CONTRASTING TWO MCMP ALTERNATIVES IN EVOLUTIONARY ALGORITHMS TO SOLVE THE JOB SHOP SCHEDULING PROBLEM

Stark N., Salto C., Alfonso H.
Laboratorio de Investigación en Sistemas Inteligentes (LISI)[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: { nstark,saltoc,alfonsoh }@ing.unlpam.edu.ar
Phone: (02302)422780/422372, Ext. 6302

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

Many researchers have shown interest to solve the job shop scheduling problem (JSSP) applying evolutionary algorithms (EAs). In a previous work we reported an enhanced evolutionary algorithm, which uses a multiplicity feature to solve JSSP. The evolutionary approach was enhanced by means of multiple crossovers on multiple parents (MCMP) and the selection of a stud among the intervening parent. Partially mapped crossover (PMX) was used on each multiple crossover operation and job based representation (permutation of jobs) was adopted as a coding technique.

The traditional MCMP approach is based on scanning crossover. But the application of this operator to permutations will yield illegal offspring in the sense that some jobs may be missed while some other jobs may be duplicated in the offspring, so some modifications to their mechanism are necessary to guarantee the offspring legality.

This paper contrasts both MCMP approaches, discusses implementation details and shows results for a set of job shop scheduling instances of distinct complexity.

Keywords: multirecombination, optimization, breeding, scanning crossover, scheduling.

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

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

1. INTRODUCTION

Evolutionary algorithms mimic the natural evolution on an abstract level. This stochastic search technique explores different regions of the search space simultaneously. Since the publications of Goldberg [13] and Davis [2], genetic algorithms are more and more used to solve scheduling problems. A job-shop scheduling problem consists of *n* jobs, or items, to be processed by *m* machines [14]. For each job is specified both a set of constraints on the order in which machines can be used and a given processing time on each machine. Our JSSP consist in finding the sequence of jobs on each machine in order to minimize the elapsed time needed to finish processing all jobs. This is the definition of deterministic job-shop scheduling problem, where all processing times are known exactly and there are no restrictions on when jobs may start.

A very important issue in building an evolutionary algorithm for the job-shop problem is to devise an appropriate representation of solutions together with problem-specific genetic operations so that all chromosomes generated in either the initial phase or the evolutionary process will produce feasible schedules. This is a crucial phase that affects all the subsequent steps of evolutionary algorithms [1].The genetic operators, which control the stochastic search have to be designed very carefully with respect to the chosen representation. Exploration and exploitation of the structure of the search space should be balanced [15]. In direct coding, the representation on the genetic level directly refers to a point within the search space [1]. A classic example of this kind of representation for JSSP is job-based representation [11]. Here a schedule (a solution of JSSP) is encoded into a chromosome, and evolutionary algorithms are used to evolve those chromosomes to determine a better schedule.

In a job shop representation we deal with permutations, so adequate genetic operators, such as *partially-mapped crossover* (PMX) [12], *order crossover* (OX) [3] and *cycle crossover* (CX) [18] should be used. In EAs the common practice of crossover is to operate once on each mating pair after selection. Such procedure is known as the SCPC (Single Crossover Per Couple) approach. But in nature when the mating process is carried out, crossover is applied many times and the consequence is a multiple and variable number of offspring. Multiple crossovers per couple (MCPC) [7,8] is a novel crossover method. It was applied to optimize classic testing functions and some harder (non-linear, non-separable) functions. For each mating pair MCPC allows a variable number of children. From all the generated offspring, the best, a randomly selected or all of them can be chosen for insertion in the next generation. In those earlier works it was noticed that in some cases MCPC found better results than those provided by SCPC. Also a reduced running time resulted when the number of crossovers per couple increased, and best quality results were obtained allowing between 2 and 4 crossover per couple. Moreover, seeking for exploitation of a greater sample of the problem space, an extended multi-recombination can be applied to a set of more than two parents. In Eiben's *multiparent* (MP) approach [5,6], offspring creation is based on a larger sample from the search space and consequently larger diversity is supplied. This can help to avoid premature convergence. Eiben used different versions of *scanning crossover* (SX) which can be categorized as multi-parent genetic operators.

These previous ideas gave rise to *multiple crossovers on multiple parents* (MCMP) [9,10], 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). MCMP provides a means to exploit good features of more than two parents selected according to their fitness by repeatedly applying one of the *scanning crossover* (SX) variants [5]: a number n_1 of crossovers is applied on a number n_2 of selected parents. From the n_2

produced offspring a number n_3 of them are selected, according to some criterion, to be inserted in the next generation. This is the method used in all our experiments.

This novel approach was successfully applied to single and multicriteria optimization. Recently it was also applied to JSSP under other representations, such as: decoders, dispatching rule representation and operation based representation [17,19, 20, 21, 22]. It was shown that a greater number of crossovers for a given number of parents provide better results.

Due to the use of permutations in the chromosome representation and aiming to apply MCMP, the traditional SX methods need to be modified in order to produce feasible offspring. Details of the original SX methods and the modifications are discussed in next sections. Also another view of MCMP on job based representation is introduced here. Instead of using the traditional SX methods, we show how to introduce PMX to a scheme of multirecombination (breeding techniques).

This work is organized as follows. Section 2 introduces job-based representations and one of the special operators developed for it: PMX. Section 3 shows the different variants of multiparent genetic operator introduced by Eiben and its further modifications to be applicable to sequence problems. Breeding techniques via the idea of a stud individual is explained in section 4. Finally, section 5 explains details of implementation and analyses of the obtained results.

2. JOB-BASED REPRESENTATION AND OPERATORS

Job-Based Representation consists of a list of *n* jobs and a schedule is constructed according to the sequence of jobs. For a given sequence of jobs, the first operation of the job under treatment is allocated in the best available processing time for the corresponding machine the operation requires; and then the second operation is allocated, and so on, until all operations of the job are scheduled. The process is repeated for each job in the list in the appropriate sequence. Any permutation of jobs corresponds to a feasible schedule.

Using this kind of representation it is necessary to consider special crossover operators. The conventional ones (one-point or multipoint crossover) generally might produce illegal offspring in the sense that some jobs may be missed while some others may be duplicated in the offspring. There are some specific crossover operators designed to apply to a permutation such as: *partial-mapped crossover* (PMX), *order crossover* (OX), *cycle crossover* (CX), *order-based crossover*, *heuristic crossover*, and so on. PMX was proposed by Goldberg and Lingle [12] and can be viewed as an extension of two-point crossover for binary string to permutation representation. It uses a special repairing procedure to solve the illegitimacy caused by the simple two-point crossover. Thus the essentials of PMX are a simple two-point crossover plus a repairing procedure. Figure 1 explains the PMX procedure [11].

For mutation a simple *interchange* operator, which randomly exchanges chosen genes of the chromosome can be used.

PMX operation

Select two positions along the string uniformly at random. The sub-strings defined by the two positions are called the *mapping sections.*

Exchange two sub-strings between parents to produce a proto-children.

Determine the *mapping relationship* between two mapping sections.

Legalize offspring by using the mapping relationship.

Figure 1: PMX procedure

3. MULTI-PARENT GENETIC OPERATORS

Eiben in his multiparent approach [4] proposed different versions of multiparent genetic operators. These operators used two or more parents to generate children. The basic mechanism used in order to define multiparent operators is *gene scanning* (figure 2). All the scanning techniques described here generate a single child from *n* parents ($n \geq 2$).

Initially, three *scanning crossover* (SX) methods were proposed, 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. Each time a gene is selected this implies updating the parent's marker. The pseudo-code in figure 2 shows the scanning mechanism.

```
initialize parent markers
for child_marker = 1 to chromosome_length do
        update parent markers
        child.allele[child_marker] = choose from values indicated by the parent markers
endfor
```

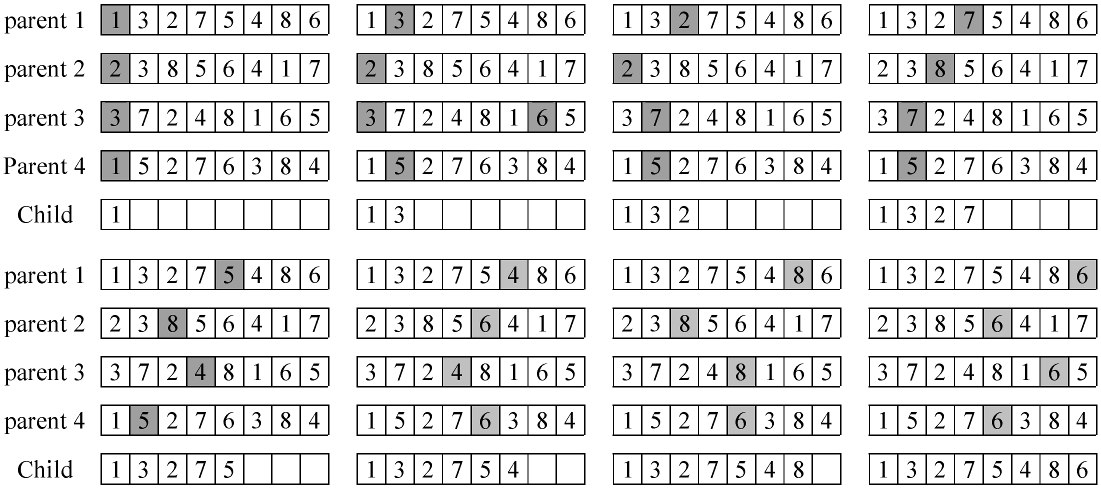
Figure 2: the general gene scanning mechanism.

The main characteristics of all gene-scanning procedures are the marker update mechanisms and the way to choose a marked gene from the parents. If the way in which the values to be inserted in the child are chosen is changed, then three specific scanning techniques can be defined, namely:

- *Uniform scanning crossover* (USX). This method is a natural extension of *uniform crossover*. Each parent has equal chance to be a donor for each gene in the child.
- *Occurrence-based scanning crossover* (OBSX). This method selects that gene value which occurs more frequently in a particular position of the parent's chromosomes. If none of the values are repeated then the gene pointed by the marker in the first parent is selected. Alternatively a random choice can be performed.
- *Fitness-based scanning crossover* (FBSX). This method chooses the gene value to inherit being proportional to the fitness value of the parents (roulette wheel restricted to selected parents subset).

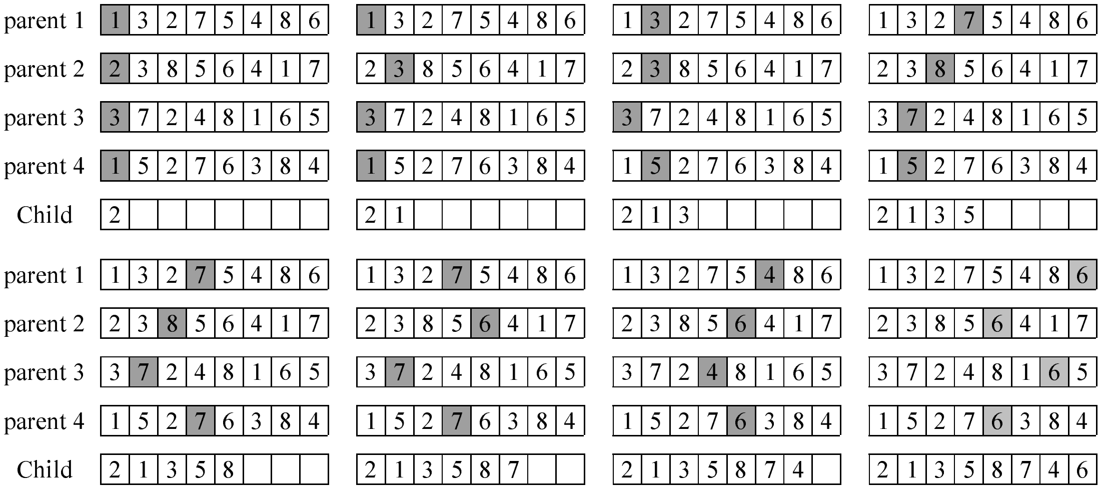
These scanning procedures might be defined for different representations types introducing chances to the marker update mechanism. This mechanism is very simple when representations with independent positions (no epistasis) are adopted; the parent markers are all initialized to the first position in each of the parents and traverses the parents from left to right by increasing it by one each step. Problems where epistasis occurs, as in our case, need a more sophisticated marker update mechanism to ensure that no value is added to the child twice. For each parent this means increasing its marker until it denotes a value that has not already been added to the child. Figure 3 shows a modification to the marker update mechanism using OBSX for permutations while figure 4 explains the mechanism using USX as a method to select the gene value to be inherited.

Another case of scanning crossover is adjacency-based crossover (ABX), which is specially designed for representations where relative positioning of values is important. Its principal difference with the other scanning procedures are the way in which the first value is selected and the marker update mechanism. The first gene value in the child is always inherited from the first gene value in the first parent. For each parent its marker is set to the first successor of the previously selected value, which does not already occur in the child.



parent 1 | 1 3 2 7 5 4 8 6 | 1 3 2 7 5 4 8 6 | 1 3 2 7 5 4 8 6 | 1 3 2 7 5 4 8 6
parent 2 | 2 3 8 5 6 4 1 7 | 2 3 8 5 6 4 1 7 | 2 3 8 5 6 4 1 7 | 2 3 8 5 6 4 1 7
parent 3 | 3 7 2 4 8 1 6 5 | 3 7 2 4 8 1 6 5 | 3 7 2 4 8 1 6 5 | 3 7 2 4 8 1 6 5
Parent 4 | 1 5 2 7 6 3 8 4 | 1 5 2 7 6 3 8 4 | 1 5 2 7 6 3 8 4 | 1 5 2 7 6 3 8 4
Child | 1 | 1 3 | 1 3 2 | 1 3 2 7
parent 1 | 1 3 2 7 5 4 8 6 | 1 3 2 7 5 4 8 6 | 1 3 2 7 5 4 8 6 | 1 3 2 7 5 4 8 6
parent 2 | 2 3 8 5 6 4 1 7 | 2 3 8 5 6 4 1 7 | 2 3 8 5 6 4 1 7 | 2 3 8 5 6 4 1 7
parent 3 | 3 7 2 4 8 1 6 5 | 3 7 2 4 8 1 6 5 | 3 7 2 4 8 1 6 5 | 3 7 2 4 8 1 6 5
parent 4 | 1 5 2 7 6 3 8 4 | 1 5 2 7 6 3 8 4 | 1 5 2 7 6 3 8 4 | 1 5 2 7 6 3 8 4
Child | 1 3 2 7 5 | 1 3 2 7 5 4 | 1 3 2 7 5 4 8 | 1 3 2 7 5 4 8 6

Figure 3: OBSX on job based representation



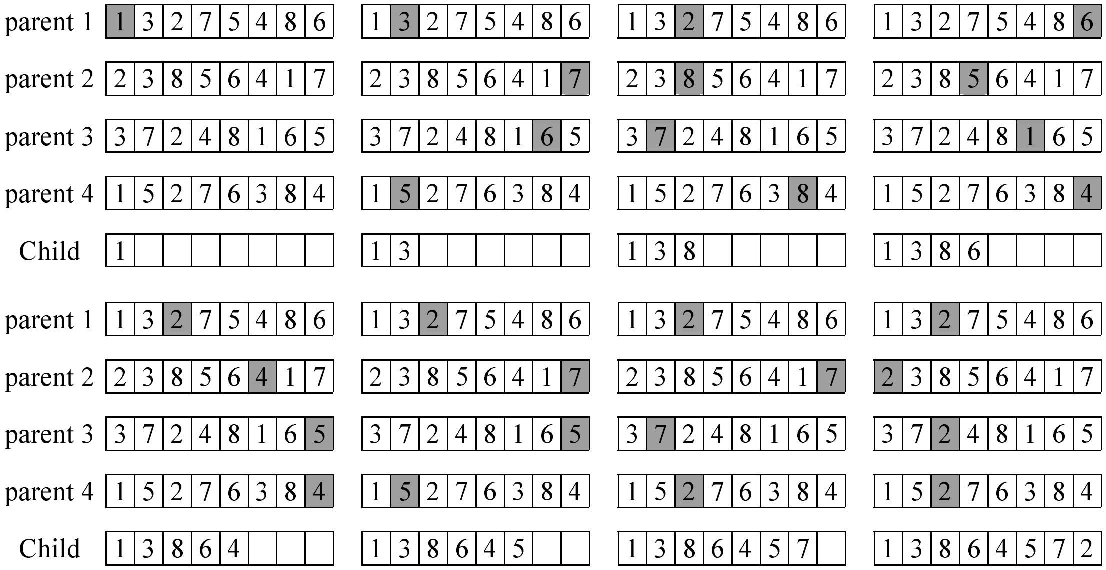
parent 1 | 1 3 2 7 5 4 8 6 | 1 3 2 7 5 4 8 6 | 1 3 2 7 5 4 8 6 | 1 3 2 7 5 4 8 6
parent 2 | 2 3 8 5 6 4 1 7 | 2 3 8 5 6 4 1 7 | 2 3 8 5 6 4 1 7 | 2 3 8 5 6 4 1 7
parent 3 | 3 7 2 4 8 1 6 5 | 3 7 2 4 8 1 6 5 | 3 7 2 4 8 1 6 5 | 3 7 2 4 8 1 6 5
parent 4 | 1 5 2 7 6 3 8 4 | 1 5 2 7 6 3 8 4 | 1 5 2 7 6 3 8 4 | 1 5 2 7 6 3 8 4
Child | 2 | 2 1 | 2 1 3 | 2 1 3 5
parent 1 | 1 3 2 7 5 4 8 6 | 1 3 2 7 5 4 8 6 | 1 3 2 7 5 4 8 6 | 1 3 2 7 5 4 8 6
parent 2 | 2 3 8 5 6 4 1 7 | 2 3 8 5 6 4 1 7 | 2 3 8 5 6 4 1 7 | 2 3 8 5 6 4 1 7
parent 3 | 3 7 2 4 8 1 6 5 | 3 7 2 4 8 1 6 5 | 3 7 2 4 8 1 6 5 | 3 7 2 4 8 1 6 5
parent 4 | 1 5 2 7 6 3 8 4 | 1 5 2 7 6 3 8 4 | 1 5 2 7 6 3 8 4 | 1 5 2 7 6 3 8 4
Child | 2 1 3 5 8 | 2 1 3 5 8 7 | 2 1 3 5 8 7 4 | 2 1 3 5 8 7 4 6

Figure 4: USX on job based representation

Two types of ABX has been defined, namely: occurrence based (OB-ABX) and fitness based (FB-ABX). This crossover uses the same mechanism for choosing the values to be inherited by the child as occurrence based and fitness based scanning, the difference resides in the marker update mechanism. Figure 5 shows OB-ABX procedure.

4. BREEDING TECHNIQUES BY MEANS OF THE STUD IDEA

Besides the possibility to introduce modifications in the scanning crossover procedure to work with permutations, in [21] another way of multi-recombination is considered. In the process for creating permutations, the PMX operator replaces the scanning method. From the parent population n_2 individuals are selected using proportional selection, the best one is called the stud for that group. As PMX proceeds, two crossover points are determined in a random way, the stud is combined with the rest of the parents selected using PMX considering those crossover points fixed. From that multi-recombination, 2*(n_2 -1) offspring are obtained, but only the best one survive. The preceding crossover process is repeated n_1 times determining other cut points each time. Finally n_1 offspring are obtained from n_2 parents and n_1 crossovers, and the best one is selected to pass to the next generation. The procedure is depicted in figure 6.



parent 1 1 3 2 7 5 4 8 6 1 3 2 7 5 4 8 6 1 3 2 7 5 4 8 6 1 3 2 7 5 4 8 6
parent 2 2 3 8 5 6 4 1 7 2 3 8 5 6 4 1 7 2 3 8 5 6 4 1 7 2 3 8 5 6 4 1 7
parent 3 3 7 2 4 8 1 6 5 3 7 2 4 8 1 6 5 3 7 2 4 8 1 6 5 3 7 2 4 8 1 6 5
parent 4 1 5 2 7 6 3 8 4 1 5 2 7 6 3 8 4 1 5 2 7 6 3 8 4 1 5 2 7 6 3 8 4
Child 1 1 3 1 3 8 1 3 8 6

parent 1 1 3 2 7 5 4 8 6 1 3 2 7 5 4 8 6 1 3 2 7 5 4 8 6 1 3 2 7 5 4 8 6
parent 2 2 3 8 5 6 4 1 7 2 3 8 5 6 4 1 7 2 3 8 5 6 4 1 7 2 3 8 5 6 4 1 7
parent 3 3 7 2 4 8 1 6 5 3 7 2 4 8 1 6 5 3 7 2 4 8 1 6 5 3 7 2 4 8 1 6 5
parent 4 1 5 2 7 6 3 8 4 1 5 2 7 6 3 8 4 1 5 2 7 6 3 8 4 1 5 2 7 6 3 8 4
Child 1 3 8 6 4 1 3 8 6 4 5 1 3 8 6 4 5 7 1 3 8 6 4 5 7 2

Figure 5. OB-ABX example.

```
Procedure: MCMP with PMX and the STUD
    Aux_child array[2*(n_2-1)]          //contain children generated by the crossover operator
    Children array[n_1]                 //contain the best child

    j=1;
    Select n_2 parents from P(t) and build mating_pool
    Sort mating_pool       //the best individual (stud) is in the first position
    while j ≤ n_1 do
    begin
      for i =2 to i = n_2 do
      PMX(mating_pool[1], mating_pool[i]) and  form aux_child
          //in each recombination, the children which preserves the stud substring are generated
      Evaluate aux_child
      Sort aux_child       // the best child is in the first position
      Children[j++]=aux_child[1];
    end while
  sort children
  children[1] is inserted in P(t+1)
end procedure
```

Figure 6. Basic structure of the Evolutionary Algorithm Including Multiplicity and Breeding.

5. EXPERIMENTS

Experiments consisted of varying parameters such as the number n_1 of crossover, the number n_2 of parents, and also the process for creating offspring.

We have decided to test the application of two alternative ways to carry out MCMP to solve the JSSP using permutations as chromosome representation. One of them consisted in the traditional application of MCMP using two versions of scanning crossover (uniform and occurrence based) together with other two versions of adjacency based crossover (uniform and occurrence based). And the other way consisted in the incorporation of breeding techniques by means of the stud idea.

We have selected four instances of different complexity of JSSP [16] with known optimal makespan to be optimized (table 1). We tested every n_1 crossover on n_2 parents versions for n_1 = 1,..,4 and n_2 = 3,.., 5 by having 50 runs and calculating performance averages for each (n_1,n_2) possible combination. In all cases the best child from the n_1 created children were inserted in the next generation. *Exchange* mutation, proportional selection and elitism to retain the best-valued individual were used. We used in the EA a 65% crossover rate and 10% mutation rate (per individual). The algorithms evolved for a minimum of 500 generations, after that a control of the mean fitness population progress began: if this value remained within a determined range for 20 consecutive generations then the evolutionary process was stopped. The population size was fixed at 100 individuals. All these values were determined as the best combination of probabilities after many initial trials.

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

Table 1. Testing 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 the best individual is 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 tells us how far the mean fitness is from that *opt_val*.
#Generation	Number of generations the EAs evolves before the fulfilment of the stop criteria.

For each series and parameter setting average values for the minimum and mean values of the performance variable were established. The following tables and figures summarize these results showing a comparison of different EAs using MCMP and also applying different ideas of multirecombination:

- ✓ MEA-Stud: Multirecombined EA using breeding techniques by means of the stud idea.
- ✓ MEA-XX: Multirecombined EA incorporating multiplicity feature in the traditional way, and using different crossover methods: OS stands for occurrence based scanning crossover, US means uniform scanning crossover, OA indicates occurrence adjacency based crossover and finally UA denotes uniform adjacency based crossover.

6. RESULTS

Analysing average minimum *Ebest* (figure 7), the lowest values are obtained by MEA-US independently of the analysed instance. For *la06*, in particular, average minimum *Ebest* is equal to 0%; in this case the optimum value is reached in all the possible (n_1, n_2) combinations at least once. The second best performer is MEA-Stud, which except for the *la12* instance, shows smaller errors than those obtained under MEA-UA, which is the third best performer. Considering the traditional

MCMP method and making a comparison among all different crossovers, MEA-UA presents good results, but with an inferior quality than MEA-US. For most instances, MEA-US presents smaller errors than MEA-OS; this observation is extensible to MEA-UA and MEA-OA. The two different kinds of occurrence based crossover reach poor results, but making a comparison between them occurrence adjacency based crossover shows in the majority of the instances smaller errors than its pair. Same remarks can be done for mean *Ebest* (figure 7).

Observing average minimum *Epop* values exhibited by the different MEA approaches (figure 8), MEA-Stud approach presents the best results. It is also noticeable a remarkable similarity between *Epop* and *Ebest* values indicating that individuals in the final population are very close to the best individual found so far.

	MEA-Stud		MEA-OS		MEA-OA		MEA-US		MEA-UA	
	Av. Min Ebest	Av. Mean Ebest	Av. Min Ebest	Av. Mean Ebest	Av. Min Ebest	Av. Mean Ebest	Av. Min Ebest	Av. Mean Ebest	Av. Min Ebest	Av. Mean Ebest
la01	5.1051	5.7518	5.2052	10.5703	5.1677	10.2085	5.1051	6.1659	5.1051	5.7895
la06	0.1260	1.0772	0.8909	5.0135	0.7289	4.7264	0.0000	0.9528	0.1890	1.2414
la12	2.4703	4.1585	4.2028	8.2899	4.2509	7.7356	1.5800	3.3466	2.2859	4.2624
la15	3.0931	4.7972	6.5175	9.6225	5.7995	9.4281	2.4234	4.3991	3.5556	5.7468

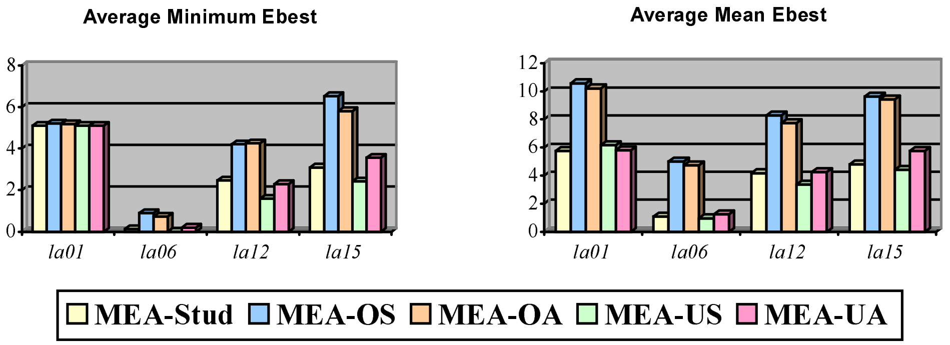


Figure 7: Average mean and minimum Ebest for all instances

	MEA-Stud		MEA-OS		MEA-OA		MEA-US		MEA-UA	
	Av. Min Epop	Av. Mean Epop	Av. Min Epop	Av. Mean Epop	Av. Min Epop	Av. Mean Epop	Av. Min Epop	Av. Mean Epop	Av. Min Epop	Av. Mean Epop
la01	5.1200	5.8046	5.5500	11.8617	6.1142	11.6652	6.1218	8.0846	5.4727	6.9946
la06	0.1379	1.0933	1.6085	6.1217	1.6415	6.0438	1.5048	3.9177	1.9708	3.9773
la12	2.5067	4.1672	4.8490	9.3459	4.8036	9.0742	4.003	7.3546	4.1433	7.5240
la15	3.0974	4.8052	7.2537	10.6986	6.3582	10.7622	4.6997	8.0839	6.2139	9.5971

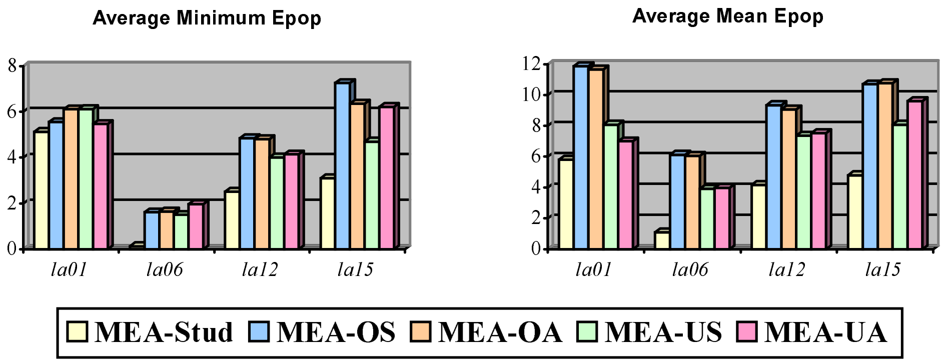


Figure 8: Average mean and minimum Epop for all instances

Figure 9 and table 2 shows ebest values obtained applying any of the different proposed MEA approaches for each possible (n_1, n_2) combination (horizontal axe) for *la06* instance. In table 2, bold-faced ebest values indicate the lowest *Ebest* values for each MEA approach. MEA-US achieves the optimum value in all possible combinations, this is concluded from an *Ebest* equal to 0%. Both MEA-Stud and MEA-UA reach in many opportunities the optimum value (8 and 7 times respectively). Regarding both occurrence based crossovers, MEA-OS and MEA-OA, they present higher ebest values, varying from 0% to 2.80% and 0% to 2.59%, respectively. Moreover, when the crossover operation is applied once on the selected parents, in most cases worst results (highest ebest values) are obtained than when this operation is repeated four times.

(n_1,n_2)	*Ebest*					*Epop*				
	MEA-Stud	**MEA-OS**	**MEA-OA**	**MEA-US**	**MEA-UA**	**MEA-Stud**	**MEA-OS**	**MEA-OA**	**MEA-US**	**MEA-UA**
(1,3)	**0.0000**	1.2959	2.5918	**0.0000**	0.5400	1.7126	11.6742	11.5900	21.9544	24.1468
(2,3)	0.1080	1.0799	0.9719	**0.0000**	**0.0000**	0.6677	8.6366	8.3887	10.6515	9.8045
(3,3)	**0.0000**	1.2959	0.2160	**0.0000**	**0.0000**	1.0366	7.4076	7.6830	3.5345	1.7720
(4,3)	**0.0000**	**0.0000**	0.7559	**0.0000**	**0.0000**	0.8257	7.6705	7.6052	2.6518	1.7457
(1,4)	**0.0000**	0.7559	0.8639	**0.0000**	0.4320	0.7257	4.9330	4.6890	1.4989	1.9438
(2,4)	0.5400	0.3240	1.1879	**0.0000**	**0.0000**	1.1015	4.2894	4.4104	0.3499	0.9114
(3,4)	**0.0000**	0.3240	**0.0000**	**0.0000**	**0.0000**	1.0626	3.8639	3.6048	1.0583	0.7883
(4,4)	0.6479	0.6479	0.2160	**0.0000**	0.3240	0.8294	4.0518	3.7559	1.5551	0.9395
(1,5)	**0.0000**	2.8078	1.1879	**0.0000**	**0.0000**	1.6328	6.3737	5.3715	0.5032	1.9914
(2,5)	**0.0000**	1.1879	**0.0000**	**0.0000**	0.5400	1.5680	5.1965	5.6242	0.3672	1.3888
(3,5)	**0.0000**	0.6479	0.2160	**0.0000**	0.4320	1.4946	4.5983	4.9482	1.2052	1.5464
(4,5)	0.2160	0.3240	0.5400	**0.0000**	**0.0000**	0.4622	4.7646	4.8553	1.6825	0.7495

Table 2: Ebest and epop values for each (n_1, n_2) combination *la06* instance

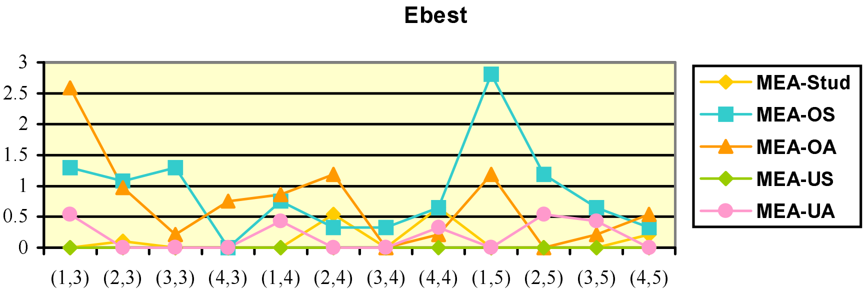


Figure 9: Ebest values for *la06* instance

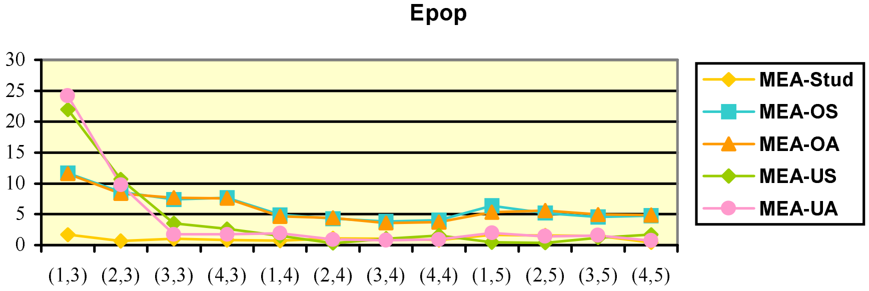


Figure 10: Epop values for *la06* instance

From figure 10 and table 2 we can see a fair similarity between *Epop* curves corresponding to MEA-OS and MEA-OA and between MEA-US and MEA-UA. MEA-Stud presents the smallest epop value for n_2=3, and as long as this value increases, *Epop* values for this approach compete with *Epop* values of MEA-UA and MEA-US. The best epop values are present when *n1* is equal to

4. Table 3 shows the number of generations the evolutionary algorithms needs to evolve until the stop criteria holds. When a single crossover is applied, the algorithms need to evolve during a greater number of generations and consequently the number of evaluations is huge, with the exception of MEA-Stud. In contrast, analysing ebest values, the best solution found is very poor in relation to the ones found applying more crossovers. When n_1 is greater than 1, in most cases the number of generation the algorithms evolve is the same, varying in this case the evaluation number, which increases as long as the number of parents and crossovers do.

(n_1,n_2)	**MEA-Stud**	**MEA-OS**	**MEA-OA**	**MEA-US**	**MEA-UA**
(1,3)	520	867	3236	5000	528
(2,3)	520	520	520	3846	523
(3,3)	520	521	520	590	526
(4,3)	520	521	520	531	521
(1,4)	520	2041	4128	4022	526
(2,4)	520	520	520	3795	524
(3,4)	520	520	520	593	523
(4,4)	520	520	520	548	525
(1,5)	520	1204	3971	5000	531
(2,5)	521	525	524	3633	527
(3,5)	520	521	523	599	520
(4,5)	520	526	523	538	520

Table 3 : Number of evaluations and generations for la06 for each number of crossover

(n_1,n_2)	***Ebest***					***Epop***				
	MEA-Stud	**MEA-OS**	**MEA-OA**	**MEA-US**	**MEA-UA**	**MEA-Stud**	**MEA-OS**	**MEA-OA**	**MEA-US**	**MEA-UA**
(1,3)	3.7536	5.9673	3.8499	2.3099	4.6198	3.7536	13.7707	14.1261	25.4473	28.6735
(2,3)	1.9249	4.1386	5.2936	1.8287	2.2137	1.9494	11.5098	11.7512	18.5891	17.2475
(3,3)	1.3474	4.3311	4.1386	1.0587	1.2512	1.7429	11.5591	11.3806	11.4223	6.7031
(4,3)	2.4062	4.5236	4.8123	1.3474	1.2512	2.4232	11.5535	10.3007	6.4110	4.2007
(1,4)	2.6949	5.2936	3.5611	2.0212	4.4273	2.6949	8.7854	8.2483	4.0115	6.2579
(2,4)	3.0799	4.8123	4.6198	2.0212	1.8287	3.0799	7.1530	7.5592	3.2339	4.2079
(3,4)	3.2724	3.5611	4.1386	1.5399	**1.1550**	3.2724	7.5842	6.6679	2.7988	3.8152
(4,4)	**0.9625**	**1.5399**	**2.6949**	1.3474	1.8287	0.9625	6.9971	6.5448	3.4495	3.2512
(1,5)	3.4649	4.1386	2.9836	2.1174	2.9836	3.4649	9.6920	8.3619	4.0808	5.9288
(2,5)	2.5024	3.6574	5.5823	**0.6737**	2.4062	2.5024	8.4081	8.8316	2.8046	3.2262
(3,5)	2.5024	4.5236	5.1011	1.2512	1.8287	2.5024	7.3975	7.4649	2.8393	3.3417
(4,5)	1.7324	3.9461	4.2348	1.4437	1.6362	1.7324	7.7401	7.6535	3.1665	3.4341

Table 4: Ebest values for each (n_1, n_2) combination in *la12* instance

Table 4 shows *Ebest* and *Epop* values obtained applying any of the different proposed MEA approaches for each possible (n_1, n_2) combination for *la12* instance. In table 4, bold-faced *Ebest* values indicate lower values for each MEA approach. In general, MEA-US achieves lower *Ebest* values than the other approaches, MEA-US errors vary from 0.67% to 2.30% while for example MEA-Stud errors vary from 0.96% till 3.75%. Moreover, when the crossover operation is applied once on the selected parents, worst results (highest ebest values) than when this operation is repeated four times are obtained in the majority of the cases. Comparing ebest and epop values for MEA-Stud approach, we can see a strong similarity between them. This affinity gives us the idea that all the individuals belonging to the final population have the same makespan, but not necessarily the same permutation of jobs.

7. CONCLUSIONS

This contribution explained the behaviour of multirecombinative approaches (MCMP) used to solve the JSSP for a set of instances of distinct complexity using a permutation representation. Due to this particular representation, the conventional scanning crossover might produce infeasible offspring. So it is necessary to introduce some modification to the general mechanism to be useful with that kind of representation. Also a further combination of MCMP with breeding techniques by means of the stud in the parent subpopulation was shown. For this last approach PMX and exchange mutation were the genetic operators chosen. This two versions of evolutionary algorithm improved with mutirecombinative features were contrasted against. After a long series of experiments we can conclude that:

- MCMP-UX is the best performer followed by MCMP-Stud, the approach with breading techniques, while MCMP-UA is the third best performer.
- In general, when 4 crossovers are applied better results than when a single crossover is applied are obtained.
- The MCMP approach with breading techniques arrives to a final population which in many cases is centred on the fittest individual, providing a set of alternative schedules, which can help to deal with system dynamics.

Now looking only the scanning methods, we can conclude that when all the parents have the same chance to be donors (independently of the update marker mechanism) better results are obtained than when the gene values is selected according to the value occurrence.

The promising results on these instances encourage us to investigation. To improve results in larger instances, further work will include self-adaptation of parameters such as (n_1, n_2) associations and hybridization of global and local search.

8. REFERENCES

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

[2] Davis, L., Handbook of Genetic Algorithms, Van Nostrand Reinhold, New York, 1991.

[3] Davis, L., Applying adaptive algorithms to domoins. In proceeding of the International Joint Conference on Artificial Intelligence, pp. 162-164, 1985.

[4] 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.

[5] Eiben A.E., van Kemenade CHM, and Kok J.N., Orgy in the computer: multi-parent reproduction in genetic algorithms, in Moran F., Moreno A., Merlo J.J., and Chacon P., editors, Proceedings of the 3rd European conference on Artificial Life, number 929 in LNAI, pages 934-945. Springer Verlag, 1995.

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

[7] Esquivel S., Leiva A., Gallard R., Multiple Crossover per Couple in Genetic Algorithms, in *Proceedings of the 4th IEEE International Conference on Evolutionary Computation* (ICEC'97), pages 103-106, Indianapolis, USA, April 1997.

[8] Esquivel S., Leiva A., 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, pages 235-241, ed. E.Alpaydin. Publishesd by ICSC Academic Press, Canada/Switzerland, February 1998.

[9] Esquivel, S., Leiva, A. And Gallard, R., Multiple Crossovers between Multiple Parents to Improve Search in Evolutionary Algorithms, en *Proceedings of the 1999 Congress on Evolutionary Computation (IEEE)*, vol. 2, pages 1589-1594, Washington DC., 1999.

[10] Esquivel, S., Leiva, A. and Gallard, R., Multiplicity in Genetic Algorithms to Face Multicriteria Optimization, en *Proceedings of the 1999 Congress on Evolutionary Computation*, IEEE Service Center, pages 85-90, Washington, D.C., 1999.

[11] Gen, M. y Cheng, R., *Genetic Algorithms and Engineering Design*, Wiley-Interscience Publication John Wiley & Sons, Inc, 1997.

[12] Goldberg, D. y Lingle, R., Alleles, Loci and the Traveling Salesman Problem, en Grefenstette, J. (editor), *Proceedings of the First International Conference on Genetic Algorithms,* Lawrence Erlbaum Associates, pages 154-159, Hillsdale, NJ, 1985.

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

[14] Husbands, P., Genetic algorithms for Scheduling, AISB Quarterly, No. 89, 1994.

[15] Kobayashi, S., Ono, L. And Yamamura, M., An efficient genetic algorithm for job shop schedulin problems, Proc. Of the 6th Conf. On Genetic Algorithms (ICGA'95), pp. 506-5011, 1995.

[16] 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.

[17] Minetti G., Salto, C., Alfonso, H., and Gallard, R., A Comparison of two Multirecombinated Evolutionary Algorithms for the Job Shop Scheduling Problem, in *Proceedings VI Congreso Argentino de Ciencias de la Computación*, CACIC 2000 Universidad Nacional de la Patagonia San Juan Bosco, 2000.

[18] Oliver I.M., Smith D.J., Holland J.R.C. A Study of Permutation Crossover Operators on the Travelling Salesman Problem , Proceedings of the 2nd Int. Conf. On Genetic Algorithms, pp. 224-230, 1987.

[19] Salto, C., Alfonso, H., and Gallard, R., Multiplicity and Incest Prevention in Evolutionary Algorithms to Deal with de Job Shop Problem, in *Proceedings of the Second ICSC Symposium on Engineering of Intelligent Systems*, University of Paisley, Scotland, pag, 451-457, 2000.

[20] Salto, C., Minetti G., Alfonso, H., and Gallard, R., A Hybrid Evolutionary Algorithm: Multirecombination with Priority Rule Base Representation for the Job Shop Scheduling Problem, in *Proceedings VI Congreso Argentino de Ciencias de la Computación*, CACIC 2000, Universidad Nacional de la Patagonia San Juan Bosco, pp. 1365-1376, 2000.

[21] Salto, C., Alfonso, H., and Gallard, R., Breeding in multirecombinated evolutionary algorithms with permutation based representation for the job shop scheduling problem, Proceedings of 4th International ICSC Symposium on soft Computing and Intelligent Systems for Industry (SOCO/ISFI2001), pp. 130, 2001.

[22] Salto, C., Alfonso, H., and Gallard, R., Performance Evaluation of Evolutionary Algorithms under Different Representations when Solving the JSS problem, Proceedings of 5th World Multi-Conference on Systemics, Cybernetics and Informatics (SCI 2001), pp. 424-428, 2001.