MCPC: ANOTHER APPROACH TO CROSSOVER IN GENETIC ALGORITHMS

ESQUIVEL S.[1], GALLARD R.[2], MICHALEWICZ Z.[3]

Grupo de Interés en Sistemas de Computación[4]
Departamento de Informática
Universidad Nacional de San Luis
Ejército de los Andes 950 - Local 106
5700 - San Luis
Argentina
E-mail:rgallard@inter2.unsl.edu.ar
gallardr@unslfm.edu.ar
Phone: 54++ 652 20823
Fax : 54++ 652 30224

ABSTRACT

Genetic algorithms (GAs) are stochastic adaptive algorithms whose search method is based on simulation of natural genetic inheritance and Darwinian strive for survival. They can be used to find approximate solutions to numerical optimization problems in cases where finding the exact optimum is prohibitively expensive, or where no algorithm is known.
The main operator, which is the driving force of genetic algorithms, is crossover. It combines the features of two parents and produces two offspring.
This paper proposes a Multiple Crossover Per Couple (MCPC) approach as an alternate method for crossover operators.

KEY WORDS: genetic algorithms, genetic operators, crossover.

---

[1] Full Professor, Head of the Informatics Department. In charge of the Intelligent Tools Branch.

[2] Full Professor at the Informatics Department. Head of the Research Group.

[3] Professor at the Computer Science Department of the University of North Carolina at Charlotte (USA), Professor at the Institute of Computer Science, Polish Academy of Sciences, Warsaw, Poland,Visitant Professor at Universidad Nacional de San Luis.

[4] The Research Group is supported by the Universidad Nacional de San Luis, the CONICET (Science and Technology Research Council) and the SSID (Subsecretary of Informatics and Development).

1. INTRODUCTION

The crossover operator provides a major contribution to the process of exchanging genetic material during the execution of a GA. Crossover combines the features of two parent chromosomes to form two similar offspring by swapping corresponding segments of the parents. For example, if the parents are represented by five-dimensional vectors $<a_1,b_1,c_1,d_1,e_1>$ and $<a_2,b_2,c_2,d_2,e_2>$ (with each element called a gene), then crossing the chromosomes after the second gene would produce the offspring $<a_1,b_1,c_2,d_2,e_2>$ and $<a_2,b_2,c_1,d_1,e_1>$.

The intuition behind the applicability of the crossover operator is information exchange between different potential solutions. Starting with the above basic idea of one-point crossover, many researchers studied the effect of different types of crossovers to improve GA performance (accuracy and speed). This research included various types of crossovers (two-point crossover, multi-point crossover, uniform crossover, etc.), and specialized crossovers, which depend on particular data structure used for a chromosome representing a potential solution to the problem. For example, Davis [2], Goldberg and Lingle [9] and Smith [13] were looking for diverse variants of crossover (PMX, OX and CX) when approaching the TSP problem using GAs. Frantz [7], Syswerda [15] and Davis [3] proposed multi-point, uniform and guaranteed-uniform crossover, respectively. More recently Eshelman and Schaffer [5] studied differentiating features of GAs testing diverse crossover variants.

The common approach to crossover is to operate once on each mating pair after selection. We devised a different approach: to allow multiple offspring per couple, as often happens in nature. Some results and perspectives are discussed below.

2. A GENERAL OUTLOOK TO GENETIC ALGORITHMS

GAs implement an elementary form of the natural selection mechanism to search a problem space using the Darwinian principle of natural selection and survival of the fittest. They were first devised by John Holland [11] and his co-workers at the University of Michigan in the 1970s and have been studied by other research groups since. GAs are today considered as a robust technique, effective across a spectrum of problems even in the presence of difficulties such as noise, multimodality, high-dimensionality and discontinuity (De Jong [4]). GAs have been applied to a wide variety of problems from pipeline engineering (Goldberg [8]), VLSI circuit layout (Davis [1]), resource scheduling (Syswerda and Palmucci [14]), machine learning (Goldberg [8]), and distributed systems allocation strategies (Esquivel et al,[6]).

Basically, GAs simulate the evolution (natural adaptation) of a population of solutions for a given problem. After the initial population is created, the evolution loop of a genetic algorithm consists of (1) evaluating all individuals in the population, (2) selecting a new, intermediate population (better individuals have better chances to be selected), and (3) alternating their genetic code. These three steps are repeated until some termination condition is satisfied.

A modeling assumption that is made with GAs is that each point in the problem space can be represented by a fixed length string of symbols. Thus each

string (*chromosome*, or individual) represents one possible solution to the problem and serves as a genetic material on which the genetic operations will be performed.

An interesting property that distinguish GAs from other search algorithms is that they maintain a population of potential solutions, which allow a highly parallel search, instead of working out on a single point of the searching space during each iteration (Michalewicz [12]).

Reproduction, **crossover** and **mutation** are basic operators in GAs. In *reproduction*, strings whose fitness value indicate that they are good solutions to the given problem will have a greater probability to be selected for mating and, thus, contributing offspring. Reproduced strings are copied into the next generation pool of members where they await the action of the other two operators, namely crossover and mutation.

Crossover is a mechanism by which chosen strings can be partly exchanged creating new strings (interchange of genetic material). Thus it can be seen that the **two** new strings 'inherit' some elements of their parents. Crossover when combined with reproduction provides an effective means of exchanging information and of contributing high quality solutions. This combination of operators gives GAs most of their search power.

Finally, the third basic operator, *mutation*, provides a GA with the ability to search different parts of the problem space thus increasing its aptitude to find near optimal solutions. This operator simply changes a symbol in some randomly chosen string at some random position, thus creating a new individual. This ensures that, in theory at least, every position in the string can take any of the alphabet values. The following scheme describes how a GA is carried out:

```
begin
    t := 0
    initialise population P(t)
    evaluate population P(t)
    repeat
        t := t + 1
        select population P(t) from P(t-1)
        modify population applying genetic operators
        evaluate new population P(t)
    until termination_condition
end.
```

3. THE MULTIPLE CROSSOVER PER COUPLE APPROACH

Conventional approaches to crossover, independently of the method being applied, involves to apply the operator only once on the selected pair of parents to create, precisely, two children. As an example, 1-pt crossover, 2-pt crossover, uniform crossover and other variations produce a pair of offspring per couple. From now on we call such procedure, the Single Crossover Per Couple (SCPC) approach.

Inspired by nature, we decided to conduct several experiments in which more than one crossover operation for each mating pair is allowed. This set of initial experiments were designed in such a way that for each mating pair, the Multiple Crossover Per Couple (MCPC) approach would allow a variable number of children (from zero up to some predefined maximum number).

Diverse scenarios can be conceived using the MCPC approach; fixed or variable population sizes with use of some "reasonable" criteria to allocate children to parent couples (outstanding couples could have more children than ordinary couples), or to allocate them by random, etc.

To start we chose one scenario where the first selected pairs are lucky and allowed to reproduce, while some other pairs can not reproduce due to a limits on population size.

A number of experimental runs, using elemental evolutionary computing techniques, were implemented in such a way that the number of children per couple is granted as a maximum number and the process of producing offsprings is controlled, for each mating pair, in order not to exceed the population size. We describe them now.

4. EMPIRICAL TESTS

The idea of multiple children per couple was implemented as multiple crossover operations once the pair was selected. Then, GAs allowing multiple crossover per (selected) couple were run for searching maximum[1] global values on functions F1, F2,F3, F4, F5, F6 and F7 where:

$F1: f(x_1, x_2) = 21.5 + x_1 \cdot \sin(4\pi x_1) + x_2 \cdot \sin(20\pi x_2)$, *for* ;
$-3.0 \le x_1 \le 12.1, \quad 4.1 \le x_2 \le 5.8$
estimated maximum value : 38.827553

$F2: f(x_1, x_2, x_3) = x_1^2 + x_2^2 + x_3^2$, *for* ;
$-5.12 \le x_i \le 5.12$, $i = 1,2,3$ (*De Jong Function F1*)
estimated maximum value : 78.6432

$F3: f(x_1, x_2) = 100(x_1^2 - x_2)^2 + (1 - x_1^2)$, *for* ;
$-2.048 \le x_i \le 2.048$, $i = 1,2$ (*De Jong Function F2*)
estimated maximum value : 3905.9213

$F4: f(x_2) = 2.0 + x_1 \cdot \sin(10\pi x_1)$, *for* ;
$-1 \le x_1 \le 2$
estimated maximum value : 3.850272

$F5: f(x_1, x_2) = 0.5 + \dfrac{\sin^2\sqrt{x_1^2 + x_2^2} - 0.5}{1.0 + 0.001(x_1^2 + x_2^2)^2}$, *for* ;
$-100 \le x_i \le 100$, $i = 1,2$ (*Schaffer Function F6*)
estimated maximum value : 0.972

$F6: f(x_1, x_2) = (x_1 \cdot sgn(x_1)) \cdot (x_2 \cdot sgn(x_2))$, *for* ;
$-1 \le x_i \le 2$, $i = 1,2$
estimated maximum value : 4.0

$F7: f(x_1, x_2, x_3) = 30 - \sum_{i=1}^{5} \text{int}(x_i)$, *for* ;
$-5.12 \le x_i \le 5.12$, $i = 1,2$ (*De Jong Function F3*)
estimated maximum value : 55.0

[1] Functions F1,F3,F5,F6 and F7 are usually known as minimization problems and were also tested as that.

A set of experiments consisting of 24 series of 10 runs each, were performed on each function, [run 1 allowed one crossover, run i (for $2 \leq i \leq 10$) allowed at most i crossovers per couple]. We make reference here only to 10 of them.

In experiments E1 to E9, initial population for each series remained constant during the corresponding ten runs, in order to compare results under the same initial conditions, for different number of crossovers allowed. In experiment E10, a variable initial population for each run (randomized with diverse seeds) was created to observe to what extents results are independent of initial conditions[1].

A simple GA, with binary coded chromosomes, elitism, one point crossover, bit-swap mutation and proportional selection was the basis for the initial experiments (advanced operators and complex evolutionary techniques are being implemented for future testing). The experiments were run, playing with parameters as follows and restricted to vary only population size, probability of crossover and probability of mutation :

Experiment	Population Size	Number of Generations	Probability of Crossover	Probability of Mutation
E1	100	200	0.7	0.001
E2	100	200	0.3	0.001
E3	100	200	0.4	0.003
E4	100	200	0.5	0.005
E5	70	200	0.7	0.001
E6	70	200	0.5	0.005
E7	70	400	0.5	0.001
E8	50	200	0.7	0.001
E9	50	200	0.5	0.005
E10	70	200	0.5	0.005

The values in the table above were selected for subsequent result comparisons and as part of a global study. For example E4, E6, E9 and E10 were thought to counterbalance expected population homogeneity introduced by MCPC by means of a relatively greater mutation probability, and to be compared for looking at the effect of population size. E7 with 400 generations, was run to be compared with similar previous experiments[2]. In general they were designed with changes in some parameters values to isolate the side effects of MCPC.

During preliminary runs of MCPC approach it was observed that an improvement on running time was found as long as the number of crossovers per couple increased. This effect is a consequence of an economy in computational effort, because lesser selections of mating pairs are done. On the other side of the coin this time reduction is paid by being more distant from known or estimated maximum value. In consequence, the following performance variables were analyzed:

[1] Comparison of experiments, effectively showed that influence of initial conditions is minimal. Discussion on this topic is out of the extent of this paper.
[2] These and other studies are out of the scope of this presentation.

Ebest (Eb) = (*max_val* - best value/*max_val*)100
It is the percentual error of the best found individual when compared with *max_val*[1] . It give us a measure of how far are we from that *max_val*.

Epop (Ep) = ((*max_val* - mean pop value)/*max_val*)100
It is the percentual error of the population mean when compared with *max_val*. It tell us how far the mean fitness is from that *max_val*.

Gbest (Gb) : Identifies the generation where the best value (retained by elitism) was found.

Dtime (Dt) = $((T_1 - T_i)/T_1)100$, where
T_1 = running time of run 1 (SCPC)
T_i = running time of run i (MCPC)
Running time difference, it is the percentual of time reduction when compared with classic crossover (single crossover per couple).

Ratio$_{b/c}$ (R) = Dtime/Ebest
It is a benefit/cost ratio, where the benefit is seen as a reduction on running time and the cost is seen as being farther from the *max_val*.
All the values analysed were mean values obtained from the 24 series for each maximum number of crossovers allowed, on each function.

5. RESULTS

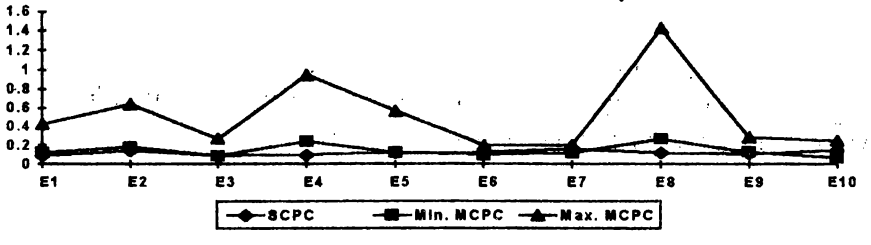
After the experimental runs were accomplished, two approaches to build statistics were carried out:

I. Minimum and maximum mean values for each of the above mentioned performance variable, on every function and experiment, were found for MCPC and thereafter, when appropriate, contrasted with the corresponding SCPC values.

II. A general overview of the MCPC effect on the performance variables was completed, also for minimum and maximum mean values.

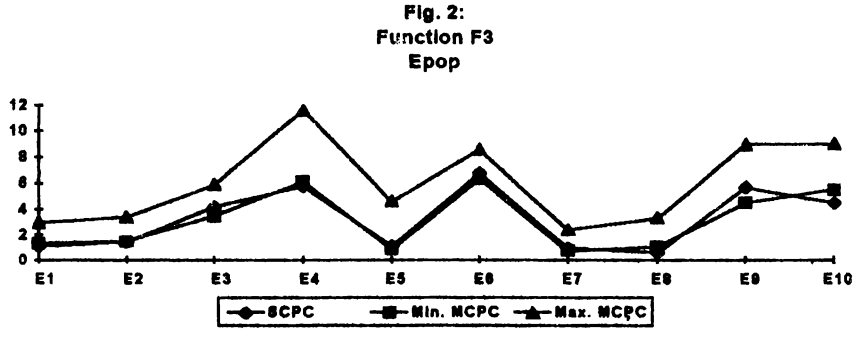
As an example for statistical approach I, figures 1 to 5 summarize results for function F3.

Fig. 1:
Function F3
Ebest

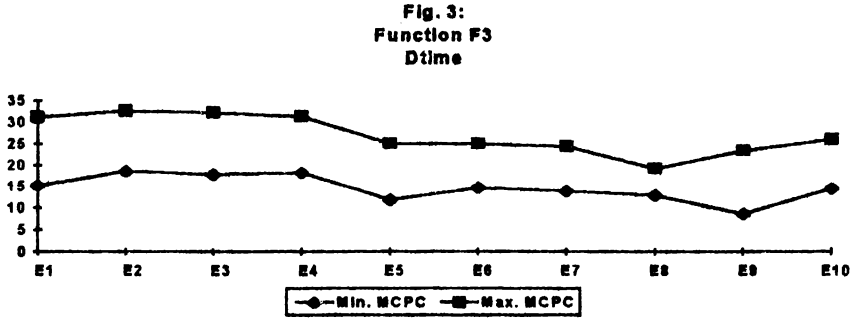


[1] *max_val* is the known, or estimated, maximum value

In Fig. 1 we can observe that minimum **Ebest** mean values for MCPC are quite coincident with **Ebest** mean value for SCPC and better for experiments E6 (0.10% vs 0.13%), E7(0.12% vs 0.16%) and E10 (0.07% vs 0.15%), while maximum **Ebest** mean values for MCPC do not exceeds 1.5% on E8.



In Fig. 2 we can observe that minimum **Epop** mean values for MCPC are quite similar to **Epop** mean value for SCPC and better for experiments E3 (3.5% vs 4.2%), E5(0.85% vs 1.1%), E6 (6.38% vs 6.80%), E7 (0.71% vs 0.95%) and E9 (4.49% vs 5.67%), while maximum **Epop** mean values for MCPC do not exceeds 12% on E4.



In Fig. 3 we can see that time reduction, **Dtime**, using MCPC goes from a minimum mean value of 8.5% for E9 to a maximum mean value of 32.7 for E2.

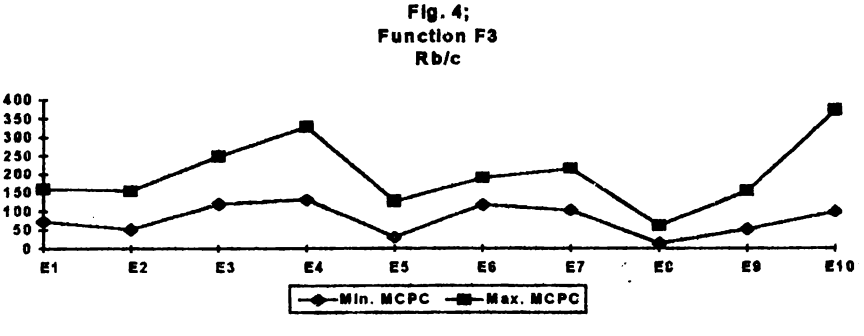
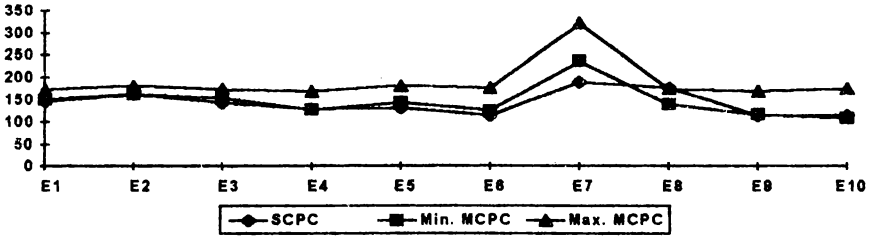


Figure 4 shows that the benefit/cost ratio, $\mathbf{Ratio_{b/c}}$, goes from 13.4 for E8 to 371.8 for E10.

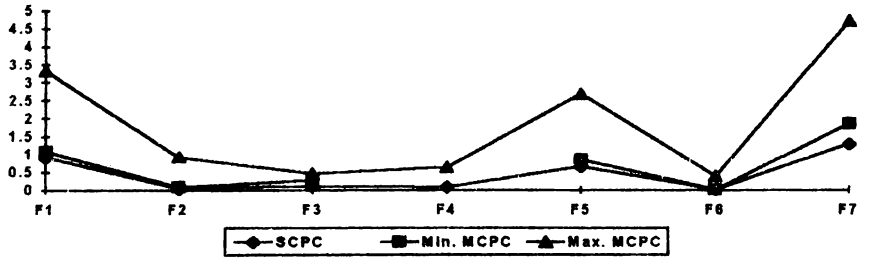
Fig. 6:
Function F3
Generations



Finally, fig. 5 shows that the minimum mean values for the number generation to find the best value, **Gbest**, under MCPC are quite similar to the mean values for SCPC, while the maximum mean values exceeded SCPC mean values in about 15% to 55%.

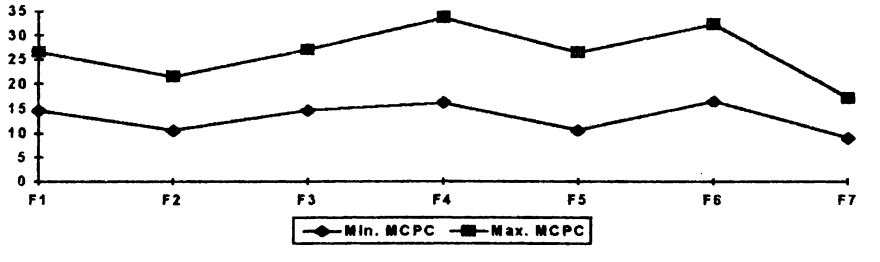
Now, as an example for statistical approach II, figures 6 and 7 summarize results for the most relevant performance variables. Here the graphs represent the effect of applying MCPC on every function, and the values are mean values obtained throughout all the experiments performed.

Fig. 6:
Ebest



In Fig. 6 we confirm that the same previous behaviour is shown for minimum mean overall **Ebest** values; they are quite similar to mean **Ebest**, SCPC values. On the other hand, maximum mean overall **Ebest** values do not surpass 5%.

Fig. 7:
Dtime



In Fig. 7 we can see that overall time reduction, **Dtime**, using MCPC goes from a minimum mean value of 9% for F7 to maximum mean value of 34% for F4.

5. CONCLUSIONS

When confronting MCPC with SMPC the following observations are remarkable:

- In some isolated cases the MCPC method finds results that are better than these found by the SCPC method.

- A dispersion factor defined as the ratio standard deviation over mean was calculated on each run; the results indicate a low dispersion on the most relevant studied variables, Ebest and Dtime.

- In some on-line applications low time complexity of an algorithm may have higher priority than the accuracy of the results; our experiments indicate that the MCPC approach might be advantageous in cases of this nature. For this reason a set of new experiments (with new strategies described briefly in section 3) is being conducted.

6. ACKNOWLEDGEMENTS

We acknowledge the cooperation of the project group for providing new ideas and constructive criticisms. Also to the UNSL, the CONICET and the Subsecretary of Informatics and Development from which we receive continuous support.

7. BIBLIOGRAPHY

[1] Davis,L., Smith,D - Adaptive Design for Layout Synthesis - (Texas Instruments internal report). Dallas:Texas Instruments. 1985.
[2] Davis, L. - Job Shop Scheduling with Genetic Algorithms - Proceedings of an International Conference on Genetic Algorithms and their Applications, 136-140, 1985.
[3] Davis L. - Adapting Operators Probabilities in Genetic Algorithms - Proceeding of the Third International Conference on Genetic Algortihms- pag. 61-69, 1989.
[4] De Jong, K. A. - Analysis of the Behavior of a Class of Genetic Adaptive Systems - Ph.D. dissertation. University of Michigan. 1975.
[5] Eshelman, L. J. and Schaffer, D. J. - Crossover Niche - proceedings of the Fifth International Conference on Genetic Algorithms, Stephanie Forrest (Editor), Morgan Kaufmann Publishers, 9-14, 1993
[6] Esquivel S., Leguizamón M., Gallard R. - A Quasi-Optimal Cluster Allocation Strategy For Parallel Program Execution In Distributed Systems Using Genetic Algorithms - ACM Press, Operating System Review, April 1995.
[7] Frantz, D. R. - Non-linearities in Genetic Adaptive Serach - Dissertation Abstracts International, 33(11), 5240B-5241B.
[8] Goldberg, D.E. - Computer-aided Gas Pipeline Operation using Genetic Algorithms and Rule Learning - Ph.D. thesis, University of Michigan. 1983.
[9] Goldberg, D.E. and Lingle, R. - Alleles, Loci, and the Traveling Salesman Problem - Proceedings of an International Conference on Genetic Algorithms and their Applications, 154-159, 1985.
[10] Goldberg, D.E. - Genetic Algorithms in Search, Optimization and Machine Learning- Addison-Wesley, Reading, MA, 1989.

[11]Holland, J.H. - Adaptation in Natural and Artificial Systems- Ann Arbor, The University of Michigan Press. 1975.
[12]Michalewicz, Z. - Genetic Algorithms + Data Structures = Evolution Programs - Springer-Verlag, 1994.
[13]Smith, D. - Bin Packing with Adaptive Search - Proceedings of an International Conference on Genetic Algorithms and their Applications, 202-206, 1985.
[14]Syswerda, G., and Palmucci, J. (1991) -The Applications of Genetic Algorithms to Resource Scheduling - Proc. of the 4th International Conference on Genetics Algorithms, 502-508. Morgan Kaufmann, San Mateo, CA, 1991.
[15]Syswerda G. - Uniform Crossover in Genetic Algorithms - Proceeding of the Third International Conference on Genetic Algortihms- 2-9, 1989.