INTERNATIONAL JOURNAL OF OPTIMIZATION IN CIVIL ENGINEERING
Int. J. Optim. Civil Eng., 2012; 2(4):607-619

AN EFFICIENT CROSSOVER OPERATOR FOR TRAVELING SALESMAN PROBLEM

M. Rajabi Bahaabadi, A. Shariat Mohaymany[*,†] and M. Babaei
Iran University of Science and Technology, Faculty of Civil Engineering, Narmak, Tehran, Iran

ABSTRACT

Crossover operator plays a crucial role in the efficiency of genetic algorithm (GA). Several crossover operators have been proposed for solving the travelling salesman problem (TSP) in the literature. These operators have paid less attention to the characteristics of the traveling salesman problem, and majority of these operators can only generate feasible solutions. In this paper, a crossover operator is presented that has the capability of generating solutions based on a logical reasoning. In other words, the solution space is explored by the proposed method purposefully. Numerical results based on 26 benchmark instances demonstrate the efficiency of the proposed method compared with the previous meta-heuristic methods.

Received: 7 August 2012; Accepted: 15 October 2012

KEY WORDS: traveling salesman problem; meta-heuristic; genetic algorithm

1. INTRODUCTION

Traveling salesman problem is one of the well-known and practical problems in the field of combinatorial optimization. The purpose of this problem is to find the shortest Hamiltonian tour in a complete graph with N nodes. A salesman starts from a city, visits a set of cities and returns to the original location in such a way that the total distance travelled becomes minimum and each city is visited exactly once. TSP is easy to understand but very hard to solve, because it belongs to the class of NP-hard problems[1]. Many practical problems can be formulated as TSP in the real world; such as: vehicle routing problem, ship scheduling,

[*] Corresponding author: A. Shariat Mohaymany, Department of Transportation Planning, Faculty of Civil Engineering, Iran University of Science and Technology (IUST), Narmak, Tehran, Iran
[†]E-mail address: Shariat@ iust.ac.ir

train driver scheduling and flowshop scheduling.

Many researchers have attempted to solve TSP during the past fifty years and various algorithms have been presented to solve it. Algorithms for solving this problem are generally classified into two categories: exact algorithms and approximate algorithms. The branch-and-bound and branch-and-cut are examples of the exact methods that are excessively time consuming especially in large-scale instances. Approximate methods are rather classified into heuristic and meta-heuristic algorithms. Heuristic algorithm for TSP can be divided into three classes : 1) tour construction methods [2-3] , 2) tour improvement methods [4-5] , and 3) composite methods [6-8]. Tour construction methods build a tour by adding an unvisited city to the solution at each step while tour improvement methods apply an initial tour and try to achieve a shorter tour in each step. Finally, composite methods combine the two previous procedures to obtain better solutions. One of the most successful heuristic methods for solving traveling salesman problem was proposed by Lin and Kernighan (1973) [8]. The Lin-Kernighan algorithm (LK) is a λ-opt algorithm where the λ can be any nonnegative integers. It determines the most suitable λ value at each step, so that, in each step, λ edges of the current tour are replaced by λ new edges to achieve a shorter tour [6].

Appearance of meta-heuristic algorithms and their applications in optimization problems started a new stage of studies. During the three last decades, various meta-heuristic search algorithms have been applied on TSP; such as: tabu search [9], genetic algorithm [10-14], memetic algorithm [15-16], ant colony optimization [17-19], particle swarm optimization [13, 20-22], neural networks[23-26], simulated annealing [27], intelligent water drops [28], chaotic ant swarm [29], and gravitational emulation search algorithm [30]. Further information about the TSP and its solution methods can be found in [1].

Genetic algorithm has attracted a great deal of attention, due to its high efficiency in solving combinatorial optimization. Therefore, various methods of solving TSP have been presented based on the genetic algorithm. One of the major differences between the genetic algorithms is in their operators. The crossover operator is one of the most effective operators used in genetic algorithms. It produces offspring by combining parents. Therefore, using an appropriate crossover operator for TSP can increase the efficiency and the speed convergence of genetic algorithm. Several crossover operators for TSP have been developed, such as: partially map crossover (PMX), cycle crossover (CX), order crossover (OX1), position-based crossover (POS), alternating-position crossover (AP), and edge recombination crossover (ER). Larranaga et al. [31] drew a comparison between the aforementioned crossover operators. Their results demonstrate the superiority of edge recombination crossover operator (ER) and order crossover operator (OX1) compared to the others. Nevertheless, the aforementioned operators have only concentrated on generating feasible solutions, and they have paid less attention to the characteristics of the traveling salesman problem. In fact, the probability of the presence of global optimum is not equal in the entire feasible space for some optimization problems. For example, longest edges of a complete graph are less likely to be in the shortest Hamiltonian tour. Therefore, one can hypothesize that if the crossover operator takes account of the likelihood of the presence of longest edges for generating offspring, it would explore the solution space more efficiently. In this regards, this paper attempts to develop a new crossover operator which takes account

of the inherent characteristics of TSP. It generates feasible solutions based on a logical reasoning by decreasing the probability of incorporating longest edges of the complete graph into the traveling salesman's tour.

The rest of this paper is organized as follows. Section 2 is dedicated to the basic concepts of genetic algorithm. Section 3 describes GA for TSP. The proposed crossover operator is described in Section 4. Some numerical results on TSP benchmarks are presented in Section 5. Finally, Section 6 concludes the paper and outlines future research perspectives.

2. BASIC CONCEPTS OF GRNRTIC ALGORITHM

The genetic algorithm is a randomized search technique which is based on the principles of natural selection and survival of the fittest chromosomes. In GA, the parameters of a solution are encoded into a numerical string called chromosome. Each chromosome represents a possible solution for the problem. Each chromosome is made up of a pre-specified number of genes. The set of chromosomes is called 'population'. Initially a random population is created to represent different points in the search space, and based on the principle of survival of the fittest individuals, some of them are selected. Biologically inspired operators like crossover and mutation are then applied on these strings to yield a new generation of strings. The process of selection, crossover and mutation continues for a fixed number of generations or until a termination condition is satisfied. The following pseudo-code describes this approach.

Algorithm 1. Genetic Algorithm

```
Set the initial population
while termination condition is not satisfied do
        Evaluate the fitness value of each chromosome in the population. /*Fitness
         Evaluation*/
         Select parents from the population. /*Selection Operation*/
         Create offspring by applying crossover operation on the parents. /*Crossover
         Operation*/
         Randomly select a number of offspring for mutation. /*Mutation
 Operation*/
end while
return the chromosome with the highest fitness
```

3. GENETIC ALGORITHM FOR TSP

In this section, we describe the detail of our GA for solving the traveling salesman problem. The flowchart of the proposed algorithm for the TSP is depicted in Figure 1. Details of the algorithm are explained in the following sections.

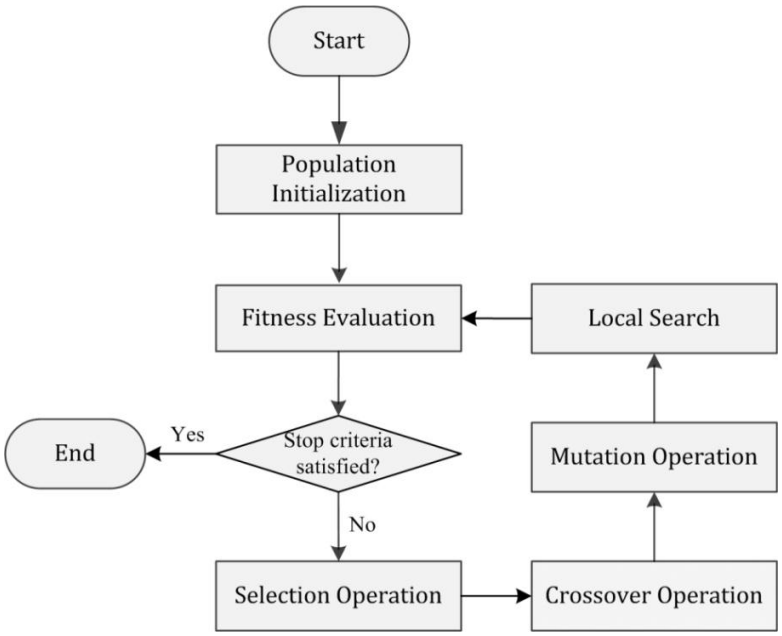

Figure 1. The flowchart of the proposed algorithm

3.1. Chromosome representation

As noted earlier, the first step in GA is to encode feasible solutions for a particular problem. Several representation methods have been used for the TSP such as: path representation, matrix representation, binary representation, adjacency representation and ordinal representation. In this paper, we use a path representation where the cities are listed in the order in which they are visited. For example, consider a tour with four cities. If a salesman goes from city 1 to city 2, city 3, and city 4 sequentially and then returns back to city 1, the path representation of the tour will be as the chromosome shown in Figure 2.

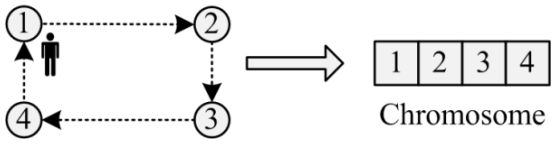

Figure 2. Path representation

3.2. Population initialization

The population initialization can be done by a random generation method or structured generation methods such as Lin-Kernighan algorithm [8]. Here, the initial population is generated randomly, where a randomized procedure is utilized to produce random permutation of nodes.

3.3. Evaluation of fitness function

Since GA is generally applied on maximization problems and the TSP is a minimization problem, the inverse of tour length is considered as the fitness function.

3.4. Selection operation

Through selection operation step, parents are selected for crossover. Several different selection methods have been developed in the previous researches, such as roulette wheel selection, tournament selection, ranking selection and proportional selection. Here, the roulette wheel selection method is selected.

3.5. Crossover operation

The main purpose of this component is to create offspring using a given pair of solutions chosen through the selection operation procedure. This paper proposes a new crossover operator. Details of the proposed crossover operator are explained in Section 4.

3.6. Mutation operation

Mutation operation increases the genetic diversity of populations. This operation avoids premature convergence and escape algorithm from local optima. In this paper, two mutation operators are used in each iteration simultaneously. The first mutation operator is the exchange mutation operator (EM) that randomly selects two cities in a tour and exchanges them (Figure 3(a)). We have designed another mutation operator which can increase the diversity of population while maintaining the optimality of sub-tours. The second operator is somewhat based on locating neighbor cities as closed as possible to each other in the resultant tour (Figure 3(b)). The pseudo-code of the proposed mutation operator is presented in the following algorithm.

Algorithm 2**.** Second mutation operator

Step 1: Divide the chromosome by two cut points into three sectors.

Step 2: Copy the third sector of the prior chromosome into the first part of the new chromosome.

Step 3: Copy the second sector of the prior chromosome into the new chromosome in front of the previous copied genes in step 2.

Step 4: Copy the inverse of the first sector of the prior chromosome into the new chromosome in front of the previous copied genes in step 3.

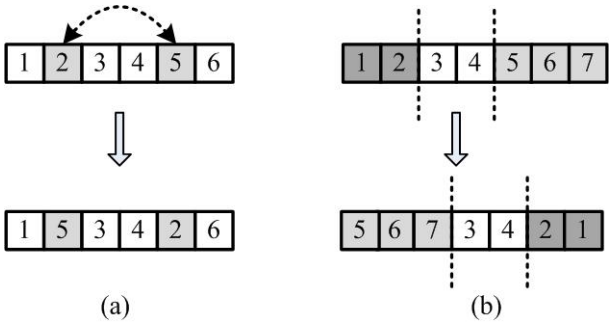

Figure 3. Mutation operators

3.7. Local search

2-opt is a well-known local search operator for solving TSP. In each step of 2-opt, two edges of the current tour are replaced by two other edges in a way that a shorter tour is achieved. This local search operator is implemented in this paper. Interested readers can refer to [8] for the details of 2-opt algorithm.

3.8. Termination criteria

The termination criteria can be characterized by the maximum number of iterations, the compution time, or the number of iterations with no improvement. In this paper, the maximum number of iterations is used as the termination criterion.

4. THE PROPOSED CROSSOVER OPERATOR

The new crossover operator presented in this section is designated to generate more logical offspring. It can increase the possibility of connecting each city to its neighboring cities. In other words, the longest edges of a complete graph are less likely to be in the shortest Hamiltonian tour. For this purpose, we have created a list of tabu cities for each city so that the possibility of direct connection of distant cities is reduced; hence, cities that their distances from city *i* are higher than a critical distance are stored in the tabu list of city *i*. The critical distance for city *i* is calculated as:

$$d_c^i = \frac{1}{B(N-1)} \sum_{j=1}^{j=N} d_{ij} \quad , B \geq 1 \tag{1}$$

where d_{ij} is the traveling distance between cities i and j, N is the number of cities in TSP and B is a positive parameter which determines the importance of the closeness of cities. Thus, as the value of B is increased the probability of connecting each city to the nearer cities is increased.

Note that storing cities in the tabu list of city *i* does not mean that the tabu cities of city *i* are prohibited from connecting to city *i* directly, but it is just a measure to decrease the possibility of the connection of faraway cities to each other.

To produce an offspring, the algorithm of the proposed crossover operator is given below.

Algorithm3. Proposed crossover operator

Step 1: Choose initial city randomly (name it city *i*) and copy it into the first gene of the offspring.

Step 2: If all of the cities are visited then go to step 6 else go to step 3.

Step 3: Denote j_1 and j_2 the cities located after the city *i* in the first and second parents respectively.

Step 4: $d_{\min} \leftarrow \min\{d_{ij_1}, d_{ij_2}\}$.

Step 5: If $(d_{\min} \leq d_c^i)$ **then**

If ($d_{ij_1} = d_{\min}$ and j_1 is an unvisited city) **then**

Copy j_1 into the next gene of the offspring and $i \leftarrow j_1$, then go to step 2.

else if ($d_{ij_2} = d_{\min}$ and j_2 is an unvisited city)

Copy j_2 into the next gene of the offspring and $i \leftarrow j_2$ then go to step2.

else

Copy the nearest city (to city *i*) that has not been chosen yet into the next gene of the offspring, name it city j_3 and $i \leftarrow j_3$ then go to step 2.

end if

else

Copy the nearest city (to city *i*) that has not been chosen yet into the next gene of the offspring, name it city j_3 and $i \leftarrow j_3$ then go to step 2.

end if

Step 6: return the offspring

Figure 4 presents an example for the application of the proposed crossover operator in a 5-city tour. Consider two parents P_1 and P_2 (see Figure 4(a)). According to the first step of the proposed crossover operator, city 1 is selected randomly and copied into the first gene of the offspring (Figure 4(b)). Then those cities that are located after city 1 in both parents are considered for the next step of the algorithm (here, $j_1 = j_2 = 2$). Since the distance between city 1 and city 2 is less than the critical distance, city 2 is copied into the second gene of the offspring and according to step 5, $i=2$ is set (Figure 4(c)). Following the same approach discussed for city 1, city 5 and city 4 are considered as the cities that are located after city 2 in the both parents. Whereas the minimal distances between city 2 and cities 5 and 4 are more than the critical distance then according to step 5, the closest city to the city 2 that has not been visited yet (city 3) is determined as the third city in this Hamiltonian tour (Figure 4(d)). In next step, cities 4 and 5 are both visited cities after city 3 in the parents. Since city 4 is closer to city 3 and the distance between city 3 and city 4 is less than the critical

distance, then this city is copied into the next gene of the offspring (Figure 4(e)). Finally, city 5 is copied into the last gene of the offspring as the last remaining city (Figure 4(f)).

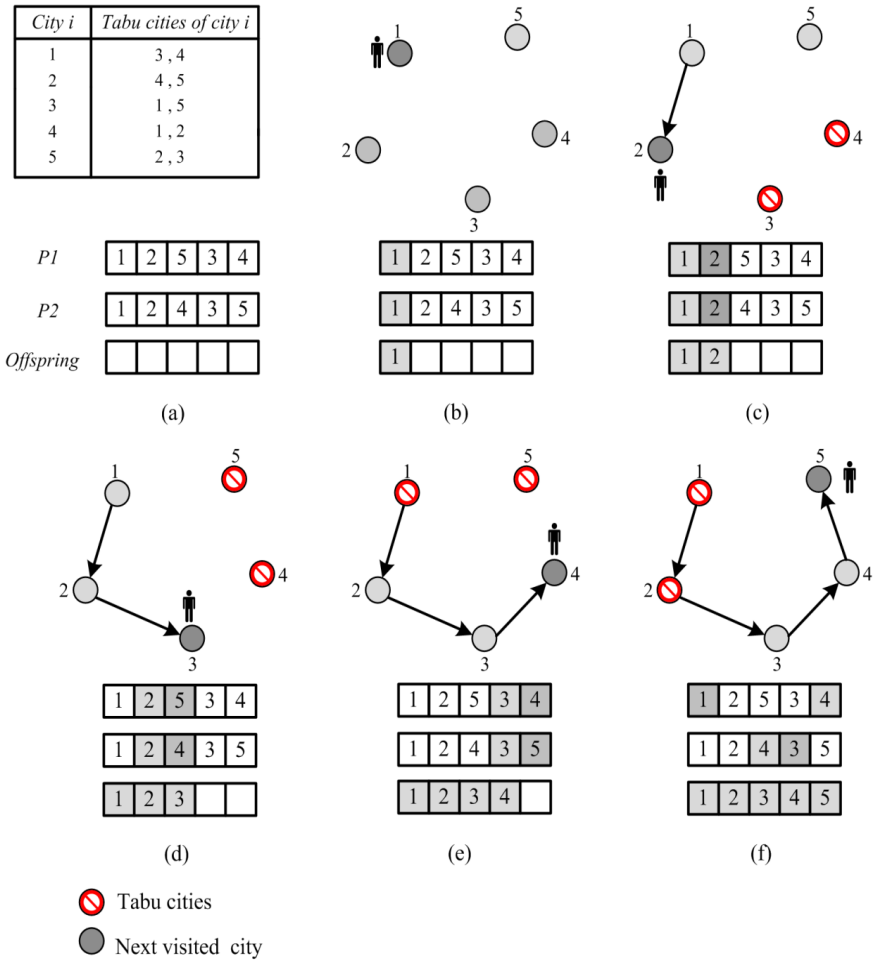

Figure 4. The proposed crossover operator

5. RESULTS AND DISCUSSION

In this section, we present the computational results of the proposed algorithm. For experiments, a computer with Intel Pentium IV 2.67 GHz processor was used and the algorithm was coded in C#.Net 2008 programming language. Moreover, in all cases a population of 100 chromosomes is used; the value of crossover rate is set 0.8 and the mutation rate is taken as 0.1.

Figure 5 compares the convergence speed of the proposed algorithm when B varies from 1 to 4. Due to the similarity of results from different test problems, here we present the results of the Eil51 problem. It can be observed from Figure 5 that better convergence speed is obtained when the value of B is 2. Besides, it can clearly be seen in Figure 6 that the

proposed algorithm finds a better solution when the value of B is 2. Therefore, the value of 2 is considered for parameter B in all analysis performed within the rest of this section.

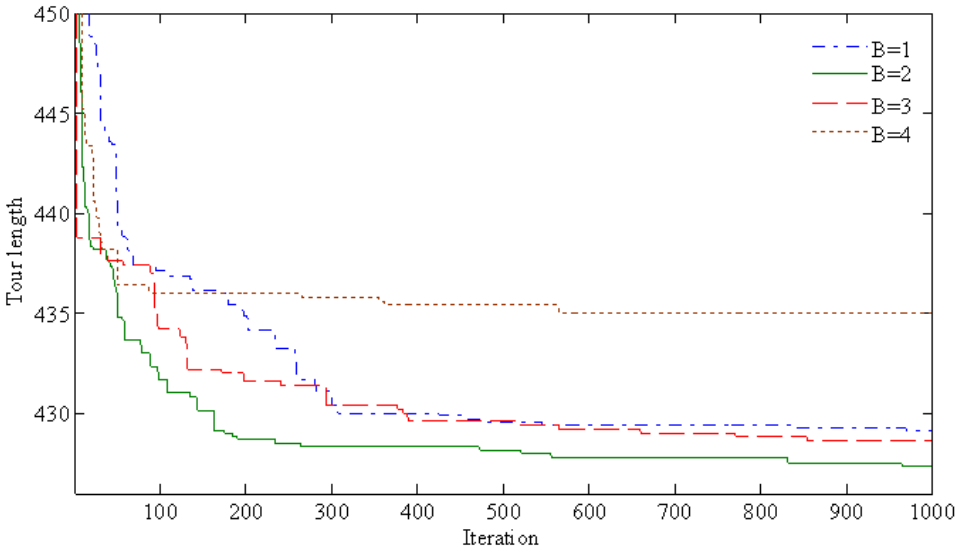

Figure 5. Convergence rates of the proposed algorithm with different values of B

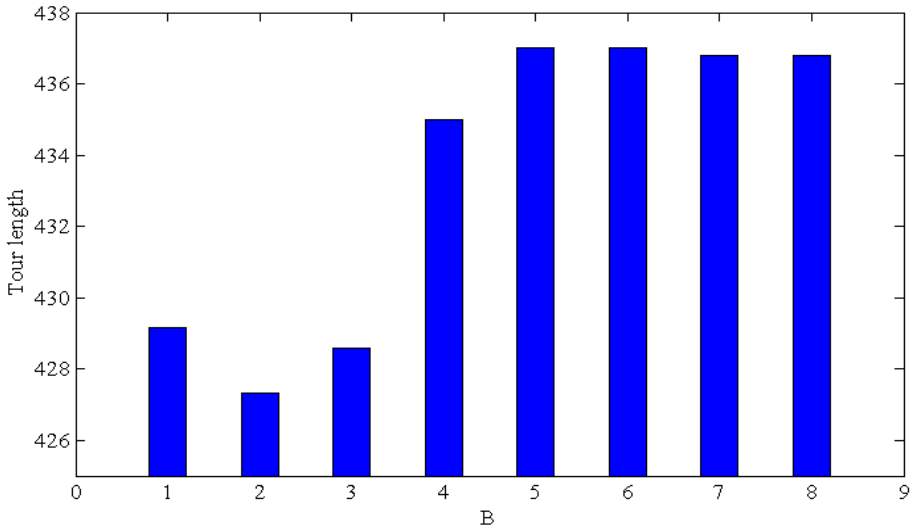

Figure 6. Average tour lengths of 15 runs on Eil51

Table 1 presents the numerical results of the proposed GA on TSP instances. The first column of the table indicates the instance name .The second column indicates the well known optimal tour length for each problem . The columns ‘‘Best”, ‘‘Average”, ‘‘Worst” and “ Std” show the best, average, worst and standard deviation of tour lengths of 15 runs, respectively. The last column indicates the average running time in seconds. The relative error, which is presented in the seventh column, is calculated as follows:

$$Err = \frac{Ave - opt}{opt} \times 100 \quad (2)$$

where *Ave* is the average tour length of all runs for each problem and *Opt* is the best-known solution.

Table1. Results of the proposed genetic algorithm for TSP instance

Problem	optimal	best	average	worst	Std	Err	Time(s)
Bruma14	3323	3323	3323	3323	0	0	0.139
F15	1105	1105	1105	1105	0	0	0.073
U16	6859	6859	6859	6859	0	0	0.450
C20	62568	62568	62568	62568	0	0	0.034
S21	60000	60000	60000	60000	0	0	0.041
U22	7013	7013	7013	7013	0	0	6.3
Wi29	27603	27603	27691.2	27750	75.9	0.319	3.95
C30	62716	62716	62716	62716	0	0	0.064
Olvir30	420	420	420.3	421	0.48	0.071	6.6
Ncit30	48873	48873	48873	48873	0	0	3.13
F32	84180	84180	84180	84180	0	0	0.071
Dj38	6656	6656	6656	6656	0	0	2.02
C40	6278	6278	6278	6278	0	0	0.071
F41	68160	68160	68160	68160	0	0	0.063
Att48	33522	33522	33522	33522	0	0	4.89
Eil50	425	426	426	426	0	0.235	3.7
Eil51	426	427	427	427	0	0.234	8.14
Mhd51	459	459	459	459	0	0	6.6
Berlin52	7542	7542	7598	7662	42	0.745	13
Ncit64	6400	6400	6400	6400	0	0	0.205
St70	675	678	680.2	684	1.98	0.77	12.6
Eil76	538	538	542.6	546	3.13	0.85	24.4
Pg88	6548	6548	6548	6548	0	0	4.4
Kroa100	21282	21334	21431.8	21542	67.5	0.7	81.5
Pr124	59030	59150	59446.7	59720	218	0.203	79
Pr226	80369	81206	81944	82363	362	1.959	112

Figure 7 shows the average convergence speed curves of the proposed crossover operator on four TSP instances: Bruma14, Dj38, Eil 51 and Pg88. It illustrates that the maximum error of the proposed algorithm is about 0.235% in the above-mentioned TSP instances.

In order to assess the performance of the proposed algorithm, some of the existing meta-heuristic algorithms are considered: genetic algorithm (GA), particle swarm optimization (PSO), artificial neural network (ANN), ant colony optimization (ACO), intelligent water drops (IWD), simulated annealing (SA) and chaotic ant swarm (CSA). Table 2 shows the capability of the proposed algorithm in comparison to the other meta-heuristic algorithms.

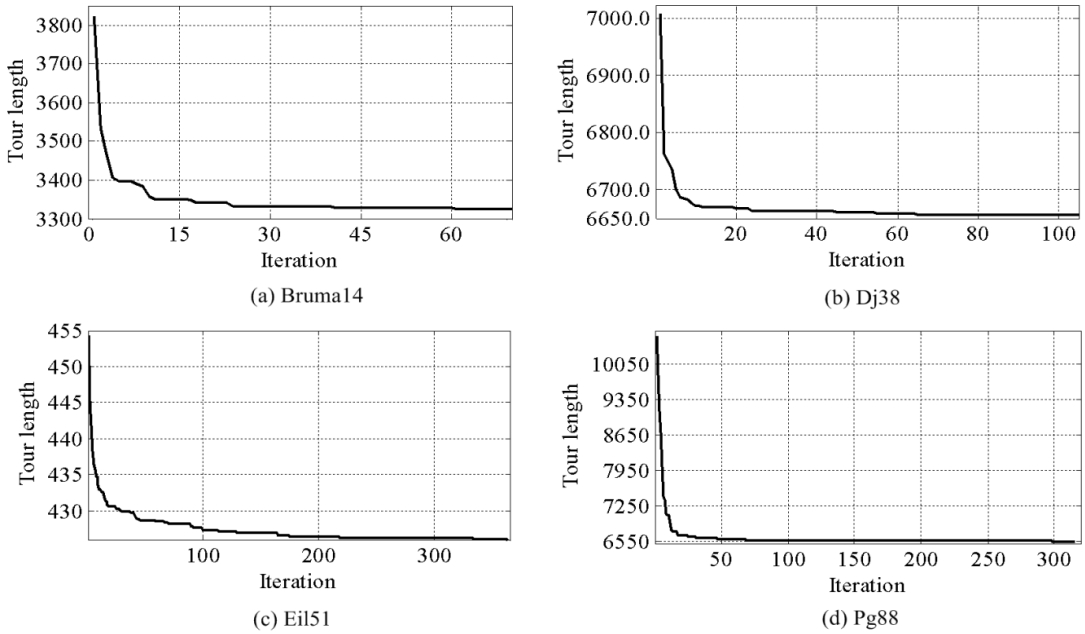

Figure 7. Convergence curves of the proposed algorithm

Table 2. Computational average results of several meta-heuristic approaches

Study	Method	Eil 51	Berlin 52	St 70	Eil 76	Kroa 100
Optimal solution	-	426	7542	676	538	21282
Proposed method	GA	**427**	**7598**	**680.2**	**542.6**	**21431.8**
Kuo et al (2010) [11]	GA	438	7738	N/A	N/A	22141
Ting et al (2010) [12]	GA	513	N/A	939	N/A	N/A
Cunkas and Ozsaglam (2009)[13]	GA	434	N/A	N/A	551	21852
Soak et al (2004) [10]	GA	429	N/A	N/A	N/A	21445
Licheng and Lei (2000) [14]	GA	446	N/A	N/A	568	22154
Cunkas and Ozsaglam (2009)[13]	PSO	436	N/A	N/A	555	22071
Shi et al (2007) [21]	PSO	436.9	7832	697.5	560.4	N/A
Shi et al (2006) [22]	PSO	444.6	7960	733.2	587.4	N/A
Li et al (2006) [20]	PSO	442	N/A	696	555	23036
Masutti et al (2009) [24]	ANN	437.4	7932.5	N/A	556.3	21522.7
Creput and Koukam (2009) [26]	ANN	435.1	N/A	681.7	553.5	21524.6
Cochrane and Beasley (2003) [23]	ANN	438	8070	N/A	561	21560
Somhom(1997) [25]	ANN	440.5	8025	N/A	562.2	21616
Puris et al (2007) [19]	ACO	N/A	7594	750	N/A	N/A
Tsai et al (2004) [17]	ACO	430	N/A	N/A	552.6	21457
Shah-Hosseini (2009) [28]	IWD	432.6	N/A	684.08	558	21904
Liu et al (2009) [27]	SA	432.5	7718.5	N/A	564	N/A
Wei (2011) [29]	CAS	439	N/A	N/A	559	21552

6. CONCLUSION

In this study a crossover operator is developed for solving TSP. The proposed method utilizes a logical reasoning for producing offspring. Computational results on 26 benchmark instances indicate that the proposed algorithm is very efficient to find the best-known solution especially for small-to-medium-scale instances. The most obvious finding to emerge from this study is that employing a crossover operator based on a logical reasoning can significantly improve the results of genetic algorithm. An important topic for future research would be the modification of the proposed crossover operator using fuzzy-logic systems, which are mostly suited to model logical reasoning issues. Finally, the overall procedure of the proposed method can be considered as a general framework and it could be extended to cover other routing and scheduling problems.

REFERENCES

1. Applegate D, Bixby R, Chvatal V, Cook W. *The Traveling Salesman Problem: A Computational Study*, Princeton University Press, 1st edition, New Jersey, USA, 2006.
2. Bentley J. Fast Algorithms for Geometric Traveling Salesman Problems, *Informs J Comput*, 1992; **4** (4): 387-411.
3. Clarke G, Wright J. Scheduling of vehicles from a central depot to a number of delivery points, *Oper Res*, 1964; **12** (4): 568-581.
4. Flood M. The traveling-salesman problem, *Oper Res* 1956; **4** (1): 61-75.
5. Croes G. A method for solving traveling-salesman problems, *Oper Res*, 1958; **6** (6): 791-812.
6. Helsgaun K. An effective implementation of the Lin-Kernighan traveling salesman heuristic, *Eur J Oper Res*, 2000; **126** (1): 106-130.
7. Marinakis Y, Migdalas A, Pardalos P. Expanding neighborhood GRASP for the traveling salesman problem, *Comput Optim Appl*, 2005; **32** (3): 231-257.
8. Lin S, Kernighan B. An effective heuristic algorithm for the traveling-salesman problem, *Oper Res*, 1973; **21** (2): 498-516.
9. Fiechter C. A parallel tabu search algorithm for large traveling salesman problems, *Discrete Appl Math*, 1994; **51** (3): 243-267.
10. Soak SM, Ahn BH. New Genetic Crossover Operator for the TSP, *Proceedings of the Seventh International Confrence on Artificial Intelligence and Soft Computing*, Zakopane, Poland, 2004, pp.480-485.
11. Kuo I , Horng SJ, Kao TW, Lin TL, Lee CL, Chen Y, Pan YI, Terano T. A hybrid swarm intelligence algorithm for the travelling salesman problem, *Expert Syst*, 2010; **27** (3): 166-179.
12. Ting CK, Su CH, Lee CN. Multi-parent extension of partially mapped crossover for combinatorial optimization problems, *Expert Syst Appl*, 2010; **37** (3): 1879-1886.
13. Çunkaş M, Ozsağlam M. A comparative study on particle swarm optimization and genetic algorithms for traveling salesman problems, *Cybern Syst*, 2009; **40** (6): 490 - 507.

14. Licheng J, Lei W. A novel genetic algorithm based on immunity, *IEEE Trans Syst Man Cybern Syst Hum*, 2000; **30** (5): 552-561.
15. Merz P, Freisleben B. Memetic algorithms for the traveling salesman problem, *Complex Syst*, 2001; **13** (4): 297-346.
16. Buriol L, França P , Moscato P. A New Memetic Algorithm for the Asymmetric Traveling Salesman Problem, *J Heuristics*, 2004; **10** (5): 483-506.
17. Tsai CF, Tsai CW, Tseng C. A new hybrid heuristic approach for solving large traveling salesman problem, *Inform Sci*, 2004; **166** (1-4): 67-81.
18. Chu SC, Roddick J, Pan JS. Ant colony system with communication strategies, *Inform Sci*, 2004; **167** (1-4): 63-76.
19. Puris A, Bello R, Martínez Y, Nowe A. Two-Stage Ant Colony Optimization for Solving the Traveling Salesman Problem, *Proceedings of the Second International Confrence on the Interplay Between Natural and Artificial Computation*, La Manga del Mar Menor, Spain, 2007, pp.307-316.
20. Li X, Tian P, Hua J, Zhong N. *A Hybrid Discrete Particle Swarm Optimization for the Traveling Salesman Problem*, Springer Berlin / Heidelberg, 2006.
21. Shi X, Liang Y, Lee H, Lu C, Wang Q. Particle swarm optimization-based algorithms for TSP and generalized TSP, *Inform Process Lett*, 2007; **103** (5): 169-176.
22. Shi X, Zhou Y, Wang L, Wang Q, Liang Y. A discrete particles swarm optimization algorithm for traveling salesman problem, *Proceedings of the First International Confrence on Computational Methods*, Singapore, Singapore, 2006, pp.1063-1068.
23. Cochrane E, Beasley J. The co-adaptive neural network approach to the Euclidean travelling salesman problem, *Neural Networks*, 2003; **16** (10): 1499-1525.
24. Masutti TA, de Castro L. A self-organizing neural network using ideas from the immune system to solve the traveling salesman problem, *Inform Sci*, 2009; **179** (10): 1454-1468.
25. Somhom S, Modares A, Enkawa T. A self-organising model for the travelling salesman problem, *J Oper Res Soc*, 1997; **48** (9): 919-928.
26. Créput J, Koukam A. A memetic neural network for the Euclidean traveling salesman problem, *Neurocomputing*, 2009; **72** (4-6): 1250-1264.
27. Liu Y, Xiong S, Liu H. Hybrid simulated annealing algorithm based on adaptive cooling schedule for TSP, *Proceedings of the first ACM/SIGEVO Summit on Genetic and Evolutionary Computation*, Shanghai, China, 2009, pp.895-898.
28. Shah-Hosseini H. The intelligent water drops algorithm: a nature-inspired swarm-based optimization algorithm, *J Bio-Inspired Comput*, 2009; **1** (1): 71-79.
29. Wei Z, Ge F, Lu Y, Li L, Yang Y. Chaotic ant swarm for the traveling salesman problem, *Nonlinear Dyn*, 2011; **65** (3): 271-281.
30. Raja Balachandar S, Kannan K. Randomized gravitational emulation search algorithm for symmetric traveling salesman problem, *Appl Math Comput* 2007; **192** (2): 413-421.
31. Larranaga P, Kuijpers C, Murga R, Inza I, Dizdarevic S. Genetic algorithms for the travelling salesman problem: A review of representations and operators, *Artific Intel Rev*, 1999; **13** (2): 129-170.