ELSEVIER

Knowledge-Based Systems 15 (2002) 13–25

Knowledge-Based SYSTEMS

www.elsevier.com/locate/knosys

Enhanced evolutionary algorithms for single and multiobjective optimization in the job shop scheduling problem

S. Esquivel[a,*], S. Ferrero[a], R. Gallard[a], C. Salto[b], H. Alfonso[b], M. Schütz[c]

[a]*Laboratorio de Investigación y Desarrollo en Inteligencia Computacional (LIDIC), Universidad Nacional de San Luis, Ejército de los Andes 950, Local 106, 5700 San Luis, Argentina*

[b]*Proyecto UNLPAM-09/F009, Departamento de Informática, Universidad Nacional de La Pampa, Calle 110 esq. 9, Local 14, 6360, General Pico, La Pampa, Argentina*

[c]*Center for Applied Systems Analysis (CASA), Informatik Centrum Dortmund (ICD), Joseph-von-Fraunhofer, Str 20, D-44227 Dortmund, Germany*

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 can pose extremely complex combinatorial optimization problems, which belong to the NP-hard family. Last enhancements on evolutionary algorithms include new multirecombinative approaches. *Multiple Crossovers Per Couple* (MCPC) allows multiple crossovers on the couple selected for mating and *Multiple Crossovers on Multiple Parents* (MCMP) do this but on a set of more than two parents. Techniques for preventing incest also help to avoid premature convergence. Issues on representation and operators influence efficiency and efficacy of the algorithm. The present paper shows how enhanced evolutionary approaches, can solve the Job Shop Scheduling Problem (JSSP) in single and multiobjective optimization.

Keywords: Evolutionary algorithms; Multiplicity of crosoovers and parents; Scheduling; Single and multiobjective optimization

1. Introduction

Due to its complexity [1] and reflecting the industrial relevance of this application domain, a variety of evolutionary schedulers based on genetic algorithms have been reported in the literature in the past [2–11]. In general, the task of scheduling is the allocation of jobs over time when limited resources are available, where a number of objectives should be optimized, and several constraints must be satisfied. A job is determined by a predefined set of operations, and the result of a scheduling algorithm is a schedule that contains the start times and allocation of resources to each operation [12]. Improvements in evolutionary algorithms (EAs) have been recently found by using a multiplicity feature, which allows multiple recombination on a couple of parents or on multiple parents [13–16]. The method was successfully applied to multimodal optimization problems. As a consequence of this approach, it was detected that all individuals of the final population are much more centered on the optimum. This is an important issue when the application requires provision of multiple alternative near-optimal solutions confronting system dynamics as in production planning.

The idea of *incest prevention*, was initially proposed by Eshelman and Shaffer [17] and it showed its benefits to avoid premature convergence. The method avoided mating of pairs showing similarities based on the parents' Hamming distance. Incest prevention was extended in an earlier work by maintaining information about ancestors within the chromosome and modifying the selection for reproduction in order to prevent mating of individuals belonging to the same 'family', for a predefined number of generations. This novel approach was also tested on a set of multimodal functions. Description of experiments and analyses of improved results can be seen in Ref. [18]. Current trends in chromosome representation are problem-dependent and genetic operators are closely related to representation.

In scheduling the quality of a schedule is measured by means of an objective function, which assigns a numerical value to a schedule. In our case, for single objective optimization the completion time of the last job abandoning the system (makespan) is optimized. For multiobjective optimization, an aggregative approach with three objectives

* Corresponding author.

E-mail addresses: esquivel@unsl.edu (S. Esquivel), swf@unsl.edu (S. Ferrero), rgallard@unsl.edu (R. Gallard), saltoc@ing.unlpam.edu.ar (C. Salto), alfonsoh@ing.unlpam.edu.ar (H. Alfonso), schuetz@icd.de (M. Schütz).

0950-7051/02/$ - see front matter
PII: S0950-7051(01)00117-4

(makespan, global earliness and weighted completion time) was first considered, then a Pareto optimality problem with two objectives (makespan, mean absolute deviation from a common due date) was studied. Section 2 discusses multi-recombination and incest prevention as means to enhance EAs. Section 3 is related to single objective JSSP optimization. Section 4 discusses aggregative and Pareto multiobjective JSSP optimizations. Section 5 discusses the conclusions.

2. The mechanisms for improvements

In this section multirecombination and incest prevention, as means to improve EAs, are discussed.

2.1. Multirecombination

In EAs the common approach is to operate once on each mating pair after selection. Such procedure is known as the *Single Crossover Per Couple* (SCPC) 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 Crossover Per Couple (MCPC) [13] 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. It is possible to choose for insertion in the next generation the best, a randomly selected or all of the generated offspring. In previous 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 [19,20], 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, three Scanning Crossover (SX) methods; *Uniform Scanning Crossover* (USX), *Occurrence Based Scanning* (OBSX) and **Fitness** *Based Scanning* (FBSX) generating a single offspring. In USX, each gene in the child is provided from any of the corresponding genes in the parents with equal probability. OBSX selects that gene value which occurs more frequently in a particular position of the parent's chromosomes. FBSX chooses the value to inherit being proportional to the fitness value of the parents. On different function optimization different versions of scanning crossover showed different behavior. Following this idea and to improve performance, *Multiple Crossovers on Multiple Parents* (MCMP) allows multiple recombination of multiple parents under *scanning crossover* (SX), expecting that exploitation and exploration of the problem space be adequately balanced [16].

2.2. Extended incest prevention (EIP)

The extension of the concept of incest is strongly related to the mating members of the same family. To prevent incest EIP allows only recombination of individuals without common ancestors. To build the new population in EIP, individuals are randomly chosen from the previous one according to the conventional *fitness proportional selection*, but they are allowed to crossover if no common ancestors are detected in earlier generations. In this way, exchange of similar genetic material is reduced and population diversity is maintained up to some convenient degree. Consequently, each individual maintains information about their ancestors. Persistent high population diversity has also a detrimental effect: to slow down the search process. To make this point clearer we have to note that by allowing crossover only on some *non-relative* individuals, we modify the effect of the selection mechanism on the population.

In the evolutionary process two important, closely related, issues exist: population diversity and selective pressure enforced by the mechanism. They are the sides of the same coin: exploration of the searching space versus exploitation of information gathered so far. Selection plays an important role here because strong selective pressure can lead to premature convergence and weak selective pressure can make the search ineffective. In this work we addressed the issue by fixing the number of generations to determine the ancestry relationship between individuals.

2.3. Multiplicity and incest prevention (MCMPIP)

The following pseudo-code delineates a procedure to prevent incest between members of the same or consecutive generations (brother–sister and parent–offspring), when a number of $n_2 > 2$ parents are used

```
procedure MCMPIP (multiple crossovers, multiple parents, incest prevention)
begin
  int mating_pool[number_of_parents]//array to store selected parents
  int children_pool[number_of_cross]//array to store created offspring
  for 1 to popsize
    select indiv-1 C(t)//C(t) is the current population
    mating_pool[1] = indiv-1
    i = 2
  while (i ≤ number_of_parents)
    repeat
    select indiv-i C(t)
    until(is_not_relative(mating_pool, indiv-i))//control of no common ancestry and
    uniqueness of parents in the mating pool
    matting_pool[i] = indiv-i
    i = i + 1
  end while
```

recombine using MCMP and mutate individuals from mating_pool to children_pool
select the best individual from children_pool building the new population $C'(t)$
end for
end procedure

3. Single objective optimization

In JSSP the main difficulty encountered is that of specifying an appropriate representation of feasible schedules. The ways of representation can be *direct* or *indirect* [21]. In direct representation, the schedule produced is itself used as a chromosome. A decoder procedure is not necessary but specific genetic operators should be designed [22,23]. In indirect representation a transition from chromosome representation to a legal schedule has to be performed by a schedule builder prior to evaluation [9,10,21,24].

In this work, two different representations are used which belong to the indirect group. They are the *decoders* and *priority rule based* representations.

3.1. Indirect representations: decoders and priority rule based representations

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.

3.1.1. Job based representation and decoders

In JSSP, a *job based representation,* consists of a list of n 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* (PMX) [25], *Order Crossover* (OX) [26], and *Cycle Crossover* (CX) [27] 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 ith component is an integer in the range $1\ldots(n-i+1)$. 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 for decoders (chromosomes) (1111) and (3121) the corresponding permutations are (1234) and (3142), respectively.

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 representations 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.

3.1.2. Priority rule based representation

In Ref. [28] a priority-rule-based genetic algorithm is proposed. Under this approach, 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. EAs 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 algorithm can be considered as the basis of priority rule based heuristics [1,29]. The main problem is to determine an effective priority rule. The most commonly priority rules used in practice are: Shortest Processing Time, Longest Processing Time, Most Work Remaining, Least Work Remaining, Most Operations Remaining, Least Operations Remaining, Earliest Due Date, First come first served and Random.

For an n jobs and m machines problem, under this representation, a chromosome is a string of $n\times m$ entries (p_1, $p_2,\ldots,p_{nm}$).Each entry represents one of the pre-specified rules. The entry p_i indicates that a conflict in the ith iteration of the Giffler and Thompson algorithm should be resolved by using the priority rule p_i (an operation from the conflict set has to be selected by rule p_i). Ties are broken at random. 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 to deduce schedule from a given chromosome (p_1, $p_2,\ldots,p_{nm}$) is given below:

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.

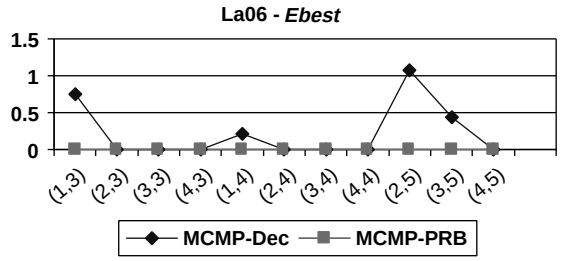


Fig. 1. *Ebest* values from both algorithms for the *la06* instance.

Increment t by one.

5. Return to step 2 until a complete schedule is generated.

3.2. Experiments and results

We refer now to design of experiments and results for single objective optimization in JSSP: minimizing the makespan.

3.2.1. MCMP-Dec and MCMPIP-Dec

First, under decode representation, MCMP-Dec and MCMPIP-Dec, were contrasted for a set of six Lawrence's instances [30] (n jobs $\times$ m machines), with known optimal makespan. They were *la06* (15×5), *la12* (20×5), *la15* (20×5), *la16* (10×10), *abz6* (10×10), and *abz7* (20×15). A total of, 72 different experiments were designed. For each instance a series of fifteen runs was performed. Experiments corresponded to different number of crossovers and number of parents (n_1, n_2) combinations, SX and multi-recombination methods. The evolutionary algorithms used proportional selection for mating and elitism to retain the best-valued individual. Parameter settings were the same for both methods except population size. In our experiments, it was fixed at 50 and 120 individuals for MCMP-Dec and MCMPIP-Dec, respectively. All three scanning crossover methods were implemented and for insertion in the next generation the best child was chosen ($n_3 = 1$). Number of crossovers and parents were set to: $1 \leq n_1 \leq 4$ and $3 \leq n_2 \leq 5$, respectively. For mutation a *big-creep* operator, replacing the gene value by another in 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. The following relevant performance variables were chosen:

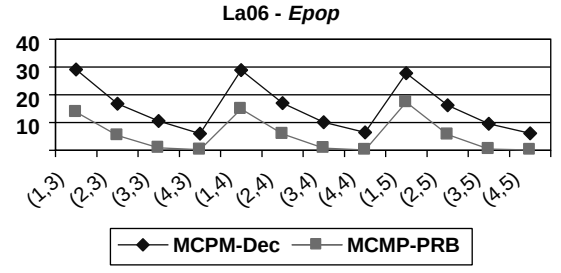


Fig. 2. *Epop* values from both algorithms for the *la06* instance.

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 are we 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*.

We discuss here results regarding the algorithms behavior on demonstrative instance *la06* only for both MCMP-Dec and MCMPIP-Dec under USX, FBSX and OBSX.

For this instance, both MCMP-Dec and MCMPIP-Dec find the optimum under any SX method in 70% of the experiments performed. The optimum is not reach in the case in which a single crossover (SCPC) is applied and better results are, in general, attained when the number n_1 of crossovers is increased, independently of the number n_2 of parents used for mating.

Ebest values range from 0 to 1.08, while the average best performance is for MCMP-Dec with FBSX followed by MCMIP-Dec with the same SX crossover method. Epop values range from 1.48 to 28.31 and minimum values ranging from 1.48 to 3.84 are found with $n_1 = 4$ and $n_2 = 5$. This means that the whole population is more concentrated around the best-found individual. A general overview indicated a loss of performance on both methods for more complex instances. In particular, for *abz6* Ebest values range from 11.03 to 11.98, and Epop values range from 14.63 to 47.99. This fact shows that as long as the complexity of the searching space augments it is more difficult for both methods to find near-optimal schedules. This is particularly true when the number of machines is incremented (*la16*, *abz6* and *abz7*). This effect can be expected because the coding space (job permutations) corresponds to only a part of the whole solution space and it is the price to be paid for straightforward providing feasible offspring. Nevertheless, the performance is better than using simple genetic algorithms (SCPC) and can be improved by alternative representations.

3.2.2. MCMP-Dec and MCMP-PRB

After these results, a new set of experiments included a comparison of multiple crossovers on multiple parents and SCPC. As MCMPIP-Dec showed to be too costly when compared against MCMP-Dec without providing sensible extra benefits the next experiments considered only *decode* (MCMP-Dec) and *priority rule based* representation (MCMP-PRB). All algorithms were contrasted for the same set of selected instances of the JSSP. A total of 60 different experiments corresponding to different (n_1, n_2)

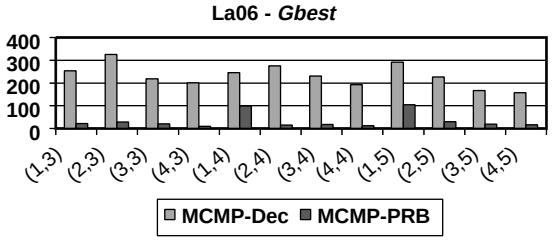


Fig. 3. *Gbest* values from both algorithms for the *la06* instance.

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. The parameter settings and performance variables were the same as in the previous described experiment. Additionally *Gbest* (the generation where the best solution was found) was recorded. Figs. 1–6 summarize results on demonstrative instances *la06* and *abz6*. The notation specifies in the horizontal edge the number of crossovers and parents, for example (1,3) references to the combination of 1 crossover on 3 parents. Fig. 1 shows that all possible combinations (n_1,n_2) under MCMP-PRB reach the optimum. This situation is present in only some combinations of MCMP-Dec, but even so errors are very small (the greater reaches 1%).

Analyzing *Epop* (Fig. 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 $n_1 = 1$ (SCPC), high peaks are observed in both algorithm (29.15% 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 algorithms, the final population turns out to be more and more closer to the optimum as long as n_1 is incremented. This is clearly indicated by the waveform of Fig. 2.

Regarding *Gbest*, (see Fig. 3) MCMP-PRB founds the optimum in a lower number of generations (10–105) than MCMP-Dec (157–326).

In MCMP-PRB, the maximum generation number to find the best value over all instances does not exceed the minimum achieved by MCMP-Dec.

Fig. 4 shows that the optimum is not found by the

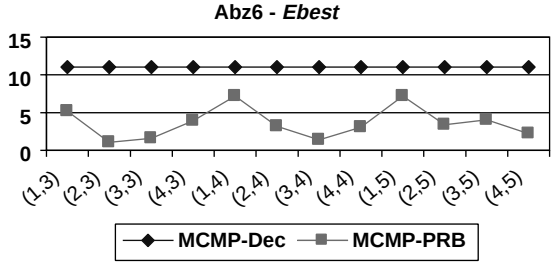


Fig. 4. *Ebest* values from both algorithms for the *abz6* instance.

algorithms but while MCMP-Dec remains in a local optimum for all (n_1, n_2) combinations (with *Ebest* equal to 11.03%) MCPC-PBR *Ebest* values are much lower (ranging from 1.06 to 7.21%). Best values are found for $n_1$between 2 and 4.

Again in Fig. 5 the same systematic behavior is observed. *Epop* diminishes as long as $n_1$is incremented. 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 three parents, *Epop* values range from 28.69 (SCPC) to 4.07% (MCMP). As it happened with *Ebest* values, MCMP-PRB presents a remarkable better behavior than MCMP-Dec providing better *Epop* values.

In Fig. 6 it is shown that MCMP-Dec needs less number of generation to find the best individual than MCMP-PRB. According to results of Fig. 4, this fact clearly show premature convergence and search stagnation.

In this section dedicated to single objective optimization a first approach with multirecombination combined with incest prevention (MCMPIP-Dec) was undertook for different scanning crossover methods under a decoder representation and contrasted against MCMP-Dec. As a general conclusion, we observe that results indicated that incest prevention is too expensive in processing time without providing noticeable extra benefits in the quality of results. Consequently, further work concentrated on hybrid approach including multiplicity of crossovers and parents (MCMP) coupled with a more problem specific representation (PRB). The same testing set was used and 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 min running time in standard workstations. In general, MCMP-PBR performs better than MCMP-Dec (in harder instances as *abz6*, the method has higher ability to scape local optima).

4. Multiobjective optimization

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. In multiobjective optimization Fonseca and Flemming [31] classified as plain aggregative approaches those methods where a single objective function resulting as a numerical combination of objectives values is to be optimized. Here decisions on multicriteria [32] are made before searching. The unique objective function obtained by aggregation

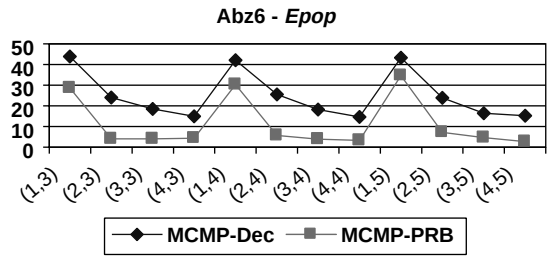


Fig. 5. *Epop* values from both algorithms for the *abz6* instance.

of multiple objectives is used to establish a total order in the solutions' space. This measure provides then a basis for selection of individuals. In this way, an evolutionary algorithm performs as usual finding the fittest individuals for that single aggregated function. Bhanu and Lee [33] and Vemuri and Cedeño [34], worked on this linear combination approach. Other approaches attempt to build the *Pareto front*, also known as the *acceptable set*, the *efficient points* and the *Pareto optimal set*. Vilfredo Pareto [35] 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 $x = (x_1, x_2, \ldots, x_n)$ and $y = (y_1, y_2, \ldots, y_n)$, the Pareto criterion says that, x dominates y iff $x_i \geq y_i \ \forall\ i$ and $\exists\ j$ such that $x_j > y_j$.

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 [36,37], Fourman [38] and then by Kursawe, [39,40] and others, *cooperative population searches (CPS) with criterion selection* [41] 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.

Next sections discuss details of MSE, an aggregative multi-stage evolutive approach to optimize an aggregation

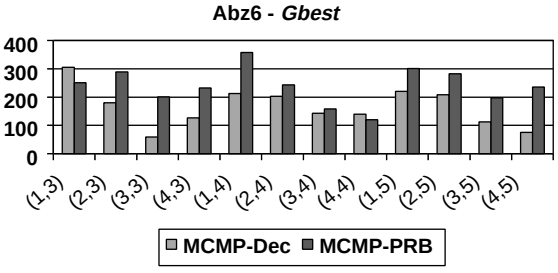


Fig. 6. *Gbest* values from both algorithms for the *abz6* instance.

of three objectives and a CPS-MCPC, a multirecombinated cooperative population search method to determine the Pareto frontier.

4.1. MSE: the multistage evolutive approach

In nature, individuals do not evolve in isolation. Instead, there is an interaction among populations (species) that evolve under diverse environmental conditions. This principle, so-called co-evolution [42], can be applied to solve multicriteria optimization for the JSSP using a plain aggregative approach. The idea is to create one sub-population for each criterion and evolve them until convergence. At this point, another evolutionary process begins actuating on the whole population whose objective is an aggregation of the partial objectives. This evolution step continues until reaching convergence. After that, the whole population is subdivided into sub-populations and the original process is started again. A final stop criterion is defined to terminate the entire process. As a result, it is expected that at the end of the multistage evolutionary process we fulfill two different purposes: to optimize the aggregated objective and obtain a set of good performers, which are near-optimal solutions on their corresponding partial objectives.

4.1.1. The multicriteria problem

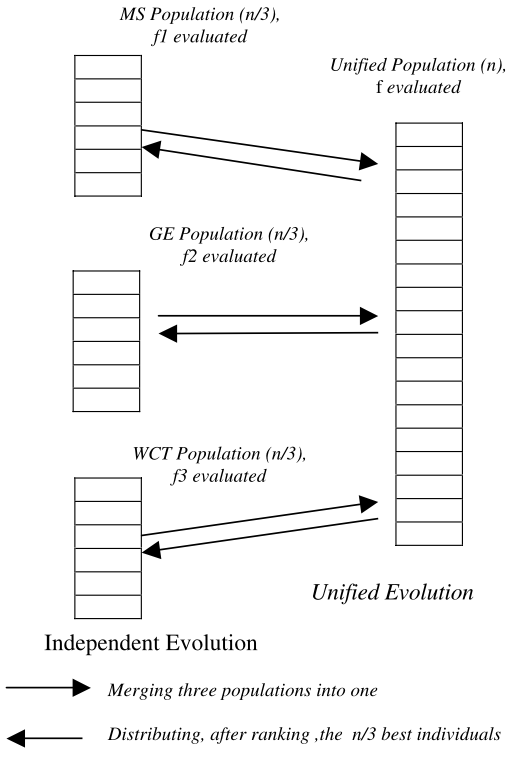
Given an instance of the JSSP, of n jobs and m machines we considered three performance parameters to minimize:

MS $= max\{C_1,\ldots,C_n\}$, where C_i is the completion time of job i. It is the makespan and it is equivalent to the completion time of the last job to leave the system.
GE $= max\{0, -GL\}$ it is the global earliness, where $GL = MS - Gd$, is the global lateness and Gd is the global due date.
WCT $= \sum_{i=1}^{n} w_i C_i$, where w_i is the weight associated to the completion time of job i. It is the weighted completion time.

Our multistage approach considered the 3-criteria problem optimizing three objective functions, f_1, f_2 and f_3, corresponding to *MS*, *GE* and *WCT*, respectively, and the aggregation $f = \alpha f_1 + \beta f_2 + \gamma f_3$.

An individual in any population is an integer vector representing a priority list. Consequently, a population evolves by creating new individuals (priority lists) optimizing some of the above-mentioned objective functions (criteria). There exist two different evolutionary processes:

1. *Independent evolution*. Here three populations evolve independently, each one optimizing one of the above-mentioned criteria. This process is performed until each population reaches convergence.
2. *Unified evolution*. After convergence of independent populations, all of them are merged into a single population and the evolutionary process optimizes now the aggregation f. That means that the independent evolved



Begin Multistage Evolutionary process

1. Initialize 3 distinct populations of size S (one for each objective)
2. Evolve independently each population until the termination criterion θ_1 holds.
3. Merge the independently evolved population into a single unified population of size 3S.
4. Evolve the unified population until the termination criterion θ_1 holds.
5. If the termination criterion θ_2 holds
   then stop
   else rank individuals according to:
   f1 and create a new population with the best one third of the unified population.
   f2 and create a new population with the best one third of the unified population.
   f3 and create a new population with the best one third of the unified population.
   go to step 2.

End of Multistage Evolutionary process.

Fig. 7. Independent and unified evolution in the multistage evolutionary process.

individuals are now submitted to a new environment. This stage involves interactions between individuals of diverse evolving populations under new environmental conditions.

Both processes are repeated until the termination criterion for the unified population holds. The whole multistage evolutionary process can be delineated as shown in Fig. 7.

Two termination criteria were used: θ_1 to control independent evolution and each stage of the multistage evolutionary processes and θ_2 to control the whole multistage process. We decided to stop these processes when after 10 consecutive generations the difference between the corresponding mean population fitness values remained less than 0.001.

4.1.2. Experiments and results

The multistage evolutionary approach (MSE) was contrasted against a conventional evolutionary approach (SGA). MSE evolved the aggregation f and individual criteria MS, GE and WCT while SGA evolved only the aggregation f. In both cases simple but non-canonical genetic algorithms, with ranking selection for mating, were used. Parameter settings were the same for both methods except population size. In our experiments with MSE, randomised initial populations of size fixed at 50 and 20 individuals for small and big instances, respectively, were used to optimize each criterion. Population sizes for SGA were augmented three times for each instance type (150 and 60). A permutation of integers represented an individual (chromosome). Elitism, *ordered crossover* and *interchange mutation* were used. The number of generations was bounded by the corresponding termination criterion and probabilities for crossover and mutation were both fixed to 0.1. These values were determined as the best combination of probabilities after many initial trials. Ten instances of two types, small and big [30], with known optimal makespan were used. Small instances were of 10 jobs and 5 machines, identified as *laX* with $X = 00, \ldots, 05$, while big instances were of 20 jobs and 10 machines identified as *laY* with $Y = 26, \ldots, 30$. As optimal values of makespan were known for each instance of the test suite benchmark, the global due date to determine GE was fixed at a value 40% greater than the corresponding optimal makespan. Coefficients α, β, and γ for the aggregation f, were set to convenient values; 0.4728, 0.5293 and 0.0170, respectively. For both the methods, the mean and best values were established. For small instances, both algorithms found the same best solution in

Table 1
Best values of f for big instances

Instance	SGA				MSE			
	MS	GE	WCT	f	MS	GE	WCT	f
la26	1705	0	20021	1047.5	1691	14	19874	1047.6
la27	1720	9	21119	1065.9	1727	2	20313	1062.3
la28	1698	4	20150	1046.5	1692	10	20119	1046.7
la29	1591	8	18823	982.9	1595	4	18568	981.7
la30	1879	18	20516	1158.6	1870	27	20412	1158.6
Aver.				1060.3				1059.4

Table 2
Best MS values for small instances

Instance	SGA		MSE		Benchmark optimum
	MS	Ebest	MS	Ebest	
la01	933	40.09	666	0.00	666
la02	870	32.82	688	5.04	655
la03	794	33.00	623	4.36	597
la04	777	31.69	611	3.56	590
la05	814	37.27	593	0.00	593
Aver.		34.98		2.59	

each of the 10 runs, corresponding to a particular instance. This fact shows no difference in the ability of the searching process of the contrasted methods in small instances.

For big instances (Table 1) MSE behaves slightly better in two instances (*la27* and *la29*) and in overall performance. The ability of MSE when optimizing one of the criteria is shown in Table 2 for the case of the makespan (the only criteria for which optimal values are known).

MSE finds better MS values than those found by SGA. Moreover in two small instances (*la01* and *la05*) the optimum was reached. For big instances, even if the error increase always remained far lower than that of SGA. This is a natural consequence of the multistage approach. Besides optimizing the aggregative function f by means of the unified evolution, MSE preserves the best individual under each criterion by means of independent evolution.

4.2. CPS-MCPC: The multirecombinated cooperative population search method

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). Consequently, under the typical 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

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

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

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

Fig. 8. An instance for a JSSP with three machines and two jobs. Elements are two-tuples of the form (machine, duration).

where C_{ik} is the completion time of job i in machine k, and C_j stands for the completion time of the last operation of job j. When minimizing function f_1, schedules tend to be shortened, 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.

The set of experiments conducted, used three basic representation schemes and contrasted the results of the proposed approach against conventional methods of recombination

4.2.1. Representations and operators

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 following the representations used to evaluate the multirecombinative approach, the corresponding operators are explained.

4.2.1.1. 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.

4.2.1.2. 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 is allocated in the best available processing time for the machine the operation requires.

4.2.1.3. 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 [43] 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-jobs m-machines problem a chromosome consists of $n \times m$ genes, where each gene has a job identifier as the

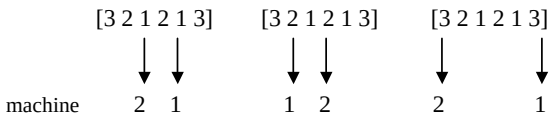


Fig. 9. Job operations and corresponding machines for the matrix instance of Fig 10.

allele value and values are repeated exactly m times in the chromosome.

For the JSSP corresponding to the instance matrix of Fig. 8 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 1 (see Fig. 9).The same interpretation is given to other gene values.

4.2.1.4. 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 select from the first parent a sub-sequence including genes from the fourth to the seventh position:

parent 1 [3 2 2 1 1 2 3 1 3] *parent* 2 [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.

[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 *parent 1* 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 *parent 2* 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.

4.2.2. Building the pareto front

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 are 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 generated 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 $O_{\text{nond}} \neq \Phi$ 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 cross-breeding among 'species'.
- Increases exploitation of good solutions previously found through multiple crossovers per couple.
- Favors 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).

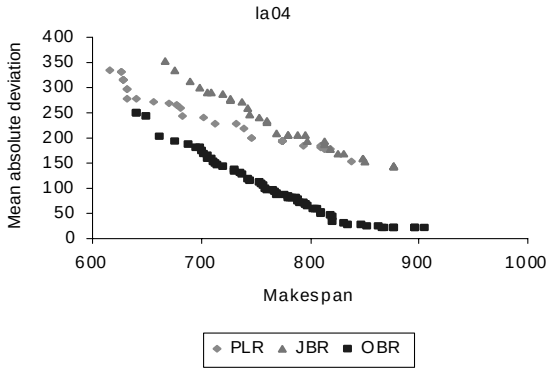


Fig. 10. Representation approaches: comparative performance for building the Pareto front under CPS-MCPC.

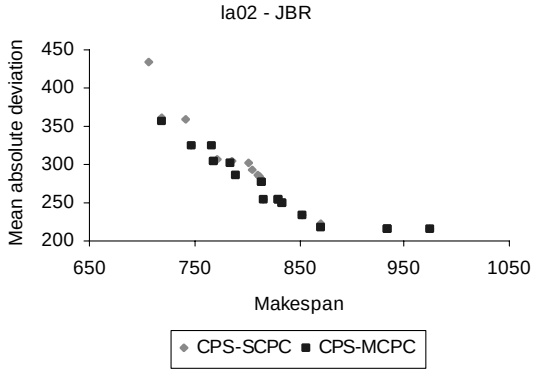


Fig. 12. Pareto fronts built under CPS-MCPC and CPS-SCPC with job-based representation, instance la02.

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.

4.2.3. Experiments

To evaluate the performance of the CPS-MCPC, the same instances indicated in section 4.1.2 were used. After many initial trials the best parameter settings, were determined as follows. Number of crossover per couple was set at $n_1 = 4$. 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 big 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. The common due date d to determine $f_2(\sigma)$ values was fixed at a value 40% greater than the corresponding optimal makespan. Section 4.2.4 shows 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.

4.2.4. 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 behavior shown in Fig. 10 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 the CPS methods, we will show only results for *la02* and *la30* as demonstrative instances for each type, because the remaining instances reveal similar findings

Figs. 11–13 show the Pareto fronts built under both recombination methods with the three chosen representations for small instance *la02*. In Fig. 11 with PLR, 48 non-dominated solutions were found under both recombination schemes. Also the quality of solutions is similar. In Fig. 12 with JBR, 15 and 19 non-dominated points were found under CPS-SCPC and CPS-MCPC, respectively. The

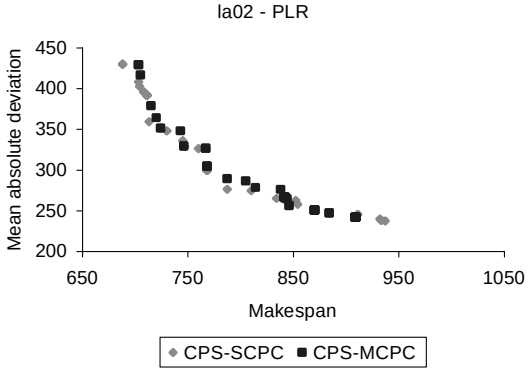


Fig. 11. Pareto fronts built under CPS-MCPC and CPS-SCPC with priority list representation, instance la02.

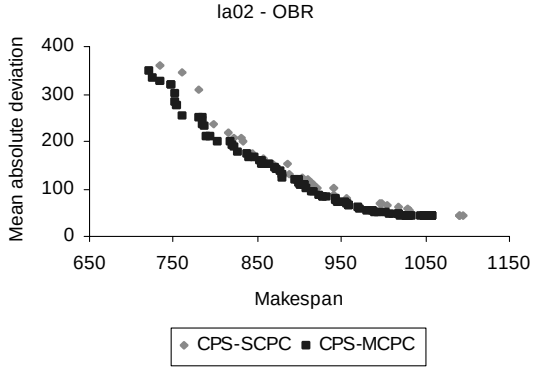


Fig. 13. Pareto fronts built under CPS-MCPC and CPS-SCPC with operation-based representation, instance la02.

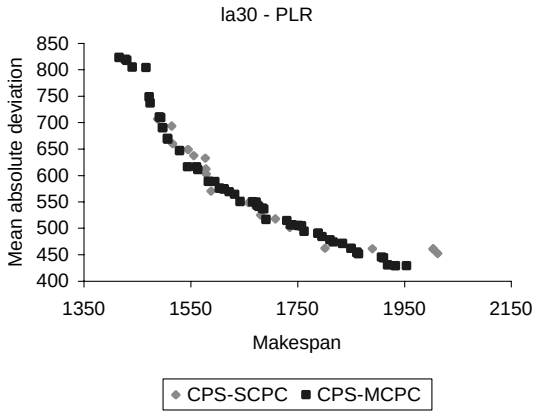


Fig. 14. Pareto fronts built under CPS-MCPC and CPS-SCPC with priority list representation, instance la30.

multirecombination approach shows better quality of results than the conventional single crossover approach. Finally in Fig. 13 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.

Figs. 14–16 show 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.

Other important evidence arises when observing at the final population attained by either recombination method. Fig. 17, shows a total of 321 points, where 125 of them are non-dominated and belong to the final $P_{current}$ sets of the corresponding recombination methods. Under CPS-SCPC the final population shows a higher diversity than under CPS-MCPC. Average individuals in both populations have

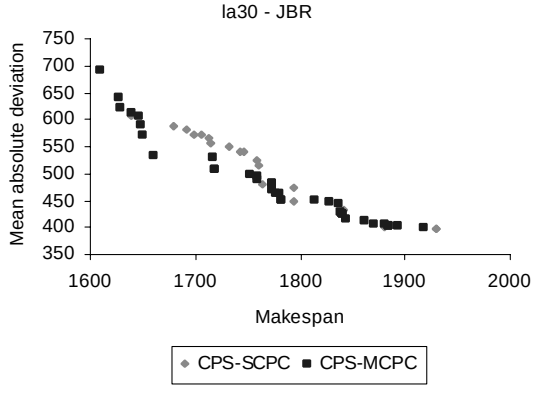


Fig. 15. Pareto fronts built under CPS-MCPC and CPS-SCPC with job-based representation, instance la30.

the following objective values

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

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

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

In this section dedicated to multiobjective optimization aggregative and Pareto optimality approaches were undertaken. In both the approaches, the enhanced evolutionary algorithms showed a better behavior than that observed for the conventional evolutionary algorithms. For the aggregative approach a multistage evolutionary algorithm (MSE) showed a slightly better overall performance than SGA and near optimal solutions, under each individual criterion, were also provided.

To build a Pareto front a multi-recombinated algorithm (CPS-MCPC), was contrasted against a single-recombinated algorithm (CPS-SCPC). Both methods were run under three distinct representations (PLR, JBR, and OBR). A novel MOX operator for crossover was designed to generate valid offspring for a problem specific representation (OBR). As results of these experiments we can conclude that, independently of the coding technique adopted, in most cases CPS-MCPC gives an indication of building better Pareto fronts.

5. Conclusions

Evolutionary algorithms have been proved as efficient tools to face scheduling problems. The effectiveness of evolutionary computation depends on the representation used for the problem solutions, the operators used and the configuration of the evolutionary algorithm. These ideas are not new and have been recognized for some time.

This contribution shows the application of enhanced evolutionary algorithms to the Job Shop Scheduling Problem in single and multiobjective optimization. Enhancements are primarily related to multirecombination (MCPC and MCMP). Also, the extended incest prevention to avoid premature convergence was implemented and the modified ordered crossover (MOX) was introduced to generate feasible offspring when operation-based representation (OBR) is used.

In single objective optimization a first approach with multirecombination combined with incest prevention was undertook for different scanning crossover methods under a decoder representation. Results indicated that USX performs similarly to other more complex methods (FBSX and OBSX) and that incest prevention is too expensive in processing time without providing noticeable extra benefits in the quality of results. Consequently, further work concentrated on a hybrid approach including multiplicity of crossovers and parents (MCMP) coupled with a more problem specific representation (PRB) that triggers a priority

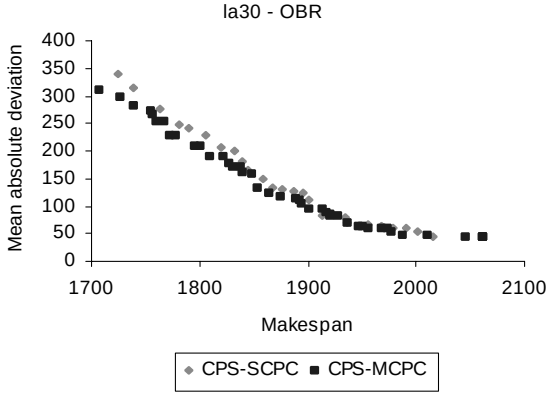


Fig. 16. Pareto fronts built under CPS-MCPC and CPS-SCPC with operation-based representation, instance la30.

dispatching rule when conflicts are to be resolved. The same testing set was used and 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.

In multiobjective optimization aggregative and Pareto optimality approaches were undertaken. For the aggregative approach, a multistage evolutionary algorithm (MSE) was implemented. Independent and unified evolution, were introduced to obtain partial optimization of three distinct criteria, f_1, f_2 and f_3, and the aggregation criterion f at the same time. Results of these preliminary experiments with MSE show some enhancements when compared with the conventional evolutionary approach (SGA). In general, the overall performance is slightly better in big instances and the near optimal solutions, under each individual criterion, are also provided. This later additional feature can be of utmost importance in the decision making process for multiobjective optimization.

To build a Pareto front a multi-recombinated cooperative population search method (CPS-MCPC), was implemented and contrasted against a single-recombinated cooperative population search method (CPS-SCPC). Both methods were run under, three well-known representations (PLR, JBR, and OBR). A novel MOX operator for crossover was designed to generate valid offspring for a problem specific representation (OBR). As results of these experiments we can conclude that, 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, and more densely and evenly distributed fronts. Moreover, the final population obtained under the novel approach is grouped around compromise solutions. The latter fact shows that the alternative solutions provided by multirecombination, attempt to balance the damage caused on the conflicting objectives when one of them is arbitrarily chosen for improvement.

These preliminary results are promising and encourage us to deep forward investigation in single and multiobjective scheduling problems by using multirecombination with better representations for the JSSP. An open remaining question is the optimal setting of the numbers $n_1$and n_2 of crossovers and parents, which could be self-adapted. Other variants of multiplicity of parents and crossovers are under study to establish the abilities and possible limitations of this approach.

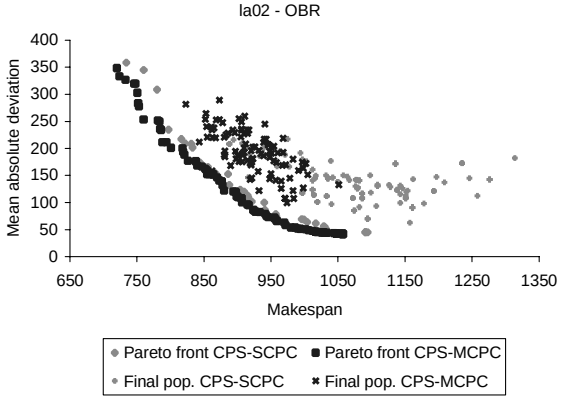


Fig. 17. Pareto fronts and final populations obtained under CPS-MCPC and CPS-SCPC with operation-based representation

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, the Center for Applied System Analysis and the ANPCYT from which we receive continuous support within the bilateral research aggreement.

References

[1] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, Freemann, San Francisco, 1979.
[2] R. Bruns, Direct chromosome representation and advanced genetic operators for production scheduling, Forrest ICGA93, 1993, pp. 352–359.
[3] F.F. Easton, N. Mansour, A distributed genetic algorithm for employee staffing and scheduling problems, Forrest ICGA93, 1993, pp. 360–367.
[4] H.-L. Fang, P. Ross, D. Corne, A promising genetic algorithm approach to job-shop scheduling, rescheduling, and open-shop scheduling problems, in: S. Forrest (Ed.), Forrest ICGA93, Proceedings of the Fifth International Conference on Genetic Algorithms, Morgan Kaufmann, San Mateo, 1993, pp. 375–382.
[5] P. Husbands, F. Mill, S. Warrington, Genetic algorithms, production plan optimisation, and scheduling, Schwefel and Maenner PPSN91, 1991, pp. 80–84.

[6] S. Lawrence, Resource constrained project scheduling: an experimental investigation of heuristic scheduling techniques-(supplement), Graduate School of Industrial Administration, Carnegie–Mellon University, Pittsburgh, 1984.
[7] I. Lee, R. Sikora, M.J. Shaw, Joint lot sizing and sequencing with genetic algorithms for scheduling: evolving the chromosome structure, Forrest ICGA93, 1993, pp. 383–389.
[8] M. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs, 3, Springer, Berlin, 1996.
[9] R. Nakano, T. Yamada, Conventional genetic algorithms for job shop problems, in: R.K. Belew, L.B. Booker (Eds.), Proceedings of the Fourth International Conference on Genetic Algorithms, Morgan Kaufmann, San Mateo, CA, 1991, pp. 474–479.
[10] G. Syswerda, Schedule optimization using genetic algorithms, in: L. Davis (Ed.), Handbook of Genetic Algorithms, Van Nostrand Reinhold, New York, 1991, pp. 332–349 Chapter 21.
[11] T. Yamada, R. Nakano, A genetic algorithm applicable to large-scale job-shop problems, in: R. Maenner, B. Manderick (Eds.), Parallel Problem Solving from Nature 2, Elsevier, Amsterdam, 1992, pp. 281–290.
[12] R. Bruns, Scheduling, in: Th Bäck, D.B. Fogel, Z. Michalewicz (Eds.), Handbook of Evolutionary Computation, Oxford University Press, Institute of Physics, New York, Bristol, 1997 Chapter F1.5.
[13] S. Esquivel, A. Leiva, R. Gallard, Multiple crossover per couple in genetic algorithms, Proceedings of the Fourth IEEE International Conference on Evolutionary Computation (ICEC'97), Indianapolis, USA, April 1997, pp. 103–106.
[14] S. Esquivel, A. Leiva, R. Gallard, Couple fitness based selection with multiple crossover per couple in genetic algorithms, in: E. Alpaydin (Ed.), Proceedings of the International Symposium on Engineering of Intelligent Systems (EIS'98), La Laguna, Tenerife, Spain, vol. 1, ICSC Academic Press, Canada, 1998, pp. 235–241.
[15] S. Esquivel, H. Leiva, R. Gallard, Multiplicity in genetic algorithms to face multicriteria optimization, Presentation in the 1999 Congress on Evolutionary Computation (IEEE). Washington DC, 2001 (in press).
[16] S. Esquivel, H. Leiva, R. Gallard, Multiple crossovers between multiple parents to improve search in evolutionary algorithms, Presentation in the 1999 Congress on Evolutionary Computation (IEEE), Washington DC, 2001 (in press).
[17] L.J. Eshelman, J.D. Schaffer, Preventing premature convergence in genetic algorithms by preventing incest, Proceedings of the Fourth International Conference on Genetic Algorithms, Morgan Kauffman, San Mateo, California, 1991, pp. 115–122.
[18] H. lfonso, P. Cesan, N. Fernandez, G. Minetti, C. Salto, L. Velazco, R. Gallard, Improving evolutionary algorithms performance by extending incest prevention, Proceedings del 4 to Congreso Argentino de Ciencias de la Computación (CACiC'98), Abstracts del Congreso, Universidad Nacional del Comahue, October 1998, pp. 323–334.
[19] A.E. Eiben, C.H.M. van Kemenade, J.N. Kok, Orgy in the computer: multi-parent reproduction in genetic algorithms, in: F. Moran, A. Moreno, J.J. Merelo, P. Chacon (Eds.), Proceedings of the Third European Conference on Artificial Life, number 929 in LNAI, Springer, Berlin, 1995, pp. 934–945.
[20] A.E. Eiben, Th. Bäck, An empirical investigation of multi-parent recombination operators in evolution strategies, Evolutionary Comput. 5 (3) (1997) 347–365.
[21] S. Bagchi, S. Uckum, Y. Miyabe, K. Kawamura, Exploring problem-specific recombination operators for job shop scheduling, Proceedings of the Fourth International Conference on Genetic Algorithms, Morgan Kaufmann, San Diego, 1991.
[22] J.J. Kanet, V. Sridharan, PROGENITOR: a Genetic Algorithm for Production Scheduling, Witschafstsinformaatik, 1991.
[23] P. Husbands, F. Mill, Simulated co-evolution as the mechanism for emergent planning and scheduling, Proceedings of The Fourth International Conference on Genetic Algorithms, Morgan Kaufmann, San Diego, 1991.
[24] L. Davis, Job shop scheduling with genetic algorithms, Proceedings of the First International Conference on Genetic Algorithms, Lawrence Erlbaum, Hillsdale, 1985, pp. 136–140.
[25] D.E. Goldberg, R. Lingle, Alleles, loci and the TSP, Proceedings of the First International Conference on Genetic Algorithms, Lawrence Erlbaum, New Jersey, 1985.
[26] L. Davis, Applying adaptive algorithms to domains, Proceedings of the International Joint Conference on Artificial Intelligence 1985, pp. 162–164.
[27] I.M. Oliver, D.J. Smith, J.R.C. Holland, A study of permutation crossover operators on the travelling salesman problem, Proceedings of the Second International Conference on Genetic Algorithms, Lawrence Erlbaum, New Jersey, 1987, pp. 224–230.
[28] U. Dorndorf, E. Pesch, Evolution based learning in a job shop scheduling environment, Comput. Oper. Res. 22 (1995) 25–40.
[29] C. Salto, H. Alfonso, R. Gallard, Multiplicity and incest prevention in evolutionary algorithms to deal with the job shop problem, Proceedings of the Second ICSC Symposium on Engineering of Intelligent Systems, University of Paisley, Scotland, 2000, pp. 451–457.
[30] S. Lawrence, Resource constrained project schedulingan experimental investigation of heuristic scheduling techniques (Supplement), Graduate School of Industrial Administration, Carnegie–Mellon University, Pittsburgh, Pennsylvania, 1984.
[31] C.M. Fonseca, P.J. Fleming, Genetic algorithms for multiobjective optimization: formulation, discussion and generalization, Proceedings of the Fifth International Conference on Genetic Algorithms, Morgan Kaufmann, Urbana-Champaign, IL, 1993, pp. 416–423.
[32] G. Leitmann, A. Marzollo, Multicriteria Decision Making-CISM No 211, Springer, Wien, NY, 1975.
[33] B. Bhanu, S. Lee, Genetic Learning for Adaptive Image Segmentation, Kluwer Academic Publishers, Boston, MA, 1994.
[34] R. Vemuri, W. Cedeño, A new genetic algorithm for multiobjective optimization in water resource management, Proceedings of the First IEEE International Conference on Evolutionary Computation (ICEC'94), Orlando, USA, 1994, pp. 495–500.
[35] V. Pareto, Cours d'Economie Politique, Rouge, Lausanne, Switzerland, 1896.
[36] J.D. Schaffer, Some experiments in machine learning using vector evaluated genetic algorithms, Doctoral dissertation, Department of Electrical Engineering, Vanderbilt University, 1984.
[37] J.D. Schaffer, Multiple objective optimization with vector evaluated genetic algorithms, in: J.J. Grefenstette (Ed.), Proceedings of the First International Conference on Genetic Algorithms 1985, pp. 93–100.
[38] M.P. Fourman, Compaction of symbolic layout using genetic algorithm, in: J.J. Grefenstette (Ed.), Proceedings of the First International Conference on Genetic Algorithms 1985, pp. 141–153.
[39] F. Kursawe, Evolutionststrategien für die vectoroptimierung, diplomarbeit, Universitat Dortmund, 1990.
[40] F. Kursawe, A Variant of Evolution Strategies for Vector Optimization, Parallel Problem solving from Nature, Lectune notes in Computer Science, 496, Springer, Berlin, 1991, pp. 193–197.
[41] J. Horn, Handbook of Evolutionary Computation, Multicriterion Decision Making, Oxford University Press, Oxford, 1997 F1.9:1–9:15.
[42] J. Paredis, Coevolutionary Constraint Satisfaction, Proceedings of the 3rd Annual Conference on Parallel Problem Solving from Nature, Springer, New York, 1994, pp. 46–55.
[43] M. Gen, Y. Tsujimura, E. Kubota, Solving job-shop scheduling problem using genetic algorithms, Proceedings of the 16th International Conference on Computers and Industrial Engineering, Ashikaga, Japan, 1994.