Decision Science Letters 4 (2015) 247–260

Contents lists available at GrowingScience

Decision Science Letters

homepage: www.GrowingScience.com/dsl

Choice of a PISA selector in a hybrid algorithmic structure for the FJSSP

Mariano Frutos[a*] and Fernando Tohmé[b]

[a]*Department of Engineering, Universidad Nacional del Sur and CONICET, Argentina*
[b]*Department of Economics, Universidad Nacional del Sur and CONICET, Argentina*

C H R O N I C L E

Article history:
Received June 10, 2014
Accepted November 22, 2014
Available online
November 22 2014

Keywords:
Flexible Job-Shop Scheduling Problem
PISA selector
Multi-Objective Hybrid Evolutionary Algorithm

A B S T R A C T

This paper analyzes the choice of a PISA selector for a Hybrid Algorithm integrating it as a Multi-Objective Evolutionary Algorithm (MOEA) with a path-dependent search algorithm. The interaction between these components provides an efficient procedure for solving Multi-Objective Problems (MOPs) in operations scheduling. In order to choose the selector, we consider both NSGA and SPEA as well as their successors (NSGAII and SPEAII). NSGAII and SPEAII are shown to be the most efficient candidates. On the other hand, for the path-dependent search at the end of each evolutionary phase we use the multi-objective version of Simulated Annealing.

1. Introduction

One of the main purposes of production planning is improving the efficiency of processes (Bihlmaier et al., 2009). A good plan in an industrial firm can be seen as a solution to a Job-Shop Scheduling Problem (JSSP) (Chao-Hsien & Han-Chiang, 2009) although this problem belongs to the NP-Hard class (Ullman, 1975; Papadimitriou, 1994). The JSSP involves the allocation of limited resources to jobs in order to optimize some given objectives (Armentano & Scrich, 2000; Storer et al., 1992). Evolutionary procedures have been designed to address these multi-objective problems (Deb et al., 2002) (Coello Coello et al., 2006). Most of the work on JSSP has been done on its single objective version, but in real-world cases multiple goals are highly frequent (Chinyao & Yuling, 2009). As indicated by T`kindt and Billaut (2006), a genuine scheduling problem requires the optimization of several simultaneous goals. Along these lines we present the result of searching for an appropriate PISA selector, from a small class of candidate Multi-Objective Evolutionary Algorithms (MOEAs) which, joint with a local search procedure (MOSA, Multi-Objective Simulated Annealing) addresses the flexible instance of JSSP (Cortés Rivera et al., 2003; Park et al., 2003; Tsai & Lin, 2003; Wu et al., 2004). In this sense, this paper provides a methodological ground for the design of such a hybrid algorithm,

* Corresponding author. Tel. : +54 0291 4595156
E-mail address: mfrutos@uns.edu.ar (M. Frutos)

doi: 10.5267/j.dsl.2014.11.001

NSGAII+MOSA, as presented in (Frutos et al., 2010). We claim that the combination of these algorithms yields a metaheuristic tool that provides a good approximation to the Pareto frontier of multi-objective JSSPs without the short-comings of the underlying MOEA. In particular, that NSGAII fares better than other alternative candidates.

1.1 Approaches to the JSSP

The large body of work on the JSSP exhibits different solution strategies ranging from priority rules to parallel branch-and-bound algorithms. While Muth and Thompson's (1964) introduced the current form of the JSSP, Jackson (1956) presented solution procedures generalizing Johnson's (1954). Akers and Friedman (1955) provided a Boolean representation of the algorithm, which was later simplified as a disjunctive graph in Roy and Sussman (1964), while Balas (1959) profited from this representation to yield another solution to the JSSP. In more contemporary times, the complexity of the JSSP permitted alternative formulations (Li et al., 2011, 2013; Della Croce et al., 2014), which allowed the application of particular algorithms like Clonal Selection (Cortés Rivera et al., 2003), Hybrid Artificial Bee Colony (Li et al., 2011), Multi-Population Interactive Coevolutionary (Xing et al., 2011), Priority Rules (Panwalker & Iskander, 1977), Shifting Bottlenecks (Adams et al., 1988) (Mönch & Zimmermann, 2011), etc. The efficiency of these meta-heuristic procedures leaves room for further improvement (De Giovanni & Pezzella, 2010) (Al-Hinai & ElMekkawy, 2011) (Shin et al., 2008).

1.2 Multi-Objective Optimization: Basic Concepts

Let us assume that several goals (objectives) have to be minimized. Thus, a vector $\bar{x}^* = (x_1^*,...,x_n^*)$ of n decision variables (real numbers) is chosen, satisfying q inequalities $g_i(\bar{x}) \geq 0$, $i = 1,...,q$ as well as p equations $h_i(\bar{x}) = 0$, $i = 1,...,p$, such that a vector of k functions, $f(\bar{x}) = (f_1(\bar{x}),...,f_k(\bar{x}))$ each one corresponding to a particular goal, attains a Pareto optimum. More precisely, the family of decision vectors satisfying the q inequalities and the p equations is denoted by Ω and each $\bar{x} \in \Omega$ is a feasible alternative. A $\bar{x}^*$ is Pareto optimal if for any $\bar{x} \in \Omega$ and every i=1,…,k, $f_i(\bar{x}^*) \leq f_i(\bar{x})$. This means that no $\bar{x}$ can improve a goal without worsening others. We say that a vector of real numbers $\bar{u} = (u_1,...,u_n)$ dominates another, $\bar{v} = (v_1,...,v_n)$ (denoted $\bar{u} \preceq \bar{v}$) if and only if for every $i \in \{1,...,k\}$, $u_i \leq v_i$ and for some $j \in \{1,...,k\}$ $u_j < v_j$. The set of Pareto optima is $P^* = \{\bar{x} \in \Omega :$ there is no $\bar{x}' \in \Omega$ such that $f(\bar{x}') \preceq f(\bar{x})\}$ and the associated Pareto frontier is $FP^* = \{f(\bar{x}) : \bar{x} \in P^*\}$. The main goal of Multi-Objective Optimization is to find the corresponding FP^*. A good approximation should yield a few feasible alternatives close enough to the frontier (Frutos & Tohmé, 2009).

2. The Flexible Job-Shop Scheduling Problem

The Job-Shop Scheduling problem amounts to organizing the execution of a class of n jobs ($\{J_j\}_{j=1}^n$) on a set of m machines ($\{M_k\}_{k=1}^m$). Each job is described as a sequence of tasks that be performed in sequence: $J_j \equiv S_1,...,S_{n_j}$ (assuming that the order of tasks is known we write $S_i \in J_j$. We denote with O_{jk}^i that the task S_i of job J_j is performed on machine M_k. O_{jk}^i requires the use of a machine M_k for a period $\tau_{jk}^i \geq 0$ (the processing time) at a cost μ_{jk}^i. The family of operations to be run on a machine M_k is denoted E_k. In the case of Flexible JSSP (FJSSP), each O_{jk}^i can be processed by any of the machines in M. A key issue here is the scheduling of activities, i.e. the determination of the starting time t_{jk}^i of each O_{jk}^i. (Table 1, FJSSP MF01 (Frutos et al., 2010)).

Table 1
A Flexible Job-Shop Scheduling Problem

MF01 / Problem 3 × 4 with 8 operations (flexible)

J_j	O^i_{jk}	M_1		M_2		M_3		M_4	
		τ^i_{j1}	μ^i_{j1}	τ^i_{j2}	μ^i_{j2}	τ^i_{j3}	μ^i_{j3}	τ^i_{j4}	μ^i_{j4}
J_1	O^1_{1k}	1	10	3	8	4	6	1	9
	O^2_{1k}	3	4	8	2	2	10	1	12
	O^3_{1k}	3	8	5	4	4	6	7	3
J_2	O^1_{2k}	4	7	1	16	1	14	4	6
	O^2_{2k}	2	10	3	8	9	3	3	8
	O^3_{2k}	9	3	1	15	2	10	2	13
J_3	O^1_{3k}	8	6	6	8	3	12	5	10
	O^2_{3k}	4	11	5	8	8	6	1	18

At the start of the process each machine is available and can only carry out an operation at a time. Furthermore, no job can use each machine more than once and has to wait until the next machine is available (Lin et al., 2011). All the setup and waiting times are included in the initial data and machines can remain unused at any step of the plan. The final state is reached when each job has completed its last operation (Heinonen & Pettersson, 2007). The FJSSP involves, in turn, two sub-problems: the allocation of the O^i_{jk} on the different M_k and the determination of the best way of sequencing them, guided by the goals to reach. That is, to find optimal levels of Processing Time (Makespan) (f_1) stated in Eq. (1), for each job J_j and Total Operation Costs (f_2) stated as Eq. (2).

$$C^j_{max} = \sum_{S_i \in J_j} \max_{M_k \in M} (t^i_{jk} + \tau^i_{kj}) \tag{1}$$

and

$$\sum_{J_j} \sum_{S_i \in J_j} \sum_{M_k \in M} x^i_{jk} \mu^i_{jk} \tag{2}$$

where $x^i_{jk} = 1$ if $O^i_{jk} \in E_k$ and 0 otherwise. On the other hand $\sum_k x^i_{jk} = 1$. Besides, starting times satisfy the following condition: $t^i_{jk} = \max\,(t^{(i-1)}_{jh} + \tau^{(i-1)}_{jh},\ t^s_{pk} + \tau^s_{pk},\ 0)$ for each pair $O^{i-1}_{jh}, O^s_{pk} \in E_k$, all machines $M_k, M_h \in M$ and tasks $S_{i-1}, S_i \in J_j$ and $S_s \in J_p$. That is, the starting time of an operation O^i_{jk} should be larger or equal than the total time spent on operation O^{i-1}_{jh} and on operation O^s_{pk}.

3. A Multi-Objective Hybrid Evolutionary Algorithm

Evolutionary algorithms have been intensively applied to optimization problems (Coello Coello et al., 2006; Gao et al., 2008; Chiang & Lin, 2013; Rabiee et al., 2012). But for the FJSSP the high rate of convergence of some of them increase the evaluation costs on multi-objective instances, leading to a low diversity in the solutions. So, poorly distributed Pareto frontiers are sometimes obtained under these procedures. But if efficient local search procedures are added in the process, very few evaluations of the fitness functions yield acceptably distributed Pareto frontiers (see Fig. 1). Our take on this issue is to present a Multi-Objective Hybrid Evolutionary Algorithm (MOHEA) for the FJSSP combining a Multi-Objective Evolutionary Algorithm (MOEA), and Multi-Objective Simulated Annealing (MOSA) (Varadharajan & Rajendran, 2005).

3.1 The Evolutionary Phase

Individuals are represented by means of a variant of (Wu et al., 2004). Given that the FJSSP has two subproblems our MOHEA operates over two chromosomes. The first one represents the allocation of

given operation O^i_{jk} to a machine M_k. For instance, for m = 4, we might have something like $0 \rightarrow M_1$, $1 \rightarrow M_2$, $2 \rightarrow M_3$ and $3 \rightarrow M_4$.

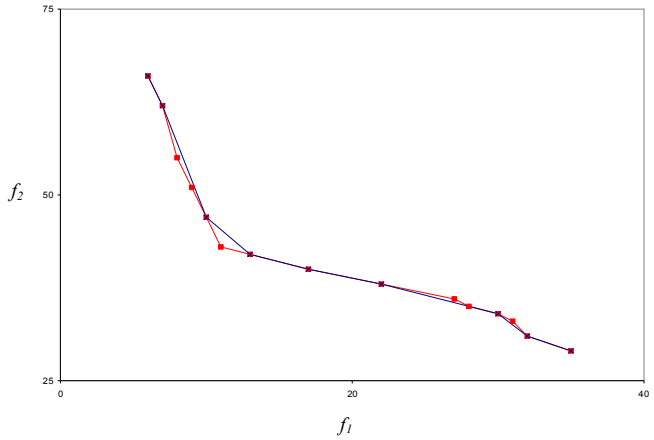

Fig. 1. Makespan vs. Total Operation Costs (MF01-3x4, (Frutos et al., 2010)). MOEA, without Local Search (×) and with Local Search (■)

The second chromosome represents the sequencing of the O^i_{jk} already assigned to a machine $M_k (\forall O^i_{jk} \in E_k)$. We denote with values between 0 and (n!-1) the sequence of J_j at a given M_k. That is, for n = 3, we may have $0 \rightarrow 1 \mid 2 \mid 3$, $1 \rightarrow 1 \mid 3 \mid 2$, $2 \rightarrow 2 \mid 1 \mid 3$, $3 \rightarrow 2 \mid 3 \mid 1$, $4 \rightarrow 3 \mid 1 \mid 2$ and $5 \rightarrow 3 \mid 2 \mid 1$ (Table 2). The initial values are generated in a random way up from uniform distributions: integer numbers between 0 and m-1, for the allocation chromosome and between 0 and n!-1, for the sequencing chromosome. After that, a crossover and a mutation operator are applied segment-wise on the population of combined allocation-sequencing chromosomes. After some preliminary runs, we selected the Uniform Crossover operator, because it yields the best results. The mutation operator is needed because the crossover alone does not allow reaching certain areas of the search space of the FJSSP. We chose the Two-Swap mutation operator, which takes the chain of integers corresponding to two chromosomes and selects at random two genes, swapping their positions.

Table 2

Allocation and Sequencing Chromosomes for the FJSSP

MF01 / Problem 3 × 4 with 8 operations (flexible)						
J_j	O^i_{jk}	M_k	M_1	M_2	M_3	M_4
		Chr.	3	3	0	5
J_1	O^1_{1k}	2				
	O^2_{1k}	1				
	O^3_{1k}	0				
J_2	O^1_{2k}	1				
	O^2_{2k}	2				
	O^3_{2k}	3				
J_3	O^1_{3k}	0				
	O^2_{3k}	3				

3.2 Simulated Annealing as a Local Search Process

Simulated Annealing provides a search procedure based on thermodynamic principles. To avoid local optimum traps, that tend to arise with traditional local search algorithms, random jumps to (possibly

worse) alternative solutions are allowed. Simulated Annealing controls the frequency of jumps by means of the probability function $e^{-(\delta/T)}$, where δ is the difference among values of the objective function, T is the "temperature" at the k-th iteration, starting at a high value (called the initial temperature) T_i, that cools down according to $T_{k+1} = \alpha T_k$ until a final temperature, T_f, is reached. Since higher temperatures increase the probability of getting poor solutions, the procedure diversifies them at its initial phase but improves them in the final stages. At the k-th iteration a class of close neighbors M (T,ω) is obtained, depending on the temperature and a control parameter ω. Each time a neighbor is generated, an acceptation criterion determines whether the current solution is kept or not. In the case of N objectives, there exist several alternative definitions of δ. We take δ as the normalized maximum deviation, $\delta = \max\left[\left(f_i(x') - f_i(x)\right)/f_i(x)\right]$. If a new solution is rejected, a slight variant is tried. The probability of accepting a bad solution makes the algorithm less prone to get caught in a local minimum. On the other hand, during the execution T decreases according to a cooling velocity α, lowering the chances of upward displacements in the space of solutions and keeping the alternatives close to the optimal ones. The algorithm stops if no improvement has been obtained after a certain number of tries or if the final temperature T_f has been reached. Van Laarhoven et al. (1992), show that under appropriate conditions, the algorithm explores efficiently the neighborhood of the actual solution. Our version of the MOSA algorithm (Multi-Objective Simulated Annealing) generates, up from a given one, a class of close-enough alternative solutions by taking one of the genes of the chromosome and changing its value at random (Frutos et al., 2010), representing the exchange of several operations on a single machine. This procedure is applied M times. The pseudo-code of the MOSA used here is presented in Fig. 2.

3.3 Combining the Algorithms

We focus here on how the aforementioned pieces are assembled (see Fig. 3). First, the memetic procedure generates the initial population. Later, to evaluate the fitness of the individuals in the population, the value of each goal is computed and a binary tournament selection is performed. The selected candidates are subject to the genetic operators and create a new and smaller population. Then, the simulated annealing procedure performs a local search on each individual, replacing it with a new one. This is repeated until a given generation number is reached.

0. Take an initial $x \in Q'_{i+1}$
1. **while** $T > T_f$
2. Compute $M = \lfloor 1/T \rfloor + \omega$
3. **for** i = *1* to *M*
4. Change *x* and obtain *x'*
5. Decodify and evaluate $f_1(x')$ and $f_2(x')$
6. **if** f_1 and f_2 *improve*
7. **then** Change *x'*
8. **if** f_1 or f_2 improve without worsening either f_2 or f_1
9. **then** Change *x'*
10. **if** either f_1 or f_2 *gets worse*
11. **then**
12. **if** $\xi(0, 1) < e^{-\delta/T}$
13. **then** Change *x'*
14. **end if**
15. **end if**
16. **end for**
17. $T = \alpha(T)$
18. **end while**
19. **end**

Fig. 2. Pseudo-code of the Simulated Annealing procedure

4. Implementation and Design of Experiments

The whole algorithm was implemented on PISA (A Platform and Programming Language Independent Interface for Search Algorithms) (Bleuler et al., 2003), an algorithm interface that distinguishes between two modules: variator and selector. The former takes all the specificities of the problem at hand to code and decode the solutions (to compute their fitness values). The selector module is independent of the problem and acts by selecting candidates. These modules exchange messages, coded as text files, independently of the programming language and the platform on which the algorithm runs. PISA provides a library of evaluations as well as statistical tools that allow evaluating and comparing alternative optimization methods (Knowles et al., 2005). For this work we considered the following MOEAs (Multi-objective Evolutionary Algorithms): the Non-dominated Sorting Genetic Algorithm (NSGA) (Srinivas, 1994), the Strength Pareto Evolutionary algorithm (SPEA) (Zitzler & Thiele, 1999) and their successors, the Non-dominated Sorting Genetic Algorithm II (NSGAII) (Deb et al., 2002) and the Strength Pareto Evolutionary algorithm II (SPEAII) (Zitzler et al., 2002).

0. Generate an initial population (P_0) of size *N*
1. Decodify and evaluate $f_1(x)$ and $f_2(x)$ on every $x \in P_0$
2. Select Parents from P_0
3. Q_0 = Cross (P_0)
4. Q'_0 = Mutate (Q_0)
5. Q''_0 =Local Search (Q'_0)
6. **for** i = 0 to G - 1 **do**
7. Decodify and evaluate $f_1(x)$ and $f_2(x)$ on every individual $x \in Q''_{i+1}$
8. Select out of $P_i \cup Q''_{i+1}$ the *N* best elements and eliminate the rest
9. Create the next generation P_{i+1}
10. Select Parents from P_{i+1}
11. Q_{i+1} = Cross (P_{i+1})
12. Q'_{i+1} = Mutate (Q_{i+1})
13. Q''_{i+1} = Local Search (Q'_{i+1})
14. **end for**
15. **end**

Fig. 3. Pseudo-code of the Multi-Objective Hybrid Evolutionary Algorithm

NSGA classifies the individuals in layers grouping all the non-dominated individuals in a single front that comprises the individuals with the same value of fitness. This value is proportional to the size of the population, providing reproduction potential for all the individuals in the front. The procedure is repeated on the remaining individuals until all the individuals in the population are classified. Since the candidates in the first front have higher fitness they get more attention than the rest of the individuals. NSGAII is a more efficient version of NSGA that applies an elitist replacement strategy choosing the best individuals from the union between parent and child generations. All the solutions are ranked in terms of their degrees of non-dominancy, being the better ones those with lowest rank. SPEA is an algorithm that at each generation keeps in memory the non-dominated individuals and deletes those that became dominated. For each individual in the external system, a strength value is computed, proportional to the number of solutions in which it is dominant. The fitness of a member of the current population is computed by adding the strengths of the external non-dominated solutions that dominate it. SPEAII instead, applies a fine-tuning procedure according to which the fitness of an individual is obtained as a balance between the number of solutions that it dominates and the number that dominate the individual. Besides, it uses the "nearest neighbor" for valuing the density of feasible solutions, leading to a more efficient search. In Figure 4 we can see the PISA architecture adapted to the FJSSP.

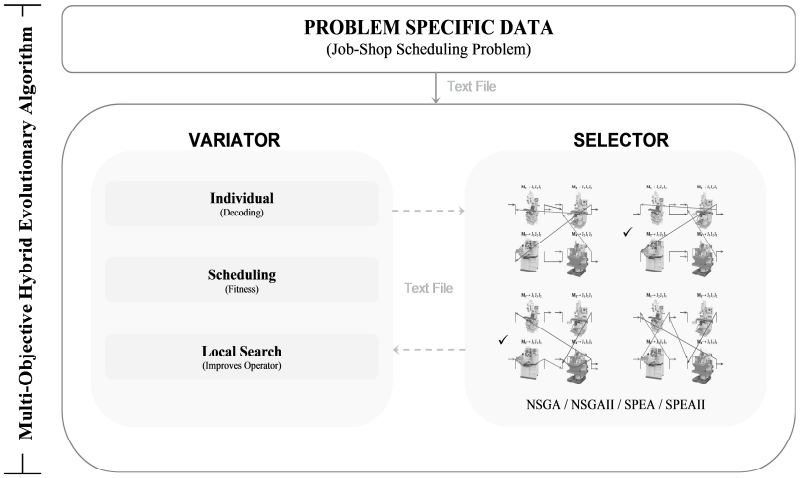

Fig. 4. The architecture of PISA

4.1 Experiments and results

A preliminary analysis of the improvement process showed that it tended to become stable at the 200th generation. We chose then the limit of 250 generations, just to leave room for any later improvement. The parameters and characteristics of the computing equipment used during these experiments were as follows: size of the population: 200, type of cross-over: uniform; probability of cross-over: 0.90, type of mutation: two-swap, probability of mutation: 0.01, type of local search: simulated annealing (T_i: 850, T_f: 0.01, α: 0.95, ω: 10), probability of local search: 0.01, CPU: 3.00 GHZ and RAM: 1.00 GB. Initially we consider four solutions, two dominated solutions (see Table 3 and Table 5) and two undominated solutions (see Table 4 and Table 6) for problem MF01 (Frutos et al., 2010).

Table 3
Scheduling of MF01 (MOHEA) (f_1: 6, f_2: 78)

J_j	O^i_{jk}	Scheduling (MF01) - f_1: 6, f_2: 78			
		A^i_{ik}	t^i_{ik}	$t^i_{ik} + \tau^i_{ik}$	μ^i_{ik}
J_1	O^1_{1k}	M_4	0	1	9
	O^2_{1k}	M_4	1	2	12
	O^3_{1k}	M_1	2	5	4
J_2	O^1_{2k}	M_2	0	1	14
	O^2_{2k}	M_2	1	4	3
	O^3_{2k}	M_4	4	6	13
J_3	O^1_{3k}	M_3	0	3	5
	O^2_{3k}	M_4	3	4	18

Table 4
Scheduling of MF01 (MOHEA) (f_1: 6, f_2: 66)

J_j	O^i_{jk}	Scheduling (MF01) - f_1: 6, f_2: 66			
		A^i_{ik}	t^i_{ik}	$t^i_{ik} + \tau^i_{ik}$	μ^i_{ik}
J_1	O^1_{1k}	M_4	0	1	4
	O^2_{1k}	M_4	1	2	12
	O^3_{1k}	M_1	3	6	4
J_2	O^1_{2k}	M_2	0	1	4
	O^2_{2k}	M_2	1	3	9
	O^3_{2k}	M_4	3	4	10
J_3	O^1_{3k}	M_3	1	4	5
	O^2_{3k}	M_4	4	5	18

Table 5
Scheduling of MF01 (MOHEA) (f1: 57, f2: 35)

J_j	O^i_{jk}	Scheduling (MF01) - f_1: 57, f_2: 35			
		A^i_{jk}	t^i_{jk}	$t^i_{jk} + \tau^i_{jk}$	μ^i_{jk}
J_1	O^1_{1k}	M_4	0	4	6
	O^2_{1k}	M_4	4	12	2
	O^3_{1k}	M_1	12	19	3
J_2	O^1_{2k}	M_2	19	23	6
	O^2_{2k}	M_2	23	32	3
	O^3_{2k}	M_4	32	41	3
J_3	O^1_{3k}	M_3	41	49	6
	O^2_{3k}	M_4	49	57	6

Table 6
Scheduling of MF01 (MOHEA) (f1: 29, f2: 35)

J_j	O^i_{jk}	Scheduling (MF01) - f_1: 29, f_2: 35			
		A^i_{jk}	t^i_{jk}	$t^i_{jk} + \tau^i_{jk}$	μ^i_{jk}
J_1	O^1_{1k}	M_4	0	4	6
	O^2_{1k}	M_4	4	12	2
	O^3_{1k}	M_1	12	19	3
J_2	O^1_{2k}	M_2	0	4	6
	O^2_{2k}	M_2	16	20	3
	O^3_{2k}	M_4	20	29	3
J_3	O^1_{3k}	M_3	0	8	6
	O^2_{3k}	M_4	8	16	6

The procedure has been applied to problems MF01 (Fig. 5 (a)), MF02 (Fig. 6 (a)), MF03 (Fig. 7 (a)), MF04 (Fig. 8 (a)) and MF05 (Fig. 9 (a)) (Kacem et al., 2002) and the non-dominated solutions (S) reached under a number of generations (G) are obtained. Then, a multi-objective analysis based on Makespan (f_1) and Total Operation Costs (f_2) is iterated 30 times. For each algorithm the sets of undominated solutions $P_1, P_2, ..., P_{30}$ were obtained as well as the super-population $P_T = P_1 \cup P_2 \cup ... \cup P_{30}$. From each superpopulation a class of undominated solutions was extracted, constituting the Pareto frontier for each algorithm: Y_{NSGAII}, Y_{NSGA}, Y_{SPEAII} and Y_{SPEA}. The mean times required for completing 250 generations by the different algorithms are shown in Table 7.

Table 7
Mean running times of the algorithms. Each algorithm is iterated 30 times

	Mean Running Time (in seconds)			
	NSGAII	NSGA	SPEAII	SPEA
MF01	112,3	98,5	110,9	102,5
MF02	195,8	175,2	185,7	181,1
MF03	221,3	197,9	214,2	204,6
MF04	402,9	360,4	382,0	372,6
MF05	531,8	475,7	514,8	491,9

The fronts obtained are shown in Fig. 5 (b) (MF01), Fig. 6 (b) (MF02), Fig. 7 (b) (MF03), Fig. 8 (b) (MF04) and Fig. 9 (b) (MF05). To obtain an approximation to the true Pareto front (Approximate Pareto Frontier, APF) we take the entire class $Y_{NSGAII} \cup Y_{NSGA} \cup Y_{SPEAII} \cup Y_{SPEA}$, from which all the dominated solutions are eliminated.

4.2 Comparison Procedure

In order to compare the results of the algorithms and establish the better option for the FJSSP, several tests were applied over the solutions.

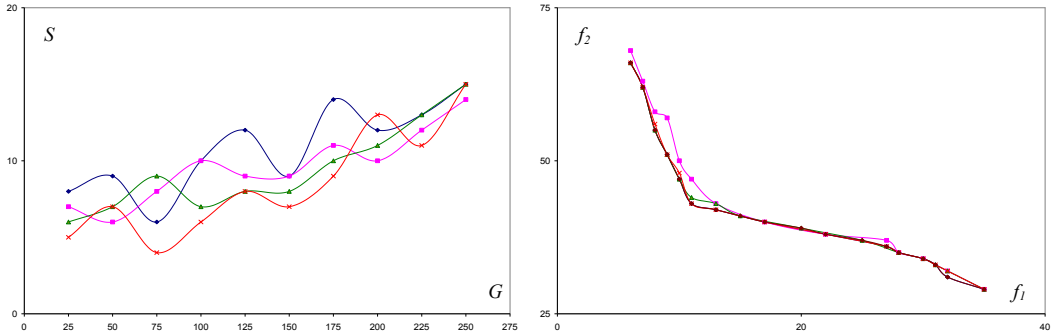

Fig. 5. MF01. NSGAII ◆, NSGA ■, SPEAII ▲, SPEA × and APF ◆

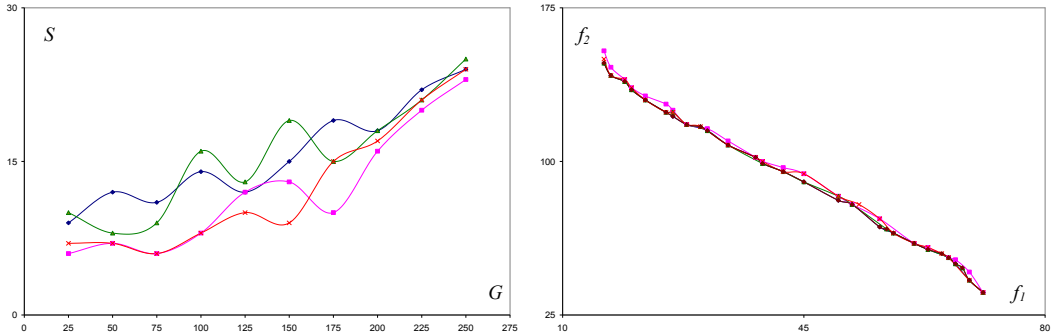

Fig. 6. MF02. NSGAII ◆, NSGA ■, SPEAII ▲, SPEA × and APF ◆

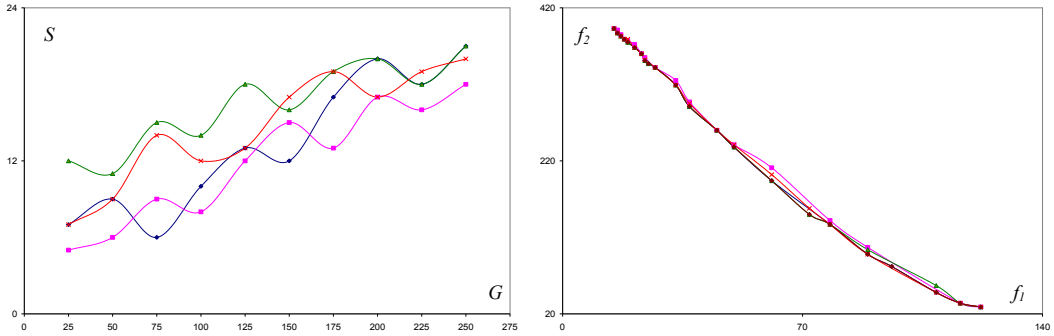

Fig. 7. MF03. NSGAII ◆, NSGA ■, SPEAII ▲, SPEA × and APF ◆

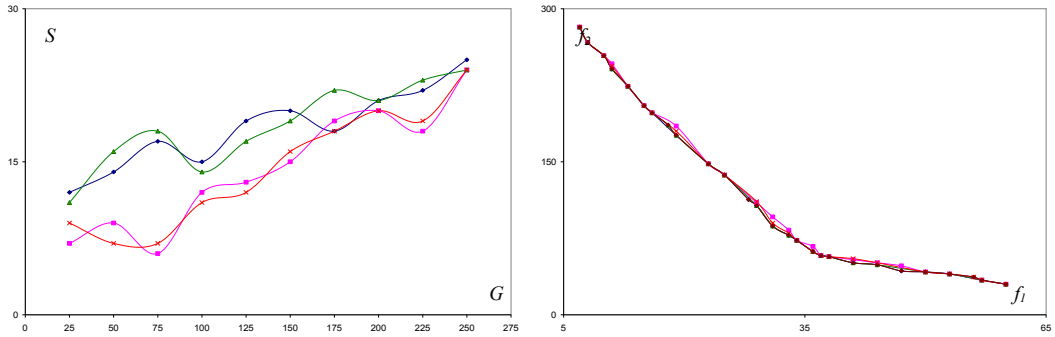

Fig. 8. MF04. NSGAII ◆, NSGA ■, SPEAII ▲, SPEA × and APF ◆

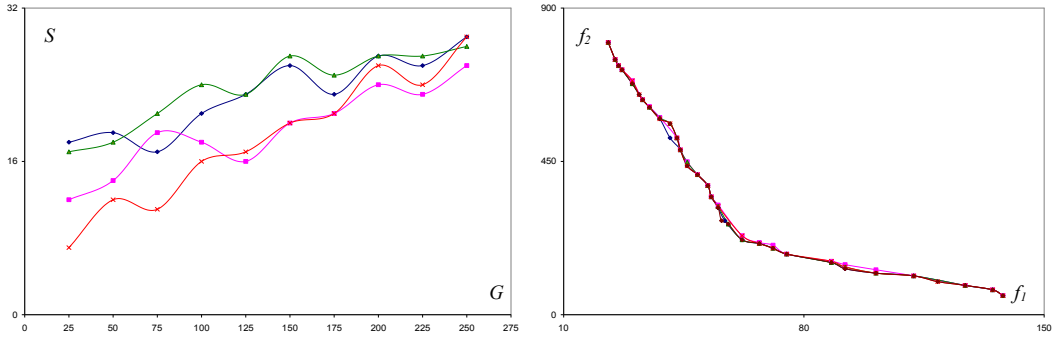

Fig. 9. MF05. NSGAII ◆, NSGA ■, SPEAII ▲, SPEA × and APF ◆

We considered unary quality indicators using normalized approximation sets. Then, we applied the unary indicators (unary epsilon indicator I_e^1, unary hypervolume indicator I_H and R indicator I_{R2}^1) on the normalized approximation sets as well as on the reference set generated by PISA (I_e^1, I_H, and I_{R2}^1, Table 8 (MF01), Table 9 (MF02), Table 10 (MF03), Table 11 (MF04) and Table12 (MF05).

Table 8
Unary epsilon indicator, unary hypervolume indicator and R indicator (MF01)

Test for Problem MF01				
		I_e^1		
	NSGAII	NSGA	SPEAII	SPEA
NSGAII	-	0,02837	0,38665	0,36578
NSGA	0,97163	-	0,72365	0,70856
SPEAII	0,61335	0,27635	-	0,49144
SPEA	0,63422	0,29144	0,50856	-
		I_H		
	NSGAII	NSGA	SPEAII	SPEA
NSGAII	-	0,02802	0,38193	0,36131
NSGA	0,97198	-	0,71481	0,69990
SPEAII	0,61807	0,28519	-	0,48543
SPEA	0,63869	0,30010	0,51457	-
		I_{R2}^1		
	NSGAII	NSGA	SPEAII	SPEA
NSGAII	-	0,02802	0,38193	0,36131
NSGA	0,97198	-	0,71481	0,69990
SPEAII	0,61807	0,28519	-	0,48543
SPEA	0,63869	0,30010	0,51457	-

Table 9
Unary epsilon indicator, unary hypervolume indicator and R indicator (MF02)

Test for Problem MF02				
I_ε^1				
	NSGAII	NSGA	SPEAII	SPEA
NSGAII	-	0,03649	0,47696	0,04688
NSGA	0,96351	-	0,89319	0,55750
SPEAII	0,52304	0,10681	-	0,59746
SPEA	0,95312	0,44250	0,40254	-
I_H				
	NSGAII	NSGA	SPEAII	SPEA
NSGAII	-	0,03726	0,48697	0,04786
NSGA	0,96274	-	0,91195	0,56921
SPEAII	0,51303	0,08805	-	0,61001
SPEA	0,95214	0,43079	0,38999	-
I_{R2}^1				
	NSGAII	NSGA	SPEAII	SPEA
NSGAII	-	0,03702	0,48387	0,04756
NSGA	0,96298	-	0,90614	0,56558
SPEAII	0,51613	0,09386	-	0,60613
SPEA	0,95244	0,43442	0,39387	-

Table 10
Unary epsilon indicator, unary hypervolume indicator and R indicator (MF03)

Test for Problem MF03				
I_ε^1				
	NSGAII	NSGA	SPEAII	SPEA
NSGAII	-	0,03739	0,48868	0,04803
NSGA	0,96261	-	0,91515	0,57121
SPEAII	0,51132	0,08485	-	0,61215
SPEA	0,95197	0,42879	0,38785	-
I_H				
	NSGAII	NSGA	SPEAII	SPEA
NSGAII	-	0,03793	0,49577	0,04873
NSGA	0,96207	-	0,92842	0,57949
SPEAII	0,50423	0,07158	-	0,62103
SPEA	0,95127	0,42051	0,37897	-
I_{R2}^1				
	NSGAII	NSGA	SPEAII	SPEA
NSGAII	-	0,03690	0,48233	0,04740
NSGA	0,96310	-	0,90326	0,56378
SPEAII	0,51767	0,09674	-	0,60420
SPEA	0,95260	0,43622	0,39580	-

Table 11
Unary epsilon indicator, unary hypervolume indicator and R indicator (MF04)

Test for Problem MF04				
I_ε^1				
	NSGAII	NSGA	SPEAII	SPEA
NSGAII	-	0,07868	0,49118	0,10167
NSGA	0,92132	-	0,92328	0,63990
SPEAII	0,50882	0,07672	-	0,10063
SPEA	0,89833	0,36010	0,89937	-
I_H				
	NSGAII	NSGA	SPEAII	SPEA
NSGAII	-	0,07679	0,47939	0,09923
NSGA	0,92321	-	0,90113	0,62454
SPEAII	0,52061	0,09887	-	0,09821
SPEA	0,90077	0,37546	0,90179	-
I_{R2}^1				
	NSGAII	NSGA	SPEAII	SPEA
NSGAII	-	0,07868	0,49118	0,10167
NSGA	0,92132	-	0,92328	0,63990
SPEAII	0,50882	0,07672	-	0,10063
SPEA	0,89833	0,36010	0,89937	-

On problems MF01, MF04 and MF05, NSGAII showed statistically significant differences with NSGA and SPEA at the α=0.05 level. On MF02 and MF03, NSGAII and SPEAII had differences with NSGA y SPEA of an overall significance level α=0.05. Thus, NSGAII and SPEAII address better the FJSSP. As a further step in the analysis, we establish the percentage of contribution of each algorithm to the Approximate Pareto Frontier (see Table 13). From this we can conclude that NSGAII is the best selector we can apply to our problem.

Table 12
Unary epsilon indicator, unary hypervolume indicator and R indicator (MF05)

Test for Problem MF05				
I_ϵ^1				
	NSGAII	NSGA	SPEAII	SPEA
NSGAII	-	0,02906	0,39616	0,37477
NSGA	0,97094	-	0,74145	0,72598
SPEAII	0,60384	0,25855	-	0,49648
SPEA	0,62523	0,27402	0,50352	-
I_H				
	NSGAII	NSGA	SPEAII	SPEA
NSGAII	-	0,02838	0,38691	0,36602
NSGA	0,97162	-	0,72413	0,70903
SPEAII	0,61309	0,27587	-	0,48488
SPEA	0,63398	0,29097	0,51512	-
I_{R2}^1				
	NSGAII	NSGA	SPEAII	SPEA
NSGAII	-	0,02869	0,39101	0,36990
NSGA	0,97131	-	0,73181	0,71654
SPEAII	0,60899	0,26819	-	0,49002
SPEA	0,63010	0,28346	0,50998	-

Table 13
Percentage of solutions contributed by NSGAII, NSGA, SPEAII and SPEA to the Approximate Pareto Frontier

Percentage of solutions in the Approximate Pareto Frontier				
	NSGAII	NSGA	SPEAII	SPEA
MF01	83,33%	27,78%	66,67%	66,67%
MF02	85,71%	21,43%	82,14%	53,57%
MF03	95,45%	31,82%	86,36%	63,64%
MF04	92,59%	55,56%	88,89%	62,96%
MF05	93,55%	48,39%	77,42%	77,42%

5. Conclusions

We presented a Multi-Objective Hybrid Evolutionary Algorithm (MOHEA) to solve the Flexible Job-Shop Scheduling Problem (FJSSP). The application of the MOHEA required the calibration of parameters to yield valid values. Our algorithm integrates two meta-heuristic procedures: a Multi-Objective Evolutionary Algorithm (MOEA) and a Multi-Objective Simulated Annealing (MOSA) algorithm. Individuals are coded in a way that facilitates the application of two basic genetic operators. Different MOEAs were tested for this task. It was shown that the performance of NSGAII is at least as good as SPEAII and it improves largely over NSGA and SPEA, validating the results in (Frutos et al., 2010). We are currently running a comparison between the MOHEA presented in this paper and recently developed approaches like the Hybrid Artificial Bee Colony Algorithm (Li et al., 2011) and the Multi-population Interactive Co-evolutionary Algorithm (Xing et al., 2011). Furthermore, we plan, in the future, to explore the performance of the MOHEA on other MOPs. We believe that it provides a strong and efficient approach to this kind of problems.

Acknowledgements

We would like to thank the economic support of the Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET) and the Universidad Nacional del Sur (UNS) for Grant PGI 24/JO56. We want also thank Dr. Ana C. Olivera for her constant support and help during this research.

References

Adams, J., Balas, E. & Zawack, D. (1998). The shifting bottleneck procedure for Job Shop Scheduling. *Management Science*, 34 (3), 391-401.

Al-Hinai, N. & ElMekkawy, T. Y. (2011) Robust and stable flexible job shop scheduling with random machine breakdowns using a hybrid genetic algorithm. *International Journal of Production Economics*, 132 (2), 279-291.

Armentano, V. A. & Scrich, C. R. (2000). Tabu Search for minimizing total tardiness in a Job-Shop. *International Journal Production Economics*, 63, 131-140.

Bihlmaier, R., Koberstein, A. & Obst, R. (2009). Modeling and optimizing of strategic and tactical production planning in the automotive industry under uncertainty. *OR Spectrum*, 31 (2), 311-336.

Bleuler, S., Laumanns, M., Thiele, L. & Zitzler, E. (2003). *PISA: a platform and programming language independent interface for search algorithms*. In: Proceedings of Evolutionary Multi-Criterion Optimization, Springer-Verlag, Berlin, 494-508.

Chao-Hsien, J. & Han-Chiang, H. (2009). A hybrid genetic algorithm for no-wait Job Shop Scheduling problems. *Expert Systems with Applications*, 36 (3), 5800-5806.

Chiang, T. C. & Lin, H. J. (2013). A simple and effective evolutionary algorithm for multiobjective flexible job shop scheduling. *International Journal of Production Economics*, 141 (1), 87-98.

Chinyao, L. & Yuling, Y. (2009). Genetic algorithm-based heuristics for an open shop scheduling problem with setup, processing, and removal times separated. *Robotics and Computer-Integrated Manufacturing*, 25 (2), 314-322.

Coello Coello, C. A., Lamont, G. B. & Veldhuizen, D. A. (2006). *Evolutionary Algorithms for Solving Multi-Objective Problems*. Genetic and Evolutionary Computation, Springer-Verlag, NY.

Cortés Rivera, D., Coello Coello, C. A. & Cortés, N. C. (2003). Use of an artificial immune system for Job-Shop Scheduling. *Lecture Notes in Computer Science*, 2787, 1-10.

De Giovanni, L. & Pezzella, F. (2010). An improved genetic algorithm for the distributed and flexible Jobshop Scheduling problem. *European Journal of Operational Research*, 200 (2), 395-408.

Deb, K., Pratap, A., Agarwal, S. & Meyarivan, T. (2002). A fast and elitist multi-objective genetic algorithm: NSGAII. *IEEE Transactions on Evolutionary Computation*, 6 (2), 182-197.

Della Croce, F., Grosso, A. & Salassa, F. (2014). A metaheuristic approach for the two-machine total completion time flow-shop problem. *Annals of Operations Research*, 213, 67-78.

Frutos, M., Olivera, A. C. & Tohmé, F. (2010). A memetic algorithm based on a NSGAII scheme for the flexible Job-Shop Scheduling problem. *Annals of Operations Research*, 181, 745-765.

Frutos, M., Méndez, M., Tohmé, F. & Broz, D. (2013). Comparison of Multiobjective Evolutionary Algorithms for Operations Scheduling under Machine Availability Constraints. *The Scientific World Journal: Bioinspired Computation and Its Applications in Operation Management*, 2013, 1-9.

Gao, J., Sun, L. & Gen, M. (2008). A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems. *Computers and Operations Research*, 35, 2892-2907.

Heinonen, J. & Pettersson, F. (2007). Hybrid ant colony optimization and visibility studies applied to a Job-Shop Scheduling problem. *Applied Mathematics and Computation*, 187 (2), 989-998.

Kacem, I., Hammadi, S. & Borne, P. (2002). Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems. *IEEE Transactions on Systems, Man and Cybernetics, Part C*, 32, 1-13.

Knowles, J., Thiele, L. & Zitzler, E. (2005). *A tutorial on the performance assessment of stochastic multiobjective optimizers*. TIK Computer Engineering and Networks Laboratory.

Li, J., Pan, Q., Xie, S. & Wang, S. (2011). A hybrid artificial bee colony algorithm for flexible Job Shop Scheduling problems. *International Journal of Computers Communications & Control*, 6 (2), 286-296.

Lin, Y., Pfund, M. & Fowler, J. (2011). Heuristics for minimizing regular performance measures in unrelated parallel machine scheduling problems. *Computers & Operations Research*, 38 (6), 901-916.

Liu, C. H., Chen, L. S. & Lin, P. S. (2013). Lot streaming multiple jobs with values exponentially deteriorating over time in a job-shop environment. International Journal of Production Research, 51 (1), 202-214.

Mönch, L. & Zimmermann, J. (2011). A computational study of a shifting bottleneck heuristic for multi-product complex job shops. *Production Planning & Control*, 22 (1), 25-40.

Panwalker, S. & Iskander, W. (1977). A survey of scheduling rules. *Operations Research*, 25 (1), 45-61.

Papadimitriou, C. H. (1994). *Computational Complexity*. Addison Wesley, NY.

Park, B. J., Choi, H. R. & Kim, H. S. A. (2003). Hybrid genetic algorithm for the Job-Shop Scheduling problems. *Computers and Industrial Engineering*, 45 (1), 597-613.

Rabiee, M., Zandieh, M. & Ramezani, P. (2012). Bi-objective partial flexible job shop scheduling problem: NSGA-II, NRGA, MOGA and PAES approaches. *International Journal of Production Research*, 50 (24), 7327-7342.

Shin, J. G., Kwon, O. H. & Ryu, C. (2008). Heuristic and metaheuristic spatial planning of assembly blocks with process schedules in an assembly shop using differential evolution. *Production Planning & Control*, 19 (6), 605-615.

Srinivas, N. (1994). Multi-objetive optimization using nondominated sorting in genetic algorithms. Master's Thesis. *Indian Institute of Technology, Kuanpur, India*.

Storer, R. H., Wu, S. D. & Vaccari, R. (1992). New search spaces for sequencing instances with application to Job-Shop Scheduling. *Management Science*, 38, 1495-1509.

T`kindt, V. & Billaut, J. C. (2006). *Multicriteria Scheduling. Theory, Models and Algorithms*, Springer-Verlag, Berlin.

Tsai, C. F. & Lin, F. C. (2003). A new hybrid heuristic technique for solving Job-Shop Scheduling problems. *In: Proceedings of the Second IEEE International Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IEEE, Kharkov, Ukraine*, 53-58.

Ullman, J. D. (1975). NP-complete scheduling problems. *Journal of Computer System Sciences*, 10, 384-393.

Van Laarhoven, P. J. M., Aarts, E. H. L. & Lenstra, J. K. (1992). Job-Shop Scheduling by simulated annealing. *Operations Research*, 40 (1), 113-125.

Varadharajan, T. K. & Rajendran, C. (2005). A multi-objective simulated-annealing algorithm for scheduling in flowshops to minimize the makespan and total flowtime of jobs. *European Journal of Operational Research*, 167, 772-795.

Wu, C. G., Xing, X. L., Lee, H. P., Zhou, C. G. & Liang, Y. C. (2004). Genetic algorithm application on the Job-Shop Scheduling problem. *In: Proceedings of the 2004 International Conference Machine Learning and Cybernetics*, 4, 2102-2106.

Xing, L. N., Chen, Y. W. & Yang, K. W. (2011). Multi-population interactive coevolutionary algorithm for flexible Job Shop Scheduling problems. *Computational Optimization and Applications*, 48, 139-155.

Zitzler, E., Laumanns, M. & Thiele, L. (2002). SPEAII: improving the strength Pareto evolutionary algorithm for multiobjective optimization. In Giannakoglou, Tsahalis, Periaux, Papailiou, and Fogarty (eds), *Evolutionary Methods for Design, Optimisation and Control, CIMNE, Barcelona, Spain*, 19-26.

Zitzler, E. & Thiele, L. (1999). Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. *IEEE Transactions Evolutionary Computation*, 3 (3), 257-271.