Multi-parent extension of sequential constructive crossover for the travelling salesman problem

Article *in* International Journal of Operational Research · July 2011

DOI: 10.1504/IJOR.2011.041347

1 author:

Zakir Hussain Ahmed
Imam Muhammad bin Saud Islamic University
25 PUBLICATIONS **267** CITATIONS

SEE PROFILE

Multi-parent extension of sequential constructive crossover for the travelling salesman problem

Zakir Hussain Ahmed

Department of Computer Science,
Al-Imam Muhammad Ibn Saud Islamic University,
P.O. Box No. 5701,
Riyadh 11432, Saudi Arabia
Fax: +96612591616
E-mail: zhahmed@gmail.com

Abstract: Crossover operator plays a vital role in genetic algorithms. This paper proposes the multi-parent sequential constructive crossover (MPSCX), which generalises the two-parent sequential constructive crossover (SCX) to a multi-parent crossover for the travelling salesman problem (TSP). Experimental results on five TSPLIB instances show that MPSCX significantly improves SCX by up to 4.60% in average tour value with maximum 4.01% away from the exact optimal solution. Finally, the efficiency of the MPSCX is compared as against multi-parent partially mapped crossover (MPPMX). Experimental results show that the MPSCX is better than the MPPMX.

Keywords: TSP; travelling salesman problem; NP-complete; heuristic; GAs; genetic algorithms; multi-parent crossover; SCX; sequential constructive crossover; selection survivor; mutation.

Reference to this paper should be made as follows: Ahmed, Z.H. (2011) 'Multi-parent extension of sequential constructive crossover for the travelling salesman problem', *Int. J. Operational Research*, Vol. 11, No. 3, pp.331–342.

Biographical notes: Zakir Hussain Ahmed is an Assistant Professor in the Department of Computer Science at Al-Imam Muhammad Ibn Saud Islamic University, Saudi Arabia. He received his MSc in Mathematics, MTech in Information Technology and PhD in Discrete Optimisation from Tezpur University (Central), Assam, India. He served in various institutions in India. His research interests include discrete optimisation, genetic algorithms, digital image processing and pattern recognition. He has publications in the fields of discrete optimisation, genetic algorithms and image processing.

1 Introduction

Genetic algorithms (GAs) are very good heuristic algorithms that have been used widely to solve a variety of combinatorial and complex optimisation problems (Ahmed, 2000; Samanlioglu et al., 2007; Sharma and Srivastava, 2009; Vasanthi and Arulmozhi, 2009). They are essentially based on mimicking the survival-of-the-fittest among the species generated by random changes in the gene-structure of the chromosomes in the evolutionary biology (Holland, 1975). A simple GA works by randomly generating an initial population of strings referred to as gene pool and then applying (possibly three)

operators to create new, and hopefully better, populations as successive generations. The first operator is reproduction, where chromosomes are copied to the next generation with some probability based on their objective function value. The second operator is crossover, where randomly selected pairs of chromosomes are mated to create new chromosomes. The third operator, mutation, is the occasional random alteration of the value at a chromosome position. The crossover operator together with reproduction is the most powerful process in the GA search. Mutation diversifies the search space and protects from loss of genetic material that can be caused by reproduction and crossover. So, the probability of applying mutation is set very low, whereas the probability of crossover is set very high (Goldberg, 1989).

Though GAs can solve some combinatorial and complex optimisation problems, but sometimes they lead to premature convergence, and if we do not use any advanced strategy, they take long time to obtain optimal solutions. Hence, they may get stuck in local minima when using traditional GAs for solving complex optimisation problems (Li and Chen, 2009). So, one has to consider some advanced strategies to improve the quality of the solution obtained by traditional GAs, especially, crossover operator, as it plays a very important role in GAs. Many researchers focus on developing new crossover operator that fits to one of the various representations for a chromosome. Unfortunately, offspring created by most of the crossover operators do not inherit enough information from its parents. Also, there have been various efforts in hybridising GAs with local search algorithms in the hope that the hybrid algorithm can explore a better trade-off between computational effort and global optimality of the solution obtained (Samanlioglu et al., 2007; Sharma and Srivastava, 2009).

Now-a-days, multi-parent extension of a crossover operator is used to improve the solution quality for optimisation problems. Traditionally, only two parents are selected at a time for crossover and produce one or two offspring. Of course, in the natural biology, no species apply multi-parent reproduction. But, in computer simulations, it is not necessary to restrict the number of parents for crossover to two only. The justification of the need for the multi-parent crossover is well described (Ting, 2005). The crossover that uses multiple parents for crossover is called multi-parent crossover and the GAs that use multi-parent crossover are called multi-parent genetic algorithms (MPGAs). With respect to the number of parents, MPGAs are generalisations of GAs. Of course, MPGAs is an old idea (Eiben et al., 1994), which has gained restored attention in the past few years (Porumbel et al., 2010; Ting et al., 2010; Wang et al., 2008; Wu et al., 2009). As reported, multi-parent crossovers were found to be better than the traditional crossover. Also, a detailed study on the usefulness of multi-parent crossover is carried out by Eiben (2002).

The travelling salesman problem (TSP) is one of the benchmark and old problems in computer science and operations research. It is proved to be NP-complete problem (Papadimitriou and Steglitz, 1997). It is an important representative of combinatorial optimisation problems, and many real-life problems can be transformed into the TSP, e.g. scheduling problem of printed circuit board components (Ho and Ji, 2009), X-ray crystallography (Bland and Shallcross, 1989), etc. GAs are very good heuristic algorithms that have been used widely to solve the TSP instances. Since, crossover operator plays a vital role in GAs, various two-parent crossover operators and few multi-parent extensions of crossovers have been developed for the TSP. However, effective multi-parent extension of crossover for solving the TSP as well as combinatorial optimisation problems is lacking. In this paper, we propose the multi-parent sequential

constructive crossover (MPSCX) for the TSP. The MPSCX generalises the sequential constructive crossover (SCX) (Ahmed, 2010), which is one of the best crossover operators for the TSP. Here, we examine whether multi-parent recombination should be avoided for practical reasons or it offers advantages for the TSP. The evaluation of our proposed MPSCX will focus on the performance on five TSPLIB instances. Finally, a comparative study is carried out between the best solution values obtained by the proposed MPSCX and the multi-parent partially mapped crossover (MPPMX) (Ting et al., 2010).

This paper is organised as follows: a literature review has been carried out for the multi-parent crossover operators for solving the TSP in Section 2. Section 3 gives a brief review of a genetic algorithm using SCX for the TSP. Section 4 describes the proposed MPSCX. Section 5 presents computational experience for five TSPLIB instances. Finally, Section 6 presents comments and concluding remarks.

2 Literature review

The TSP can be stated as:

A network with 'n' nodes, with node 1 as 'headquarters' and a travel cost (or distance, or travel time, etc.) matrix $C = [c_{ij}]$ of order n associated with ordered node pairs (i, j) is given. The problem is to find a least cost Hamiltonian cycle, i.e. to obtain a tour $\{1 = \alpha_0, \alpha_1, \alpha_2, \ldots, \alpha_{n-2}, \alpha_{n-1}, \alpha_n = 1\} = \{1 \to \alpha_1 \to \alpha_2 \to \ldots \to \alpha_{n-2} \to \alpha_{n-1} \to 1\}$ for which the total travel cost, $C(1 = \alpha_0, \alpha_1, \alpha_2, \ldots, \alpha_{n-2}, \alpha_{n-1}, \alpha_n = 1) = \sum_{i=0}^{n-1} c(a_i, a_{i+1})$, is minimum.

On the basis of the structure of the cost matrix, the TSPs are classified as symmetric and asymmetric. The TSP is symmetric if $c_{ij} = c_{ji}, \forall i, j$ and asymmetric otherwise.

Several multi-parent crossovers and MPGAs have been proposed for various optimisation problems and have shown their superiority over GAs. In binary-coded GAs, Eiben et al. (1994) and Eiben and van Kemenade (1997) proposed scanning crossover and diagonal crossover as the generalisation of uniform crossover and one-point crossover, respectively. As reported, their multi-parent extensions of crossovers are found better than the two-parent versions for some test functions. For real-coded GAs, Tsutsui and Ghosh (1998) presented a series of multi-parent crossovers: centre of mass crossover, multi-parent feature-wise crossover and seed crossover. As reported, multi-parent crossovers were found to be better. Many multi-parent crossover operators have been developed for solving various problems, e.g. Shekel problems (Lin et al., 2007), multi-modal function optimisation (Wang et al., 2008), complex function optimisation (Li and Chen, 2009), bounded diameter minimum spanning tree problem (Binh and Nghia, 2009), object shape matching problem (Wu et al., 2009) and graph colouring problem (Lü and Hao, 2010; Porumbel et al., 2010). In all these studies multi-parent crossovers have shown their superiority over classic two-parent crossovers.

However, the above multi-parent crossover operators do not work on the TSP, and the literature for the multi-parent crossover for the TSP is very few. Ting (2007) proposed two variants of multi-parent edge recombination (MPEX) to enhance order-based GAs using edge recombination for the TSP. The experimental results on two TSPLIB instances show that MPEX outperforms the two-parent edge recombination crossover. Ting et al. (2010) proposed MPPMX that generalises the partially mapped crossover to

multi-parent crossover. The experimental results on five TSPLIB instances show that MPPMX significantly improves two-parent partially mapped crossover. Although there is a significant improvement, the best solutions are at least 20.62% away from the exact solutions.

3 Genetic algorithm using SCX

To apply GA for TSP, one has to think a way for encoding solutions as feasible chromosomes so that the crossovers of feasible chromosomes result in feasible chromosomes. Most of the literature considers the order representation for a chromosome, which simply lists the label of nodes. For example, let {1, 2, 3, 4, 5} be the labels of nodes in a 5-node instance, then a tour $\{1 \rightarrow 3 \rightarrow 4 \rightarrow 2 \rightarrow 5 \rightarrow 1\}$ may be represented as (1, 3, 4, 2, 5). We consider this order representation for the chromosomes.

The GAs are used for maximisation problem and since, TSP is a minimisation problem; most of the literature consider the fitness function $F(x) = 1/f(x)$, where $f(x)$ calculates cost (or value) of the tour represented by a chromosome. Reproduction/selection is the first operation of a GA that copies some chromosomes into next generation mating pool with a probability associated with their fitness value. By assigning to next generation a higher portion of the highly fit chromosomes, reproduction mimics the Darwinian survival-of-the-fittest in the natural world. In this phase, no new chromosome is produced. The commonly used reproduction operator is the proportionate reproduction operator, where a chromosome is selected for the mating pool with a probability proportional to its fitness value. We have considered the stochastic remainder selection method (Deb, 1995) for our GAs.

The basic crossover operators, such as one-point, multi-point and uniform crossover operators, do not directly support for the TSP. To obtain valid chromosome after crossover, many crossover operators have been proposed. SCX is one of the best crossover operators for the TSP (Ahmed, 2010). The algorithm for the SCX is as follows:

Step 1 Start from 'node 1' (i.e. current node $p = 1$).

Step 2 Sequentially search both of the parent chromosomes and consider the first 'legitimate node' (the node i.e. not yet visited) appeared after 'node p' in each parent. If no 'legitimate node' after 'node p' is present in any of the parent, search sequentially the nodes {2, 3, …, n} and consider the first 'legitimate' node, and go to Step 3.

Step 3 Suppose the 'node α' and the 'node β' are found in 1st and 2nd parent, respectively, then for selecting the next node go to Step 4.

Step 4 If $c_{p\alpha} < c_{p\beta}$, then select 'node α', otherwise, 'node β' as the next node and concatenate it to the partially constructed offspring chromosome. If the offspring is a complete chromosome, then stop, otherwise, rename the present node as 'node p' and go to Step 2.

Let us illustrate the SCX through the example given as cost matrix in Table 1. Let a pair of selected chromosomes be P_1: (1, 5, 7, 3, 6, 4, 2) and P_2: (1, 6, 2, 4, 3, 5, 7) with values 312 and 331, respectively.

Table 1 The cost matrix

Node	*1*	*2*	*3*	*4*	*5*	*6*	*7*
1	999	75	99	9	35	63	8
2	51	999	86	46	88	29	20
3	100	5	999	16	28	35	28
4	20	45	11	999	59	53	49
5	86	63	33	65	999	76	72
6	36	53	89	31	21	999	52
7	58	31	43	67	52	60	999

Select 'node 1' as the 1st gene. The 'legitimate' nodes after 'node 1' in P_1 and P_2 are 'node 5' and 'node 6', respectively, with $c_{15} = 35$ and $c_{16} = 63$. Since $c_{15} < c_{16}$, we accept 'node 5'. So, the partially constructed chromosome will be (1, 5). The 'legitimate' node after 'node 5' in both P_1 and P_2 is 'node 7'. So, we accept the 'node 7', and the partially constructed chromosome will be (1, 5, 7). The 'legitimate' node after 'node 7' in P_1 is 'node 3', but none in P_2. So, for P_2, we consider the first 'legitimate' node in the set {2, 3, 4, 5, 6, 7}, i.e. 'node 2'. Since $c_{72} = 31 < 43 = c_{73}$, we accept 'node 2'. Thus, the partially constructed chromosome will be (1, 5, 7, 2). Proceeding in this way, it may lead to the complete offspring chromosome as (1, 5, 7, 2, 4, 3, 6) with value 266 which is less than value of both the parent chromosomes.

Now-a-days after performing crossover operation, survivor selection method is used for selecting next generation population. Traditionally, the survivor selection of GA considers only the fitter chromosomes. In natural biology, the survivability of an individual is, however, concerned with many factors in addition to fitness. For example, breeding of related individuals will cause a decrease in fitness. To avoid this decrease, humans generally avoid marriage in blood relations. The survivor selection of GA considers two kinds of chromosomes for the next generation:

1 parents in current population of size m

2 offspring that are generated by crossover of size n.

The procedure for $(\mu + \lambda)$ survivor selection method combines chromosomes in (1) and (2), sorts them in ascending order according to their fitness, and considers the first m chromosomes for the next generation. In worst case, all the μ parents in the present generation will survive into the next generation.

Mutation is the third operator of GA that randomly selects a position in a chromosome and changes the corresponding gene, thereby modifies the information. The basic mutation operator does not produce valid chromosome for the TSP. For this investigation, we have considered the reciprocal exchange mutation that selects two nodes randomly and swaps them.

4 Multi-parent SCX

In this section, we propose the MPSCX to extend SCX into a multi-parent crossover to obtain better quality of the solution. Unlike the MPPMX (Ting et al., 2010), there is no any modification of the two-parent crossover to legalise the offspring. Of course, there

are little modifications in the steps for generalisation. Step 2 is modified to have more options in selecting next node; whereas the other steps are the generalised steps. The algorithm for the MPSCX is as follows:

Step 1 Start from 'node 1' (i.e. current node $p = 1$).

Step 2 Sequentially search all of the parent chromosomes and consider the first 'legitimate node' appeared after 'node p' in each parent. If no 'legitimate node' after 'node p' is present in any of the parent, search sequentially from the beginning of the chromosome and consider the first 'legitimate node', and go to Step 3.

Step 3 Suppose the nodes $\{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ are found in 1st, 2nd, …, nth parent chromosomes, respectively, then for selecting the next node go to Step 4.

Step 4 Select the 'node α'= $\min\{c_{p,\alpha_1}, c_{p,\alpha_2}, \ldots, c_{p,\alpha_n}\}$ as the next node and concatenate it to the partially constructed offspring. Now, if the offspring is a complete chromosome, then stop, otherwise, rename the present node as 'node p' and go to Step 2.

Let us illustrate the MPSCX for three parents through the same example given as the cost matrix in Table 1. Let the selected three chromosomes be P_1: (1, 5, 7, 3, 6, 4, 2), P_2: (1, 6, 2, 4, 3, 5, 7) and P_3: (1, 4, 2, 3, 6, 7, 5) with values 312, 331 and 365, respectively.

Select 'node 1' as the 1st gene. The 'legitimate' nodes after 'node 1' in P_1, P_2 and P_3 are nodes 5, 6 and 4, respectively, with $c_{15} = 35$, $c_{16} = 63$ and $c_{14} = 9$. Since c_{14} is the smallest, we accept 'node 4'. Thus, the partially constructed chromosome will be (1, 4). The 'legitimate' nodes after 'node 4' in P_1, P_2 and P_3 are 2, 3 and 2, respectively, with $c_{42} = 45$ and $c_{43} = 11$. Since c_{43} is the smallest, we accept 'node 3'. So, the partially constructed chromosome will be (1, 4, 3). The 'legitimate' nodes after 'node 3' in P_1, P_2 and P_3 are 6, 5 and 6, respectively, with $c_{36} = 35$ and $c_{35} = 28$. Since c_{35} is the smallest, we accept 'node 5'. So, the partially constructed chromosome will be (1, 4, 3, 5). The 'legitimate' nodes after 'node 5' in P_1, P_2 and P_3 are 7, 7 and 2, respectively, with $c_{57} = 72$ and $c_{52} = 63$. Since c_{52} is the smallest, we accept 'node 2'. So, the partially constructed chromosome will be (1, 4, 3, 5, 2). The 'legitimate' nodes after 'node 2' in P_1, P_2 and P_3 are 7, 7 and 6, respectively, with $c_{27} = 20$ and $c_{26} = 29$. Since c_{27} is the smallest, we accept 'node 7'. So, the partially constructed chromosome will be (1, 4, 3, 5, 2, 7). Finally, the only legitimate node is 6 that leads to the complete offspring chromosome as (1, 4, 3, 5, 2, 7, 6) with value 227 which is less than value of all the parent chromosomes.

5 Computational experience

For examining the performance of MPSCX, we consider the same five TSPLIB instances which are considered for MPPMX (Ting et al., 2010): eil51, st70, pr76, lin105 and d198. Initial population is generated randomly. The following parameters are selected for our algorithm and also by Ting et al. (2010): population size is 100; crossover probability is 1.0 for SCX, and MPSCX with number of parents ranges from 2 to 5 (except for lin105); six mutation probabilities: 0, 0.01, 0.02, 0.03, 0.04 and 0.05; maximum 5,000 generations as termination condition and 30 independent runs for each setting.

Table 2 reports the mean and standard deviation (in parenthesis) of the best tour values over 30 trials of SCX and MPSCX with 2–10 parents on lin105 for different

mutation probabilities. The italic values denotes significant improvement of MPSCX over SCX. Due to the similarity of results from different TSPLIB instances, we present results of the lin105 instance only.

In terms of the respective best results for all test mutation probabilities, MPSCX (14,928.5) improves SCX (15,630.2) by 4.49% in solution quality. It is also seen that suitable number of parent depends on mutation probability. For different mutation probabilities (except 0), MPSCX using 2–10 parents can outperform SCX and finds best performance with $n = 2$ or 3 or 4. Of course, for $P_m = 0.0$, MPSCX outperforms SCX with $n = 9$. It is observed that as the mutation probability increases the number of parents for which MPSCX gives better solution than SCX decreases. The improvements of MPSCX over SCX range from 0.91% to 40.21%, and all have statistical significance. However, the increase of parents in MPSCX slows the convergence and the algorithm tends to be greedy. SCX and MPSCX obtain best solutions at mutation probabilities 0.04 and 0.01, respectively. Anyway, two-parent MPSCX outperforms SCX for all mutation probabilities.

Figure 1 plots the mean of best tour values obtained from SCX, and MPSCX with 2–10 parents for mutation probabilities from 0.01 to 0.04. The figure shows that MPSCX with 2–10 parents outperforms SCX for the mutation probability 0.01. Overall, MPSCX with 2–4 parents can lead to significant improvement of MPSCX over SCX. Therefore, for the other four instances we consider MPSCX with 2–5 parents only.

Figure 2 compares anytime behaviour of SCX and MPSCX with 2–5 parents at $P_m = 0.01$ and maximum 1,000 generations for lin105. The figure shows that GAs using MPSCX converges faster than that using SCX in terms of solution quality, but gets stuck in local minima quickly. Of course, in terms of computational time, SCX is the fastest and as number of parents increases for MPSCX computational time also increases.

Table 3 summarises the best tour values for SCX and *n*-parent MPSCX among all test mutation probabilities for five TSPLIB instances. It reports mean, standard deviation (in parenthesis) and the mutation probability (mark by an asterisk) corresponding to the best tour value over 30 trials of SCX and *n*-parent MPSCX. The italic values denotes the best results with respect to the TSP instances. The experimental results show that MPSCX outperforms SCX in solution quality by 2.23% for eil51, 3.50% for st70, 2.97% for pr76, 4.49% for lin105 and 4.60% for d198.

Table 2 Mean and standard deviation of the tour values of SCX and *n*-parent MPSCX on lin105

Crossover	$P_m = 0$	$P_m = 0.01$	$P_m = 0.02$	$P_m = 0.03$	$P_m = 0.04$	$P_m = 0.05$
SCX	28,588.7	16,116.4	15,822.8	15,682.6	15,630.2	16,168.8
	(3,245.7)	(592.5)	(633.9)	(519.1)	(433.3)	(558.3)
MPSCX ($n = 2$)	*26,166.6*	*16,097.6*	*15,818.4*	*15,597.6*	*15,444.3*	*16,021.8*
	(2,893.4)	(745.4)	(660.3)	(612.3)	(559.5)	(590.0)
MPSCX ($n = 3$)	*19,647.1*	*14,928.5*	*15,331.3*	*15,287.3*	15,631.6	16,248.5
	(1,084.5)	(168.0)	(303.9)	(204.7)	(290.7)	(444.6)

Table 2 Mean and standard deviation of the tour values of SCX and *n*-parent MPSCX on lin105 (continued)

Crossover	$P_m = 0$	$P_m = 0.01$	$P_m = 0.02$	$P_m = 0.03$	$P_m = 0.04$	$P_m = 0.05$
MPSCX ($n = 4$)	*18,177.8*	*15,424.3*	*15,256.1*	*15,651.2*	16,808.3	16,893.7
	(484.0)	(263.8)	(210.2)	(354.7)	(828.1)	(587.2)
MPSCX ($n = 5$)	*17,395.1*	*15,353.1*	*15,335.6*	16,034.3	16,711.3	17,004.9
	(696.6)	(327.3)	(298.6)	(352.3)	(757.5)	(514.3)
MPSCX ($n = 6$)	*17,475.4*	*15,487.8*	*15,469.5*	16,354.6	16,829.0	17,569.5
	(553.9)	(218.1)	(288.2)	(732.7)	(909.5)	(116.7)
MPSCX ($n = 7$)	*17,092.9*	*15,569.2*	16,189.8	16,285.2	17,154.1	17,165.0
	(506.3)	(256.3)	(407.6)	(548.1)	(522.7)	(644.0)
MPSCX ($n = 8$)	*17,123.1*	*15,528.0*	16,467.4	16,542.1	17,238.7	17,689.7
	(719.8)	(341.1)	(382.3)	(487.7)	(523.1)	(349.2)
MPSCX ($n = 9$)	*17,091.3*	*15,595.7*	16,815.9	16,832.7	16,883.9	17,978.3
	(507.7)	(27.8)	(733.0)	(666.9)	(536.7)	(316.0)
MPSCX ($n = 10$)	*17,292.5*	*15,484.3*	16,162.6	16,722.8	17,549.9	17,399.3
	(518.4)	(285.4)	(418.0)	(734.3)	(445.1)	(573.0)

Figure 1 Mean of best tour values obtained from SCX and *n*-parent MPSCX for lin105

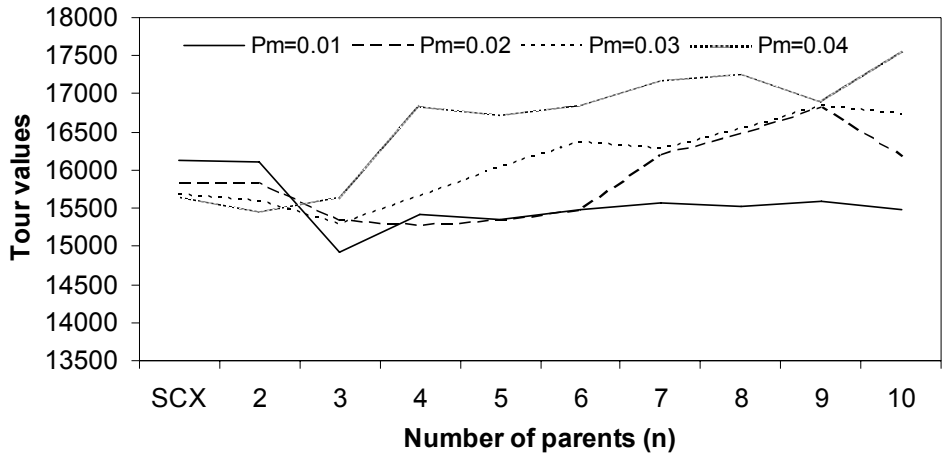

Figure 2 Convergence of GAs using SCX and *n*-parent MPSCX at $P_m = 0.01$ for lin105

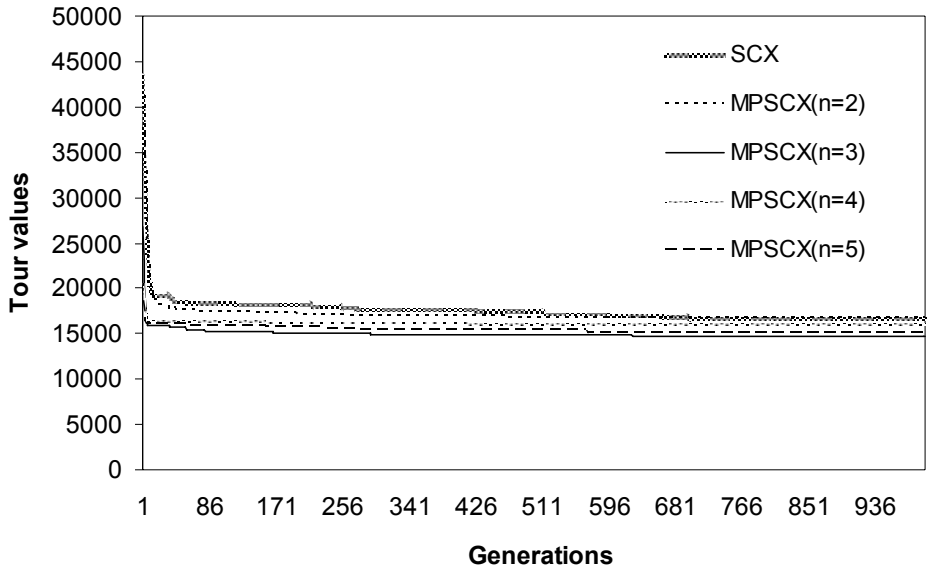

Table 3 Summary of the best tour values for SCX and *n*-parent MPSCX on five instances

MPSCX	*eil51*	*st70*	*pr76*	*lin105*	*d198*
SCX	444.1 (9.39)	720.6 (16.65)	115,902.0 (3,700.00)	15,630.2 (433.27)	17,203.2 (300.5)
	0.05*	0.05*	0.04*	0.04*	0.03*
MPSCX (n = 2)	445.1 (7.47)	713.2 (15.77)	115,095.9 (3,274.86)	15,444.3 (559.5)	17,203.0 (343.5)
	0.03*	0.04*	0.04	0.04*	0.03*
MPSCX (n = 3)	437.7 (2.92)	712.2 (15.37)	*112,454.3* (1,710.09)	*14,928.5* (168.0)	17,122.4 (182.4)
	0.03*	0.03*	0.02*	0.01*	0.03
MPSCX (n = 4)	438.8 (5.71)	*695.4* (12.70)	114,918.4 (1,475.73)	15,256.1 (210.2)	*16,412.1* (231.6)
	0.03*	0.04*	0.02*	0.02*	0.02*
MPSCX (n = 5)	*434.2* (4.40)	728.0 (19.50)	114,957.2 (1,146.02)	15,335.6 (298.6)	16,982.7 (191.4)
	0.05*	0.02*	0.03*	0.02*	0.02*

The significance of the improvement of MPSCX over SCX is further validated by the statistical one-tailed *t*-test with a confidence level 0.05. Table 4 shows the *p*-values of the *t*-test with confidence level 0.05 on the best tour value between SCX and *n*-parent MPSCX for the five instances. The italic values denotes the significant improvement of MPSCX over SCX. The table clearly indicates that MPSCX with 2–5 parents can lead to significant improvement over SCX on these five test instances.

We finally compare MPSCX and MPPMX, as reported by Ting et al. (2010), using same setting of parameters. Table 5 reports best mean solution value, optimal number of parents, mutation probability and solution quality, for the five test instances obtained by MPPMX and MPSCX. The solution quality is measured by the percentage of excess

above the exact optimal solution value reported in TSPLIB (1995) website, as given by the formula

$$\text{Excess}(\%) = \frac{\text{Solution value} - \text{Optimal solution value}}{\text{Optimal solution value}} \times 100$$

Table 5 shows that the solution quality obtained by MPPMX ranges from 20.61% to 104.75% away from the optimal solution, whereas our MPSCX ranges from only 1.92% to 4.01%. Also, the best solutions are obtained using at least five parents by MPPMX, whereas by MPSCX, best solutions are obtained using at most five parents. That means, the computational time would be more using MPPMX than using MPSCX. Hence, it can be conclude that our MPSCX is far better than MPPMX.

Table 4 *P*-values of *t*-test on the best tour values between SCX and *n*-parent MPSCX

n	*eil51*	*st70*	*pr76*	*lin105*	*d198*
2	3.49×10^{-01}	8.89×10^{-02}	2.24×10^{-01}	1.53×10^{-01}	4.99×10^{-01}
3	4.54×10^{-03}	9.11×10^{-02}	6.00×10^{-04}	4.21×10^{-07}	1.93×10^{-01}
4	1.45×10^{-02}	1.54×10^{-06}	1.47×10^{-01}	1.49×10^{-03}	4.15×10^{-09}
5	3.10×10^{-04}	1.25×10^{-01}	1.46×10^{-01}	1.52×10^{-02}	5.76×10^{-03}

Table 5 A comparative study between MPPMX and MPSCX

		MPPMX (Ting et al., 2010)				*MPSCX*			
Instance	*Optimal value*	*Best mean value*	*No. of parents*	P_m	*%*	*Best mean value*	*No. of parents*	P_m	*%*
eil51	426	513.86	5	0.04	20.62	434.2	5	0.05	1.92
st70	675	939.41	9	0.03	39.17	695.4	4	0.04	3.02
pr76	108,159	152,737.9	9	0.04	41.22	112,454.3	3	0.02	3.97
lin105	14,379	24,444.9	10	0.005	70.00	14,928.5	3	0.01	3.82

6 Conclusions

In this paper, we developed a MPSCX, which is an extension of SCX. Unlike the MPPMX (Ting et al., 2010), there is no modification required for the SCX to legalise the offspring. Of course, we modified the existing SCX to improve the solution quality and we achieved the goal. A series of experiments were carried out to evaluate the performance of MPSCX with different number of parents at different mutation probabilities. Experimental results on five TSPLIB instances show that MPSCX significantly improves SCX by up to 4.60% in average tour value with maximum 4.01% away from the exact optimal solution. The significance of such improvement is validated by a *t*-test. It can be concluded that multi-parent crossover is superior to the traditional two-parent crossover for the TSP. Experimental results also show that there is a relation between number of parents and mutation probability. Finally, in terms of solution quality, the comparative study shows that MPSCX is better than MPPMX.

Although the results show that the performance of MPSCX increases as the number of parents is raised above 2, however, the tests provide no solid ground to draw conclusion on optimal number of parents, for instances, the optimal number of parents for the five instances are 5, 4, 3, 3 and 4, respectively, as shown in Table 5. Of course, considering larger number of parents does not guarantee the better performance of MPSCX, rather it becomes greedy. The performance improves up to a certain number of parents and decreases afterwards, or even fluctuates. So, a further investigation for deciding the optimal number of parents in different stages of GA could be done. To understand phenomenon of MPSCX, we need to see the uniformity of the selected chromosomes for crossover, and the power of producing better offspring by measuring parent–child fitness ratio. By introducing incest prevention, the performance of MPSCX might be improved. In addition, computational time of using more than two parents in MPSCX and the effects of hybridising with some local search need to be considered. Finally, the advantage and usefulness of MPSCX should be verified on other combinatorial optimisation problems.

Acknowledgements

The author is thankful to the honourable anonymous reviewer for his constructive comments and suggestions, which helped him to improve this paper.

References

Ahmed, Z.H. (2000) 'A sequential constructive sampling and related approaches to combinatorial optimization', PhD Thesis, Tezpur University, Assam, India.

Ahmed, Z.H. (2010) 'Genetic algorithm for the traveling salesman problem using sequential constructive crossover', *Int. J. Biometrics and Bioinformatics*, Vol. 3, No. 6, pp.96–105.

Binh, H.T.T. and Nghia, N.D. (2009) 'New multi-parent recombination in genetic algorithm for solving bounded diameter minimum spanning tree problem', *Proceedings of 1st Asian Conference on Intelligence Information and Database Systems*, pp.283–288.

Bland, R.G. and Shallcross, D.F. (1989) 'Large traveling salesman problems arising form experiments in x-ray crystallography – a preliminary report on computation', *Operations Research Letters*, Vol. 8, pp.125–128.

Deb, K. (1995) *Optimization for Engineering Design: Algorithms and Examples*. New Delhi, India: Prentice Hall of India Pvt. Ltd.

Eiben, A. (2002) 'Multi-parent recombination in evolutionary computing', in A. Ghosh and S. Tsutsui (Eds.), *Advances in Evolutionary Computing*. Heidelberg: Springer, pp.175–192.

Eiben, A., Rouě, P-E. and Ruttkay, Z. (1994) 'Genetic algorithms with multi-parent recombination', *Parallel Problem Solving from Nature – PPSN III*. LNCS 866, Springer, pp.78–87.

Eiben, A. and van Kemenade, C. (1997) 'Diagonal crossover in genetic algorithms for numerical optimization', *Journal of Control and Cybernetics*, Vol. 26, No. 3, pp.447–465.

Goldberg, D.E. (1989) *Genetic Algorithms in Search, Optimization, and Machine Learning*. New York: Addison-Wesley.

Ho, W. and Ji, P. (2009) 'An integrated scheduling problem of PCB components on sequential pick-and-place machine: mathematical models and heuristic solutions', *Expert Systems with Applications*, Vol. 36, No. 3, pp.7002–7010.

Holland, J. (1975) *Adaptation in Natural and Artificial Systems*. Ann Arbor, Michigan, USA: University of Michigan Press.

Li, Y. and Chen, S. (2009) 'A multi-stage evolutionary algorithm for solving complex function optimization problems', *Second International Conference on Computer and Electrical Engineering*, pp.516–519.

Lin, G., Kang, L., Chen, Y., McKay, B. and Sarker, R. (2007) 'A self-adaptive mutations with multi-parent crossover evolutionary algorithm for solving function optimization problems', *Lecture Notes in Computer Science*, Vol. 4683, pp.157–168.

Lü, Z. and Hao, J-K. (2010) 'A memetic algorithm for graph coloring', *European Journal of Operational Research*, Vol. 203, No. 1, pp.241–250.

Papadimitriou, C.H. and Steglitz, K. (1997) *Combinatorial Optimization: Algorithms and Complexity*. India: Prentice Hall of India Private Limited.

Porumbel, D.C., Hao, J-K. and Kunz, P. (2010) 'An evolutionary approach with diversity guarantee and well-informed grouping recombination for graph coloring', *Computers and Operations Research*, Vol. 37, No. 10, pp.1822–1832.

Samanlioglu, F., Kurz, M.B., Ferrell, W.G. and Tangudu, S. (2007) 'A hybrid random-key genetic algorithm for a symmetric traveling salesman problem', *Int. J. Operational Research*, Vol. 2, pp.47–63.

Sharma, R. and Srivastava, K. (2009) 'A new hybrid evolutionary algorithm for the MinLA problem', *Int. J. Operational Research*, Vol. 5, pp.229–249.

Ting, C-K. (2005) 'Design and analysis of multi-parent genetic algorithms', PhD Thesis, University of Paderborn, Germany.

Ting, C-K. (2007) 'Multi-parent extension of edge recombination', *GECCO'07*, London, England, p.1535.

Ting, C-K., Su, C-H. and Lee, C-N. (2010) 'Multi-parent extension of partially mapped crossover for combinatorial optimization problems', *Expert Systems with Applications*, Vol. 37, pp.1879–1886.

Tsutsui, S. and Ghosh, A. (1998) 'A study on the effect of multi-parent recombination in real coded genetic algorithms', *Proceedings of International Conference on Evolutionary Computation*, pp.828–833.

TSPLIB (1995) Available at: http://www.iwr.uni-heidelberg.de/iwr/comopt/software/TSPLIB95/.

Vasanthi, T. and Arulmozhi, G. (2009) 'Optimal allocation problem using genetic algorithm', *Int. J. Operational Research*, Vol. 5, No. 2, pp.211–228.

Wang, H., Wu, Z. and Liu, Y. (2008) 'Particle swarm optimization with a novel multi-parent crossover operator', *Proceedings of 4th International Conference on Natural Computation*, pp.664–668.

Wu, A., Tsang, P.W.M., Yuen, T.Y.F. and Yeung, L.F. (2009) 'Affine invariant object shape matching using genetic algorithm with multi-parent orthogonal recombination and migrant principle', *Applied Soft Computing*, Vol. 9, pp.282–289.