Expert Systems with Applications 37 (2010) 1879–1886

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Multi-parent extension of partially mapped crossover for combinatorial optimization problems

Chuan-Kang Ting [a,*], Chien-Hao Su [b], Chung-Nan Lee [b]

[a] Department of Computer Science and Information Engineering, National Chung Cheng University, Chia-Yi 62102, Taiwan
[b] Department of Computer Science and Engineering, National Sun Yat-sen University, Kaohsiung 80424, Taiwan

ARTICLE INFO

Keywords:
Genetic algorithms
Multi-parent crossover
Partially mapped crossover (PMX)
Combinatorial optimization
Traveling salesman problem (TSP)

ABSTRACT

This paper proposes the multi-parent partially mapped crossover (MPPMX), which generalizes the partially mapped crossover (PMX) to a multi-parent crossover. The mapping list and legalization of PMX are modified to deal with the issues that arise from the increase of parents in PMX. Experimental results on five traveling salesman problems show that MPPMX significantly improves PMX by up to 13.95% in mean tour length. These preferable results not only demonstrate the advantage of the proposed MPPMX over PMX, but also confirm the merit of using more than two parents in crossover.

1. Introduction

Genetic algorithms (GAs) (Holland, 1975) have effectively solved a variety of combinatorial and numerical optimization problems. The operators in GAs include the selection, crossover, mutation, and survivor. Crossover is the most salient GA operator. It produces offspring by recombining parental genetic material. Traditionally, the number of parents used in crossover is two. This idea is reasonable because, to the best of our knowledge, no organisms on Earth apply multi-parent reproduction. In computer simulations, nevertheless, it is not necessary to limit the number of parents for crossover to two. This idea, to use more than two parents in crossover, is then implemented as multi-parent crossover. The GAs that use multi-parent crossover are named multi-parent genetic algorithms (Ting, 2005).

Beyond two parents in binary-coded GAs, Eiben et al. proposed scanning crossover (Eiben, Raué, & Ruttkay, 1994) and diagonal crossover (Eiben & van Kemenade, 1997; Eiben, van Kemenade, & Kok, 1995) as the generalization of uniform crossover and one-point crossover, respectively. Their experimental results on several test functions show that, in terms of success rate, both scanning crossover and diagonal crossover outperform their two-parent versions, namely uniform crossover and one-point crossover. Mühlenbein, Schomisch, and Born (1991) and Voigt and Mühlenbein (1995) introduced the concept of global recombination into GAs as gene pool recombination. Instead of two parents, gene pool recombination samples the genes for crossover from the gene pool, which consists of several pre-selected parents. Studies indicate that gene pool recombination and its variants are easier to analyze and can converge faster than two-parent recombination. Tsutsui and Jain (1998) proposed multi-cut crossover and seed crossover, wherein multi-cut crossover generalizes the classic two-point crossover and achieves empirically better performance compared to diagonal crossover.

For real-coded GAs, Tsutsui and Ghosh (1998) presented a series of multi-parent crossovers: center of mass crossover, multi-parent feature-wise crossover, and seed crossover. They showed that these multi-parent crossovers can lead to better performance, though this performance is problem-dependent. Another multi-parent crossover, simplex crossover (Tsutsui, Yamamura, & Higuchi, 1999), generates offspring using the simplex sampled from multiple parents. Experimental results show that this method performs well with three or four parents on multimodal and epistatic problems. Kita, Ono, and Kobayashi (1999) introduced multiple parents into unimodal normal distribution crossover to enhance the diversity of the offspring. This multi-parent extension of unimodal normal distribution crossover exhibits an improvement in search ability on highly epistatic problems. Gong and Ruan (2004) proposed the fitness-weighted crossover. It gives the fitter parents a bigger influencing factor, which is used to determine the contribution of parents to their offspring.

The above literature demonstrates the superiority of multi-parent crossover over two-parent crossover. The effectiveness of these multi-parent crossover operators is validated mostly on numerical optimization problems (Eiben, 2002; Ting, 2005; Tsutsui & Jain, 1998). As for combinatorial optimization problems, there exists only the adjacency based crossover (Eiben et al., 1994). Nevertheless, experimental results point out that using more than two parents in adjacency based crossover has no tangible benefit. Effective

* Corresponding author.
E-mail addresses: ckting@cs.ccu.edu.tw (C.-K. Ting), b8934018@student.nsysu.edu.tw (C.-H. Su), cnlee@cse.nsysu.edu.tw (C.-N. Lee).

0957-4174/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2009.07.082

multi-parent crossover for combinatorial optimization problems is still lacking.

This paper proposes the *multi-parent partially mapped crossover* (MPPMX) for combinatorial optimization problems. Specifically, MPPMX generalizes the partially mapped crossover (PMX) (Goldberg & Lingle, 1985), which is widely used for combinatorial optimization problems, e.g., Drechsler, Becker, and Göckel (1997), Skliarova and Ferrari (2002) and Tseng, Wang, and Shih (2007). The traveling salesman problem (TSP) is an important representative of combinatorial optimization problems. Many practical problems, such as scheduling (Ho & Ji, 2009; Pan & Huang, 2009), manufacturing control system (Skliarova & Ferrari, 2002), and bioinformatics (Ezziane, 2006), can be transformed into the TSP. In this paper, the evaluation of the proposed MPPMX will focus on the performance on the TSP.

The remainder of this paper is organized as follows. Section 2 gives a brief review of GAs for combinatorial optimization problems. In Section 3, we describe in detail the proposed MPPMX. Section 4 presents a performance evaluation. Finally, conclusions are drawn in Section 5.

2. Genetic algorithms for combinatorial optimization problems

The basic idea of GAs is to enhance candidate solutions by simulating the mechanisms of natural evolution, such as selection, crossover, and mutation. The operation of crossover is subject to chromosome representation, which can be binary, integer, real, or order.

Order (or permutation) is the most common representation of chromosomes with combinatorial optimization problems that GAs have to tackle. In a 9-city TSP, for example, a visiting schedule 5–9–7–4–1–2–8–3–6 can be simply represented as a chromosome in the form of order (5,9,7,4,1,2,8,3,6). GAs that use order representation for chromosomes are called order-based GAs.

Even though order representation facilitates GAs to handle combinatorial optimization problems, it also causes an intrinsic constraint in the operation of chromosomes, because no duplicate numbers are allowed in a chromosome. Therefore, the crossover for binary-coded GAs, such as one-point, two-point, and uniform crossovers, cannot be directly applied to order-based GAs. Fig. 1 illustrates the failure of two-point crossover for the TSP. The two-point crossover yields duplicate genes 1, 2, 5 in Offspring 1 and 4, 6, 7 in Offspring 2. Thus, both offspring are illegal, i.e. infeasible for being a visiting schedule.

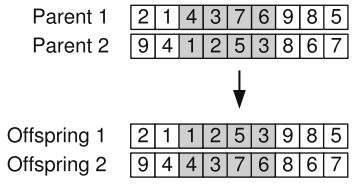

Fig. 1. Failure of two-point crossover in order-based GAs.

To address the issue of the legality of an order, several crossover operators for order-based GAs are proposed. Partially mapped crossover (PMX) (Goldberg & Lingle, 1985) is one of the most popular and effective crossovers for order-based GAs to deal with combinatorial optimization problems, especially the TSP. In view of the operation, PMX can be regarded as a modification of two-point crossover but additionally uses a mapping relationship to legalize offspring that have duplicate numbers. The algorithm of PMX is given below.

Algorithm 1. Partially mapped crossover (PMX)

1. **Substring selection:** Cut each parent into two substrings, and then select one substring for each parent at random.
2. **Substring exchange:** Exchange the two selected substrings to produce proto-offspring.
3. **Mapping list determination:** Determine the mapping relationship based on the selected substrings.
4. **Offspring legalization:** Legalize proto-offspring with the mapping relationship.

Fig. 2 illustrates how PMX legalizes the offspring in Fig. 1. Assume that the selected substrings in Step 1 are [4 3 7 6] for Parent 1 and [1 2 5 3] for Parent 2. These two substrings are then exchanged to produce proto-offspring in Step 2. Note that the proto-offspring are possibly illegal. Steps 3 and 4 in PMX then fix the illegal offspring. In Step 3, the mapping relationship is established according to the selected substrings, e.g., '1' to '4', '2' to '3' until '6', and '5' to '7' in Fig. 2. To legalize the proto-offspring, the fourth step of PMX replaces the duplicates genes with the corresponding genes in the mapping relationship.

Mutation is another important operator in GAs. This operator changes a small amount of genes to activate the population diversity. Several mutation operators have been proposed for order-based GAs. This study adopts the well-known swap mutation (Syswerda, 1991), which swaps genes at two randomly chosen loci.

3. Multi-parent partially mapped crossover

In light of the considerable success of PMX in combinatorial optimization problems, this paper proposes the multi-parent partially mapped crossover (MPPMX) to extend PMX into a multi-parent crossover for better performance. Satisfying the legality of offspring is key to the design of the crossover for order-based representation. The increase of parents, however, complicates the determination of the mapping relationship and the legalization

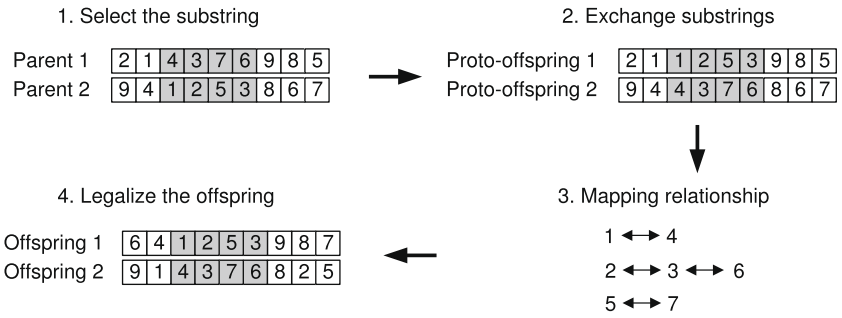

Fig. 2. Example of PMX.

process in PMX. For this, we have devised an approach for MPPMX, wherein it constructs a unique mapping relationship in order to legalize the proto-offspring reproduced from more than two parents.

The proposed MPPMX follows the four-step procedure of PMX, as shown in Algorithm 1. Nonetheless, MPPMX constructs a unique circular list as the basis of mapping the relationship for all parents, which is different from the mapping list of PMX. Detailed descriptions of each step are given in the following subsections.

3.1. Substrings selection

For an n-parent MPPMX, the substrings selection imitates n-point crossover. It randomly selects n crossover points to cut each parent into $n+1$ substrings. Fig. 3 illustrates the case of three parents. The substrings selection cuts each parent into four substrings at three random crossover points.

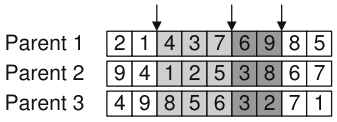

Fig. 3. Substrings selection.

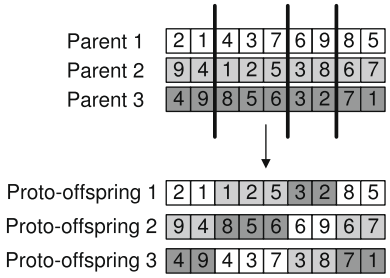

Fig. 4. Substrings exchange.

3.2. Substrings exchange

This step exchanges the selected substrings to create proto-offspring. The method for exchanging substrings is analogous with that of multi-cut crossover (Tsutsui & Jain, 1998). As shown in Fig. 4, MPPMX picks and recombines the substrings from each parent in a diagonal way to form an offspring. Note that both PMX and MPPMX have one-child and multiple-children versions, where the one-child version can be simply obtained by randomly choosing one from n children.

3.3. Mapping list determination

Determining the mapping list is the key point in MPPMX. To construct a mapping list for more than two parents, MPPMX must first randomly choose a parent as the *start parent*, and a locus as the *start locus*. Second, a parent different from the start parent is then chosen at random as the *end parent*. The *end locus* is the locus in which the gene in the end parent is the same as the gene at the start locus. Next, a parent different from the end parent is picked at random to serve as the start parent for the next round. The gene of the new start parent at the locus corresponding to the end locus is determined as the element of the mapping list. This process repeats until the mapping list is completed. The pseudocode of the mapping list determination is presented in Fig. 5.

Fig. 6 gives an example of mapping list determination in MPPMX. First, Parent 2 is randomly chosen as the start parent and the second locus as the start locus. The gene '4' at the start locus is therefore the first element of the mapping list. Afterwards, Parent 1 is randomly chosen as the end parent, and its third locus is the end locus because the gene at this locus corresponds to '4'.

```
procedure MappingListDetermination ( )
begin
1  startparent := Randomly chosen from parents;
2  ptr := Randomly chosen from {1, . . . ,chromosome length};
3  temp_value := parents[startparent, ptr];
4  for j := 1 to chromosome length do
5  begin
6       mapping_list[j] := temp_value;
7       boolean[temp_value] := true;
8       generate an array[1..n] for 1 to n with random order;
9       if array[1] = startparent then
10            array[1] is exchanged with a random sequent element;
11      endparent := array[1];
12      ptr := pointer of endparent that points to temp_value;
13      k : = 1;
14      repeat
15            k := k + 1;
16            startparent := array[k];
17      until boolean[parents[startparent, ptr]] = false or k = parent number;
18      if boolean[parents[startparent, ptr]] = true and k = parent number then
19            repeat
20                  startparent := Randomly chosen from parents;
21                  ptr := Randomly chosen from {1, . . . ,chromosome length};
22            until boolean[parents[startparent, ptr]] = false;
23      temp_value := parents[startparent, ptr];
24 end;
end;
```

Fig. 5. Pseudocode of mapping list determination.

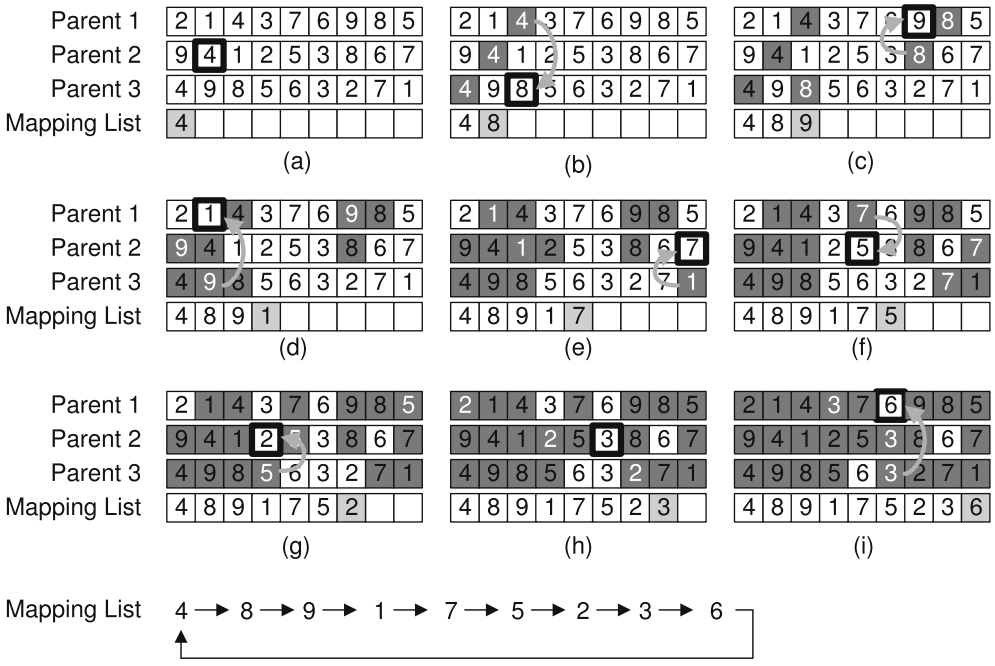

Fig. 6. Example of mapping list determination in MPPMX.

Next, we randomly choose Parent 3 as the second start parent. By mapping from Parent 1 to 3 at the third locus, gene '8' is determined to be the second element of the mapping list, as shown in Fig. 6b. Through this process, we can obtain a unique mapping list for the three parents.

In MPPMX, a deadlock may occur when there is no parent qualified to be a start parent. When this happens, MPPMX breaks the deadlock by randomly choosing a gene from all that are available as the next start parent and start locus. Fig. 6h shows that MPPMX encounters a deadlock when there is no suitable start parent for the element '2'. MPPMX would then randomly choose gene '3' at Parent 2 as the next start parent and start locus. Subject to the way of breaking the deadlock, two-parent MPPMX is not exactly equivalent to PMX. More precisely, MPPMX is not an exact, but an approximate, generalization of PMX in terms of the number of parents.

3.4. Offspring legalization

In this paper, MPPMX starts legalization from the second substring even though the starting substring can be any of those available. As a result, it is not necessary for the second substring of the proto-offspring to be legalized since there are no duplicate genes in it at all. The remainder of the proto-offspring, on the other hand, needs to be legalized through the mapping list and order array. The *order array* is an integer string used to indicate the sequence of loci in a certain substring that needs to be checked for legality. Here, the array is generated randomly in order to diversify the legalization results. The process of legalization examines the offspring genes in the sequence indicated in the order array. If a gene appears only once in the proto-offspring, it is then *legal* and so will bypass the legalization process. Otherwise, the gene is *illegal* and needs to be legalized. MPPMX will legalize it with an unused subsequent gene from the mapping list.

Fig. 7 gives an example of offspring legalization. According to the order array, MPPMX first checks the first element of the third substring for legality, namely gene '3'. Since gene '3' appears only once in the offspring, it is determined to be legal and thus bypasses legalization. The next gene, '2', is illegal in that it also appears in the second substring. To legalize gene '2', MPPMX searches the genes subsequent to '2' in the mapping list for an unused one. Gene '6' would therefore be picked as the gene in place of '2'. By applying the same procedures on the first substring, as shown in Figs. 7d–f, we can legalize the complete proto-offspring.

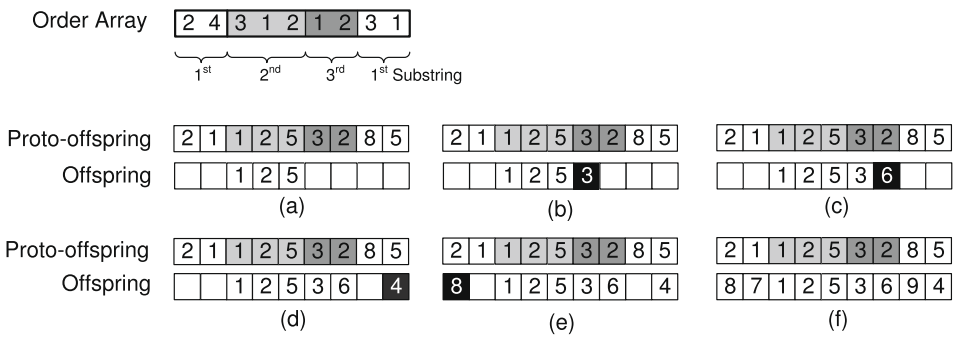

Fig. 7. Offspring legalization in MPPMX.

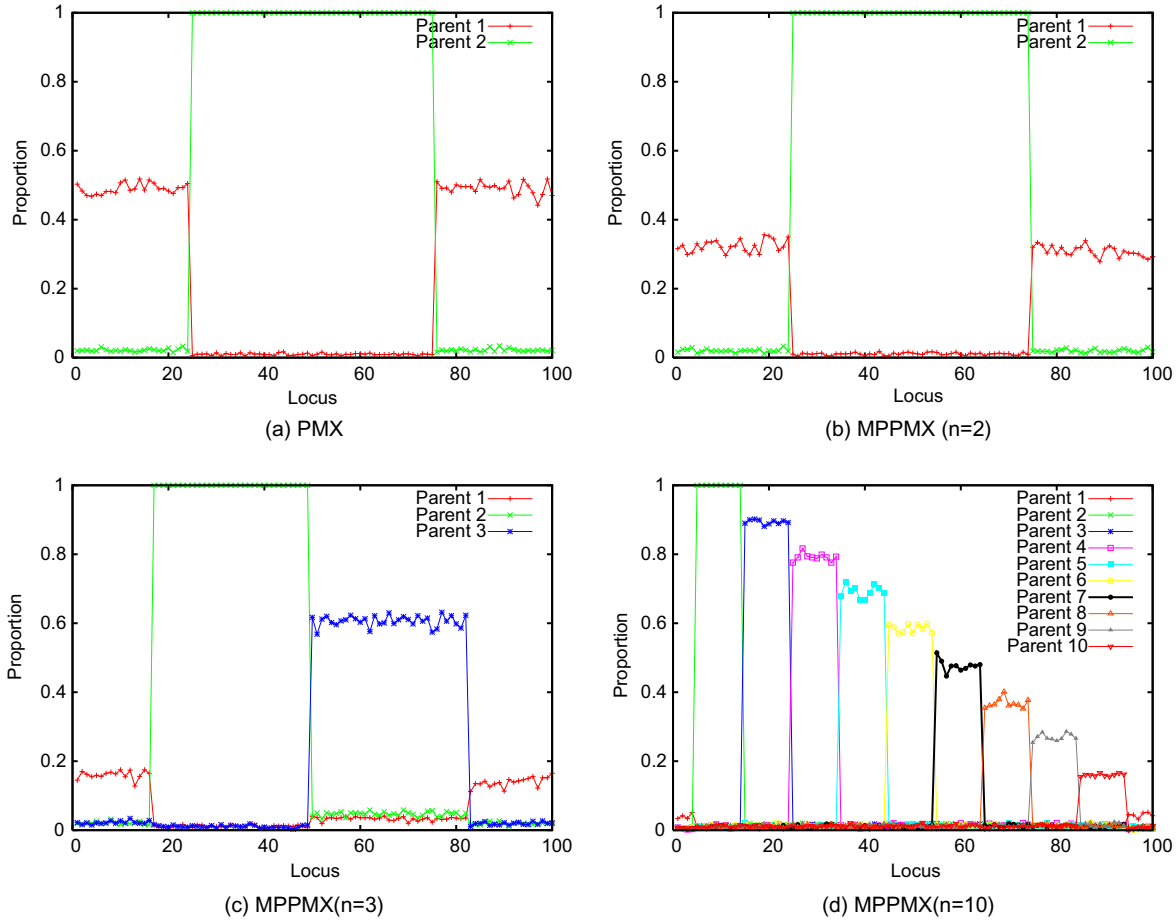

Fig. 8. Inheritance proportions from each parent for one offspring reproduced by PMX and MPPMX using 2, 3, and 10 parents.

3.5. Complexity of MPPMX

In this paper, we conduct a worst-case complexity analysis of MPPMX. With respect to complexity, the key steps of MPPMX are mapping list determination and offspring legalization. In the former, the worst case of creating the list needs $\mathcal{O}(nl)$ steps for n parents with chromosome length l. In the latter, the worst case is when n substrings have equal length l/n, wherein all genes need to be legalized except those in the second substring. Thus, the number of times that MPPMX needs to check the mapping list, in the worst case, is

$$\frac{l}{n}\cdot\frac{l}{n}+\frac{l}{n}\cdot\frac{2l}{n}+\cdots+\frac{l}{n}\cdot\frac{(n-1)l}{n}=\frac{n(n-1)l^2}{2n^2},$$

which accounts for the complexity of $\mathcal{O}(l^2)$. To summarize the complexity in mapping list determination and offspring legalization for $n < l$, we have the worst-case complexity $\mathcal{O}(l^2)$ in MPPMX.

4. Performance evaluation

This work conducted a series of experiments to analyze the inheritance and evaluate the performance of MPPMX. First, we empirically analyzed the inheritance proportion of genes in PMX and MPPMX in Section 4.1. Second, the performance of MPPMX is evaluated by experiments on the traveling salesman problem (TSP). The experimental results and discussion are presented in Section 4.2.

4.1. Empirical analysis

The *inheritance proportion* is defined as the percentage of genes that the offspring inherit from their parents. This number serves as a measure of a crossover's level of exploitation. This study analyzes the inheritance proportion of chromosomes with length $l = 100$ and fixed crossover points for equal-length substrings.

Fig. 8 shows that, in PMX, all of the genes of the second substring inherit from Parent 2. Only half of the genes of the first substring inherit from Parent 1, while the other half is generated through legalization process. This situation also occurs in two-parent MPPMX, but in the inheritance proportion, 30% of the first substrings are lower than 50% of PMX. As explained in Section 3.4, this difference is caused by the deadlock in MPPMX, which decreases the inheritance proportion. In addition, the inheritance proportion decreases with the order of substrings since the succeeding substrings are legalized later and thus more likely to induce deadlock.

4.2. Experimental results on the TSP

This section examines the performance of MPPMX on five TSP instances from TSPLIB (Reinelt, 1995): eil51, st70, pr76, lin105, and d198. Table 1 lists the setting[1] for the GA employed in our experiments. The number of parents for MPPMX ranges from 2 to 10. Seven mutation rates (r_m) are tested: 0, 0.005, 0.01, 0.02, 0.03, 0.04, and 0.05. Each setting includes 30 independent runs.

[1] For a more detailed description of these operators, see Bäck and Michalewicz (1997) and Eiben and Smith (2003).

Table 1
Parameter setting for GA.

Parameter	Value
Representation	Order
Population size	100
Population model	Generational
Parent selection	Roulette wheel selection
Crossover	PMX
	MPPMX ($n = 2, \ldots, 10$)
Crossover rate r_c	1.0
Mutation	Swap
Mutation rate r_m	0, 0.005, 0.01, 0.02, 0.03, 0.04, and 0.05
Survivor selection	$\mu + \lambda$
Termination	5000 generations
Number of runs	30 runs

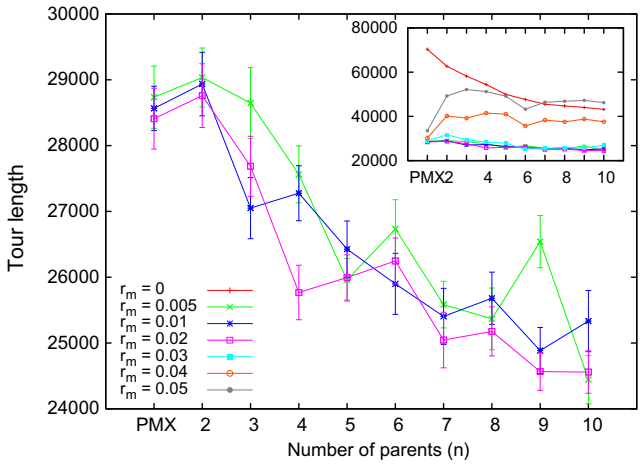

Fig. 9. Mean and standard error of the best tour length for PMX and n-parent MPPMX at different mutation rates on lin105.

Fig. 9 compares the mean best tour length obtained from PMX and MPPMX with 2–10 parents. Due to the similarity of results from different test problems, we present here the results of the lin105 problem only. Here, the results of three representative mutation rates are compared: MPPMX's best ($r_m = 0.005$), PMX's best ($r_m = 0.02$), and the common setting ($r_m = 1/l \approx 0.01$). The miniature figure at the right top of Fig. 9 applies to all test mutation rates. The figure demonstrates that MPPMX with more than two parents outperforms PMX and two-parent MPPMX. Roughly, the performance of MPPMX improves along with the increase of parents from the three representative mutation rates. This discrepancy between PMX and two-parent MPPMX reveals the influences of deadlock in the mapping list determination of MPPMX.

Table 2 lists the mean and standard deviation of the best tour lengths over 30 trials of PMX and MPPMX using different mutation rates and numbers of parents. In terms of the respective best results for all test mutation rates, MPPMX (24444.9) enhances PMX (28409.2) by 13.95% in solution quality. In addition, the experimental results show that the suitable numbers of parents are dependent on the adopted mutation rate. For mutation rates r_m =0–0.02, MPPMX using 3–10 parents can outperform PMX as well as achieve the best performance with $n = 9$ or 10. The improvements of MPPMX over PMX range from 12.88% to 38.63%, and all have statistical significance, which is validated by the one-tailed t-test with a confidence level of 0.05. However, for $r_m = 0.04$ or 0.05, MPPMX is inferior to PMX due to excessive mutation. These phenomena occur in population sizes of 100 and 500 in our experiments.

Table 3 summarizes the best tour lengths for PMX and n-parent MPPMX among all test mutation rates on the five TSP instances with population size of 100. The best numbers of parents for MPPMX on each test TSP instance are shown in boldface. The experimental results show that MPPMX can outperform PMX in solution quality by 3.68% (eil51), 7.53% (st70), 5.82% (pr76), 13.95% (lin105), and 13.32% (d198). The significance of the improvement of MPPMX over PMX is further validated by the statistical one-tailed t-test. With a confidence level 0.05, the p-values in Table 4 indicate that MPPMX with 4–10 parents can lead to significant improvement over PMX on all test TSP instances.

Fig. 10 depicts the convergence of PMX and MPPMX at $r_m = 0.01$. Owning to the similarity of results in different test problems, only the results of the lin105 problem are presented here. The figure shows that GAs using PMX converge faster than those using MPPMX at the cost of solution quality. On the other hand, the increase of parents in MPPMX slows the convergence but gives better solution quality.

Table 2
Mean and standard deviation (in parenthesis) of the tour length over 30 trials of PMX and n-parent MPPMX on lin105. The boldface denotes significant improvement of MPPMX over PMX.

	r_m						
	0	0.05	0.01	0.02	0.03	0.04	0.05
PMX	70339.0	28735.3	28565.3	28409.2	28989.6	30243.9	33499.4
	(4918.5)	(2592.6)	(1846.1)	(2527.5)	(2055.0)	(2225.7)	(2405.7)
MPPMX ($n = 2$)	**62663.0**	29034.4	**28935.1**	28759.3	31685.1	40199.3	49268.3
	(4287.9)	(2454.5)	(2641.1)	(2652.8)	(2384.5)	(2887.3)	(3090.8)
MPPMX ($n = 3$)	**58222.9**	**28645.5**	**27049.8**	**27685.8**	29429.3	39213.7	52148.4
	(2919.5)	(2956.8)	(2545.3)	(2499.0)	(2681.6)	(3170.2)	(4337.3)
MPPMX ($n = 4$)	**54437.1**	**27563.0**	**27276.8**	**25768.0**	28360.8	41508.8	51193.6
	(4561.1)	(2374.4)	(2290.8)	(2267.6)	(2453)	(5226.8)	(5011.1)
MPPMX ($n = 5$)	**49943.8**	**25958.1**	**26427.1**	**25995.8**	27913.7	40989.5	49122
	(3608.5)	(1776.2)	(2343.2)	(1875.9)	(3405.5)	(4795.5)	(4729.6)
MPPMX ($n = 6$)	**47667.9**	**26733.7**	**25899.2**	**26247.1**	**25111.6**	35654.2	43268.4
	(3034.5)	(2434.6)	(2536.7)	(1912.6)	(2636.9)	(6101.7)	(4495.6)
MPPMX ($n = 7$)	**45480.1**	**25584.5**	**25404.7**	**25045.3**	**25507.5**	38338.2	46421.7
	(3207.1)	(1941.3)	(2330.2)	(2310.7)	(2948.2)	(5309.6)	(4638.3)
MPPMX ($n = 8$)	**44761.6**	**25366.8**	**25680.7**	**25176.9**	**25810.9**	37521	46825.8
	(2647.2)	(2572)	(2168.1)	(2045.2)	(2923.0)	(5885.7)	(5033.2)
MPPMX ($n = 9$)	**44035.7**	**26540.7**	**24885.6**	**24566.2**	**25730.8**	38755.2	47253.6
	(2902.8)	(2173.3)	(1923.8)	(1561.9)	(2225.1)	(5542.9)	(4689.7)
MPPMX ($n = 10$)	**43164.6**	**24444.9**	**25333.8**	**24558.8**	**26953.8**	37574.3	46225.3
	(2655.6)	(2013.7)	(2546.0)	(1765.1)	(4135.0)	(4768.5)	(4240.5)

Table 3
Mean, standard deviation (in parenthesis), and the mutation rate (marked by an asterisk) corresponding to the best tour length over 30 trials of PMX and n-parent MPPMX. The boldface denotes the best result with respect to the TSP instance.

	eil51	st70	pr76	lin105	d198
PMX	533.52 (31.56) 0.04*	1015.90 (68.79) 0.05*	162172.2 (9898.4) 0.03*	28409.2 (2527.5) 0.02*	37275.8 (4472.8) 0.005*
MPPMX ($n = 2$)	541.71 (30.74) 0.06*	1039.51 (72.28) 0.03*	163586.6 (10820.0) 0.03*	28759.3 (2652.8) 0.02*	37838.7 (4518.4) 0.006*
MPPMX ($n = 3$)	527.18 (28.93) 0.05*	1002.48 (73.47) 0.05*	161547.9 (11214.7) 0.03*	27049.8 (2545.3) 0.01*	36049.1 (2803.7) 0.01*
MPPMX ($n = 4$)	522.28 (28.93) 0.06*	973.89 (67.54) 0.04*	157436.3 (11099.9) 0.04*	25768.0 (2267.6) 0.02*	34812.9 (3669.4) 0.01*
MPPMX ($n = 5$)	**513.86** (28.31) 0.04*	962.79 (66.38) 0.04*	154115.7 (10660.5) 0.04*	25958.1 (1776.2) 0.005*	34220.0 (3371.5) 0.01*
MPPMX ($n = 6$)	520.19 (32.03) 0.05*	959.57 (63.46) 0.05*	156310.9 (11335.8) 0.04*	25111.6 (2636.9) 0.03*	34163.2 (3004.0) 0.01*
MPPMX ($n = 7$)	517.77 (26.58) 0.06*	956.28 (81.60) 0.05*	155252.7 (10066.9) 0.04*	25045.3 (2310.7) 0.02*	33398.7 (3253.1) 0.005*
MPPMX ($n = 8$)	516.41 (28.27) 0.06*	951.51 (66.56) 0.05*	155301.8 (10731.1) 0.04*	25176.9 (2045.2) 0.02*	33371.0 (3240.0) 0.01*
MPPMX ($n = 9$)	519.09 (35.58) 0.05*	**939.41** (63.71) 0.03*	**152737.9** (9697.3) 0.04*	24566.2 (1561.9) 0.02*	**32309.6** (3459.1) 0.004*
MPPMX ($n = 10$)	514.01 (30.48) 0.06*	947.07 (59.42) 0.04*	153161.7 (11058.6) 0.04*	**24444.9** (2013.7) 0.005*	33365.0 (3912.9) 0.004*

Table 4
The p-values of t-test on the best tour length between n-parent MPPMX and PMX in the TSP. The boldface denotes significant improvement of MPPMX over PMX. The negative p-values mean that MPPMX is inferior to PMX.

n	eil51	st70	pr76	lin105	d198
2	−3.2E−02	−9.5E−03	−1.7E−01	−3.0E−01	−3.1E−01
3	7.0E−02	9.2E−02	3.4E−01	**2.1E−02**	1.0E−01
4	**4.6E−03**	**1.1E−05**	**8.4E−04**	**3.8E−05**	**1.1E−02**
5	**3.2E−06**	**4.4E−08**	**4.8E−08**	**3.2E−05**	**2.1E−03**
6	**1.7E−03**	**4.2E−09**	**6.8E−05**	**3.6E−06**	**1.3E−03**
7	**9.1E−05**	**3.9E−08**	**9.9E−07**	**7.1E−07**	**1.7E−04**
8	**3.9E−05**	**9.1E−11**	**2.4E−06**	**6.1E−07**	**1.5E−05**
9	**1.4E−03**	**2.0E−14**	**5.8E−11**	**2.6E−09**	**6.1E−06**
10	**7.3E−06**	**7.3E−13**	**3.2E−09**	**5.3E−09**	**3.3E−04**

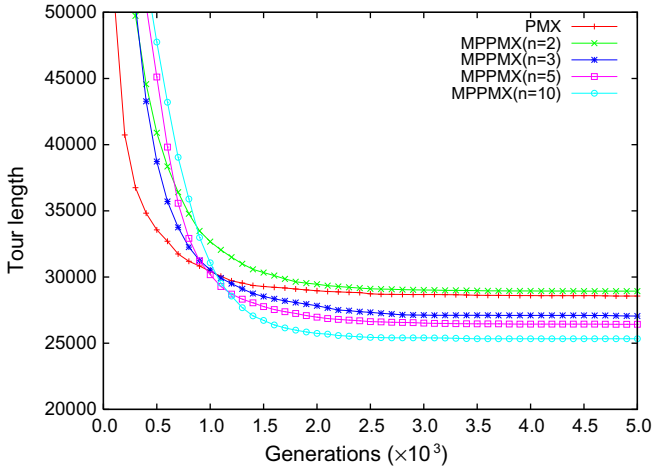

Fig. 10. Convergence of GAs using PMX and n-parent MPPMX at $r_m = 0.01$ on the lin105.

5. Conclusions

This paper presents MPPMX, a multi-parent extension of PMX. A key point for extending PMX into a multi-parent version is to determine the mapping relationship between more than two parents. For this, a novel approach to establish a global mapping list for all parents is proposed. The resultant mapping list provides a basis for offspring legalization in MPPMX. The analysis shows that MPPMX has the worst-case complexity of $\mathcal{O}(l^2)$.

A series of experiments were carried out to evaluate the performance of MPPMX. Experimental results on the TSP demonstrated that, when comparing the best performance in tour length, MPPMX can improve PMX by 3.68–13.95% on the five test TSP instances. The significance of such improvement is validated by a t-test. Additionally, the experimental results reveal that a suitable number of parents for MPPMX is closely related to mutation rate.

In conclusion, MPPMX is capable of further improving the present performance of GAs using PMX on combinatorial optimization problems. Nonetheless, experiments on more problems, such as scheduling problems, are necessary to verify the advantage and versatility of MPPMX. The reduction of time complexity in MPPMX would be beneficial to enhance it, and this could be studied in future works.

References

Bäck, T., & Michalewicz, Z. (1997). Test landscapes. In *Handbook of evolutionary computation*. Institute of Physics Publishing and Oxford University Press. pp. B2.7:14–B2.7:20.

Drechsler, R., Becker, B., & Göckel, N. (1997). A genetic algorithm for RKRO minimization. *Expert Systems with Applications, 12*(1), 127–139.

Eiben, A. (2002). Multiparent recombination in evolutionary computing. In *Advances in evolutionary computing* (pp. 175–192). Springer.

Eiben, A., Raué, P.-E., & Ruttkay, Z. (1994). Genetic algorithms with multi-parent recombination. In *Parallel problem solving from nature – PPSN III. LNCS* (Vol. 866, pp. 78–87). Springer.

Eiben, A., & Smith, J. (2003). *Introduction to evolutionary computing. Natural computing*. Springer-Verlag.

Eiben, A., & van Kemenade, C. (1997). Diagonal crossover in genetic algorithms for numerical optimization. *Journal of Control and Cybernetics, 26*(3), 447–465.

Eiben, A., van Kemenade, C., & Kok, J. (1995). Orgy in the computer: Multi-parent reproduction in genetic algorithms. In *Proceedings of the 3rd European conference on artificial life* (Vol. 929, pp. 934–945).

Ezziane, Z. (2006). Applications of artificial intelligence in bioinformatics: A review. *Expert Systems with Applications, 30*(1), 2–10.

Goldberg, D., & Lingle, R. (1985). Alleles, loci, and the traveling salesman problem. In *Proceedings of the 1st international conference on genetic algorithms and their applications* (pp. 154–159). Lawrence Erlbaum Associates, Publishers.

Gong, G., & Ruan, X. (2004). New multi-parent recombination genetic algorithm. In *Proceedings of the 5th world congress on intelligent control and automation* (Vol. 3, 2099–2104).

Ho, W., & Ji, P. (2009). An integrated scheduling problem of PCB components on sequential pick-and-place machines: Mathematical models and heuristic solutions. *Expert Systems with Applications, 36*(3P2), 7002–7010.

Holland, J. (1975). *Adaptation in natural and artificial systems*. University of Michigan Press.

Kita, H., Ono, I., & Kobayashi, S. (1999). Multi-parental extension of the unimodal normal distribution crossover for real-coded genetic algorithms. *Proceedings of the congress on evolutionary computation* (Vol. 2, pp. 1581–1588). IEEE Press.

Mühlenbein, H., Schomisch, M., & Born, J. (1991). The parallel genetic algorithm as function optimizer. In *Proceedings of the 4th international conference on genetic algorithms* (pp. 271–278). Morgan Kaufmann.

Pan, J., & Huang, H. (2009). A hybrid genetic algorithm for no-wait job shop scheduling problems. *Expert Systems with Applications, 36*(3P2), 5800–5806.

Reinelt, G. (1995). TSPLIB. <http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/>.

Skliarova, I., & Ferrari, A. (2002). FPGA-based implementation of genetic algorithm for the traveling salesman problem and its industrial application. *Lecture notes in computer science* (Vol. 2358, pp. 77–87). Springer.

Syswerda, G. (1991). Schedule optimization using genetic algorithms. In L. Davis (Ed.), *Handbook of genetic algorithms* (pp. 332–349). Van Nostrand Reinhold.

Ting, C.-K. (2005). *Design and analysis of multi-parent genetic algorithms*. Ph.D. thesis, University of Paderborn, Germany.

Tseng, H., Wang, W., & Shih, H. (2007). Using memetic algorithms with guided local search to solve assembly sequence planning. *Expert Systems with Applications, 33*(2), 451–467.

Tsutsui, S., & Ghosh, A. (1998). A study on the effect of multi-parent recombination in real coded genetic algorithms. In *Proceedings of international conference on evolutionary computation* (pp. 828–833).

Tsutsui, S., & Jain, L. (1998). On the effect on multi-parent recombination in real coded genetic algorithms. In *Proceedings of the 2nd international conference on knowledge-based intelligent electronic systems* (pp. 155–160).

Tsutsui, S., Yamamura, M., & Higuchi, T. (1999). Multi-parent recombination with simplex crossover in real coded genetic algorithms. *Proceedings of the genetic and evolutionary computation conference* (Vol. 1, pp. 657–664). Morgan Kaufmann.

Voigt, H.-M., & Mühlenbein, H. (1995). Gene pool recombination and the utilization of covariances for the breeder genetic algorithm. In *Proceedings of the 2nd IEEE international conference on evolutionary computation* (pp. 172–177). IEEE Press.