On Genetic Crossover Operators for Relative Order Preservation

Pablo Moscato

Caltech Concurrent Computation Program 158-79.
California Institute of Technology.
Pasadena, CA 91125, U.S.A.

ABSTRACT

Genetic Algorithms is a population-based optimization strategy that has been successfully applied to many *real world* and *accademic* optimization problems. The crossover operator plays a very important role in genetic algorithms since it is the element that generates the exchange of information between individuals during the search. This paper shows how the careful design of these crossover operators should be essential to avoid loss of information. This is shown graphically using the Travelling Salesman Problem as an example of one application of genetic algorithms where the relative order preservation of the structure of the configurations is extremly important when the diversity of the individuals is small.

1 Introduction

With the name Genetic Algorithms (GA), we understand a population-based optimization strategy that can also be regarded as part of evolutionary techniques to deal with optimization problems. Their remarkable property is the fact that they are robust methods, which can deal with a huge variety of problems. Although they are inspired is based on biological evolution, they have applications in a large number of different fields and in many *real world* problems such as the transient optimization of gas pipelines [1] communications network link size optimization [2] steady-state optimization of oil pump-pipeline systems [3] VLSI layout [4] [5] bin packing and graph coloring problems [6] aircraft landing strut weight optimization [7] recursive adaptive filter design [8] and also in more *academic* problems such as nonlinear equation solving for fitting potential surfaces [9] Axelrod's works in Social Sciences [10] [11], etc. This list, which is very incomplete, only pretends to show the diversity of applications of this approach. We recommend looking at the more extensive list of applications given by Goldberg in his most recent book [12].

Usually, a GA is composed of three operators, Reproduction, Crossover and Mutation. The latter is viewed as the mechanism that introduces the neccesary amount of noise to improve the results of the search while Crossover is viewed as the operator that has the mission of interchanging structural information developed between different individuals during the search. This paper will try to have a different look at the problem of having less "noisy" crossover operators for problems in which the relative order in the string is important. New results of GA in one of these problems, the Travelling Salesman Problem, also motivates this study and this discussion using it as a particular example [13].

2 The Travelling Salesman Problem

An optimization problem exists which is an important for both its academic and its real-world origin aspects, the Travelling Salesman Problem, (TSP). The TSP is generally defined as the task of finding the *cheapest* way of connecting N cities in a closed tour where a cost is associated with each link between cities. One version of it, which belongs to the NP-complete class, the Euclidean TSP in two dimensions, has been one of the most studied optimization problems. In the Euclidean version, the cost of linking two cities is proportional to the Euclidean distance between them. It is interesting since belonging to the NP-Complete class of combinatorial optimization problems, it is conjectured that it can not be solved by any polynomial-time algorithm. The TSP has been studied using GA [12] and it is a common test-bed for many optimization algorithms.

The TSP, as some other optimization problems, involves an ordering problem in which N cities must be ordered in a ring-like array. Representing it with strings, this N different elements, the city names, the task consist of finding the string that has the minimun length. To a given string, there are $2N$ equivalent strings that have the same length and are constructed by shifting all the elements or by inversing the order. So an iterative search must deal with a huge configuration space, since the number of tours is $(N-1)!/2$.

This paper will discuss some new crossover operators for problems where it is important that the mating of two individuals should not produce a new one which is much worse, in the sense of the cost function we are trying to optimize, than the parents. The TSP will be an example of a problem where we want to preserve, in the created individual, the relative order found in the parents.

3 The OX operator

To describe the different types of crossover operators, we can start with the sequence described by Goldberg (see Ref. [12]) for the OX operator. The two parents A and B (see Fig. 1 and Fig. 2) are the following

A = 9 8 4 5 6 7 1 3 2 0
B = 8 7 1 2 3 0 9 5 4 6

As Goldberg describes, we will swap a substring from one of the parents that will be called the donor. This substring is selected randomly, and in Goldberg's example, the substring 5 6 7 of donor A is the one that will be mapped into the receiver B. A *matching section* is defined and is graphically represented between two symbols " ' ".

A = 9 8 4 '5 6 7' 1 3 2 0
B = 8 7 1 '2 3 0' 9 5 4 6

The swapping of substring '5 6 7' will be performed in replacement of the '2 3 0' substring. Since the new individual should have the ten different cities, that is, after the insertion of the substring we must delete the cities of the receptor that were included in the new substring added. Instead of presenting the final result, we will show all its steps. Before swapping, we will delete the cities in the receptor.

A = 9 8 4 '5 6 7' 1 3 2 0
B = 8 * 1 '2 3 0' 9 * 4 *

We fill the *holes* with stars. Now to preserve the relative order in the receiver, we will make a *sliding motion* to leave the holes in the matching section marked in the receiver. Goldberg (Gold. page 174) makes this sliding motion starting in the second crossover site, so after the rearrangement we have

A = 9 8 4 '5 6 7' 1 3 2 0
B = 2 3 0 '* * *' 9 4 8 1

After that, the stars are replaced with the city names taken from the donor A. We get the configuration OX(B,A,2) (see Fig. 3)

A = 9 8 4 '5 6 7' 1 3 2 0
OX(B,A,2) = 2 3 0 '5 6 7' 9 4 8 1

For the purpose of this explanation, we can stop the discussion here, but in the description of the OX operator, the creation of the complementary cross must be added. This would give

OX(A,B,2) = 5 6 7 '2 3 0' 1 9 8 4
OX(B,A,2) = 2 3 0 '5 6 7' 9 4 8 1

4 The MPX operator.

We will return to our original parents A and B. The matching section is selected in the donor.

A = 9 8 4 '5 6 7' 1 3 2 0
B = 8 7 1 '2 3 0' 9 5 4 6

Before the swapping can take place, and inversion makes the beginning and the end of the donor's matching section lay together in the receiver. Inversion can either bring the 5 nearby the 7 or the 7 nearby the 5, so two different configurations can result from this inversion. For clarity reasons we will also display B

B = 8 7 1 '2 3 0' 9 5 4 6
C = 5 7 1 '2 3 0' 9 8 6 4
D = 6 4 1 '2 3 0' 9 5 7 8

Now the substring from the donor may be inserted, between the beginning and the end cities

A = 9 8 4 '5 6 7' 1 3 2 0
C = 5 '5 6 7' 7 1 '2 3 0' 9 8 6 4
D = 6 4 1 '2 3 0' 9 5 '5 6 7' 7 8

The repeated cities are replaced by stars.

C = * '5 6 7' * 1 '2 3 0' 9 8 * 4
D = * 4 1 '2 3 0' 9 * '5 6 7' * 8

And then with the sliding motion, the two possible configurations are (see Fig. 4 and Fig. 5)

C = 5 6 7 1 2 3 0 9 8 4
D = 4 1 2 3 0 9 5 6 7 8

M. Gorges-Schleuter [13] notes that individual C has no implicit mutations while D has two. By an implicit mutation we understand those links that are neither the parent A nor the parent B. In their procedures, they include them as mutations done when considering the mutation rate.

5 Some remarks about the matching section and the insertion procedure

In order to understand the potential advantages of using the MPX operator over the OX operator, we include some drawings of the tours that result from them. Let us consider two parents A and B, which are drawn in Figures 1 and 2. As it is the most common case, they have similar lenghts but one of them is shorter, in this case tour A. They have the same sequence we used before so we will refer to them using the same letters for designating the offspring.

In Figure 3 we show the tour OX(B,A,2) that results from the mating. Figures 4 and 5 show the result of applying the MPX operator that gives the tours C and D. It is interesting to remark that tour C is indeed better that its parents and also better than the OX(B,A,2). Using the OX operator very few times we had offspring better than the parents, but it is a fact that there are some that are better and this combined with the selection mechanisms of GA can explain some earlier results given by Grenfesette in the framework of *classical* GA.

As is the case for the MPX, the OX operator can also give two different offsprings if we consider that we have fixed the **second** crossover position in the receptor when doing the sliding motion. What can happen if we use the **first** crossover position is displayed in Figure 6 and the tour obtained is coded OX(B,A,1).

OX(B,A,1) = 4 8 1 5 6 7 2 3 0 9

The result is a remarkably good tour.

At this point we must have another look to see which of the improvements are given by the use of the MPX over the OX. M. Gorges-Schleuter notice one property of this operator. He said "...it keeps the relative city order as far as possible." and then he adds "The gain is, that most edges are preserved or if this is not possible at least the order of the receiver string is preserved". If we remember its name, OX is the abbreviation of "order crossover", so the interest of having an operator that preserves relative orderings is not new. But what must be noticed in the case of the MPX is the way it deals with the insertion procedure in the receiver. It is here where there is a different philosophy in the way a crossover operator must be designed for this problem. The OX operator creates a matching section in the receiver just by selecting the same positions of the substring of the donor. This does not take into account that the structure in the TSP is a ring while we are using a string representation, so the OX can destroy considerably information. It seems that the OX operator has some flavor of the swapping of genes, a case where they do not need to be in a particular order. On the contrary, we must look at this problem as the creation of two programs by merging one part of it into another. We can not do it in any place. What we are doing here is comparable to the merging of two genes, an operation that does not have the "crossover flavor".

These things can be in part corrected by using a kind of *shifted order crossover* (SOX), that is a crossover operator that takes the shift invariance into account. The first crossover point in the receiver should be chosen as the position of the city that is in the first position of the matching section of the donator, so in the case of the two parents A and B described above, the matching sections would be

A = 9 8 4 '5 6 7' 1 3 2 0
B = 8 7 1 2 3 0 9 '5 4 6'

A SOX can take into account the N tours that can be obtained from a given one by shifting each city name a certain number of positions in the string. But B is also equivalent to the tour

inv(B) = 6 4 '5 9 0' 3 2 1 7 8

we can see that in this case, the matching section in the donor is very different. There is a way to take this into account. Given any operator that swaps a substring of a donor to a receiver, we can take, for example, the OX, we can define another operator, XO, that before insertion of the substring s it checks whether the insertion of s or its inverse inv(s) gives the tour with less lenght and creates it. Analogously, we can create the XOS which must have this the two facts into account. Although for the tours A and B, all these operators do not give a better tour than the OX(B,A,1), in the general case it should perform better.

6 A 20-cities-tour example

Suppose we are given a donor A and a receiver B (see Fig. 7 and Fig. 8) given by

A = 3 7 18 '13 1 5 19 14 15 4' 20 8 12 11 6 16 17 2 9 10
B = 3 9 11 '6 16 2 17 8 12 4' 20 15 14 19 5 1 18 7 13 10

where we have marked the matching sections for the application of the OX. We have (see Fig. 9)

OX(B,A,2) = 17 8 12 13 1 5 19 14 15 4 20 18 7 10 3 9 11 6 16 2

and the OX(B,A,1) (see Fig. 10) is

OX(B,A,1) = 3 9 11 13 1 5 19 14 15 4 6 16 2 17 8 12 20 18 7 10

Now we will see what happens after application of the MPX. If we move the 13 in the inversion to bring it nearby the 4 we can have (see Fig 11)

MPX(B,A,13,4) = 3 9 11 6 16 2 17 8 12 13 1 5 19 14 15 4 7 18 20 10

but in the generation of XPM(B,A,13,4) we would have also create the tour (see Fig. 12)

XPM(B,A,13,4) = 3 9 11 6 16 2 17 8 12 4 15 14 19 5 1 13 7 18 20 10

It is very interesting to notice, regarding the figures displayed, what can happen if instead of putting the 13 nearby city 4, we make the inversion to put city 13 nearby city 20, so the link of the receiver between cities 4 and 20, is not broken. We can write it XPM(B,A,13,20) or a better notation S1XPM(B,A,13,4) (see Fig 13) that stands for *shifted one* XPM operator. This gives

S1XPM(B,A,13,4) = 3 9 11 6 16 2 17 8 12 4 20 15 14 19 5 1 13 7 18 10

The benefit of using this operator is quite obvious. Since we expect that these moves do not destroy important information that has been found before by the insertion of a new substring, we would like to preserve the receiver as far as possible. For quite developed tours, we expect that the link between cities 4 and 20 is one that has a good structure for the receiver. In addition, the connection between 20 and 13 should be good since we can suppose that the connection between 4 and 13 is also a plausible one. This can not be the case for a non-euclidean and asymetric TSP but it is not the case we are studying here.

7 Another 20-cities-tour example

Another example can clarify the subject a little more. Suppose that we are given two parents (see Fig. 14 and Fig. 15)

A = 3 9 2 '17 10 7 13 18 1 14 19 15' 20 8 12 5 16 6 4 11
B = 3 9 2 '17 10 18 7 13 1 5 14 19' 15 20 4 8 12 16 6 11

We have determined the matching section for the OX operator. This example is a little tricky since we have two parents ordered starting with the same sequence. We are also swapping a good substring from donor A into a not well developed part of receiver B. So we would expect a good result of the OX operator. In fact OX(B,A,2) (see Fig. 16) gives

OX(B,A,2) = 9 2 5 '17 10 7 13 18 1 14 19 15' 20 4 8 12 16 6 11 3

which is a very good combination. If instead of B we would have its inverse

inv(B) = 11 6 16 '12 8 4 20 15 19 14 5 1' 13 7 18 10 17 2 9 3

direct application of the OX operator would have given (see Fig. 17)

OX(inv(B),A,2) = 4 20 5 '17 10 7 13 18 1 14 19 15' 2 9 3 11 6 16 12 8

and from the figure we can see the effect of inserting a substring without taking into account the possible shifting and without doing an inversion while inserting. To see the effect of not considering the shift, we will use as the receiver a shifted version of B, that is

Bs = 18 7 13 '1 5 14 19' 15 20 4 8 12' 16 6 11 3 9 2 17 10

and the mating using OX gives (see Fig. 18)

OX(Bs,A,2) = 4 8 12 '17 10 7 13 18 1 14 19 15' 16 6 11 3 9 2 5 20

The result is a tour that clearly shows the consequence of not taking a shift into account when performing the matching section.

We can also show the result of performing the MPX operator. It gives (see Fig. 19)

MPX(B,A,15,17) = 4 20 5 17 10 7 13 18 1 14 19 15 2 9 3 11 6 16 12 8

8 ROX: Relative Order Crossover

With the lessons learned with these examples, we can introduce a different crossover with the good features displayed. Again we can work an example given two previous parents considered (see Fig. 7 and Fig. 8)

A = 3 7 18 '13 1 5 19 14 15 4' 20 8 12 11 6 16 17 2 9 10
B = 3 9 11 6 16 2 17 8 12 4 20 15 14 19 5 1 18 7 13 10

and we will consider the same matching section as before. We look at the city name of the first position in the matching section, in this case is city 13. So, we can define two different matching sections. They are called left (L) and right (R) (see Fig. 20 and Fig. 21) .

L = 3 9 11 6 16 2 17 8 12 4 20 15 '14 19 5 1 18 7 13' 10
R = 3 9 11 6 16' 2 17 8 12 4 20 15 14 19 5 1 18 7 '13 10

the holes are created

L = 3 9 11 6 16 2 17 8 12 * 20 * '* * * * 18 7 *' 10
R = 3 9 11 6 16' 2 17 8 12 * 20 * * * * * 18 7 '* 10

and now we fill without doing the sliding motion. In the case of L, we start from city 10 and inserting city 13 between city 10 and city 7, cities 7 and 18 rest in their places, and then we fill the subsequent holes with cities 1, 5, 19, 14, 15, city 20 rest in that position and city 4 goes between 20 and 12. So we have

L = 3 9 11 6 16 2 17 8 12 4 20 15 '14 19 5 1 18 7 13' 10

and similarly we have

R = 3 9 11 6 16' 2 17 8 12 1 20 5 19 14 15 4 18 7 '13 10

so then we compute the length of L and R and we accept as the offspring the one that has the best value. In this case is L, and we note it ROX(B,A,13,4).

9 Conclusions

We have presented a new set of genetic operators for problems where it is important to preserve the relative order founded by the parents as a result of the evolution of previous generations. They have been discussed using the TSP as an example but the generalization is quite obvious. It is also important to remark that we have a different point of view for the problems discussed, regarding them as the merging of programs instead to as the swapping of genes, which should be understood as the swapping of independent codes. The gain obtained encourages to use them in further research in GA and represents a lesson for designing more effective crossover operators in different problems.

10 Aknowledgements

This work would have not been possible without the support of a Rotary Foundation Scholarship. It is also a pleasure to aknowledge Dr. Scott John from the Department of Biology at Caltech, Dr. Michael Norman form the Department of Physics, Edinburgh, and Dr. Geoffrey Fox from the Caltech Concurrent Computation Program at Caltech, from which I have benefited with helpful discussions.

References

[1] D.E. Goldberg, Doctoral Dissertation, University of Michigan, *Dissertation Abstracts International*, 44(10), 3174B. University Microfilms No. 8402282, (1983).

[2] L. Davis and S. Coombs, "Genetic algorithms and communication link speed design: theoretical considerations", *Genetic algorithms and their applications: Proceedings of the Second International Conference on Genetic Algorithms*, pp. 252-256, (1987).

[3] D.E. Goldberg and C.H. Kuo, "Genetic algorithms in pipeline optimization", Paper presented at the 1985 meeting of the Pipeline Simulation Interest Group, Alburquerque NM (1985).

[4] L. Davis and D. Smith, "Adaptive design for layout synthesis", Texas Instruments internal report; Dallas, Texas Instruments, (1985).

[5] M.P Fourman, "Compactation of symbolic layout using genetic algorithms", *Proceedings of an International Conference on Genetic Algorithms and Their Applications*, pp. 141-153, (1985).

[6] L. Davis, "Appliying adaptive algorithms to epistatic domains", *Proccedings of the 9th International Joint Conference on Artificial Intelligence*, pp 162-164, (1985).

[7] A.K Minga, "Genetic Algorithms in Aerospace Design", Paper presented at the AIAA Southeastern Regional Student Conference, Huntsville AL, (1986).

[8] D.M. Etter, M.J. Hicks and K.H. Cho, "Recursive adaptive filter design using an adaptive genetic algorithm", *Proceedings of IEEE International Conference in Acoustics, Speech and Signal Processing*, 2, pp. 635-638, (1982).

[9] J.D. Schaffer, "Multiple objective optimization with vector evaluated genetic algorithms", *Proceedings of an International Conference on Genetic Algorithms and Their Applications*, pp. 93-100, (1985).

[10] R. Axelrod, "Modelling the evolution of norms", Paper presented at the American Political Science Association Meeting, New Orleans, LA, (1985).

[11] R. Axelrod, "The simulations of genetics and evolution", Paper presented at A Conference on Evolutionary Theory in Biology and Economics, University of Biefield, Federal Republic of Germany, (1985).

[12] D.E. Goldberg, *Genetic Algorithms in Search Optimisation and Machine Learning* Addison Wesley (1989).

[13] M. Gorges-Schleuter, "ASPARAGOS, Asyncronous Parallel Genetic Optimization Strategy", Paper to be presented at the Third International Conference of Genetic Algorithms, Farifax VA, (1989).