MULTIRECOMBINATED EVOLUTIONARY ALGORITHMS TO SOLVE MULTIOBJECTIVE JOB SHOP SCHEDULING*

ESQUIVEL S. C., FERRERO S.W., GALLARD R.H.

Proyecto UNSL-338403[1]
Departamento de Informática
Universidad Nacional de San Luis (UNSL)
Ejército de los Andes 950 - Local 106
5700 - San Luis, Argentina.
E-mail: {esquivel, swf, rgallard}@unsl.edu.ar
Phone: + 54 652 20823
Fax : +54 652 30224

Abstract

Multiobjective optimization, also known as vector-valued criteria or multicriteria optimization, have long been used in many application areas where a problem involves multiple objectives, often conflicting, to be met or optimized. Scheduling problems is one of such application areas whose importance lays on its economical impact and its complexity.
The present paper proposes CPS-MCPC, a cooperative population search method with multiple crossovers per couple. The cooperative search CPS is implemented with individuals of a single population, which are selected for recombination using alternatively each criterion. MCPC a multirecombination approach is used to exploit good features of both selected parents. To test the potentials of the novel method for building the Pareto front regular and non-regular objectives functions were chosen: the makespan and the mean absolute deviation of job completion times from a common due date (an earliness/tardiness related problem). The set of experiments conducted, used three basic representation schemes and contrasted results of the proposed approach against conventional methods of recombination. Details of implementation and results are discussed.

Keywords: Evolutionary Computation, Job shop scheduling, multiobjective optimization, multirecombination.

---

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

1 Introduction

In a multiobjective optimization problem, a solution has a number of objective values, one per each optimizing criteria (attributes). As many of these criteria can be in conflict it is impossible to optimize any of the objective functions without degrading at least one of the remaining criteria. When *m* objectives are involved, the search space can be seen as an *m*-dimensional space and therefore each solution is an *m*-vector of attribute components. This leads to a decision making problem for choosing a suitable solution (or set of solutions) according to higher level organization goals [24].

Vilfredo Pareto [26] established that there exists a partial ordering in the searching space of a multiobjective problem. The Pareto criterion simply states that a solution is better than another one if it is so good in all attributes, and better in at least one of these attributes. For instance, in a maximization problem given two solutions $x = (x_1, x_2, ..., x_n)$ and $y = (y_1, y_2, ..., y_n)$, the Pareto criterion says that, x dominates y iff $x_i \geq y_i \; \forall i$ and $\exists \, j$ such that $x_j > y_j$.

In the problem space some solutions will not be dominated by any other solution and they conform the *Pareto front*, also known as the *acceptable set*, the *efficient points* and the *Pareto optimal set*. Knowledge of the Pareto front is of utmost importance when search is applied before decision making. This information provides to the judgement of a human decision maker with the trade-offs to establish interactions between different criteria, hence simplifying the decision process to choose an acceptable range of solutions for a multicriteria problem. Implemented first by Schaffer [28], [29], Fourman [16] and then by Kursawe, [21], [22] and others, *cooperative population searches (CPS) with criterion selection* [19] was used to build the Pareto front in selected multicriteria problems. The central idea in CPS, is to make a parallel single criterion search, where all members of the population of an evolutionary algorithm are involved in a cooperative search to build the Pareto front.

A job shop can be seen as a multi-operation model where jobs follow fixed routes, but not necessarily the same for each job. Here there exist facilities, which produce goods according to specified production plans under several domain-dependent constraints. Job Shop Scheduling (JSS) attempts to provide optimal schedules. Common variables to optimize are total completion time (makespan), mean flowtime, mean lateness, percentage of tardy jobs, etc. All of them are regular performance measures in the sense that they are non-decreasing in completion times. When both earliness and tardiness are penalized [1] this gives rise to non-regular performance measures of current interest in just-in-time production.

Due to their implicit parallel search, evolutionary algorithms (EAs) are suitably fitted to deal with JSSP [35], [25] [14] as well as seeking solutions in multiobjective optimization [5], [7], [9], [11], [15], [31], [33], [34].

The present work investigates the ability of the CPS-MCPC method, a cooperative population search approach with multirecombination allowing multiple crossovers per couple, to find non-dominated points and contrasts its performance against conventional recombination when building the Pareto front under different representations.

2 The job shop scheduling problem

The job scheduling problem (JSSP) is related to the allocation of limited resources (machines) to jobs over time. Complexity of scheduling problems [17] and their economical impact motivated extensive research [2], [3], [4], [20], [30], [32]. This is a decision making process that has as a goal the opti-

mization of one or more objectives. The model considered here assumes that the system consists of a number of different machines and only one job may execute on a machine at a time. All schedules and jobs are non-preemptive. Jobs can have distinct priorities and all of them are available at production initiating time. Each job visits all machines, only once, following a predetermined sequence of machines, called a *route*. Consequently, a job can be represented by a vector where its components are the successive operations to be performed. These components are 2-tuples of the form (machine, duration), specifying the machine where the job must be allocated and the time spent in that machine. An instance of the JSSP is a matrix where the rows specify the jobs as above described. This matrix is called the *instance matrix* for the specific JSSP. Figure 1 shows the instance matrix for a given JSSP, with three jobs and two machines.

	O_1	O_2
J_1	(2,4)	(1,2)
J_2	(1,3)	(2,8)
J_3	(2,7)	(1,3)

Fig. 1. An instance for a JSSP

2.1.Conflicting objectives

In this paper we selected f_1 as the *makespan*, and f_2 as the *mean absolute deviation* of job completion times from a common due date *d,* as the conflicting criteria to minimize. When minimizing function f_1, schedules tend to be shor-tened, usually implying high utilization of machines. When minimizing function f_2 earliness and tardiness are penalized at the same rate for all jobs and schedules are built so that *d* is in the middle of the job completion times, which usually derives in lower inventory costs.

Cheng, Gen, and Tsujimura [6] proposed a formulation for a better representation of precedence constraints in a JSSP with *n* jobs and *m* machines, minimizing the makespan, as follows. Given:

- C_{jk} and p_{jk} the completion time and the processing time of job *j* in machine *k*, respectively.
- a_{ihk} and x_{ijk}, binary indicator coefficients defined as follows:

$$a_{ihk} = \begin{cases} 1, \text{ if processing on machine } h \text{ precedes that on} \\ \quad \text{machine } k \text{ for job } i \\ 0, \text{ otherwise} \end{cases}$$

$$x_{ijk} = \begin{cases} 1, \text{ if job } i \text{ precedes job } j \text{ on machine } k \\ \\ 0, \text{ otherwise} \end{cases}$$

- M a large positive number.

Then the problem is formulated as shown below:

$$min \quad max_{1 \le k \le m} \{ max_{1 \le i \le n} \{ C_{ik} \} \} \qquad (1)$$

s.t.

$$C_{ik} - p_{ik} + M(1 - a_{ihk}) \ge C_{ik} \qquad (2)$$

$$C_{jk} - C_{ik} + M(1 - x_{ijk}) \ge p_{jk} \qquad (3)$$

$$a_{ihk} = 0 \text{ or } 1 \qquad (4)$$

$$x_{ijk} = 0 \text{ or } 1 \qquad (5)$$

$$i, j = 1,.., n \quad , \quad k, h = 1,..., m \qquad (6)$$

Operation precedence constraint (2) guarantees that the sequencing of operations for each job corresponds to the prescribed order. *Operation non-overlapping* constraint (3) ensures that each machine can process only one job at a time. Consequently, under the same set of constraints for the JSSP and given a due date *d*, common to all jobs, our multiobjective optimization problem can be formulated as follows:

Minimize $f_1(\sigma)$ and $f_2(\sigma)$ where sought solutions σ are feasible schedules and

$$f_1(\sigma) = max_{\,1\le k\le m}\{max_{\,1\le i\le n}\{C_{ik}\}\} \qquad (7)$$

$$f_2(\sigma) = \frac{1}{n}\sum_{j=1}^{n}\left|C_j - d\right| \qquad (8)$$

In expression (8) C_j stands for the completion time of the last operation of job *j*.

3 Representations and operators in evolutionary algorithms for the JSSP

It is a well-known problem in evolutionary computation the limitations a particular representation (encoding) of solutions imposes on the genetic operators to be used. This issue is mainly addressed to the creation of valid offspring avoiding the use of penalties or repair algorithms. In the following the representations used to evaluate the multirecombinative approach, and the corresponding operators are explained.

Priority list representation (PLR)

Under this representation associated with the instance matrix is a job priority list, which is used by the schedule builder at the building stage of a schedule to solve conflicts between jobs requiring the same resource. At each step, subjected to precedence and non-overlapping constraints, resources are allocated to those job operations which are not in conflict. When conflicts on a requested resource arise the allocation is done following the priority list. By using different priority lists different schedules can be built. As a priority list is a permutation of jobs, a chromosome is represented as a permutation of integer job identifiers.

Job-based representation (JBR)

Here also a chromosome consists of a list of *n* jobs. Following the sequence in the list, all operations of the first job are scheduled, then all the operations of the second job are considered, and so on, until all jobs are scheduled. Each operation of the job being scheduled are allocated in the best available processing time for the machine the operation requires.

Operation-based representation (OBR)

Here a schedule is encoded in the chromosome as a sequence of operations. Due to the existence of precedence constraints among operations of a particular job, the assignment of natural numbers to identify operations and the use of a permutation representation can lead to unfeasible schedules. To avoid this problem Gen, Tsujimura and Kubota [18] proposed a representation where each operation is identified by the job number to whom it belongs and the order of occurrence in the sequence. For an *n*-job *m*-machine problem a chromosome consists of *n* x *m* genes, where each gene have a job identifier as the allele value and values are repeated exactly *m* times in the chromosome. For the JSSP corresponding to the instance matrix of Fig. 1 and a given chromosome [3 2 1 2 1 3], allele values 1, 2 and 3 stand for jobs J_1, J_2 and J_3, respectively. Because each job has two operations these values appear twice in the chromosome. The first occurrence of a '1' refers to the first operation of job J_1, which should be allocated to machine 2, and the second occurrence refers to the second operation of job J_1, which should be allocated to machine 2.The same interpretation is given to other gene values.

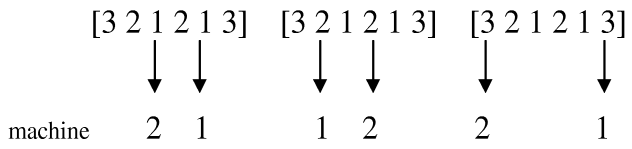

Fig. 2. Job operations and corresponding machines for the matrix instance of Fig 1.

Genetic operators

As PLR and JBR deal with permutations our experiments used *order crossover* (OX) and *exchange mutation* for both representations. In the case of OBR we propose a *modified order cross*over (MOX). Here, to build a valid offspring a sub-sequence of one parent is inserted in the same position in the offspring and the rest of allele values are copied from the second parent in the order they are appearing controlling the number of allele repetitions.

For example, consider a JSSP with $n = m = 3$. Given two parents and selecting from the first parent a sub-sequence including genes from the 4^{th} to the 7^{th} position :

parent1 [3 2 2 1 1 2 3 1 3] *parent2* [1 2 3 2 1 3 3 1 2]

Once the selected sub-sequence from the first parent was inserted in the offspring the remaining genes are extracted from the second parent, beginning from the position next to the last one in the sub-sequence, in the order they appear. The process for creating an offspring is delineated below.

parent2 [1 **2 3** 2 1 **3** 3 **1 2**]

Boldfaced genes in Parent2 are the ones to be inserted in the offspring.

[x x x 1 1 2 3 x x] Sub-sequence from *parent1* is inserted.

[x x x 1 1 2 3 **1 2**] The number of 1's is completed.

[**2** x x 1 1 2 3 **1 2**] The first '1' of *parent2* is skipped and the number of 2's is completed.

[**2 3** x 1 1 2 3 **1 2**] Another '3' is inserted.

[**2 3 3**1 1 2 3 **1 2**] The number of 3's is completed.

offspring [2 3 3 1 1 2 3 1 2]

For mutation a *modified exchange mutation* was implemented in order to ensure that the exchange effectively changes the allele values.

4 Multirecombination in multicriteria optimization (CPS-MCPC)

Conventional approaches to crossover, independently of the method being used, involve applying the operator only once on the selected parents. In this paper such a procedure will be known as the Single Crossover Per Couple (SCPC) approach.

In earlier works [12], [13], we devised a different approach: to allow multiple offspring per couple, as often happens in nature. In order to deeply explore the recombination possibilities of previously found solutions, we decided to conduct several experiments in which more than one crossover operation for each mating pair was allowed.

The number of children per couple was fixed or granted as a maximum number and the process of producing offspring was controlled, for each mating pair, in order not to exceed the population size.

The idea of multiple children per couple was tested on a set of well-known testing functions (De Jong functions F_1, F_2 and F_3 [8], Schaffer F_6 [27] and other functions). A simple genetic algorithm, with conventional operators and parameter values, was the basis of those initial experiments.

During the first studies of the MCPC approach it was observed that in many cases MCPC found better results than SCPC and best quality results were obtained allowing between 2 and 4 crossovers per couple. These effects were a consequence of a greater exploitation of the recombination of good, previously found, solutions.

For multiobjective optimization initial experiments with CPS-MCPC were implemented executing exactly n_1 crossovers, providing $2\ n_1$ children per couple ($2 \leq n_1 \leq 4$). Basically this novel approach:

1) Maintains a single population of solutions which is separately ranked by each criterion.
2) Uses ranking selection to select one parent per criterion.
3) Uses *multiple crossovers per couple* (MCPC), and the corresponding crossover and mutation operators to generate multiple offspring.
4) After each mating, for insertion in the next population, selects those offspring, which are classified so far, as *globally* non-dominated. If none fulfilling this condition exists then n_1 (half) of the newly ge-nerated offspring are inserted, selecting first those that are non-dominated within the new offspring subset and completing the n_1 insertions by random selection if necessary.

The last point above mentioned, implies to maintain the updated set of solutions found so far as belonging to the Pareto front. Let us call it $P_{current}$. This set is updated at the end of each generation cycle. To build the new population, each time the new offspring are created by application of MCPC, we apply the following procedure:

While the new population is created
do
  By using ranking selection select one parent per criterion,
  Apply MCPC with the corresponding crossover to obtain the set O of $2\ n_1$ offspring and mutate,
  By consulting $P_{current}$ determine the subset O_{nond} of O that are globally nondominated,
  If $O_{nond} \neq \Phi$ then insert O_{nond} into the new population
    else insert n_1 offspring selecting first those that are non-dominated in O.
      Complete n_1 insertions by random selection if necessary.
od

The number n_1 of crossovers is a parameter of the EA. Essentially the proposed CPS-MCPC,

- Augments implicit parallel search by encouraging crossbreeding among "species".
- Increases exploitation of good solutions previously found through multiple crossovers per couple.
- Favours for insertion in the next generation those solutions which are, at the present stage, non-dominated (globally, at $P_{current}$ level, or locally, at O subset level). If none is found then genetic diversity is favoured by random selection.

Consequently, it is expected a contribution of the method to speed the search and to find a larger set size when seeking the Pareto optimal set.

5 Experiments

To evaluate the performance of the CPS-MCPC, the problem of minimizing $f_1(\sigma)$ and $f_2(\sigma)$, as explained in 2.1 was used for experiments. Ten instances of two types, small and medium (size) from the Lawrence's benchmark set [23], with known optimal makespan were used. Small instances were of 10 jobs and 5 machines, identified as *laX* with *X*=00,..,05, while medium instances were of 20 jobs and 10 machines identified as *laY* with *Y*=26,..,30.

After many initial trials best parameter settings were determined as follows. Number of crossover per couple n_1 = 4. Probabilities for crossover and mutation were fixed at 0.7 and 0.05 respectively. Population size fixed at 100 and 20 individuals, and maximum number of generations fixed at 1000 and 5000 for small and medium instances, respectively. To establish the raw potential of the method we use neither insertion of 'seeds' (good individuals provided by other conventional heuristics) within the initial randomized population nor any hybrid approach during the evolutionary process. Elitism was used to retain the best individual found so far under each criterion. As optimal values of makespan were known for each instance of the test suite, the common due date *d* to determine $f_2(\sigma)$ values was fixed at a value 40% greater than the corresponding optimal makespan. Next section show comparative results of the proposed CPS-MCPC and the conventional CPS method, applying a single crossover per couple, which from now on will be called CPS-SCPC.

6 Results

All the above mentioned Lawrence's instances were tested for each representation and corresponding genetic operators under CPS-MCPC and CPS-SCPC. In general, from the representation point of view OBR outperformed both other coding techniques, and PLR was better than JBR. This behaviour shown in Fig. 3 for instance la04, was expected because OBR is a more problem-specific representation while PLR and JBR coding spaces correspond to only a part of the whole solution space. For the discussion of the compared performance of both CPS methods we will show only results for la02 and la30 as demonstrative instances for each type, because the remaining instances reveal similar findings

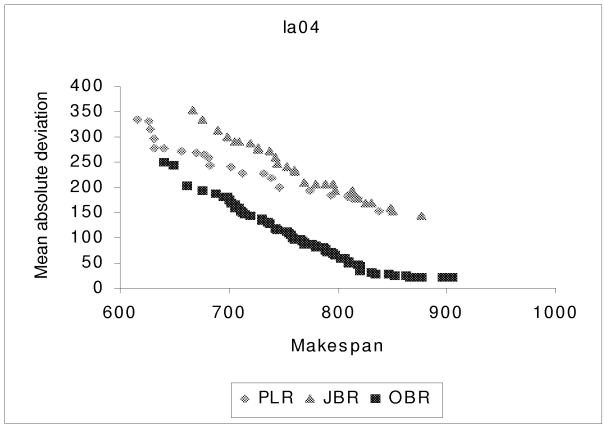

Fig. 3. Representation approaches: comparative performance for building the Pareto front under CPS-MCPC.

Figures 4 to 6 show the Pareto fronts built under both recombination methods with the three chosen representations for small instance la02. In Fig. 4 with PLR, 48 non-dominated solutions were found under both recombination schemes. Also, the quality of solutions is similar. In Fig. 5 with JBR, 15 and 19 non-dominated points were found under CPS-SCPC and CPS-MCPC, respectively. The multirecombination approach shows better quality of results than the conventional single crossover approach. Finally in Fig. 6 with OBR, 34 and 91 non-dominated points were found under CPS-SCPC and CPS-MCPC, respectively. A better Pareto front is achieved here also under CPS-MCPC.

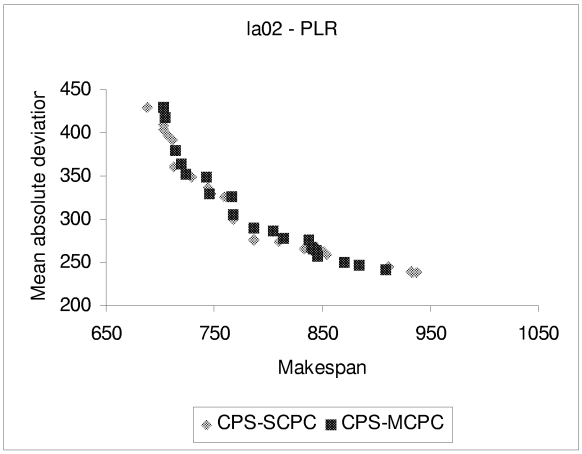

Fig. 4. Pareto fronts built under CPS-MCPC and CPS-SCPC with priority list representation, instance la02.

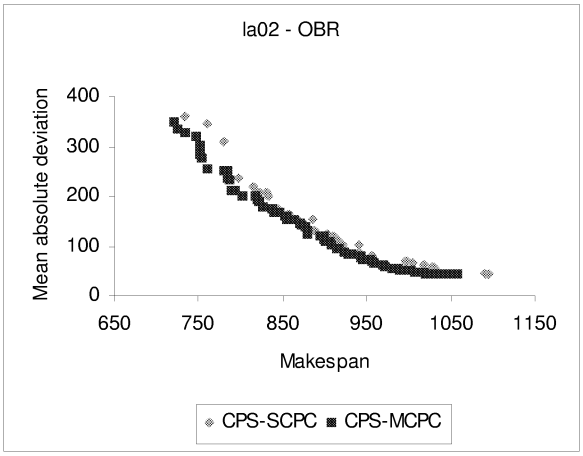

Fig. 6. Pareto fronts built under CPS-MCPC and CPS-SCPC with operation-based representation, instance la02.

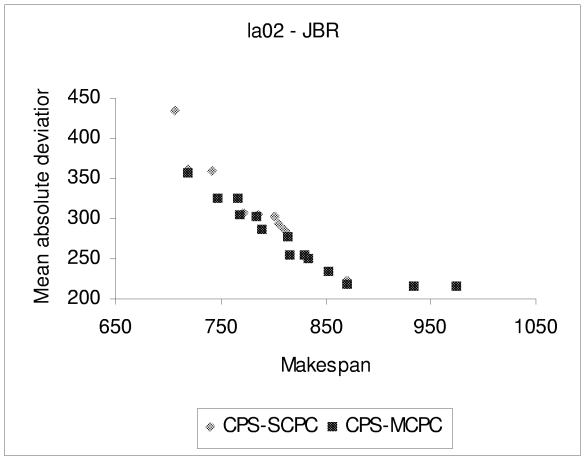

Fig. 5. Pareto fronts built under CPS-MCPC and CPS-SCPC with job-based representation, instance la02.

Figures 7 to 9 shows the Pareto fronts obtained for big instances. All of them clearly show better set of efficient points under CPS-MCPC for any representation. These figures show 23 and 58, 19 and 38, and 25 and 44 non-dominated solutions found under CPS-SCPC and CPS-MCPC, respectively for the corresponding coding techniques.

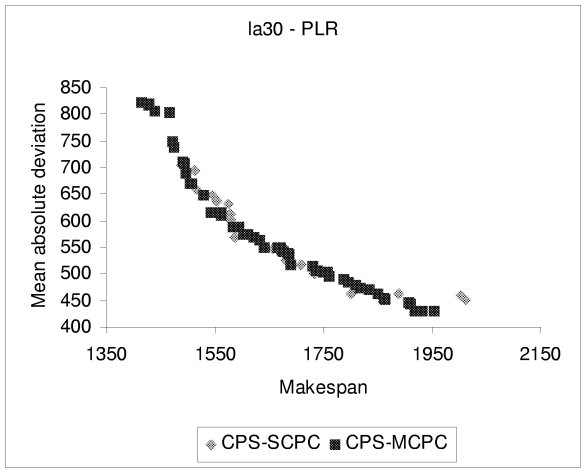

Fig. 7. Pareto fronts built under CPS-MCPC and CPS-SCPC with priority list representation, instance la30.

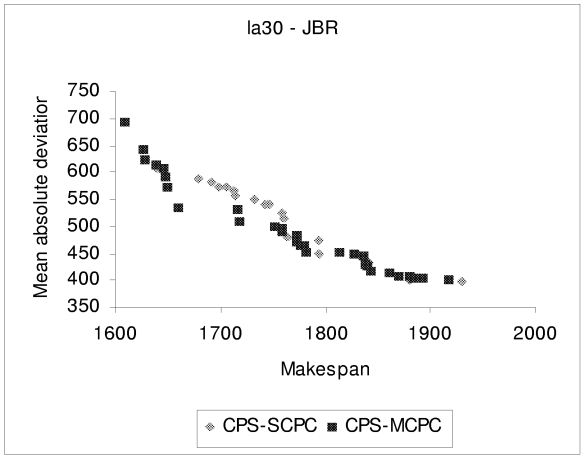

Fig. 8. Pareto fronts built under CPS-MCPC and CPS-SCPC with job-based representation, instance la30.

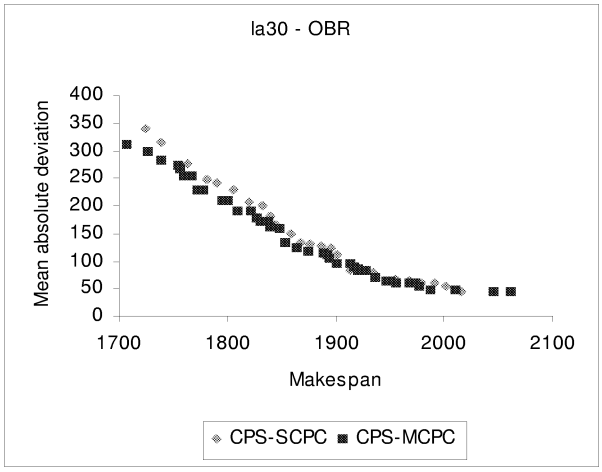

Fig. 9. Pareto fronts built under CPS-MCPC and CPS-SCPC with operation-based representation, instance la30.

Other important evidence arises when observing at the final population attained by either recombination method. Fig. 10, shows a total of 321 points, where 125 of them are non-dominated and belong to the final $P_{current}$ sets of the co-rresponding recombination methods.

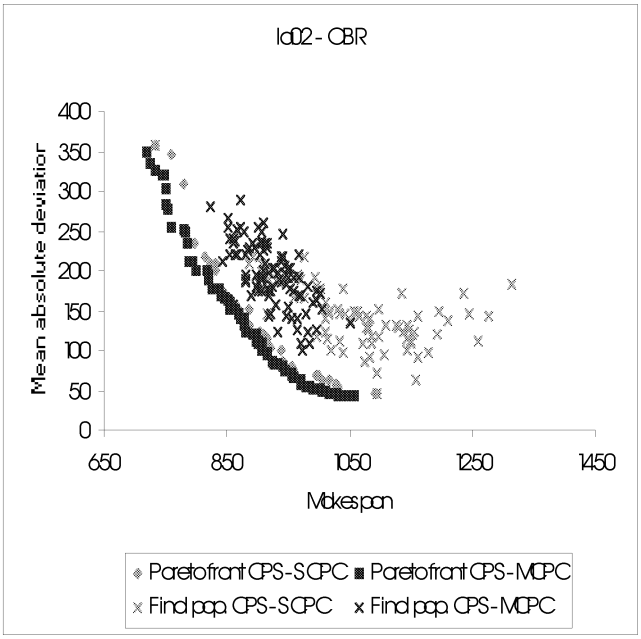

Fig. 10. Pareto fronts and final populations obtained under CPS-MCPC and CPS-SCPC with operation-based

Under CPS-SCPC the final population show a higher diversity than under CPS-MCPC. Average individuals in both populations have the following objective values,

$(\bar{f}_1(\sigma_{CPS-SCPC}), \bar{f}_2(\sigma_{CPS-SCPC}))$ = (1073,138)
$(\bar{f}_1(\sigma_{CPS-MCPC}), \bar{f}_2(\sigma_{CPS-MCPC}))$ = (922,194)

This means that the final population under CPS-MCPC is nearer of a compromise (mean) solution.

7 Conclusions

This work reports experience on multiobjective optimization applied to the Job Shop Scheduling problem using a co-operative population searches method with multirecombination. Experiments to contrast multiple versus single recombination were performed using three basic representations for the JSSP. Preliminary runs determined that varying the number n_1 of crossover from 2 to 4 better results were found for higher values of n_1. Consequently, exhaustive experiments were performed with $n_1 = 4$, for the set of selected instances.

Independently of the coding technique adopted, in most cases CPS-MCPC gives an indication of building better Pareto fronts. This was shown by the achievement of improved, more densely and evenly distributed fronts. More-over the final population obtained under the novel approach is grouped around compromise solutions. This fact shows that the alternative solutions provided by multirecombination attempt to balance the damage caused on the conflicting objectives of the multicriteria problem.

These preliminary results are promising and encourage us to deep forward investigation in multiobjective scheduling problems by using multirecombination with better representations for the JSSP. An open remaining question is the optimal setting of the number n_1 of crossovers, which could be self-adapted. Other variants such as multiplicity of parents and crossovers [10], [11] are under study to establish the abilities and possible limitations of this approach.

Acknowledgements

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

Bibliography

[1] Baker K., Scudder G., *Sequencing with earliness and tardiness penalties: A review*. Operations Research, vol. 38, pp 22-36, 1990.

[2] Bhanu B, Lee S., *Genetic learning for adaptive image segmentation*, (Boston MA: Kluwer). 1994.

[3] 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.

[4] Bruns R., *Direct chromosome representation and advanced genetic operators for production scheduling*. In Forrest ICGA93 , pages 352-359.

[5] Coello Coello C., *An updated survey of evolutionary multiobjective optimization*, Proceedings of the 1999 Congress on Evolutionary Computation (IEEE). Washington DC, pp 3-13.

[6] Chen R., Gen M., Tsujimura Y., *A tutorial survey of job shop scheduling problems using genetic algorithms: part I, representation*, International Journal of Computers and Industrial Engineering, vol. 30, no. 4, pp. 983-997, 1996.

[7] Deb K., *Multi-objective genetic algorithms: problem difficulties and construction of test problems*. Evolutionary Computation, 1999 by the MIT.

[8] De Jong K. A., *Analysis of the Behavior of a Class of Genetic Adaptive Systems*, PhD Dissertation, University of Michigan, 1975.

[9] Eiben A., Lis J., *A multisexual genetic algorithm for multiobjective optimization*, 4th IEEE International Conf. on Evolutionary Computation (ICEC'97), pp 59-64, Indianapolis, USA, April 1997.

[10] Eiben A., Raue P.-E., Ruttkay Zs., *Genetic Algorithm with multiparent recombination*, Proc of Parallel Solving from Nature Nr 3, LNCS 866, Springer Verlag, pp 78-87, 1994.

[11] Esquivel S., Leiva H.,.Gallard R., *Multiplicity in genetic algorithms to face multicriteria optimization*, Proceedings of the 1999 Congress on Evolutionary Computation (IEEE). Washington DC, pp 85-90.

[12]Esquivel S., Gallard R., Michalewicz Z., *MCPC: Another Approach to Crossover in Genetic Algorithms*, Proceeding of Primer Congreso Argentino de Ciencias de la Computación, pp 141 - 150, 1995.

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

[14] Fang H.-L., Ross P., and D. Corne. *A promising genetic algorithm approach to job-shop scheduling, rescheduling, and open-shop scheduling problems*. ICGA93 , pages 375-382. Proceedings of the Fifth International Conference on Genetic Algorithms . Morgan Kaufmann Publishers, San Mateo, CA, 1993.

[15] Fonseca C. M., Fleming P. J. *Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization* Proc. of the 5th In. Conf. on Genetic Algorithms, pp 416-423, Urbana- Champaign, IL, Morgan Kaufmann, 1993.

[16] Fourman M. P., *Compaction of symbolic layout using genetic algorithm*, Proc. of the 1st Int. Conf. on Genetic Algorithms, ed Grefenstette J.J pp 141-153, Pittsburgh, PA, 1985.

[17] Garey M. R. and D. S. Johnson. *Computers and Intractability . A Guide to the Theory of NP-Completeness*. Freemann & Co., San Francisco, CA, 1979.

[18] Gen M., Tsujimura Y., Kubota E., *Solving job-shop scheduling problem using genetic algorithms,* Proceedings of the 16th International Conference on Computers and Industrial Engineering, Ashikaga, Japan, 1994.

[19]Horn.J., *Multicriterion decision making*, Handbook of Evolutionary Computation. F1.9:1-9:15, Oxford University Press, 1997.]

[20] Husbands P., F. Mill, and S. Warrington. *Genetic algorithms, production plan optimisation, and scheduling*. In Schwefel and Maenner PPSN91 , pages 80-84.

[21] Kursawe F., *Evolutionststrategien für die vectoroptimierung*, diplomarbeit, universitat Dortmund, 1990

[22] Kursawe F., *A variant of evolution strategies for vector optimization*, Parallel Problem solving from Nature, Lectune notes in Computer Science 496 Berlin:Springer, pp 193-197. 1991.

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

[24] Leitmann G., Marzollo A., *Multicriteria Decision Making*, CISM No 211, Springer Verlag, Wien , NY, 1975.

[25] Nakano R. and Yamada T., *Conventional genetic algorithms for job shop problems*. Proceedings of the Fourth International Conference on Genetic Algorithms , pages 474-479. Morgan Kaufmann Publishers, San Mateo, CA, 1991.

[26] Pareto V., *Cours d'Economie Politique*, 1896, Switzerland, Lausanne: Rouge.

[27]Schaffer J., Caruana R., Eshelman R., Das R., *A Study of Control Parameters Affecting Online Performance of Genetic Algorithms for Function Optimization*, in Third International Conference on Genetic Algorithms, pages 51 - 60, 1989.

[28] Schaffer J. D., *Some experiments in machine learning using vector evaluated genetic algorithms*, Doctoral dissertation, Department of Electrical Engineering, Vanderbilt University. 1984.

[29] Schaffer J. D., *Multiple objective optimization with vector evaluated genetic algorithms*, Proc. of the 1st Int. Conf. on Genetic Algorithms, ed Grefenstette J.J pp 93-100, Pittsburgh, PA, 1985.

[30] Shaw K. J., Northcliffe A. L., Thompson M., Love J., Fleming P.J., Fonseca C. M, *Assessing thge performance of multiobjective Genetic Algorithms for Optimization of a batch process scheduling problem*, Proceedings of the 1999 Congress on Evolutionary Computation (IEEE). Washington DC, pp 37-45.

[31] Srinivas N., Deb K., *Multiobjective optimization using nondominated sorting in genetic algorithms*, Evolutionary Compt. 2, pp 221-248, 1995

[32] Syswerda G.. *Schedule optimization using genetic algorithms*. In L. Davis, editor, Handbook of Genetic Algorithms , chapter 21, pages 332-349. Van Nostrand Reinhold, New York, 1991.

[33] Tamaki H., Nishino E., Abe S., *A genetic algorithm approach to multiobjective scheduling problems with earliness and tardiness penalties*, Proceedings of the 1999 Congress on Evolutionary Computation (IEEE). Washington DC, pp 46-52.

[34] Vemuri R., Cedeño W., *A new genetic algorithm for multiobjective optimization in water resource management*, Proc. of the 1st IEEE International Conf. on Evolutionary Computation (ICEC'94), pp 495-500, Orlando, USA, 1994.

[35] Yamada T. and Nakano R.. A genetic algorithm applicable to large-scale job-shop problems. In R. Maenner and B. Manderick, editors, Parallel Problem Solving from Nature 2 , pages 281-290. Elsevier, Amsterdam, 1992.