.

New Crossover Methods for Sequencing Problems[1]

Tolga Aşveren and Paul Molitor

Abstract

In this paper we present two new crossover operators that make use of macro-order information and neighborhood information in sequencing problems. None of them needs local information, thus making them usable for a wide area of applications, e.g., optimal variable orders for binary decision diagrams, scheduling problems, seriation in archeology. The experimental results are promising. Especially they show that macro-order and neighborhood information is very important.

1 Introduction

Genetic Algorithms (GAs) are one of the stochastic search algorithms based on evolutionary and biological processes that enable organisms to adapt more to their environment over many generations. They are being successfully applied to problems in business, engineering and science [1]. They work on a set of possible solutions which is called the population. The most basic operations used by GAs are selection, crossover, and mutation. The selection operator identifies

[1]The paper will appear in *Proceedings of the 4th International Conference on Parallel Problem Solving from Nature*, held in Berlin, Germany, September 1996. The paper will also be presented at the *5th Turkish Symposium on Artificial Intelligence and Neural Networksl*, held in Istanbul, Turkey, June 1996.

the individuals of the current population which will serve as parents for the next generation. Crossover randomly chooses pairs of individuals to combine properties of them by creating childs. Mutation is usually considered as a secondary operator, which makes small changes on single individuals to restore diversity of the population.

Hybrid Genetic Algorithms (HGA) uses local information during the process, i.e., information which is specific to the problem investigated in order to improve the solutions found by pure GAs. Of course, HGAs are not universal. Whenever the nature of the local information changes, a new algorithm has to be designed.

A very prominent and interesting class of combinatorial optimization problems is the class of sequencing problems. A sequencing problem is defined as follows: Given a finite set I of items and a cost function c which assigns costs to permutations of I, compute a minimum cost permutation of I. Problems as, e.g., the traveling salesman problem (TSP), the problem of computing good variable orders of Binary Decision Diagrams (BDD) which has to be solved for circuit verification and logic synthesis, scheduling problems, or seriation in archeology, are representatives of this class.

In this paper, we discuss two new crossover operators for sequencing problems, namely the Neighborhood Relationship Crossover Operator (NRX) and the Meta-Ordering Crossover Operator (MOX). Both do not use local information, i.e., they are applicable to any sequencing problem.

The experimental results obtained for TSP (as representative of the class of sequencing problems) are promising. The new crossover operators outperform the operators known from literature by far in most cases. We have worked with parallel implementations of GAs using PVM 3.1 [11] in order to overcome long convergence times which are typical for GAs.

The paper is structured as follows. In Section 2 we outline the TSP problem and motivate 'blind' search methods, once again. In Section 3 the overall algorithm for sequencing problems is presented. In Section 4 the new crossover operators are introduced. We present experimental results in Section 5.

2 Traveling Salesman Problem

The most prominent member of the rich set of combinatorial optimization problems is undoubtly the traveling salesman problem (TSP). TSP is defined as follows: Given n cities, compute a minimal cost tour on which the salesman visit each city once. Obviously, TSP is a sequencing problem as defined above.
Many researchers have used TSP as test problem to measure the effectiveness of their GAs [2-4].
The most famous technique used to improve the solutions found by GAs for TSP is an algorithm named 2-Opt. A 2-Opt move consists of eliminating two edges and reconnecting the two resulting paths in a different way to obtain a new tour. If the fitness of the modified tour is better than that of the original one, the change is kept, otherwise it is discarded. 2-Opt examines in this manner each of the $\frac{n(n-1)}{2}$ pairs of edges. If the edge lengths are known, this local information can be used by GA to compute the fitness of the new generated tour in constant time in order to determine whether the new fitness is better than the fitness of the original tour. However, this method will not be applicable to some other sequencing problems, as edge lengths (or something of this kind) are not available in any case (e.g., in optimal variable ordering in BDDs).
As we aim at efficient crossover operators applicable to any sequencing problem, we do not use local information in the following. We will only use the fact that the problem handled is a sequencing problem.

3 Algorithm

In the past few years, parallel genetic algorithms(PGAs) have been widely used. The basic motivation behind this was to reduce the processing time needed to reach an acceptable solution by hard problems. This was accomplished implementing PGAs on different parallel architectures [8-10]. In addition, it was noted that in some cases the PGAs found better solutions than comparably sized serial GAs.
We have implemented our algorithms by a network of workstations interconnected via Ethernet. We have used PVM 3.1, which is a software package that allows a heterogeneous network of parallel and serial computers to appear as a single

concurrent computational resource [11].

We have used a coarse grained parallel method. In coarse grained PGA the population is divided into isolated subpopulations that exchange individuals by migration. Depending on the exchange model, the approach is called island model or stepping stone model. In the island model, migration can occur between any of the subpopulations. In the stepping stone model migration is restricted to neighboring subpopulations. The convergence time in the island model is less than in the stepping stone model. But in the stepping stone model the population diversity is higher. Thus the probability to hang up on a local optima is less than in the island model. We have used stepping stone model because we think that the effectiveness of crossover operators can be analyzed better in a model with longer convergence time.

Mutation plays an important role in our algorithm. In conventional GAs mutation is either not used or used with a small probability for every individual in the population. But with the time the idea has changed. Mühlenbein [12] has shown that mutation is a very important component of a GA and that it is a common mistake to believe that mutation is not important because the mutation rate is so small. Eshelman [12] has used a mutation operator to avoid decreasing diversity in the population. Whenever the population converges, the population is partially randomized for a restart by using the best individual so far as a template and creating new individuals by mutating this template.

In our algorithm, when two subpopulations exchange their best individuals, the entire subpopulation is discarded and the new individuals are created by mutating the exchanged individuals and the best individual of this subpopulation. This method has the advantage that more new areas of the fitness landscape can be searched. Deletion of the old individuals of the subpopulation is reasonable because the "good" information of these individuals is already collected by the best individual of this subpopulation. We have used insertion, replacement, and 2-Opt as mutation operators. It should be noted that our 2-Opt is not a local improvement technique because all new individuals created by mutation are accepted for the new subpopulation, no matter what their fitness values are (i.e., the fitness has not to be computed during our 2-Opt).

As selection scheme, we have used a steady state, elitist selection. In each iteration a random pair is selected as parents and their offspring is created. Here each

individual has equal chance to be selected. An offspring is accepted if it is not worse than the worst individual. In this case the offspring replaces the worst individual in the population.
Our method can be viewed as a parallel mixed Evolutionary / Genetic Algorithm method. In the subpopulations GA is used. The method used by the 'meta-population' can be stated as a evolutionary-like algorithm: only the best individuals are kept alive for the emigrant exchange and, after exchanging emigrants, the emigrants are mutated which is an operator like the neighborhood operator in Evolutionary Algorithms.

4 Crossover Operators

We will use terminology of TSP in the following. However, note that the ideas can be applied to any other sequencing problem, too.
The use of GAs on TSP presents specific difficulties: ommission or duplication of a city cause an illegal solution. Thus the traditional crossover and mutation operators cannot be used. For this reason many special crossover operators are implemented for TSP such as order crossover (OX) [3], cycle crossover (CX) [3], partially mapped crossover (PMX) [14], edge recombination (ER) [5].
We have designed two crossover operators for sequencing problems that make use of macro-order and neighborhood information. The experimental results at the end of the paper will show that they outperform the crossover operators mentioned above by far in most cases.

4.1 NRX Operator

Our first operator is called Neighborhood Relationship Crossover Operator (NRX) which is defined as follows: One city is chosen arbitrary (e.g., city A) and selected as a reference point. Then for each city x the number of cities through which the salesman has to run in order to travel from reference city A to city x is computed. The two distances of each city x which correspond to the two parents selected are added. In this operation the distances are multiplied with the fitness values of the parents. The child tour is computed according to the following algorithm.

`for` $i = 0$ `to` *city number*

```
for j = i + 1 to city number
  if distance sum of city[j] < distance sum of city[i]
  then exchange city[i] and city[j];
```

Example

Let the tours A D G H C B F E and C G H A B D F E be the parents $parent_1$ and $parent_2$ selected. Assume the fitness of $parent_1$ to be 5 and that of $parent_2$ to be 2. Let city A be the reference city. The corresponding distance values of the cities are given in Table 1.

city	$parent_1$	$parent_2$	distance sum
A	0	0	0
B	5	1	27
C	4	5	30
D	1	2	9
E	7	4	43
F	6	3	36
G	2	6	22
H	3	7	29

Table 1 Distance sums of the cities

The child, i.e., the new tour, is created by the process shown in Table 2.

step	$city[i]$	$city[j]$	$sum(city[i])$	$sum(city[i])$	new tour
1	B	C	27	30	ABCDEFGH
2	B	D	27	9	ADCBEFGH
3	B	E	27	43	ADCBEFGH
4	B	F	27	36	ADCBEFGH
5	B	G	27	22	AGCBEFDH
6	B	H	27	29	AGCBEFDH
7	C	D	30	9	AGBCEFDH
8	C	E	30	43	AGBCEFDH
9	C	F	30	36	AGBCEFDH
10	C	G	30	22	AGDCEFBH
11	C	H	30	29	AGHCEFBD
12	D	E	9	43	AGHCEFBD
13	D	F	9	36	AGHCEFBD
14	D	G	9	22	AGHCEFBD
15	D	H	9	29	AGHCEFBD
16	E	F	43	36	AGHCFEBD
17	E	G	43	22	AGHCBEFD
18	E	H	43	29	AGHCDEFB
19	F	G	36	22	AGHCDFEB
20	F	H	36	29	AGHCDBEF
21	G	H	22	29	AGHCDBEF

Table 2 The NRX approach

Note that only the cities in the tour are exchanged, not their distance sum values. This allows the operator to make many replacements of the cities, thus it is very disruptive. However, replacements are only made for those cities whose distance sums are below a threshold value. This is the reason that the NRX operator works well if the cities that are to be replaced have high neighborhood relationship between each other. Neighborhood relationship of two cities is defined by the number of their common neighbors. A more accurate definition is given in the experimental part of this paper where the effect of neighborhood is analyzed in more detail. The crucial idea is that during the exchange phase the new generated solutions remain good if the neighborhood relationship is high, because the cities exchanged are in the neighborhood of their new adjacent cities in the tour with high probability.

4.2 MOX Operator

Our second crossover operator is named Meta-Ordering Crossover Operator (MOX) as it uses meta-order information. It can be defined by the following algorithm:

$count_1 = 0$
$count_2 = 0$

BEGIN
Select the first b cities from $parent_1$ beginning by $count_1$;
Select only cities, which aren't already in the child tour;
Order these cities according to their order in $parent_2$;
if the child tour is completed then exit;
$count_1$ = last selected city by $parent_1$;

Select the first b cities from $parent_2$ beginning by $count_2$;
Select only cities, which aren't already in the child tour
Order these cities according to their order in $parent_1$
if the child tour is completed exit;
$count_2$ = last selected city by parent2

goto BEGIN

If a parent encounters a city, which is already in the child tour, then he skips this city and goes to the next city. For the second child, the algorithm begins by $parent_2$ to select the first b cities.

Example

Let the tours D G I K B F H E C A J and G A D E K J I H F C B be the parents $parent_1$ and $parent_2$ selected. Let b be equal to 3. Then, the first child is created by the process shown in Table 3.

step	parent no	selected cities	new order	child tour
1	1	DGI	GDI	GDI
2	2	AEK	KEA	GDIKEA
3	1	BFH	HFB	GDIKEAHFB
4	2	CJ	CJ	GDIKEAHFBCJ

Table 3 The MOX approach

This operator has a parameter b. Obviously, the behavior of the MOX operator depends on b. However, it is very difficult to determine an optimal value of b for a given problem instance such that the MOX operator performs best. Therefore, we have coded b as a gene of each individual. b values which produce better results have greater chance to be selected as emigrants. Of course, b-genes are also mutated after changing emigrants.

5 Experimental Results

We have tested the NRX and MOX operator on the TSP problems given in Table 4 which we have taken from TSPLIB [14].

problem	city number
d198.tsp	198
kroA200.tsp	200
a280.tsp	280
lin318.tsp	318
d1291.tsp	1291

Table 4 Benchmarks for TSP

We compared our operators to PMX, which is one of the best performing operators in real time known so far [16].
We have implemented PMX in two different ways. In PMX1, we have used a look-up table for cities in order to find directly the position of a city in a tour. In PMX2, such a table is not available.
We made our experiments with 20 subpopulations each of 60 individuals and 50 subpopulation steps. A meta-generation is defined to be a process which starts by initialising 20 subpopulations at the parallel machine and stops after having obtained the best individuals from each of these sub-GAs which have run for 50 steps each.
Table 5 presents the average running times of one meta-generation for each of the crossover operators. By the experiments we have used a network of 10 workstations (SPARCclassic).

problem	PMX1	PMX2	NRX	MOX
d198.tsp	≈0.62s	≈0.68s	≈1.91s	≈0.64s
kroA200.tsp	≈0.60s	≈0.65s	≈1.86s	≈0.62s
a280.tsp	≈0.80s	≈0.91s	≈3.09s	≈0.81s
lin318.tsp	≈0.87s	≈1.00s	≈3.66s	≈0.88s
d1291.tsp	≈3.82s	≈5.08s	≈38.2s	≈3.74s

Table 5 Running times of a meta-generation

Figures 1 to 5 summarize the results for the TSP instances themselves. The horizontal axis denotes the running time in units of time needed by one meta-generation using PMX1. The vertical axis denotes the tour length. In Figure 4, e.g., the GA which uses NRX (△) computes a tour of length about 100.000 in time in which the GA which uses PMX1 computes about 200 meta-generations. We have taken average of 10 runs.

Note that MOX performs always better than PMX, i.e., computes better tours in the same time. NRX is the winner by far, except for kroA200.tsp.

In order to better understand what happens when using NRX, we have computed the neighborhood relationships in the benchmarks. (Note that this is not done by our GAs!)

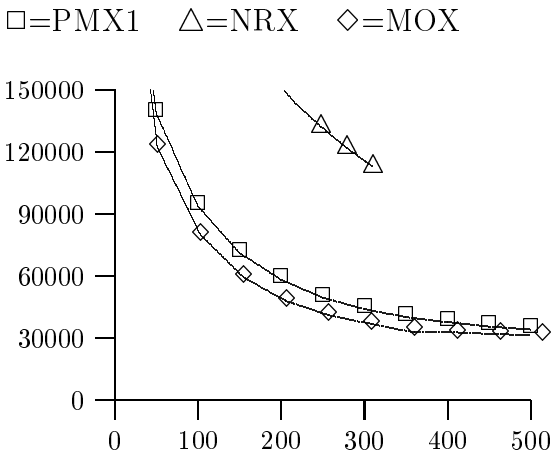

Figure 1 kroA200.tsp

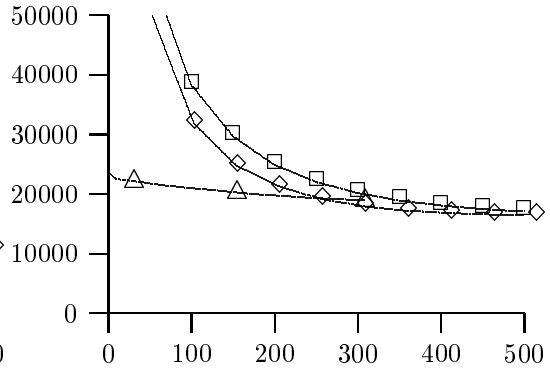

Figure 2 d198.tsp

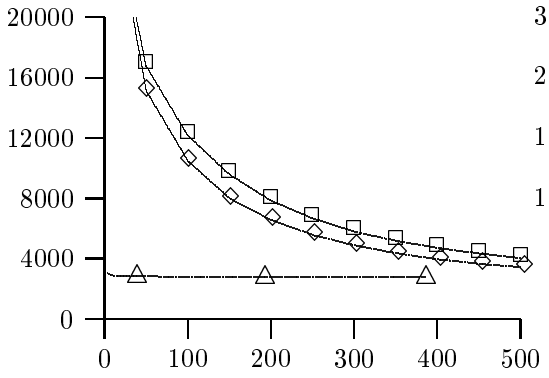

Figure 3 a280.tsp

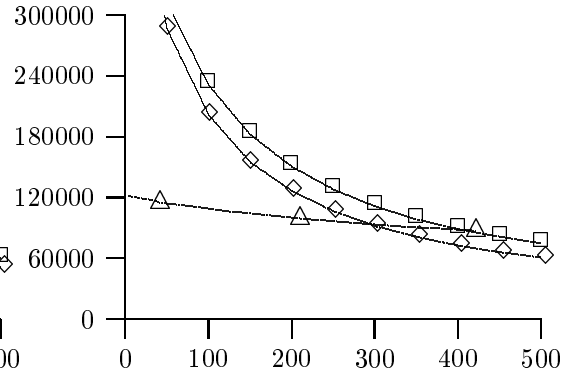

Figure 4 lin318.tsp

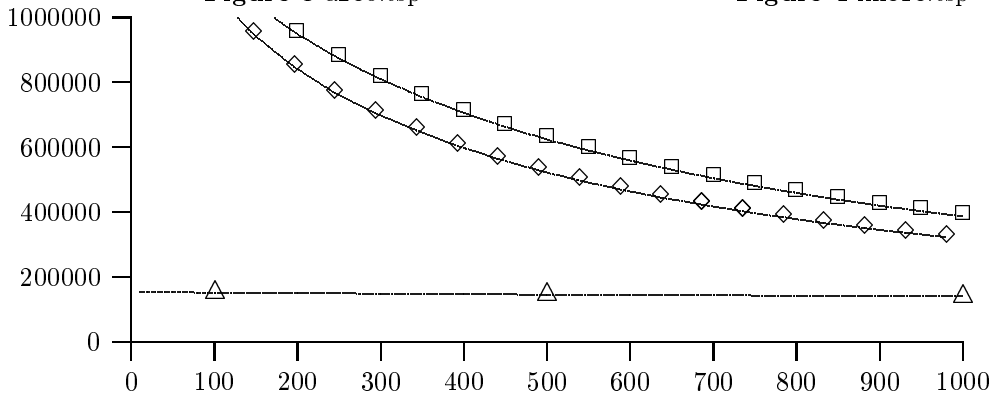

Figure 5 d1291.tsp

Let $f_n(city_1, city_2)$ be the number of cities which are among the n closest cities to city $city_1$ and among the n closest cities to city $city_2$ divided by the number of cities which are among the n closest cities to at least one of both. In the case that the set of the n closest cities is not well defined, we have chosen a permissible set at random. Now, the neighborhood relationship $g_n(city_1, ..., city_m)$ for a given problem instance is defined as

$$g_n(city_1, ..., city_m) = \frac{\sum_{i=1,j=1}^{m,m} f_n(city_i, city_j)}{\sum_{i=1,j=1}^{m,m} f_2(city_i, city_j)}.$$

By dividing $f_2(city_i, city_j)$ we are filtering out the effect of adjacent cities. Figure 6 illustrates the neighborhood relationship of the benchmark problems. The horizontal axis denotes n, the vertical one the neighborhood relationship. It seems to be that NRX is the more successful the more the neighborhood relationship is monotonously increasing.

Another observation seems to be that the effect of NRX grows with the number of cities. Comparing Figure 2 and 4 we note that they have similar shapes although

the increasing in neighborhood relationship for d198.tsp is much higher than for lin318.tsp. However, note that the number of cities in lin318.tsp is much greater than the number of cities in d198.tsp. The analogous observation can be made by a comparison of a280.tsp and d1291.tsp.

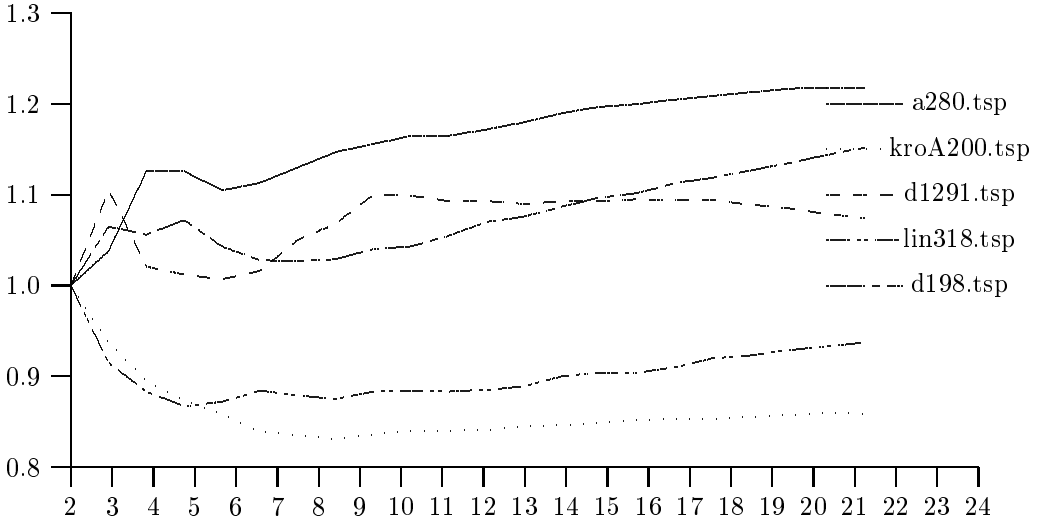

Figure 6 Neigborhood relationship

The experimental results show that NRX computes excellent solutions very fast in most cases. Then, NRX improves the populations just very slow. Thus, it seems to be a good idea to use NRX only in the first steps and then to switch to another operator.

6 Conclusion

The new crossover operators NRX and MOX make use of neighborhood relationship and meta-order information, that have been neglected by the most of the known crossover operators. When the neighborhood relationship is high enough NRX outperforms PMX and MOX by far.

Experimental results seem to imply that it may be a good idea to use NRX with another operator: use in the first steps NRX, then switch to the other operator, e.g., MOX.

MOX has outperformed PMX in all of the test problems. With growing city number, it needs less time for a meta-generation than PMX.

NRX and MOX are applicable to each sequencing problem as they do not make use of local information. Currently we apply NRX and MOX to the problem of

finding good variable orders for binary decision diagrams.

References

[1] D.E. Goldberg (1994) Genetic and Evolutionary Algorithms Come of Age, Communications of the ACM,37(3)

[2] D.E. Goldberg and J.R. Lingle (1985) Alleles,Loci and The Traveling Salesman Problem, 1. Int. Conf. on Genetic Algorithms and Their Applications, Erlbaum Associates

[3] I.M. Oliver, D.J. Smith, J.R.C. Holland (1987) A Study of Permutation Crossover Operators on The Traveling Salesman Problem, 2. Int. Conf. on Genetic Algorithms and Their Applications, MIT

[4] J. Prasanna, Y.S. Jung, D.V. Gucht (1989) The Effects of Population Size, Heuristic Crossover and Local Improvement on a Genetic Algorithm For The Traveling Salesman Problem, 3. Int. Conf. on Genetic Algorithms, Morgan Kaufmann

[5] D. Whitley, T. Starkweather, D. Fuquay (1989) Scheduling Problems and Traveling Salesman: The Genetic Edge Recombination operator, 3. Int. Conf. on Genetic Algorithms, Morgan Kaufmann

[6] K. Mathias, D. Whitley (1992) Genetic Operators, The Fitness Landscape and The Traveling Salesman Problem, PPSN 2, Elsevier Science Pub.

[7] N. Ulder, E. Aarts, H. Bandelt, P. van Laarhoven, E. Pesch (1991) Genetic Local Search Algorithms for the Traveling Salesman Problem. PPSN, Springer Verlag.

[8] M.G. Schleuter (1989) ASPARAGOS:An Asynchronous Parallel Genetic Optimization Strategy, 3. Int. Conf. on Genetic Algorithms, Morgan Kaufmann

[9] T.Fogarty (1991) Implementing the Genetic Algorithm on Transputer Based Parallel Processing Systems, PPSN, Springer Verlag

[10] T. Maruyma, A. Kanagaya, and I. Konishi (1992) An Asynchronous Fine-Grained Parallel Genetic Algorithm, PPSN 2, Elsevier Science Pub.

[11] A. Geist, A. Beguelin, J. Dongorra, W. Jiang, R. Manchek, and V. Sunderam, PVM3 Users's Guide and Reference Manual

[12] H. Mühlenbein (1991) Evolution in Time and Space-The Parallel Genetic Algorithm, Foundations of Genetic Algorithms, Morgan Kaufmann

[13] L.J. Eshelman (1991) The CHC Adaptive Search Algorithm, Foundations of Genetic Algorithms, Morgan Kaufmann

[14] D.E. Goldberg (1989) Genetic Algorithms in Search, Optimization, and Machine Learning, Addison Wesley

[15] B. Bixby and G. Reinelt (1990) TSPLIB

[16] B.R. Fox, M.B. McMahon (1991), Genetic Operators for Sequencing Problems, Foundations of Genetic Algorithms, Morgan Kaufmann