLE I. PARAMETERS OF THE TWO ALGORITHMS

LPCOEA	Initial inver-over mutation rate (p)	0.02 (changed in every generations according to the formula)
	Inver-over crossover rate	1-p
	The segment replace crossover rate	0.05
	Critical velocity	5000
	Generations	Number of citys *2000
IOEA	Inver-over mutation rate (p)	0.02
	Inver-over crossover rate	1-p
	Generations	Number of citys *2000

TABLE II. COMPARISON ON GENERAL RESULTS

Instance	**Algorithm**	**The best solution**	**The worst solution**	**Average**	**Average generations of no change in best solution at the end of run**	**Standard deviation**	**Average of running time**	**T-test**
rat195	LPCOEA	2341	2364	2350.7	176356.8	6.47	83.02	-1.39
	IOEA	2389	11186	3106.8	382819.3	2427.87	9.23	
gil262	LPCOEA	2397	2424	2407.2	253062.7	7.61	132.95	-4.87
	IOEA	2462	21876	10912.4	520061.2	7813.84	9.75	
lin318	LPCOEA	42439	43100	42773.6	376191.4	173.73	135.08	-1.40
	IOEA	42604	317741	63270.6	618646.3	65452.65	31.12	
pcb442	LPCOEA	51410	52188	51774.3	456809.2	194.28	318.87	-4.85
	IOEA	52849	589083	275861.6	875820.2	206659.50	39.25	

REFERENCES

[1] Z. Michalewicz and D. B. Fogel, How to solve it: Modern Heuristics, Berlin:Springer, 2000.

[2] Universität Heidelberg. TSPLIB.

[3] T. Guo and Z. Michalewicz, "Inver-over Operator for the TSP," Proc. Parallel Problem Solving from Nature (PPSN V), Springer, 1998, pp. 803-812,.

[4] S. M. Soak and B. H. Ahn, "New Genetic Crossover Operator for the TSP," Proc. Artificial Intelligence and Soft Computing (ICAISC 2004), Springer, 2004, pp. 480-485, doi: 10.1007/978-3-540-24844-6_71

[5] K. Smith-Miles, J. van Hemert, X. Y. Lim, "Understanding TSP Difficulty by Learning from Evolved Instances," Proc. 4th Conference on Learning and Intelligent Optimization (LION 4), Springer, 2010, pp. 266-280, doi: 10.1007/978-3-642-13800-3_29

[6] Y. S. Zhang, Z. F. Hao, "Runtime Analysis of (1+1) Evolutionary Algorithm for a TSP Instance," Proc. 1st International Conference on Swarm, Evolutionary, and Memetic Computing (SEMCCO 1), Springer, 2010, pp. 296-304, doi: 10.1007/978-3-642-17563-3_36