INFLUENCE OF CROSSOVER OPERATORS IN EVOLUTIONARY SCHEDULING UNDER MULTIRECOMBINED SCHEMES

De San Pedro M., Pandolfi D., Villagra A, Lasso M.
Proyecto UNPA-29/B032[1]
División Tecnología
Unidad Académica Caleta Olivia
Universidad Nacional de La Patagonia Austral
Ruta 3 Acceso Norte s/n
(9011) Caleta Olivia – Santa Cruz - Argentina
e-mail: {edesanpedro,dpandolfi,avillagra,mlasso}@uaco.unpa.edu.ar
Phone/Fax : +54 0297 4854888

Gallard R.
Laboratorio de Investigación y Desarrollo en Inteligencia Computacional (LIDIC)[2]
Departamento de Informática
Universidad Nacional de San Luis
Ejército de los Andes 950 - Local 106
(5700) - San Luis -Argentina
e-mail: rgallard@unsl.edu.ar
Phone: +54 2652 420823
Fax : +54 2652 430224

ABSTRACT
In evolutionary algorithms based on genetics, the crossover operation creates individuals by interchanging genes. On the other side selection mechanisms aim to favour reproduction of better individuals imposing a direction on the search process: copies of better ones replace worst individuals. Consequently, part of the genetic material contained in these worst individuals disappears forever. This *loss of diversity* can lead to a premature convergence. To prevent a premature convergence to a local optimum under the same selection mechanism and multirecombined scheme then, either a larger population size or adequate crossover and mutation operators are needed.

In this work we are showing the effect on genetic diversity, quality of results and required computational effort, when applying different crossover methods to a set of very hard instances of the weighted tardiness scheduling problem in single machine environments. For these experiments we are using multirecombined approaches which allow multiple crossover operations on multiple parent each time a new individual is generated. A description of each method, experiments and preliminary results are reported.

KEYWORDS: Evolutionary Scheduling, Weighted Tardiness, Crossover Operators, genetic diversity.

[1] The Research Group is supported by the Universidad Nacional de La Patagonia Austral.
[2] The LIDIC is supported by the Universidad Nacional de San Luis and the ANPCYT (National Agency to Promote Science and Technology).

1. INTRODUCTION

To achieve higher customer satisfaction current trends in manufacturing are focussed today on production policies, which emphasizes minimum weighted tardiness. Under this approach jobs to be delivered in a production system are usually weighted according to clients requirements and job relevance. Among other heuristics [4, 5, 18, 22], evolutionary algorithms (EAs) have been successfully applied to solve scheduling problems [14, 23, 24]. Current trends in evolutionary algorithms make use of multiparent and multirecombinative approaches. The latter we called, *multiple-crossovers-on-multiple-parents* (MCMP). Instead of applying crossover once on a pair of parents this feature applies n_1 crossover operations on a set of n_2 parents. In order to improve the balance between exploration and exploitation in the search process [17] a variant called MCMP-SRI [20, 21] recombines a breeding individual (stud) by repeatedly mating individuals that randomly immigrate to a mating pool. Under this approach the random immigrants incorporate exploration and the multi-mating operation with the stud incorporates exploitation to the search process. Due to its good performance, previous works using MCMP-SRI always adopted PMX and proportional selection as the standard mechanisms. In this work we mainly study the influence of different crossover operators on genetic diversity of the individuals which intervene in the multirecombined evolutionary process: population, studs, immigrants and descendants. The weighted tardiness scheduling problem was selected to face this study because its inherent difficulty. Next sections describe the scheduling problem, the multirecombinative approach, the crossover operators selected and discuss the results obtained.

2. THE SINGLE MACHINE WEIGHTED TARDINESS SCHEDULING PROBLEM

The single-machine total weighted tardiness problem [18, 22] can be stated as follows: n jobs are to be processed without interruption on a single machine that can handle no more than one job at a time. Job j (j = 1,...,n) becomes available for processing at time zero, requires an uninterrupted positive processing time p_j on the machine, has a positive weight w_j, and a due date d_j by which it should ideally be finished. For a given processing order of the jobs, the earliest completion time C_j and the tardiness $T_j = \max\{C_j - d_j, 0\}$ of job j can readily be computed. The problem is to find a processing order of the jobs with minimum total weighted tardiness $\sum_{j=1}^{n} w_j T_j$. Even with this simple formulation, this model leads to an optimization problem that is NP-Hard [22].

3. MULTIRECOMBINATION OF STUDS AND IMMIGRANTS

The conventional approach to crossover, independently of the method being applied, involves applying the operator only once on the selected parents. Such a procedure is known as the Single Crossover Per Couple approach (SCPC). An alternative approach, Multiple Crossover per Couple (MCPC) implies the repeated application of crossover to exploit the good features of a pair of parents. Implementation and results are discussed elsewhere [11, 12]. To improve MCPC performance, by using the *multiparent approach* of Eiben [8, 9, 10], the method was extended to MCMP [13] where the multiple crossovers are applied to a set of multiple parents. Results obtained in diverse single and multiobjective optimization problems indicated that the searching space is efficiently exploited by the multiple application of crossovers and efficiently explored by the greater number of samples provided by the multiple parents.

Attempting to achieve a better balance between exploration and exploitation we devised MCMP-SRI. Here, the process for creating offspring is performed as follows (see figure 1). From the old population an individual, designated as the stud, is selected by means of proportional selection. The number of n_2 parents in the mating pool is completed with randomly created individuals (random immigrants). The stud mates every other parent, the couples undergo crossover and $2*n_2$ offspring are created. The best of these $2*n_2$ offspring is stored in a temporary children pool. The crossover operation is repeated n_1 times, for different cut points each time, until the children pool is completed. Children may or may not be exposed to mutation. Finally, the best offspring created from n_2 parents and n_1 crossover is inserted in the new population.

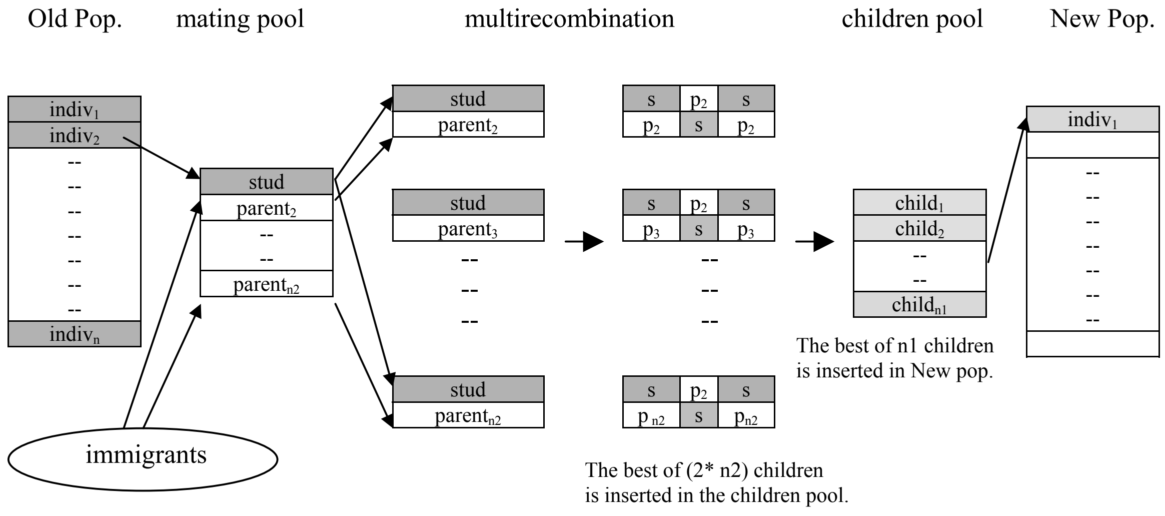

Fig. 1. The stud and random immigrants multirecombination process.

4. CROSSOVER OPERATORS FOR PERMUTATION REPRESENTATION

As a result of selection and recombination during the process of an evolutionary algorithm, copies of better ones replace worst individuals. Consequently, part of the genetic material contained in these worst individuals disappears forever. This *loss of diversity* is defined as the proportion of the population that is not selected for the next generation and can lead to a premature convergence. To prevent a premature convergence to a local optimum then, either a larger population size or adequate crossover and mutation operators are needed. In this work we concentrate in alternative crossovers which are adequate to permutation representation. The influence of the following methods is studied.

PMX (*Partial Mapped Crossover*): [15] it can be seen as an extension of the popular TPX (two point crossovers) for binary or integer representation. Besides, it uses a repair procedure to solve the appearance of illegal individuals that TPX may cause, as follows.

- Selects two random cutting points in the chromosome, which enclose sub-chains (the mapping sections between both parents).
- Interchange both sub-strings between parents to produce a proto-child.
- Determines the mapping relation between both mapping sections.
- By means of the mapping relation, creates legal offspring

OX1 *(Order Crossover 1):* [6] proposed by Davis in 1995, it is a variant of PMX, which uses a different repair mechanism:

- Selects two random cutting points, enclosing sub-chains in both parents.
- The sub-chain of the first parent is copied in the child.

- The remaining alleles in the child are copied from the first parent according to the order they appear in the second parent and beginning from the second cutting point.

OX2 (*Order Crossover 2):* [7] also proposed by Davis in 1991. The procedure to create an offspring is similar to that of OX1, but in the third step the remaining alleles are copied from the beginning of the first parent.

CX *(Circle Crossover):* [19] proposed by Oliver in 1987 proceed as follows. It begins by putting in the first position of the child, the allele value, which is in the first position of the first parent. Then the allele value found in the same (first) position of the second parent is used as an index to retrieve an allele value from the first parent and put it into the offspring. Again, the allele value found in the same position of the second parent is used as an index to retrieve the next allele value to put into the offspring. This process is repeated until the cycle is finished. Then, if some gene positions are to be filled, the remaining alleles are extracted from the second parent

OCPX *(one cut point crossover)***:** [23] was proposed by Reeves, based on OPX (one point crossover). Here a cutting point is randomly selected. A child is built by following the sequence of the first parent until the cutting point, then the rest of the alleles are taken from the beginning of the second parent in the order they appear.

OBX *(Order Based Crossover)*: [14] Syswerda proposed this crossover operator for the TSP, and proceeds as follows.

- A binary mask is randomly generated to determine which genes are copied from the first and which from the second parent.
- Genes from the first parent, which are in the same position as the 1´s in the mask, are copied in the child.
- The remaining positions in the child are copied in the positions corresponding to 0´s in the mask, preserving the order they appear in the second parent.

PPX *(Precedence Preservative Crossover):* [3] the precedence relationships that appear in the parents are preserved in the child. The method proceeds as follows.

- A binary mask is randomly generated to determine which genes are copied from the first and which from the second parent.
- Bits set to 1 indicate that genes from the first parent will be copied in the child and bits set to 0 indicate that genes from the second parent will be copied in the child. Genes from the parents are copied, from left to right, in the same order they appear in either parent.

OSX *(One Segment Crossover):* [15] it is also based in TPX, and proceeds as follows.

- To determine which genes are copied from the first or second parent two cutting points P_1 and P_2 are randomly selected.
- Genes from the first parent are copied into the child from its first position until P_1.
- Positions between P_1 and P_2 are copied in the child from the second parent avoiding repetitions.
- Positions P_2+1 to l (chromosome length) are copied from the first parent avoiding repetitions.

5. THE BIAS, A MEASURE FOR GENETIC DIVERSITY

Bäck and Hoffmeister [1], introduced the *population diversity* in terms of the bias measure defined by Grefenstette [16] as follows;

$$b(P(t)) = \frac{1}{l \cdot \mu} \sum_{j=1}^{l} max \left(\sum_{\substack{i=1 \\ a^t_{i,j}=0}}^{\mu} (1 - a^t_{i,j}), \sum_{\substack{i=1 \\ a^t_{i,j}=1}}^{\mu} a^t_{i,j} \right)$$

where l is the chromosome length, μ is the population size and $a^t_{i,j}$ denotes the allele value at time (generation) t. The bias b $(0.5 \leq b \leq 1.0)$ indicates the average percentage of the most outstanding value in each position of the individuals. Smaller values of b indicate higher genotypic diversity and vice versa. The bias b can be used to formulate an adequate termination criterion.

In this work we try to determine the EA behaviour under different crossover methods. Diversity of individuals is crucial to convergence. Consequently, we studied this diversity in the different type of individuals, which participate in the multirecombined evolutionary process: population, studs, immigrants and offspring.

In our multirecombinative scheme, during a generation, each one of these four pools has the following size,

Pool Type	Pool size (μ)
Population pool size:	number of individuals in the population.
Studs pool size:	number of individuals in the population.
Immigrants pool size:	(n2 – 1) * number of individuals in the population.
Offspring pool size:	(n2 – 1) * 2 * n1* number of individuals in the population.

Bias b of Grefenstette was established for binary representation of chromosomes and to calculate it we can proceed as follows. A vector of length l is built for each allele value (0 or 1), indicating the number of occurrences of that allele value in each gene position. Then the greatest occurrence values, of either allele value, are added and averaged to the number of genes in the population ($l.\ \mu$).

As in our scheduling problem an individual is represented by a permutation of jobs we extend the bias concept as follows:

$$b(Pool(t)) = \frac{1}{l \cdot \mu} \sum_{i=1}^{\mu} \max_{j=1,\dots,l} \left(occ^t_{i,j} \right)$$

where l is the chromosome length, μ is the pool size and $occ^t_{i,j}$ denotes the number of occurrences of allele value i in position j at time t. The bias b , ranges from $\frac{1}{number\_of\_allele\_values}$ to 1.0 and indicates the average percentage of the most outstanding value in each position of the chromosome in the considered pool. Smaller values of b indicate higher genotypic diversity and vice versa. To study the diversity of each pool we build an l x μ matrix, *OCC*, and again the greatest occurrence values, of all allele values, are added and averaged to the number of genes in the pool type ($l.\ \mu$).

Each *OCC* matrix associated with a pool type, stores the information of individuals through a generation and it is re-initiated in the next generation. At the end of each generation the bias b is re-calculated.

6. EXPERIMENTAL TESTS AND RESULTS

The evolutionary algorithms were tested for five, 40 job problem size, hard instances (19, 41, 46, 56 and 116) extracted from OR-library benchmarks [2] for the single machine weighted tardiness scheduling problem. We performed a series of runs for each instance. The maximum number of generations was fixed at 500. Population size was fixed at 15 individuals. Probabilities for crossover were set to 0.65 and for mutation (exchange) were set to 0.05, in all experiments. The number n_1, of crossovers and the number n_2, of parents, were set to 18 and 20 respectively, in all experiments.

To compare the algorithms, the following relevant performance variables were chosen:

Ebest = ((best value - *opt_val*)/*opt_val*)100. It is the percentile error of the best-found individual when compared with the known, or estimated, optimum value *opt_val*. It gives us a measure on how far the best individual is from that *opt_val*.

Gbest. It is the generation where the best individual was found.

Evals: It is the number of evaluations necessary to obtain the best-found individual in a run.

The following tables summarize mean values for the performance variables through the selected instances under each crossover operator. Boldfaced-italic values indicate the best performer(s) for each instance. At the bottom of the tables, average, minimum and maximum mean values of the corresponding performance variable, are indicated.

Instance	Upper Bound	PMX	OX1	OX2	CX	OCPX	OBX	PPX	OSX
19	77122	0.33	0.61	***0.31***	0.67	6.87	17.21	18.51	0.55
41	57640	***0.11***	2.59	0.30	1.45	12.30	24.86	25.65	1.21
46	64451	0.03	0.17	***0.01***	0.55	7.34	18.93	18.71	0.08
56	2099	8.06	6.42	***6.10***	8.78	29.50	52.46	109.19	13.68
116	46770	***0.33***	1.92	0.58	1.56	17.80	37.69	41.87	1.28
	Avg	**1.77**	**2.34**	**1.46**	**2.60**	**14.76**	**30.23**	**42.79**	**3.36**
	Min	**0.03**	**0.17**	**0.01**	**0.55**	**6.87**	**17.21**	**18.51**	**0.08**
	Max	**8.06**	**6.42**	**6.10**	**8.78**	**29.50**	**52.46**	**109.19**	**13.68**

Table 1. Mean *Ebest* for each instance under different crossover operators.

Table 1 summarizes mean *Ebest* values through all instances and operators. Results show that on average, the percentile error of the best found individual when compared with the best known objective value is the smallest (1.46 %) under OX2 and the greatest (42.79 %) under PPX. We can also see that OX2, PMX, OX1, CX, and OSX are of good or acceptable performance (in that order) while OCPX, OBX, and PPX are the worst performers.

Instance	Upper Bound	PMX	OX1	OX2	CX	OCPX	OBX	PPX	OSX
19	77122	355.2	445.8	***279.8***	477.8	461.3	361.3	330.2	331.1
41	57640	371.0	456.8	***299.8***	480.4	461.6	319.3	376.8	392.5
46	64451	371.0	468.4	***303.9***	472.2	457.9	327.4	397.4	426.3
56	2099	***83.2***	210.1	112.9	343.0	260.8	338.3	309.5	234.5
116	46770	441.0	468.7	390.0	484.3	427.4	355.2	***272.4***	456.3
	Avg	**324.28**	**409.96**	**277.28**	**451.54**	**413.80**	**340.30**	**337.26**	**368.14**
	Min	**83.20**	**210.10**	**112.90**	**343.00**	**260.80**	**319.30**	**272.40**	**234.50**
	Max	**441.00**	**468.70**	**390.00**	**484.30**	**461.60**	**361.30**	**397.40**	**456.30**

Table 2. Mean *Gbest* for each instance under different crossover operators.

Table 2 summarizes mean *Gbest* values through all instances and operators. Results show that on average, the minimum number of generations required to find the best individual is 277.28 under OX2, while the maximum is 451.54 under CX. PMX also shows a low *Gbest* value of 324.28.

This is reflected in table 3 too, where the corresponding mean *Evals* values are 2,828,256, 4,605,708 and 3,307,656 for OX2, CX and PMX, respectively.

Instance	Upper Bound	PMX	OX1	OX2	CX	OCPX	OBX	PPX	OSX
19	77122	3623040	4547160	***2853960***	4873560	4705260	3685260	3368040	3275220
41	57640	3784200	4659360	***3057960***	4900080	4708320	3256860	3843360	4003500
46	64451	3784200	4777680	***3099780***	4816440	4670580	3339480	4053480	4348260
56	2099	***848640***	2143020	1151580	3498600	2660160	3450660	3156900	2391900
116	46770	4498200	4780740	3978000	4939860	4359480	3623040	***2778480***	4654260
	Avg	**3307656**	**4181592**	**2828256**	**4605708**	**4220760**	**3471060**	**3440052**	**3734628**
	Min	**848640**	**2143020**	**1151580**	**3498600**	**2660160**	**3256860**	**2778480**	**2391900**
	Max	**4498200**	**4780740**	**3978000**	**4939860**	**4708320**	**3685260**	**4053480**	**4654260**

Table 3. Mean *Evals* for each instance under different crossover operators.

The following figures resume information on genetic diversity, and are related only to instance 19, because it is demonstrative for all instances.

In the multirecombinative processes of MCMP-SRI, immigrants ensure to maintain genetic diversity incorporating exploration while the studs guide the search to promising areas, incorporating exploitation.

This assertion is clearly indicated in the following two figures. In figure 2, the high diversity of the immigrant's pool is shown. In average the bias *b* ranges from 0.0486 to 0.0530. In figure 3, the low diversity of the stud's pool is shown. In average the bias ranges from 0.5516 (in PPX) to 0.8419 (in CX).

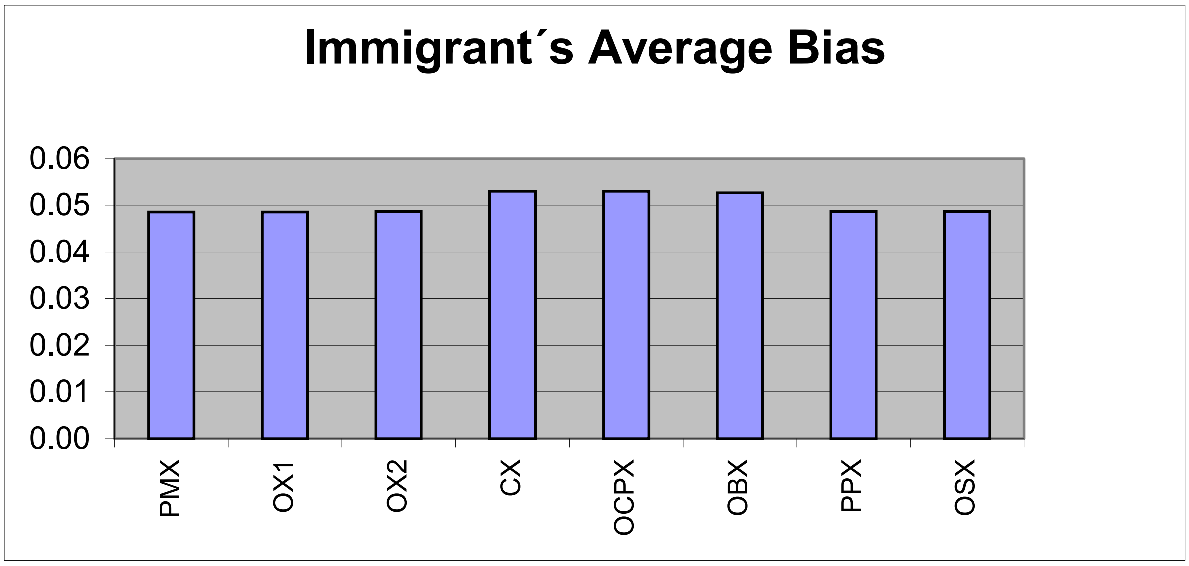

Fig. 2. Average bias of immigrants over 500 generations for each series of experiments

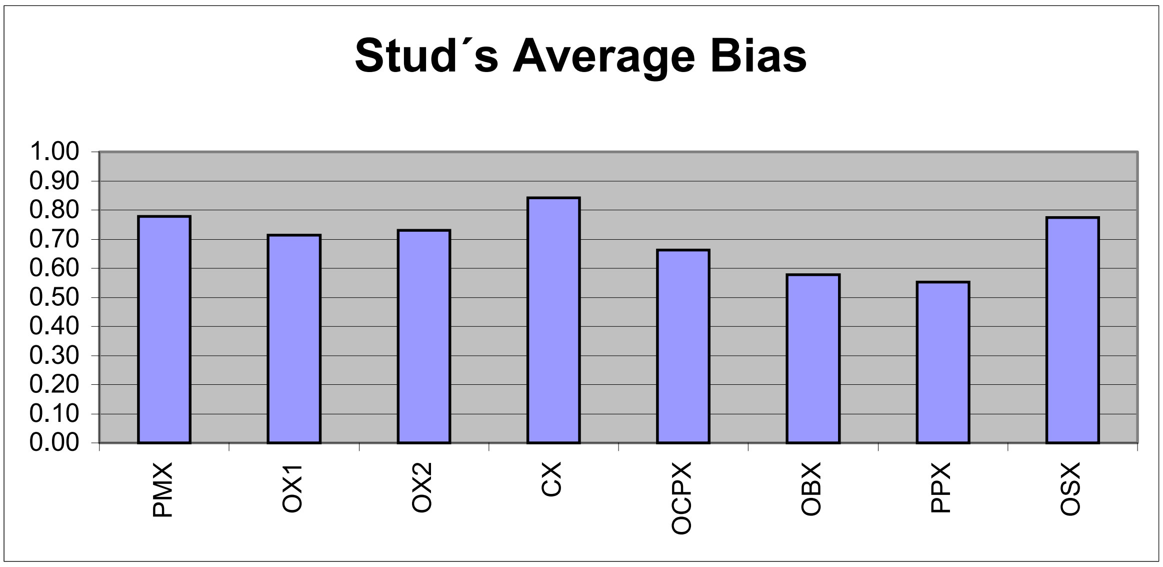

Fig. 3. Average bias of studs over 500 generations for each series of experiments

PMX	OX1	OX2	CX	OCPX	OBX	PPX	OSX
0.33	0.61	0.31	0.67	6.87	17.21	18.51	0.55

Table 4. Mean Ebest values under each crossover method for instance 19

From table 4 and figure 3 we can see how the stud´s genetic diversity affects quality of results. Best performers are PMX, OX1, OX2, CX and OSX, with lower pool diversity while worst performers are OCPX, OBX and PPX, showing higher pool diversity.

The stud is the breeding individual, which provides good genetic material into the mating pool for SRI multirecombinations. Some of the crossover operators (with $b \geq 0.7$) preserve better this genetic material than others. This observation allows us to conjecture that under this condition the algorithm will better exploit the stud's neighbourhood creating higher quality populations.

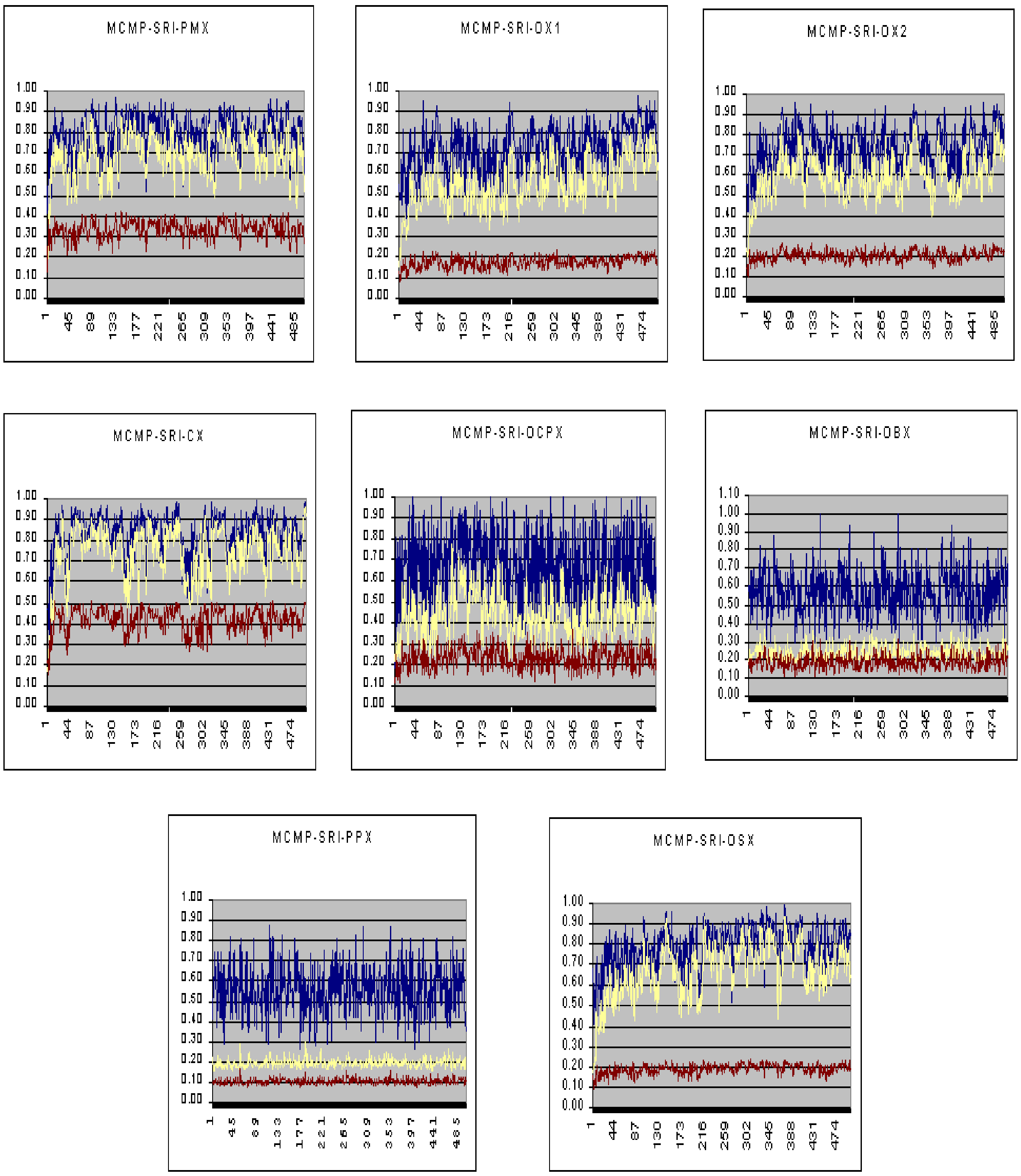

Fig. 4. Bias b values for the stud's, population's and offspring's pools, under each crossover method for instance 19, through 500 generations.

Figure 4 shows bias b values after each generation and for each pool type. The black, white and grey curves correspond to the stud's, population's and offspring's pools, respectively, through the

500 generations. We can observe that for the best performers (PMX, OX1, OX2, CX and OSX) the bias *b* has similar (high) ranges for the stud's and the population's pools showing low diversity and differs substantially from that of the offspring's pool which shows higher diversity. On the other hand for the worst performers (OCPX, OBX and PPX) the population's pool shows a higher diversity which, in the extreme cases of OBX and PPX, is in the range of that of offspring's pool. At this point there is a clear indication that a correlation exists between the similar variability of genetic diversity for studs and population, and the quality of results. That is to say: if the crossover operator does not destroy the guiding ability of the stud, the convergence is performed towards better individuals.

7. CONCLUSIONS

MCMP-SRI, showed its good performance by providing new benchmark values for the E-T scheduling problem [20] or reaching existent benchmarks in other related due-date objectives in single machine environments. That performance was achieved by using PMX crossover, selected as a good method in our initial trials.

In this work we selected the most difficult instances of one of the hardest scheduling problem for that machine environment: the weighted tardiness problem. A variety of crossover methods suitable for permutation representation were implemented. Their effect on diversity and quality of results were studied.

We can remark the following observations:

- In SRI multirecombinations we must adopt crossover operators, which better preserve the genetic material embedded in the stud. This allows the algorithm to better exploit the stud's neighbourhood creating higher quality populations.

- If the crossover operator does not destroy the guiding ability of the stud, then the individuals in the population can perform their search towards the promising areas of the problem space providing better final solutions. This can be seen by the existent correlation between higher quality of results and similarity of population and studs (low) diversity.

- We determined that for the fitness landscape associated to weighted tardiness problem, SRI multirecombination performs better under OX2 and PMX.

- The study of pools diversity offers a framework to determine the influence of other operators and mechanism in multirecombination.

Future work will be devoted to study the effect of different operators and mechanisms in multirecombination schemes in a variety of optimization problems.

8. ACKNOWLEDGEMENTS

We acknowledge the co-operation of the project group for providing new ideas and constructive criticisms. Also to the Universidad Nacional de San Luis, the Universidad Nacional de La Patagonia Austral, and the ANPCYT from which we receive continuous support.

9. REFERENCES

[1]. Bäck T., Hoffmeister F.: *Extended selection mechanisms in genetic algorithms*. Proc. of the 4th Int. Conf. on Genetic Algorithms, pp 92-97, Morgan Kaufmann, 1991.

[2]. Beasley J. E. Weighted Tardiness Scheduling, OR Library, http://mscmga.ms.ic.ac.uk/

[3]. Bierwirth C., Mattfeld D.C., Kopfer H.: *On Permutation Representations for Scheduling Problems*. In Proceedings of Parallel Problem Solving from Nature. Lecture Notes in Computer Science, Vol. 1141. Springer-Verlag, Berlin Heidelberg New York (1996) 310-318.

[4]. Crauwels H.A.J., Potts C.N. and Van Wassenhove L.N. *Local search heuristics for the single machine total weighted tardiness scheduling problem*, Informs Journal on Computing 10, 341-350. 1998.

[5]. Chen T. and Gupta M., Survey of scheduling research involving due date determination decision, European Journal of Operational Research, vol 38, pp. 156-166, 1989.

[6]. Davis L., *Applying adaptive algorithms to domains*, in Proceedings of the International Joint Conference on Artificial Intelligence, pp. 162-164, 1985.

[7]. Davis L., *Handbook of Genetic Algorithms,* New York: Van Nostrand Reinhold Computer Library, 1991.

[8]. Eiben A.E., Raué P-E., and Ruttkay Zs., *Genetic algorithms with multi-parent recombination*. In Davidor, H.-P. Schwefel, and R. Männer, editors, Proceedings of the 3rd Conference on Parallel Problem Solving from Nature, number 866 in LNCS, pages 78-87. Springer-Verlag, 1994

[9]. Eiben A.E., van Kemenade C.H.M., and Kok J.N., *Orgy in the computer: Multi-parent reproduction in genetic algorithms*. In F. Moran, A. Moreno, J.J. Merelo, and P. Chacon, editors, Proceedings of the 3rd European Conference on Artificial Life, number 929 in LNAI, pages 934-945. Springer-Verlag, 1995.

[10]. Eiben A.E. and. Bäck Th., An empirical investigation of multi-parent recombination operators in evolution strategies. Evolutionary Computation, 5(3):347-365, 1997.

[11]. Esquivel S., Leiva A., Gallard R., - *Multiple Crossover per Couple in Genetic Algorithms*, Proceedings of the Fourth IEEE Conference on Evolutionary Computation (ICEC'97), pp 103-106, ISBN 0-7803-3949-5, Indianapolis, USA, April 1997.

[12]. Esquivel S., Leiva A., Gallard R.: *Couple Fitness Based Selection with Multiple Crossover per Couple in Genetic Algorithms*. Proceedings of the International Symposium on Engineering of Intelligent Systems (EIS´98), La Laguna, Tenerife, Spain Vol. 1, pp 235-241, ed. E.Alpaydin. Publishesd by ICSC Academic Press, Canada/Switzerland, February 1998.

[13]. Esquivel S., Leiva H.,.Gallard R., *Multiple crossovers between multiple parents to improve search in evolutionary algorithms*, Proceedings of the 1999 Congress on Evolutionary Computation (IEEE). Washington DC, pp 1589-1594

[14]. Gen M. and Cheng R.: *Genetic Algorithm & Engineering Optimization*. John Wiley & Sons (1997).

[15]. Goldberg, D. and R. Lingle, *Alleles, loci and the traveling salesman problem*, in Proceeding of the First International Conference on Genetic Algorithms, Lawrence Eribaum Associates, pp. 154-159, Hillsdale, NJ, 1987.

[16]. Grefenstette J.J.: A user's guide to GENESIS. Navy Center for Applied Research in Artificial Intelligence, Washington D.C., 1987.

[17]. Michalewicz, M., *Genetic Algorithms + Data Structures = Evolution Programs*. Springer, third revised edition, 1996.

[18]. Morton T., Pentico D*., Heuristic scheduling systems*, Wiley series in Engineering and technology management. John Wiley and Sons, INC, 1993.

[19]. Oliver, I., D. Smith, and J. Holland, *A study of permutation crossover operators on the traveling salesman problem*, in European Journal of Operational Research, pp. 224-230, 1986.

[20]. Pandolfi D., Vilanova G., De San Pedro M., Villagra A.: *Multirecombining studs and immigrants in evolutionary algorithm to face earliness-tardiness scheduling problems*. Proceedings of the International Conference in Soft Computing. University of Paisley, Scotland, U.K., June2001.

[21]. Pandolfi D.; Vilanova G.; De San Pedro M.; Villagra A. Gallard R.: *Solving the Single-Machine Common Due Date Problem Via Stud and Immigrants in Evolutionary Computation* vol III pp 409-413 World Multiconference on Systemics, Cybernetics and Informatics Orlando 2001

[22]. Pinedo, Michael., *Scheduling: Theory, Algorithms and System*. Prentice Hall, first edition, 1995. pp 44-48, 143-145.

[23]. Reeves C., *A genetic algorithm for flow shop sequencing*, Computers and Operations Research, vol 22, pp5-13, 1995.

[24]. Tsujimura Y., Gen M., Kubota E.: *Flow shop scheduling with fuzzy processing time using genetic algorithms*. The 11th Fuzzy Systems Symposium pp 248-252. Okinawa. 1995.