A COMPARISON OF TWO MULTIRECOMBINATED EVOLUTIONARY ALGORITHMS FOR THE JOB SHOP SCHEDULING PROBLEM

Minetti G., Salto C., Alfonso H.
Proyecto UNLPAM-09/F009[1]
Departamento de Informática - Facultad de Ingeniería
Universidad Nacional de La Pampa
Calle 110 esq. 9
(6360) General Pico – La Pampa – Rep. Argentina
e-mail: {minettig,saltoc,alfonsoh }@ing.unlpam.edu.ar
Phone: (02302)422780/422372, Ext. 6302

Gallard R.
Proyecto UNSL-338403[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

Over the past few years, a continually increasing number of research efforts have investigated the application of evolutionary computation techniques for the solution of scheduling problems. Scheduling problems can pose extremely complex combinatorial optimization problems, which belong to the NP-hard family.

This work shows how an evolutionary approach using different chromosome representations with multiplicity feature MCMP can efficiently solve the JSSP.

Keywords: evolutionary algorithms, multiplicity, scheduling, optimization, chromosome representation

---

[1] The Research Group is supported by the Universidad Nacional de La Pampa.

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

1. Introduction

Scheduling is an economically very important yet computationally extremely difficult task. This problem can be identified in several different application areas such as production planning, flexible manufacturing systems, computer design, logistics, communications, and soon on. Scheduling problems are prominent combinatorial optimization problems. The task of scheduling is the allocation of jobs over time to limited resources, where a number of objectives should be optimized and several constraints must be satisfied. A job is completed by a predefined sequence of operations. The result of scheduling is a schedule showing the assignment of start times and resources to each operation. The quality of a schedule is measured by means of an objective function, which assigns a numerical value to a schedule. In our case the completion time of the last job abandoning the system (makespan) is optimized. The determination of an optimal solution to a scheduling problem belongs to the class of NP-hard problems, which means than no deterministic algorithm is known yet for solving the problem in polynomial time. In recent years, several efforts have been sought to investigate and exploit the application of evolutionary computation techniques to various scheduling problems[4, 16, 3, 22]. The main difficulty encountered is that of specifying an appropiarte representation of feasible schedules. The ways of representation can be direct or indirect [1].

- *Indirect Representation*: a transition from chromosome representation to a legal production schedule has to be performed by a schedule builder prior to evaluation [18, 21, 1, 4].
- *Direct Representation*: the schedule produced itself is used as a chromosome. Decoder procedure is not necessary but specific genetic operators should be designed [17, 14].

In this work, two different representations are used which belong to the indirect group. They are the *decoders* and *priority rule based* representations. Both are domain-independent indirect representation, then they do not contain auxiliary information of the particular scheduling problem.

The application of the multiple crossovers on multiple parents [8, 9, 10, 11] MCMP have been successfully applied in uni and multimodal optimization problems. Recently the applicability to JSSP has been studied in [19].

The next section is devoted to an indirect representation description before mentioned. Later the multiple crossovers on multiple parents feature is detailed. Finally the performed experiments and their results are explained.

2. Indirect Representations: Decoders and priority rule based representation

Under this approach the evolutionary algorithm performs a blind reproduction of encoded solutions by applying the conventional operators. The domain knowledge remains separated within the evaluation procedure to determine the fitness.

2.1. Decoders

In JSSP a *job based representation* consists of a list of jobs and a schedule is built according to the sequence of jobs. Here we deal with permutations, and if chromosomes are encoded as permutations adequate genetic operators, such as *partially-mapped crossover*, *order crossover* and *cycle crossover* should be used. Nevertheless, another way to face a problem involving permutations is by the use of *decoders*. Here a chromosome is an n-vector where the i^{th} component is an integer in the range 1..(n-i+1). The chromosome is interpreted as a strategy to extract items from an ordered list L and build a permutation. For example if L = (1, 2, 3, 4) then table 1, indicates the correspondence among chromosomes and permutations.

chromosome	**permutation**
1 1 1 1 1 1	1 2 3 4 5 6
4 3 4 1 1 1	4 3 6 1 2 5
4 2 3 2 1 1	4 2 5 3 1 6

Table 1. Chromosome-permutation correspondence

A decoder is a mapping from the representation space into a feasible part of the solution space, which includes mappings between representations that evolve and representation that constitute the input for the evaluation function. This simplifies implementation and produces feasible offspring under different conventional crossover methods, avoiding the use of penalties or repair actions. Next sections introduce incest prevention, multiplicity of parents and crossovers, implementation details and results.

2.2. PRIORITY RULE BASED REPRESENTATION

In [5] a priority-rule-based genetic algorithm is proposed, where a chromosome is encoded as a sequence of dispatching rules for job assignment and a schedule is built with a priority dispatching heuristic based on the sequence of dispatching rules. The genetic algorithms are used here to evolve those chromosomes improving the sequences of dispatching rules. Priority dispatching rules are frequently applied heuristics for solving scheduling problems due to their ease of implementation and low time complexity. Giffler and Thompson´s algorithms can be considered as the basis of priority rule based heuristics [12, 19]. The main problem is to determine an effective priority rule. In table 1 priority rules commonly used in practice are shown.

For an n job and m machine problem a chromosome is a string of $n \times m$ entries ($p_1, p_2, \ldots, p_{n\,m}$). Each entry, represents one of the prespecified rules. The entry in the i^{th} position indicates that a conflict in the i^{th} iteration of the Giffler and Thompson algorithm should be resolved by using the priority rule p_i. Ties are broken by a random choice. More precisely, an operation from the conflict set has to be selected by rule p_i.

Let

PS_t	=	a partial schedule containing t scheduled operations.
S_t	=	the set of schedulable operations at iteration t, corresponding to PS_t.
σ_i	=	the earliest time at which operation $i \in S_t$ could be started.
ϕ_i	=	the earliest time at which operation $i \in S_t$ could be completed.
C_t	=	the set of conflicting operations in iteration t.

The procedure *builder* deduces a schedule from a given chromosome ($p_1, p_2, \ldots, p_{n\,m}$):

Procedure: builder (Deduce a schedule for Priority-Rule-Based Encoding)

1. Let $t = 1$ and begin with PS_t as the null partial schedule and let S_t include all operations with no predecessors.
2. Determine $\phi^*_t = \min_{i \in S_t} \{ \phi_i \}$ and the machine m^* on which ϕ^*_t could be realized. If more than one such machine exists, the tie is broken by a random choice.
3. Form a conflicting set C_t which includes all operations $i \in S_t$ with $\sigma_i < \phi^*_t$ that require machine m^*. Select one operation from C_t by the priority rule p_t and add this operation to PS_t as early as possible, thus creating a new partial schedule PS_{t+1}. If more than one operation exists according to the priority rule p_t, the tie is broken by a random choice.

4. Update PS_{t+1} by removing the selected operation from S_t and adding the direct successor of the operation to S_t. Increment t by one.
5. Return to step 2 until a complete schedule is generated.

RULE		DESCRIPTION
SPT	Shortest Processing Time	Select an operation with the shortest processing time
LPT	Longest Processing Time	Select an operation with the longest processing time
MWR	Most Work Remaining	Select an operation for the job with the most total remaining processing time
LWR	Least Work Remaining	Select an operation for the job with the least total remaining processing time
MOR	Most Operations Remaining	Select an operation for the job with the greatest number of operations remaining
LOR	Least Operations Remaining	Select an operation for the job with the smallest number of operations remaining
EDD	Earliest Due Date	Select a job with the earliest due date
FCSF	First come, first served	Select the first operation in the queue of jobs for the same machines
RND	Random	Select an operation at random

Table 2. Dispatch rules for a Job Shop

3. MULTIPLICITY OF CROSSOVERS AND PARENTS (MCMP)

One of the latest contributions in the theoretical field of Evolutionary Computation known as *the multiplicity feature* is related to new proposed multirecombination methods:

- ✓ MCPC: Multiple Crossovers per Couple which reinforces the exploitation of features of previously found (good) solutions.
- ✓ MCMP: Multiple Crossovers on Multiple Parents [6,7,8,9] which provides a balance in exploitation and exploration because the searching space is efficiently exploited (by the multiple application of crossovers) and explored (by a greater number of samples provided by multiple parents).

This novel approach was successfully applied to single and multicriteria optimization. Recently it was also applied to JSSP with decoder representation [18]. It was shown that a greater number of crossovers for a given number of parents provides better results.

In MCMP we extended the multiparent approach of Eiben [5]. Here he used, initially, three *scanning crossover* (SX) methods, which essentially take genes from parents to contribute in building the offspring. The SX general mechanism assigns a marker to each parent and to the offspring. The offspring's marker traverses all of its positions from left to right. At each step parent's markers are updated each time a gene is selected. The main characteristics of all gene-scanning procedures are the update marker mechanisms and the way to choose a marked gene from the parents. Only one offspring is generated. In our work we choose *uniform scanning crossover* (USX*)*.This method is a natural extension of *uniform crossover*. Consequently, each gene in the child is provided from any of the corresponding genes in the parents with equal probability.

Multiple crossovers on multiple parents (MCMP), the method used here, allows multiple recombination of multiple parents under any of the SX variants, expecting that exploitation and exploration of the problem space be adequately balanced. MCMP provides a means to exploit good features of more than two parents selected according to their fitness by repeatedly applying one of the SX variants: a number n_1 of crossovers is applied on a number n_2 of selected parents. From the n_1 produced offspring a number n_3 of them are selected, according to some criterion, to be inserted in the next generation.

4. EXPERIMENTAL TESTS

Two different genetic algorithms were devised using MCMP: one of them with priority rule based representation (MCMP-PRB), and the other with decoders (MCMP-Dec). Both algorithms were contrasted for a set of selected instances of the Job Shop Scheduling Problem (JSSP). A total of 60 different experiments corresponding to different (n_1, n_2) combinations, USX and multi-recombination methods were designed. For each instance a series of ten runs was performed. The evolutionary algorithms used proportional selection for mating and elitism to retain the best valued individual. The population size was fixed at 50 individuals. For insertion in the next generation the best child was chosen (n_3 = 1). The number of crossover and parents were set: $1 \leq n_1 \leq 4$ and $3 \leq n_2 \leq 5$, respectively. For mutation a *big-creep* operator, replacing the gene value by another en the permitted range was used. The maximum number of generations was fixed at 500 and probabilities for crossover and mutation were fixed at 0.8 and 0.01, respectively. These values were determined as the best combination of probabilities after many initial trials. For these experiments, except EDD, all the rules listed in table 2 were used. Six instances [15], with known optimal makespan were used. They were:

Instance	Size	Optimum
abz6	10×10	943
la01	10×5	666
la06	15×5	926
la12	20×5	1039
la15	20×5	1207
ft10	10×10	930

Table 3. Instances

The following relevant performance variables were chosen:

Ebest = (Abs(*opt_val* - best value)/*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 of how far we are from that *opt_val*.

Epop = (Abs(*opt_val*- pop mean fitness)/*opt_val*)100
It is the percentile error of the population mean fitness when compared with *opt_val*. It tell us how far the mean fitness is from that *opt_val*.

Gbest **=** It identify the generation where the best individual (remained by elitism) were found.

5. Results

Figures 1 to 9 summarise the information on demonstrative instances *abz6*, *la06* and *la12*. The used notation specifies in the horizontal edges the number of crossovers and parents, for example (1,3) references to the combination of 1 crossover on 3 parents.

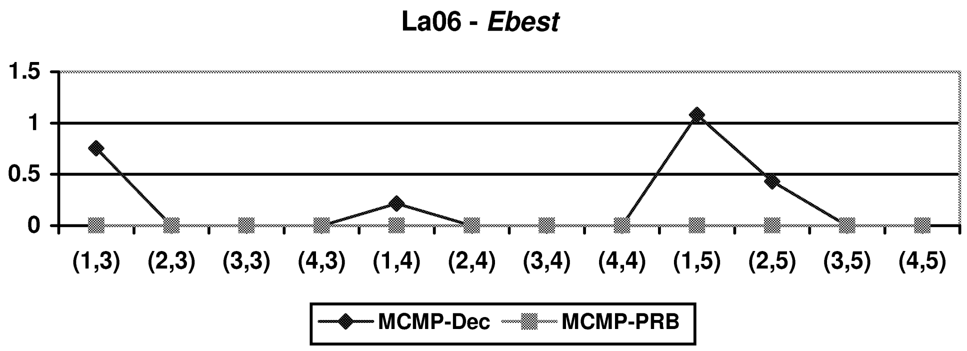

Figure 1. *Ebest* values from both algorithms for the *la06* instance

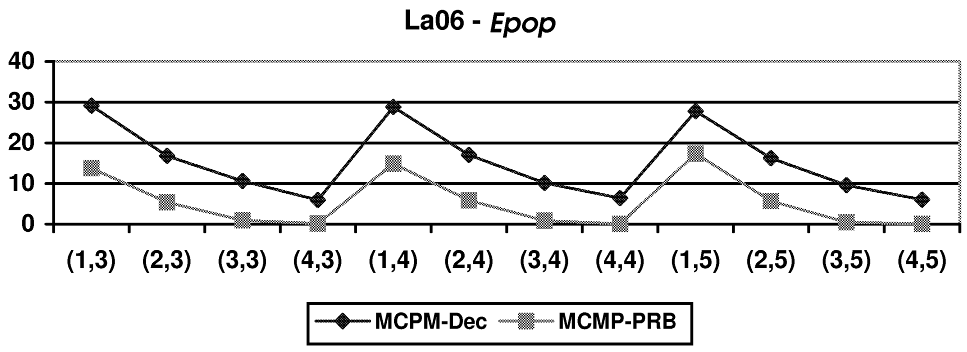

Figure 2. *Epop* values from both algorithms for the *la06* instance

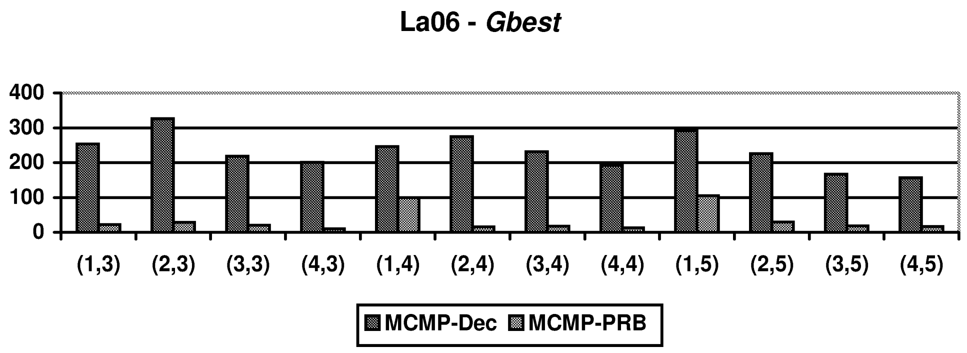

Figure 3. *Gbest* values from both algorithms for the *la06* instance

Figure 1 shows that all possible combinations (n_1, n_2) under MCMP-PRB reach the optimum at least once. This situation is present in only some combinations of MCMP-Dec, but even so errors are very small (the greater reaches 1%).

Analysing *Epop* (figure 2), we can see that there is a relation between the number of crossover applied and the quality of obtained solutions. That is, lower population errors are obtained when the number of crossover is incremented. When $n_1 = 1$, high peaks are observed in both algorithm (29.00% in MCMP-Dec and 17.38% in MCMP-PBR). In MCMP-Dec, *Epop* varies in a range from 6% to 29.15% while for MCMP-PRB it ranges from 0.084% to 17.38%. In both algorithm, the final population turn out to be more and more closer to the optimum as long as n_1 is incremented. This is clearly indicated by the wave form of figure 2.

Regarding *Gbest*,(see fig. 3) MCMP-PRB founds the optimum in a lower number of generations than MCMP-Dec. In MCMP-PRB, the maximum generation number to find the best value over all instances does not exceed the minimum achieved by MCMP-Dec.

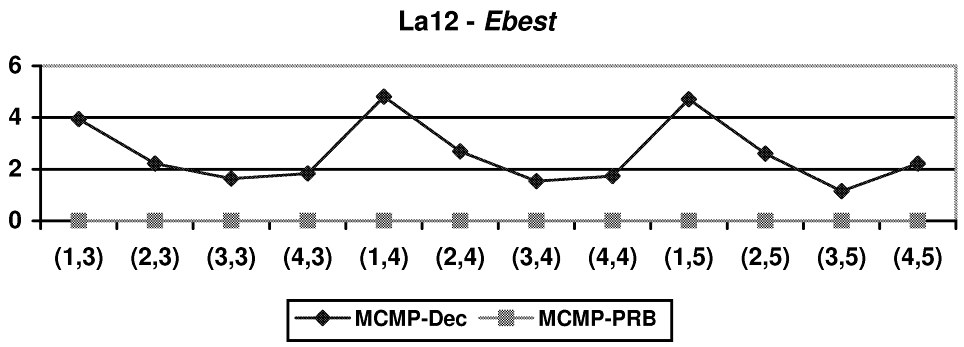

Figure 4. *Ebest* values from both algorithms for the *la12* instance

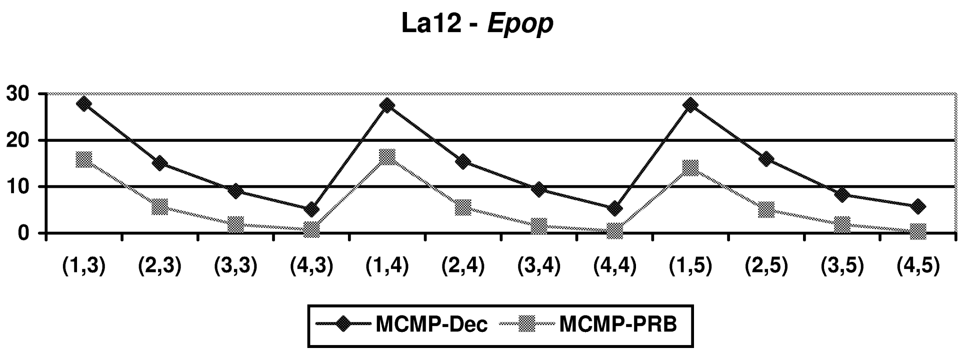

Figure 5. *Epop* values from both algorithms for the *la12* instance

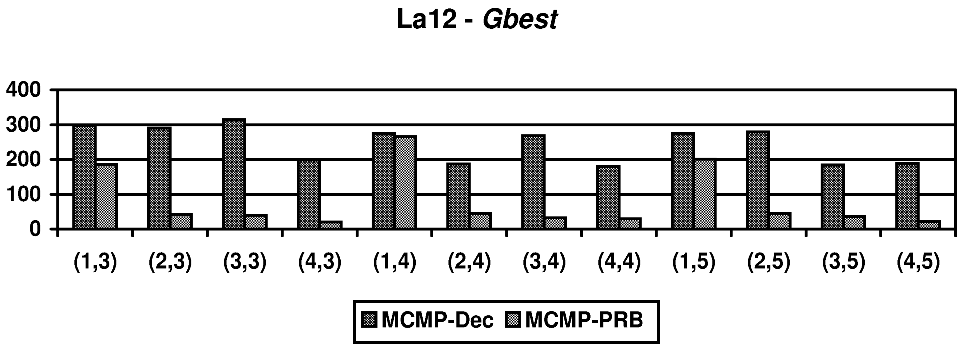

Figure 6. *Gbest* values from both algorithms for the *la12* instance

For instance *la12,* figure 4 shows that MCMP-Dec never achieves the optimum (but best individuals are very closed to it with a percentile error varying between 1.15 and 4.8 %) while MCMP-PBR finds the optimum for every combination (n_1, n_2) at least once. When n_1 is augmented (independently of n_2) until 4, MCMP-Dec presents, in general, a clear *Ebest* decrement.

Regarding *Epop*, in the figure 5 MCMP-PRB presents a better performance at population level. Again, as expected both methods have big errors when $n_1 = 1$ (27% for MCMP-Dec and 16.8% for MCMP-PRB) and they decline as n_1 is incremented. Under MCMP-PRB, the individuals of the final population are closer to the best found individual for each of the possible combinations (n_1, n_2).

In figure 6 it is shown that MCMP-PRB reaches the optimum in less generations than MCMP-Dec. Values of *Gbest* for both algorithms are closer only when a single crossover is applied.

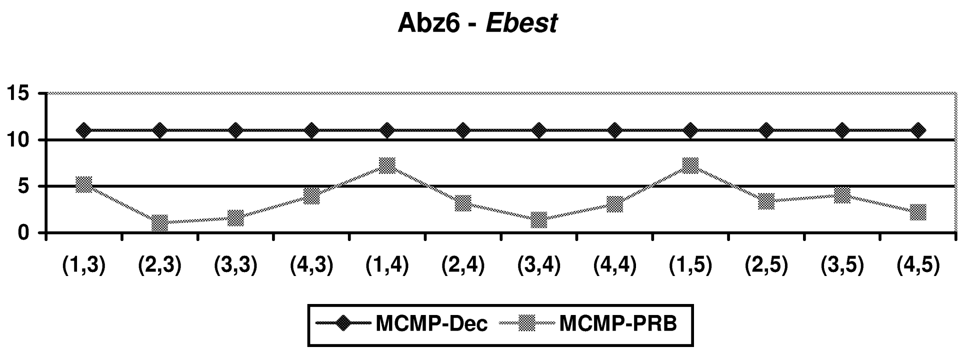

Figure 7. *Ebest* values from both algorithms for the *abz6* instance

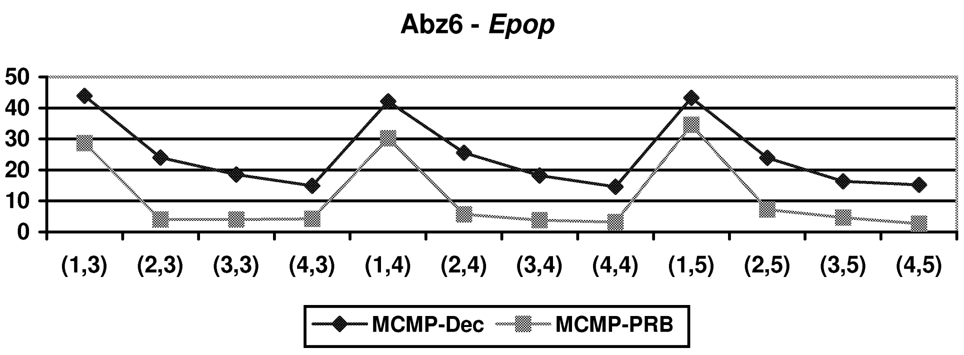

Figure 8. *Epop* values from both algorithms for the *abz6* instance

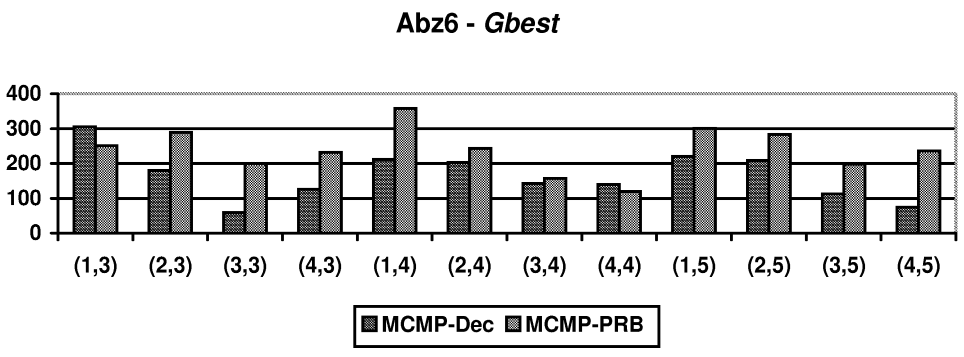

Figure 9. *Gbest* values from both algorithms for the *abz6* instance

Figure 7 shows that the optimum is not found by both algorithms but while MCMP-Dec stays in a local optimum for all (n_1, n_2) combinations (with *Ebest* equal to 1.0286%) MCPC-PBR *Ebest* values are much lower. Under this later approach greater *Ebest* values do not exceed 7.2%. Best values are found for n_1 between 2 and 4.

Again in figure 8 the same systematic behaviour is observed. *Epop* diminishes as long as n_1 is incremeted. Larger *Epop* changes are produced when n_1 varies from one to two. From $n_1 = 2$, the *Epop* values start to be more stable. For example, in MCMP-PRB and 3 parents, *Epop* values range from 28.69% to 4.07%. As it happened with *Ebest* values, MCMP-PRB presents a notable better behaviour than MCMP-Dec providing better *Epop* values.

In figure 9 it is shown that MCMP-Dec needs less number of generation to find the best individual than MCMP-PRB. However, these solutions are not the same quality (see figure 7). This suggest the convergence to local optima.

For the remaining instances results show the same characteristics. The exception is given for the *la15* instance, because the optimum is found once by MCMP-PRB in (4,4) combination. Even though MCMP-Dec finds good suboptimal solutions. The *Gbest* analysis indicates that the generation needed to find the best value are similar for both algorithm.

6. Conclusiones

This paper shows two evolutionary approaches to solve the JSSP. Both of them apply a novel recombination method; the multirecombination of parents by replication of crossover operations. They essentially differ in the indirect representation adopted. MCMP-PRB express the individual as a string specifying dispatching rules which are applied when a conflict arises under the Giffler and Thompson algorithm. The later is used in the evolutionary process to build feasible schedules and evaluate solutions. MCMP-Dec represents an individual as a decoder and the algorithm is restricted to search within the space of all permutations. Valid offspring are guaranteed for conventional operators here and no repair algorithms neither penalty functions are longer necessary.

After a series of trials the analysis of results suggests that:

- ✓ MCMP-PBR reaches the optimum for any (n_1, n_2) combination for *la06*, *la01*, *la12* and *la15* instances.
- ✓ When instance complexity increases it became harder for both algorithms to find the optimum and this problem is stronger under MCMP-Dec where a tendency to stagnate the search is detected.
- ✓ Both algorithms find near optimal solutions in less than 3 minutes running time in standard workstations.
- ✓ In general, MCMP-PBR performs better than MCMP-Dec.

At the light of these results future work will be oriented to deep the application of MCMP-PBR on different sets of JSS problems by using different parameter settings and self-adaptation of (n_1, n_2) combinations.

7. Acknowledgements

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

8. BIBLIOGRAPHY

[1]. Bagchi, S., Uckum, S., Miyabe, Y., and Kawamura, K., *Exploring Problem-Specific Recombination Operators for Job Shop Scheduling*, in Proceedings of the fourth International Conference on Genetic Algorithms, San Diego, Morgan Kaufmann Publishers, 1991.

[2]. Bruns, R., *Scheduling*, en Th. Bäck, D. B. Fogel, and Z. Michalewicz, editors. Handbook of Evolutionary Computation , chapter F1.5, pp. F1.5:1-F1.5:9. Oxford University Press, New York, and Institute of Physics Publishing, Bristol, 1997

[3]. Cleveland, G.A and Smith, S.F., *Using Genetic Algorithms to Schedule Flow Shop Releases*, in Proceedings 3rd International Conference on Geneitc Algorithms and tehir Applications, pp. 160-169, 1989.

[4]. Davis, L., *Job Shop Scheduling with Genetic Algorithms*, in Proceedings of the First International Conference on Genetic Algorithms, Lawrence Erlbaum Associates, Hillsdale, pp. 136-140, 1985

[5]. Dorndorf, U., and Pesch, E., *Evolution based learning in a job shop scheduling environment*, Computers and operations Research, vol. 22, pp. 25-40, 1995.

[6]. Eiben, A.E., *A Method for Designing Decision Support Systems for Operational Planning*, PhD Thesis, Eindohoven University of Technology, 1991.

[7]. Eiben, A.E and van Kemenade, C.H.M, *Diagonal crossover in Genetic Algorithms for numerical Optimization*, in Journal of Control and Cybernetics, 26(3), pp. 447-465, 1995.

[8]. Esquivel S., Leiva A., and Gallard R., - *Multiple crossover per couple in genetic algorithms*, in Proceedings of the 4th IEEE International Conf. on Evolutionary Computation (ICEC'97), pp 103-106, Indianapolis, USA, April 1997.

[9]. Esquivel S., Leiva A., and Gallard R.: *Couple Fitness Based Selection with Multiple Crossover per Couple in Genetic Algorithms*, in 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.

[10]. Esquivel S., Leiva H., and Gallard R., *Multiplicity in genetic algorithms to face multicriteria optimization,* in Proceedings of the 1999 Congress on Evolutionary Computation (IEEE). Washington DC, 1999.

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

[12]. Garey, M., and Johnson, D., *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman, 1979.

[13]. Giffler, B., and Thompson, G., *Algorithms for solving production scheduling problems*, Operations Research, vol. 8, no. 4, pp. 487-503, 1960.

[14]. Husbands, P., and Mill, F., *Simulated Co-Evolution as The Mechanism for Emergent Planning and Scheduling*, in Proceedings of The Fourth International Conference on Genetic Algorithms, San Diego, Morgan Kaufmann Publishers, 1991.

[15]. Lawrence, S., *Resource constrained project scheduling: an experimental investigation of heuristic scheduling techniques*, (Supplement), Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, Pennsylvania, 1984.

[16]. Liepins, G.E. and Hilliard, M.R., *Greedy Genetics*, in Proceedings Second International Conference on Genetic Algorithms and their Applications, pp. 90-99, 1987.

[17]. Kanet, J.J., and Sridharan, V., PROGENITOR: *a Genetic Algorithm for Production Scheduling*, Witschafstsinformaati*k*, 1991.

[18]. Nakano, R., and Yamada, T., *Conventional genetic algorithms for job-shop problems*, in Proceedings of the Fourth International Conference on Genetic Algorithms, Morgan Kaufmann Publishers, Sap Mateo, CA, pp. 477-479, 1991.

[19]. Salto, C., Alfonso, H., and Gallard, R., *Multiplicity and incest prevention in evolutionary algorithms to deal with the job shop problem*, in Proceedings of the Second ICSC Symposium on Engineering of Intelligent Systems, University of Paisley, Scotland, pp, 451-457, 2000.

[20]. Storer, R., Wy, S., and Vaccari, R., *New search spaces for sequencing problems with application to job shop scheduling*, Management Science, vol. 38, no. 10, pp. 1495-1510, 1992.

[21]. Syswerda, G., *Schedule Optimization Using Genetic Algorithms*, in Handbook of Genetic Algorithms, Van Nostrand Reinhold, New York, 1991.

[22]. Whitley, D., Starkweather, T., and Fuquay, D., *Scheduling Problems and Traveling Salesman: The Genetic Edge Recombination Operator*, in Proceedings Third International Conference on Genetic Algorithms and their Applications, pp. 133-140, 1989.