A Multi-objective Memetic Algorithm for the Job-Shop Scheduling Problem

Mariano Frutos & Fernando Tohmé

Operational Research
An International Journal

ISSN 1109-2858
Volume 13
Number 2

Oper Res Int J (2013) 13:233-250
DOI 10.1007/s12351-012-0125-y

Springer

Oper Res Int J (2013) 13:233–250
DOI 10.1007/s12351-012-0125-y

ORIGINAL PAPER

A Multi-objective Memetic Algorithm for the Job-Shop Scheduling Problem

Mariano Frutos · Fernando Tohmé

Received: 1 October 2011 / Revised: 3 March 2012 / Accepted: 9 April 2012 /
Published online: 25 April 2012

Abstract Planning means, in the realm of production activities, to design, coordinate, manage and control all the operations involved in the production system. Many MOPs (multi-objective optimization problems) are generated in this framework. They require the optimization of several functions that are usually very complex, which makes the search for solutions very expensive. Multi-objective optimization seeks Pareto-optimal solutions for these problems. In this work we introduce, a Multi-objective Memetic Algorithm intended to solve a very important MOP in the field, namely, the Job-Shop Scheduling Problem. The algorithm combines a MOEA (Multi-Objective Evolutionary Algorithm) and a path-dependent search algorithm (Multi-objective Simulated Annealing), which is enacted at the genetic phase of the procedure. The joint interaction of those two components yields a very efficient procedure for solving the MOP under study. In order to select the appropriate MOEA both NSGAII and SPEAII as well as their predecessors (NSGA and SPEA) are pairwise tested on problems of low, medium and high complexity. We find that NSGAII yields a better performance, and therefore is the MOEA of choice.

Keywords Multi-objective optimization · Production · Job-Shop Scheduling Problem · Multi-objective Memetic Algorithm

M. Frutos (✉)
Department of Engineering, Universidad Nacional del Sur and CONICET,
Av. Alem 1253, B8000CPB Bahía Blanca, Argentina
e-mail: mfrutos@uns.edu.ar

F. Tohmé
Department of Economics, Universidad Nacional del Sur and CONICET,
12 de Octubre y San Juan, B8000CPB Bahía Blanca, Argentina
e-mail: ftohme@criba.edu.ar

1 Introduction

The main purpose of production planning activities in firms is to improve the efficiency of processes (Bihlmaier et al. 2009). One way of conceiving a good plan in an industrial firm is as a solution to a Job-Shop Scheduling Problem (JSSP) (Chao-Hsien and Han-Chiang 2009). The main possible drawback of this approach is that this problem is hard to solve, being a member of the class NP-Hard (Ullman 1975; Papadimitriou 1994). The JSSP is concerned with the allocation of limited resources to scheduled jobs in order to optimize one or more objectives (Armentano and Scrich 2000; Storer et al. 1992). Evolutionary algorithms have become popular to handle these multi-objective problems (Deb et al. 2002; Coello Coello et al. 2006). Following this trend, we present here a Multi-Objective Evolutionary Algorithm (MOEA) linked to a local search procedure (MOSA, Multi-Objective Simulated Annealing) in order to solve a JSSP (Cortés Rivera et al. 2003; Park et al. 2003; Tsai and Lin 2003; Wu et al. 2004). Even if the literature on JSSP is already large, most of the work has focused on a single objective, despite that in applications multiple goals are pervasive (Chinyao and Yuling 2009). As T'kindt and Billaut (2006) claim, a genuine scheduling problem essentially involves the optimization of many simultaneous objectives.

1.1 JSSP treatments: state of the art

The huge literature on the topic presents a variety of solution strategies that go from simple priority rules to sophisticated parallel branch-and-bound algorithms. A particular variety of scheduling problem is the JSSP. Muth and Thompson's 1964 book Industrial Scheduling presented the JSSP, basically in its currently known form. Even before, Jackson in 1956 generalized the flow-shop algorithm of Johnson (1954) to yield a job-shop algorithm. In 1955, Akers and Friedman gave a Boolean representation of the procedure, which later Roy and Sussman (1964) described by means of a disjunctive graph, while Egon Balas, already in 1959, applied an enumerative approach that could be better understood in terms of this graph. Giffler and Thompson (1960) presented an algorithm based on rule priorities to guide the search. For these reasons, the problem was already part of the folklore of Operations Research years before its official inception. The JSSP generated a huge literature. Its resiliency made it an ideal problem for further study. Besides, its usefulness made it a problem worth to scrutinize. Due to its complexity, several alternative presentations of the problem have been tried (Sadeh and Fox 1995; Chinyao and Yuling 2009; Della Croce et al. 2011; Lin et al. 2011), in order to apply particular algorithms like Priority Rules (Panwalker and Iskander 1977), Shifting Bottlenecks (Adams et al. 1998), Simulated Annealing (Van Laarhoven et al. 1992), Tabu Search (Dell Amico and Trubian 1993; Armentano and Scrich 2000; Nowicki and Smutnicki 2005), the Branch and Bound Algorithm (Brucker et al. 1994), Genetic Algorithms (Zalzala and Flemming 1997), Ant Colony Optimization (Merkle and Middendorf 2001), Clonal Selection (Cortés Rivera et al. 2003), Solution-Guided Multi-Point Constructive Search (Beck (2007), Hybrid Algorithms (Zhang et al. 2008), etc. The performance of these meta-heuristic procedures varies, and some

seem fitter than others (De Giovanni and Pezzella 2010), although unlike our treatment of the problem, most of them focus on a single objective (usually the minimization of makespan). Since in our case we consider an additional objective, that may or may not be aligned with the minimization of makespan, most of the approaches mentioned above may improve the latter worsening the former.

1.2 Multi-objective optimization: basic concepts

Our goal in this section is to characterize the general framework in which we will state the Job-Shop problem. We assume, without loss of generality, that there are several goals (objectives) to be minimized. Then, we seek to find a vector $\vec{x}^* = [x_1^*, \ldots, x_n^*]^T$ of decision variables, satisfying q inequalities $g_i(\vec{x}) \geq 0$, $i = 1, \ldots, q$ as well as p equations $h_i(\vec{x}) = 0$, $i = 1, \ldots, p$, such that $\vec{f}(\vec{x}) = [f_1(\vec{x}), \ldots, f_k(\vec{x})]^T$, a vector of k functions, each one corresponding to an objective, defined over the decision variables, attains its minimum. The class of the decision vectors satisfying the q inequalities and the p equations is denoted by Ω and each $\vec{x} \in \Omega$ is a feasible alternative. A $\vec{x}^* \in \Omega$ is Pareto optimal if for any $\vec{x} \in \Omega$ and every $i = 1,\ldots,k, f_i(\vec{x}^*) \leq f_i(\vec{x})$. That is, if there is no $\vec{x}$ that improves some objectives without worsening the others. To simplify the notation, we say that a vector $\vec{u} = [u_1, \ldots, u_n]^T$ dominates another, $\vec{v} = [v_1, \ldots, v_n]^T$ (denoted $\vec{u} \prec \vec{v}$) if and only if $\forall i \in \{1, \ldots, k\}$, $u_i \leq v_i \wedge \exists i \in \{1, \ldots, k\} : u_i < v_i$. Then, the set of Pareto optima is $P^* = \{\vec{x} \in \Omega \,|\, \neg \exists \vec{x}' \in \Omega, \vec{f}(\vec{x}') \prec \vec{f}(\vec{x})\}$ while the corresponding Pareto frontier is $PF^* = \{\vec{f}(\vec{x}), \vec{x} \in P^*\}$. The search of the Pareto frontier is the main goal of Multi-Objective Optimization. Given the complexity of the task, a good approximation consists in finding a few feasible alternatives, close enough to the frontier. These alternatives should be uniformly distributed, just to facilitate the interpretation of the results.

2 The Job-Shop Scheduling Problem

Optimal scheduling is a problem of high theoretical and practical importance that has been extensively analyzed during the last decades. The formal description of the JSSP requires considering *n* jobs to be carried out on *m* machines. In turn, each job is decomposed in operations (Frutos et al. 2010). As initial data we have the sequence of machines for each task and the processing time of each operation (Adams et al. 1998) (Table 1).

We assume that at the start of the process each machine is available and that it can only carry out an operation at a time. Furthermore, no job can use more than one time each machine. A job has to wait until the next machine is available (De Giovanni and Pezzella 2010). 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 and Pettersson 2007). While there might exist many objectives for the JSSP, we will focus on the minimal time required in completing a job (Makespan, see Eq. 1) and the minimal mean time for completing all the jobs (Mean Flow Time, see Eq. 2).

Table 1 A JSSP [la02-10 × 5 (Beasley 1990)]

		Operations									
		1		2		3		4		5	
		M_k	τ_{ij}^k	M_k	τ_{ij}^k	M_k	τ_{ij}^k	M_k	τ_{ij}^k	M_k	τ_{ij}^k
Jobs	1	1	20	4	87	2	31	5	76	3	17
	2	5	25	3	32	1	24	2	18	4	81
	3	2	72	3	23	5	28	1	58	4	99
	4	3	86	2	76	5	97	1	45	4	90
	5	5	27	1	42	4	48	3	17	2	46
	6	2	67	1	98	5	48	4	27	3	62
	7	5	28	2	12	4	19	1	80	3	50
	8	2	63	1	94	3	98	4	50	5	80
	9	5	14	1	75	3	50	2	41	4	55
	10	5	72	1	18	2	37	4	79	1	61

We can see that each operation of a job is allocated to a given machine. Besides, the length of each operation and its place in the sequence is represented in the table

$$f_1 : C_{max}^j = \sum_{i \in O(j)} \max_{k \in M} \left(t_{ij}^k + \tau_{ij}^k \right) \tag{1}$$

$$f_2 : \overline{F} = \frac{1}{n} \sum_{j \in J} \max_{h,k \in M} \left(t_{l_j j}^h + \tau_{l_j j}^h - t_{l_j}^k \right) \tag{2}$$

s.t.

$t_{ij}^k \geq 0, \quad \forall j \in J, \quad \forall i \in O(j), \quad \forall k \in M$

(Starting time constraint)

$t_{ij}^k - t_{sj}^h \geq \tau_{sj}^h, \quad if \quad O_{sj}^h \quad precedes \quad O_{ij}^k, \quad \forall j \in J, \quad \forall i,s \in O(j), \quad \forall k,h \in M$

(Precedence constraint)

$t_{ij}^k - t_{sp}^k \geq \tau_{sp}^k \quad or \quad t_{sp}^k - t_{ij}^k \geq \tau_{ij}^k, \quad if \quad O_{sp} \quad \text{and} \quad O_{ij}, \quad \text{require} \quad k \in M,$
$\forall j,p \in J, \quad \forall i \in O(j), \quad \forall s \in O(p), \quad \forall k \in M$

(Disjunctive constraint)

where J is the class of n jobs, O(j) the class of operations to be carried out for job $j \in J$ and M the class of m machines. O_{ij}^k the operation $i \in O(j)$ carried out on machine k, t_{ij}^k is the starting time of operation $i \in O(j)$ on machine k, l_j is the last operation in job j, while τ_{ij}^k is the time that takes carrying out operation $i \in O(j)$ on machine k.

3 The Multi-objective Memetic Algorithm

Evolutionary algorithms have been widely applied to optimization problems, due to their many advantages (Coello Coello et al. 2006). But for the JSSP, in particular,

their high rate of convergence generate also high evaluation costs when many objectives are at stake, leading to a loss of diversity in the class of solutions. This is reflected by poorly distributed Pareto frontiers. But if evolutionary algorithms are complemented by efficient local search procedures the whole multi-objective procedure requires a very few evaluations of the fitness functions while yielding a better distributed Pareto frontier (see Fig. 1). This kind of procedure, mixing an evolutionary algorithm and local search, is called a Multi-objective Memetic Algorithm (Ishibuchi et al. 2003). As in the realm of Biology, in such procedure the chromosome evolves, but is also subject to alterations that are bequest to it descendants.

We introduce here a Multi-objective Memetic Algorithm for the treatment of the JSSP combining a Multi-Objective Evolutionary Algorithm (MOEA), and Multi-Objective Simulated Annealing (MOSA) (Varadharajan and Rajendran 2005) for improving the individuals in the population.

3.1 Evolutionary process

Individuals in the population are possible candidates to be solutions of the JSSP. In the literature chromosomes, describing the ordering and timing of operations, are usually of size m x n. Interventions inside the chromosomes along the critical path have shown to improve the efficiency in the optimization of Makespan. But with the addition of the goal of optimizing Mean Flow Time, those interventions make no longer sense. This is why after some experiments Croce and Tadei's (1995) m-sized chromosome representation was chosen. Each entry indicates an ordered list of integers between 0 and n! − 1, standing for the order of tasks assigned to each machine. For instance, in la02, m = 5 and n = 10 while, say, 0 is the sequence 1|2|3|4|5|6|7|8|9|10, whereas 1 → 1|2|3|4|5|6|7|8|9|10| 9, 2 → 1|2|3|4|5|6|7|8|9, …. If the chromosome is coded as [0 2 1 0 1], the decoding process yields the sequences

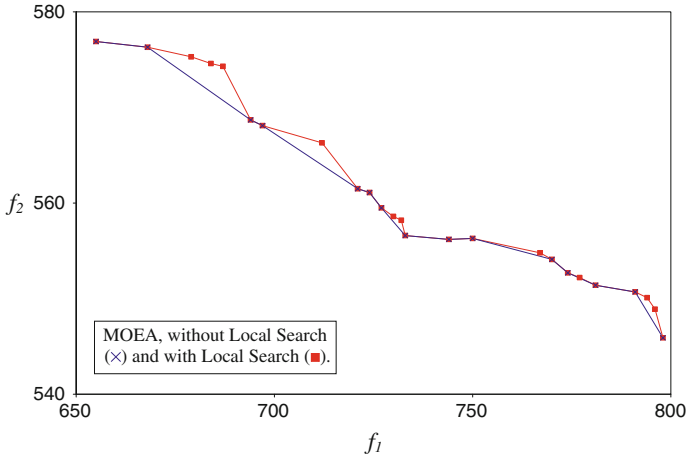

Fig. 1 Makespan versus mean flow time [la02-10 × 5 (Beasley 1990)]. MOEA, without Local Search (*times*) and with Local Search (*red square*) (color figure online)

Table 2 Sequences obtained from decoding the results of (la02)

		Operations				
		1 M_k	2 M_k	3 M_k	4 M_k	5 M_k
Jobs	1	$1^{(1)}$	$4^{(1)}$	$2^{(1)}$	$5^{(1)}$	$3^{(1)}$
	2	$5^{(2)}$	$3^{(2)}$	$1^{(2)}$	$2^{(2)}$	$4^{(2)}$
	3	$2^{(3)}$	$3^{(3)}$	$5^{(3)}$	$1^{(3)}$	$4^{(3)}$
	4	$3^{(4)}$	$2^{(4)}$	$5^{(4)}$	$1^{(4)}$	$4^{(4)}$
	5	$5^{(5)}$	$1^{(5)}$	$4^{(5)}$	$3^{(5)}$	$2^{(5)}$
	6	$2^{(6)}$	$1^{(6)}$	$5^{(6)}$	$4^{(6)}$	$3^{(6)}$
	7	$5^{(7)}$	$2^{(7)}$	$4^{(7)}$	$1^{(7)}$	$3^{(7)}$
	8	$2^{(10)}$	$1^{(8)}$	$3^{(8)}$	$4^{(8)}$	$5^{(8)}$
	9	$5^{(10)}$	$1^{(9)}$	$3^{(10)}$	$2^{(8)}$	$4^{(9)}$
	10	$5^{(9)}$	$3^{(9)}$	$2^{(9)}$	$4^{(10)}$	$1^{(10)}$

shown in Table 2. The most costly activity in the algorithm involves decoding the chromosome, because it requires a large number of lookups into the table of codes. Each change in a gene affects the sequence of operations in the machines, allowing the use of basic genetic operators.

But that is not a real problem, since the initial population, which is rather small (only one in fifty alternatives is chosen) is generated in a random way. The algorithm assigns an integer value between 0 and $n! - 1$ to the first machine. It is chosen from a uniform distribution on the values. Then, the algorithm repeats this for the $m - 1$ remaining machines. The crossover operator works as usual, using the parents obtained from the selection process. After some tests, we selected the Uniform Crossover operator, because it yields the best results. We included also a mutation operator, since in JSSP the crossover alone does not allow to reach certain areas of the search space. We applied the mutation consisting in swapping two genes (Two-Swap). The operator takes the chain of integers from a child and selects at random two genes, swapping their positions, leaving the machine order intact, which is consistent with the fact that the problem studied here is the fixed-order version of the JSSP.

3.2 Simulated annealing as a local search process

Simulated Annealing provides a search procedure that applies a probabilistic acceptance criterion, based on thermodynamic principles. To avoid getting trapped in a local optimum, something that tends to happen with traditional local search algorithms, random jumps to possible worse solutions are allowed. On the other hand, to lead the search to better solutions, some control has to be exerted on these jumps. Simulated Annealing does this by controlling the frequency of jumps by means of the probability function $e^{-(\delta/T)}$, where δ is the difference among the 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 at a higher temperature more probable is to get poorer solutions, the procedure induces more diversity in the beginning but focusing on improving 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 is applied to check out whether the current solution is kept or not. In the case of N objectives, there exist several alternative definitions of δ. We will, in particular, take δ as the normalized maximum deviation, $\delta = \max[(f_i(x') - f_i(x))/f_i(x)]$.

If a new solution were rejected, a slight variation of the previous one would be tried. The non-zero probability of accepting a worse solution frees the algorithm from the possibility of getting caught in a local minimum. Through 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 (more involved stopping criteria can be also implemented). Van Laarhoven et al. (1992), show that under appropriate conditions, the algorithm explores efficiently the neighborhood of the actual solution. In fact, a crucial element in the MOSA algorithm (Multi-Objective Simulated Annealing) is the procedure that generates the class of close-enough alternative solutions to a given one. In our work we do this by taking one of the genes of the chromosome and changing it value at random (Frutos et al. 2010). This alteration allows, in a single stroke, to exchange several

Simulated Annealing Algorithm

```
0.  Take an initial x ∈ Q'_{i+1}
1.  while T > T_f
2.      Compute M = ⌊1/T⌋ + ω
3.      for i = 1 to M
4.          Change x and obtain x'
5.          Decodify and evaluate f1(x') and f2(x')
6.              if f1 and f2 improve
7.                  then Change x'
8.              if f1 or f2 improve without worsening either f2 or f1
9.                  then Change x'
10.             if either f1 or f2 gets worse
11.                 then
12.                     if ξ(0, 1) < e^{-δ/T}
13.                         then Change x'
14.                     end if
15.             end if
16.     end for
17.     T = α(T)
18. end while
19. end
```

Fig. 2 Pseudo-code of the simulated annealing procedure

Multi-objective Memetic Algorithm

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 $x \in Q_i''$
8. Select out of $P_i \cup Q_i''$ 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 Memetic Algorithm

operations on a single machine. This procedure is applied M times. The pseudo-code of the version of MOSA used here is presented in Fig. 2.

3.3 Putting all the pieces together

In this section we will explain how all the aforementioned pieces are assembled (Fig. 3) First, the memetic procedure generates the initial population. Later, in order to evaluate the fitness of the individuals in the population, it is necessary to calculate the value of each one of the objectives. Afterward, a binary tournament selection is performed, and the genetic operators are used to create the population. Then, the simulated annealing procedure performs a local search for each individual, replacing it with a new individual. This is repeated until a given generation number is reached.

4 Implementation and design of experiments

The algorithm was implemented on PISA (A Platform and Programming Language Independent Interface for Search Algorithms) (Bleuler et al. 2003), a search algorithm interface that distinguishes between two modules: variator and selector.

The former takes all the specificities of the problem at hand to codify and decode the solutions (computing 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 languages and the operations system 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). In this work we consider for MOEAs (Multi-

objective Evolutionary Algorithms): Non-dominated Sorting Genetic Algorithm II (NSGAII), Strength Pareto Evolutionary algorithm II (SPEAII) (Zitzler et al. 2002), as well as their predecessors, Non-dominated Sorting Genetic Algorithm (NSGA) (Srinivas 1994) and Strength Pareto Evolutionary algorithm (SPEA) (Zitzler and Thiele 1999).

NSGA classifies the individuals in layers or fronts, by grouping all the non-dominated individuals in a single front, with the same value of fitness for each individual. This value is proportional to the size of the population as to provide reproduction potential for all the individuals in the front. This procedure is repeated on the remaining individuals (those outside the non-dominated front) and so on until all the individuals in the population are classified. Since the ones 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, applying an elitist replacement strategy that chooses the best individuals from the union between parent and child generations. NSGAII classifies the fronts, such that the solutions on the main front are the non-dominated ones, the solutions on the second front those that are non-dominated in absence of the first front, etc. 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 memorizes the non-dominated individuals and deletes from memory 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. In SPEA, 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, has a fine-tuning procedure according to which the fitness of an individual is obtained as a balance between the number of solutions that are dominated by it and the number that dominate the individual.

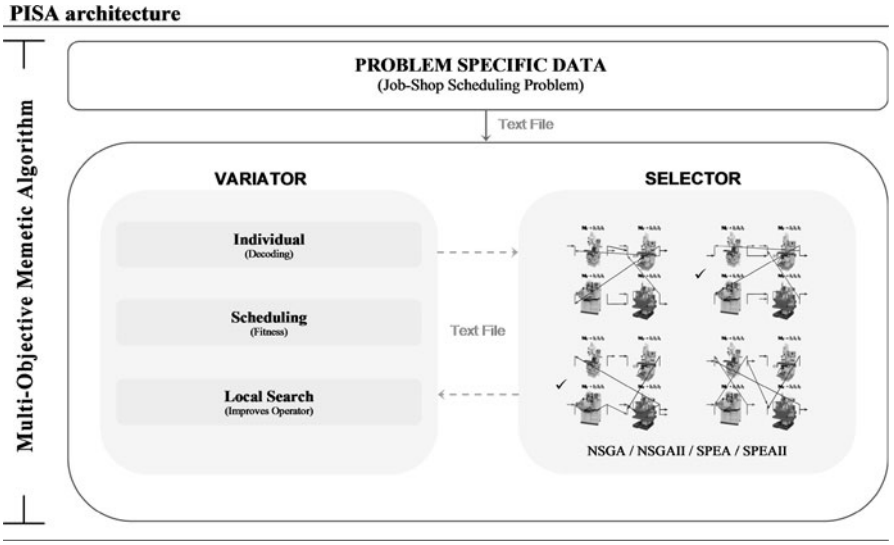

Fig. 4 The architecture of PISA

Besides, it uses the "nearest neighbor" for valuing the density of feasible solutions and thus leading to a more efficient search. In Fig. 4 we can see the PISA architecture adapted to the JSSP.

4.1 Experiments and results

The parameters and characteristics of the computing equipment used during these experiments were as follows: size of the population: 200; number of generations: 250; 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. For the problems la02, la03, la07, la26, la32 and la40 (Beasley 1990), we show the results for the multi-objective analysis based on Makespan and Mean Flow Time. They were obtained by running each algorithm 20 times. For each algorithm the sets of undominated solutions $P_1, P_2, \ldots, P_{20}$ were obtained as well as the super-population $P_T = P_1 \cup P_2 \cup \ldots \cup P_{20}$. From each superpopulation a class of undominated solutions was extracted, constituting the Pareto frontier for each

Table 3 Mean times to complete the process

Mean running time (in seconds)	NSGAII	NSGA	SPEAII	SPEA
la02	0.180	0.158	0.178	0.164
la03	1.510	1.324	1.491	1.379
la07	0.390	0.342	0.385	0.356
la26	0.670	0.588	0.662	0.612
la32	14.560	12.769	14.381	13.293
la40	491.210	430.791	485.168	448.475

Each algorithm (NSGAII, NSGA, SPEAII and SPEA) runs 20 times

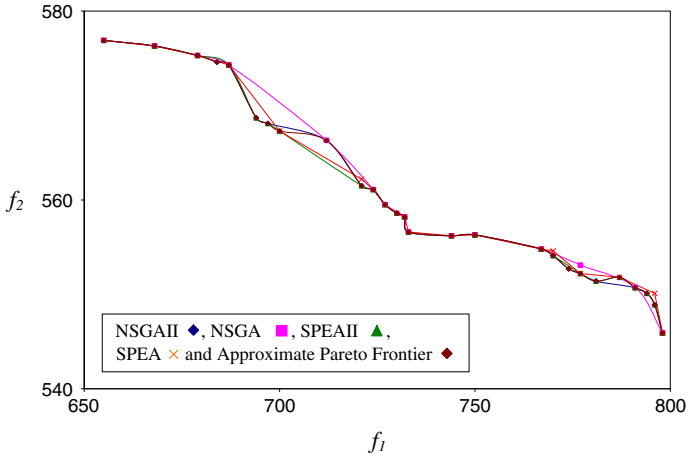

Fig. 5 Solutions (la02). NSGAII (*blue diamond*), NSGA (*rose square*), SPEAII (*green triangle*), SPEA (*orange times*) and approximate Pareto frontier (*brown diamond*) (color figure online)

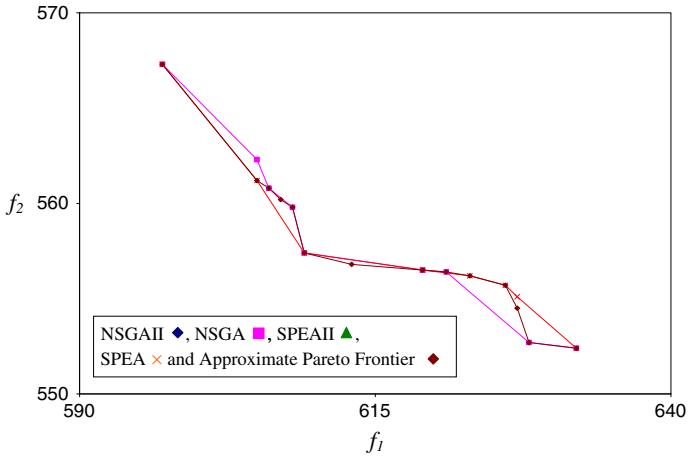

Fig. 6 Solutions (la03). NSGAII (*blue diamond*), NSGA (*rose square*), SPEAII (*green triangle*), SPEA (*orange times*) and approximate Pareto frontier (*brown diamond*) (color figure online)

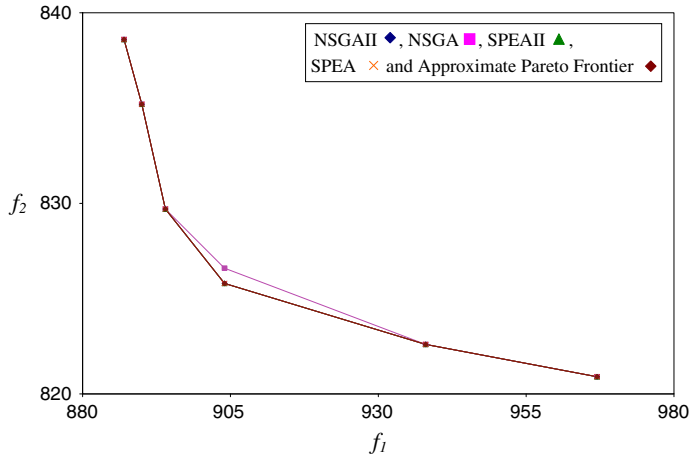

Fig. 7 Solutions (la07). NSGAII (*blue diamond*), NSGA (*rose square*), SPEAII (*green triangle*), SPEA (*orange times*) and approximate Pareto frontier (*brown diamond*) (color figure online)

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 3.

The corresponding results are shown in Fig. 5 (la02), Fig. 6 (la03), Fig. 7 (la07), Fig. 8 (la26), Fig. 9 (la32) and Fig. 10 (la40). To obtain an approximation to the true Pareto front we take the entire class $Y_{NSGAII} \cup Y_{NSGA} \cup Y_{SPEAII} \cup Y_{SPEA}$, from which all the dominated solutions are eliminated. Even before testing the performance of the algorithms it can be assessed that la02, la03 and 07 are easy to solve. On the other hand in la26, la32 and la40 the complexity increases and the results are clearly different.

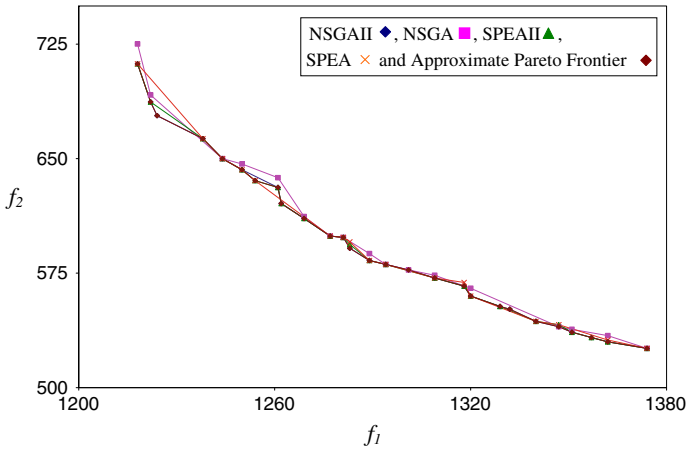

Fig. 8 Solutions (la26). NSGAII (*blue diamond*), NSGA (*rose square*), SPEAII (*green triangle*), SPEA (*orange times*) and approximate Pareto frontier (*brown diamond*) (color figure online)

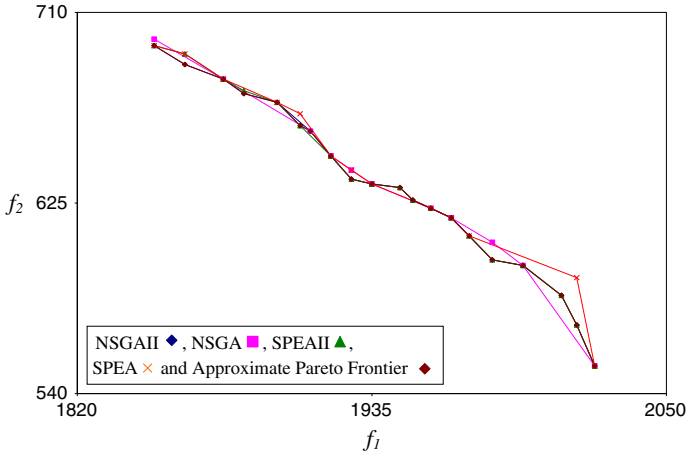

Fig. 9 Solutions (la32). NSGAII (*blue diamond*), NSGA (*rose square*), SPEAII (*green triangle*), SPEA (*orange times*) and approximate Pareto frontier (*brown diamond*) (color figure online)

4.2 Comparison procedure

In order to compare the results of the algorithms and establish the better option for solving the JSSP, several tests were applied over the solutions. We considered unary quality indicators using normalized approximation sets. Then, we applied the unary indicators (unary hypervolume indicator I_H, unary epsilon indicator I_e^1 and R indicator I_{R2}^1) on the normalized approximation sets as well as on the reference set

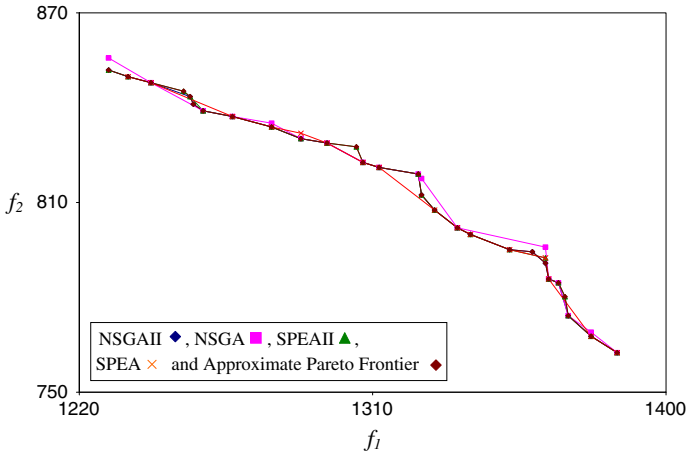

Fig. 10 Solutions (la40). NSGAII (*blue diamond*), NSGA (*rose square*), SPEAII (*green triangle*), SPEA (*orange times*) and approximate Pareto frontier (*brown diamond*) (color figure online)

Table 4 Unary hypervolume indicator, unary epsilon indicator and R indicator (la02)

Test for problem la02	NSGAII	NSGA	SPEAII	SPEA
I_e^1				
NSGAII	–	0.46688	0.49838	0.47918
NSGA	0.53312	–	0.43697	0.48266
SPEAII	0.50162	0.56303	–	0.55780
SPEA	0.52082	0.51734	0.44220	–
I_H				
NSGAII	–	0.48494	0.49348	0.48066
NSGA	0.51506	–	0.45387	0.52343
SPEAII	0.50652	0.54613	–	0.57938
SPEA	0.51934	0.47657	0.42062	–
I_{R2}^1				
NSGAII	–	0.45604	0.48681	0.45547
NSGA	0.54396	–	0.42682	0.51670
SPEAII	0.51319	0.57318	–	0.54485
SPEA	0.54453	0.48330	0.45515	–

generated by PISA (I_H, I_e^1 and I_{R2}^1, Table 4 (la02), Table 5 (la03), Table 6 (la07), Table 7 (la26), Table 8 (la32) and Table 9 (la40)).

For problems la02, la03, la07 and la26, the differences among algorithms are not significant at an overall significance level $\alpha = 0.05$. For la32 and la40, NSGAII and SPEAII showed differences with NSGA and SPEA significant at $\alpha = 0.05$. This indicates that NSGAII and SPEAII are more appropriate candidates for solving the

Table 5 Unary hypervolume indicator, unary epsilon indicator and R indicator (la03)

Test for Problem la03	NSGAII	NSGA	SPEAII	SPEA
I_e^I				
NSGAII	–	0.46081	0.49190	0.47295
NSGA	0.53919	–	0.43129	0.47638
SPEAII	0.50810	0.56871	–	0.55055
SPEA	0.52705	0.52362	0.44945	–
I_H				
NSGAII	–	0.49426	0.50385	0.49076
NSGA	0.50574	–	0.46260	0.53443
SPEAII	0.49615	0.53740	–	0.59051
SPEA	0.50924	0.46557	0.40949	–
I_{R2}^I				
NSGAII	–	0.46786	0.49943	0.46208
NSGA	0.53214	–	0.43789	0.52419
SPEAII	0.50057	0.56211	–	0.55897
SPEA	0.53792	0.47581	0.44103	–

Table 6 Unary hypervolume indicator, unary epsilon indicator and R indicator (la07)

Test for Problem la07	NSGAII	NSGA	SPEAII	SPEA
I_e^I				
NSGAII	–	0.50597	0.49479	0.52082
NSGA	0.49403	–	0.51304	0.51126
SPEAII	0.50521	0.48696	–	0.52509
SPEA	0.47918	0.48874	0.47491	–
I_H				
NSGAII	–	0.51331	0.50197	0.52837
NSGA	0.48669	–	0.52047	0.51867
SPEAII	0.49803	0.47953	–	0.53271
SPEA	0.47163	0.48133	0.46729	–
I_{R2}^I				
NSGAII	–	0.49940	0.48836	0.51405
NSGA	0.50060	–	0.50637	0.50461
SPEAII	0.51164	0.49363	–	0.51827
SPEA	0.48595	0.49539	0.48173	–

JSSP. As a further step in the analysis, we establish the percentage of contribution of each algorithm to the Approximate Pareto Frontier (Table 10). From this we can conclude that NSGAII is the best selector we can apply to our problem.

Table 7 Unary hypervolume indicator, unary epsilon indicator and R indicator (la26)

Test for Problem la26	NSGAII	NSGA	SPEAII	SPEA
I_e^I				
NSGAII	–	0.07583	0.31397	0.09576
NSGA	0.92417	–	0.91287	0.61674
SPEAII	0.68603	0.08713	–	0.09698
SPEA	0.90424	0.38326	0.90302	–
I_H				
NSGAII	–	0.07693	0.31852	0.09715
NSGA	0.92307	–	0.92611	0.62568
SPEAII	0.68148	0.07389	–	0.09839
SPEA	0.90285	0.37432	0.90161	–
I_{R2}^I				
NSGAII	–	0.07485	0.30989	0.09451
NSGA	0.92515	–	0.90101	0.60872
SPEAII	0.69011	0.09899	–	0.09572
SPEA	0.90549	0.39128	0.90428	–

Table 8 Unary hypervolume indicator, unary epsilon indicator and R indicator (la32)

Test for Problem la32	NSGAII	NSGA	SPEAII	SPEA
I_e^I				
NSGAII	–	0.02867	0.48836	0.03083
NSGA	0.97133	–	0.95422	0.51395
SPEAII	0.51164	0.04578	–	0.03162
SPEA	0.96917	0.48605	0.96838	–
I_H				
NSGAII	–	0.03055	0.49403	0.03286
NSGA	0.96945	–	0.98080	0.54775
SPEAII	0.50597	0.01920	–	0.03338
SPEA	0.96714	0.45225	0.96662	–
I_{R2}^I				
NSGAII	–	0.02929	0.49901	0.03151
NSGA	0.97071	–	0.97502	0.52516
SPEAII	0.50099	0.02498	–	0.03231
SPEA	0.96849	0.47484	0.96769	–

5 Conclusions

We presented a Multi-objective Memetic Algorithm for solving the Job-Shop Scheduling Problem (JSSP), one of the most avidly studied problems in the class

Table 9 Unary hypervolume indicator, unary epsilon indicator and R indicator (la40)

Test for Problem la40	NSGAII	NSGA	SPEAII	SPEA
I_e^I				
NSGAII	–	0.02904	0.49479	0.03124
NSGA	0.97096	–	0.96679	0.52072
SPEAII	0.50521	0.03321	–	0.03204
SPEA	0.96876	0.47928	0.96796	–
I_H				
NSGAII	–	0.03011	0.48696	0.03239
NSGA	0.96989	–	0.96678	0.53992
SPEAII	0.51304	0.03322	–	0.03290
SPEA	0.96761	0.46008	0.96710	–
I_{R2}^I				
NSGAII	–	0.02869	0.48874	0.03086
NSGA	0.97131	–	0.95497	0.51435
SPEAII	0.51126	0.04503	–	0.03164
SPEA	0.96914	0.48565	0.96836	–

Table 10 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 (%)
la02	92.59	40.74	88.89	59.26
la03	92.86	57.14	85.71	50.00
la07	100.00	83.33	100.00	100.00
la26	92.59	25.93	81.48	48.15
la32	90.00	40.00	85.00	45.00
la40	89.29	46.43	89.29	50.00

NP-Hard. Our algorithm integrates two meta-heuristic procedures: a Multi-Objective Evolutionary Algorithm (MOEA) with Multi-Objective Simulated Annealing (MOSA). The individuals were coded as to facilitate the application of 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. This means that our algorithm provides a useful tool for solving the JSSP. We plan, in the future, to explore its performance on other MOPs. We believe that it provides a strong and efficient approach to solving this kind of problems.

Acknowledgments 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. Manage Sci 34(3):391–401

Armentano VA, Scrich CR (2000) Tabu search for minimizing total tardiness in a job-shop. Int J Prod Econ 63:131–140

Beasley JE (1990) OR-library: distributing test problems by electronic mail. J Oper Res Soc 41(11):1069–1072

Beck JC (2007) Solution-guided multi-point constructive search for job-shop scheduling. J Artif Intell Res 29:49–77

Bihlmaier R, Koberstein A, Obst R (2009) Modeling and optimizing of strategic and tactical production planning in the automotive industry under uncertainty. OR Spectr 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, pp 494–508

Brucker P, Jurisch B, Sievers B (1994) A branch and bound algorithm for the job-shop scheduling problem. Discret Appl Math 49:107–127

Chao-Hsien J, Han-Chiang H (2009) A hybrid genetic algorithm for no-wait job shop scheduling problems. Expert Syst Appl 36(3):5800–5806

Chinyao L, Yuling Y (2009) Genetic algorithm-based heuristics for an open shop scheduling problem with setup, processing, and removal times separated. Robot Comput Integr Manuf 25(2):314–322

Coello Coello CA, Lamont GB, Veldhuizen DA (2006) Evolutionary algorithms for solving multi-objective problems. Genetic and evolutionary computation. Springer, New York

Cortés Rivera D, Coello Coello CA, Cortés NC (2003) Use of an artificial immune system for job-shop scheduling. In: Proceedings of the second international conference on artificial immune systems, Edinburgh. Scotland, Lecture notes in computer science, vol. 2787. Springer, New York, pp 1–10

Croce F, Tadei R (1995) A genetic algorithm for the job-shop problem. Comput Oper Res 22(1):15–24

De Giovanni L, Pezzella F (2010) An improved genetic algorithm for the distributed and flexible jobshop scheduling problem. Euro J Oper Res 200(2):395–408

Deb K, Pratap A, Agarwal S, Meyarivan T (2002) A fast and elitist multi-objective genetic algorithm: NSGAII. IEEE Trans Evol Comput 6(2):182–197

Dell Amico M, Trubian M (1993) Applying Tabu search to the job-shop scheduling problem. Ann Oper Res 41:231–252

Della Croce F, Grosso A, Salassa F (2011) A matheuristic approach for the two-machine total completion time flow-shop problem. Ann Oper Res. Springer. Online First™, 6 July 2011

Frutos M, Olivera AC, Tohmé F (2010) A memetic algorithm based on a NSGAII scheme for the flexible job-shop scheduling problem. Ann Oper Res 181:745–765

Giffler B, Thomson GL (1960) Algorithms for solving production scheduling problems. Oper Res 8:487–503

Heinonen J, Pettersson F (2007) Hybrid ant colony optimization and visibility studies applied to a job-shop scheduling problem. Appl Math Comput 187(2):989–998

Ishibuchi H, Yoshida T, Murata T (2003) Balance between genetic search and local search in memetic algorithms for multiobjective permutation flow-shop scheduling. IEEE Trans Evol Comput 7(2): 204–223

Johnson SM (1954) Optimal two- and three-stage production schedules with setup times included. Naval Res Logist Quart 1:61–68

Knowles J, Thiele L, Zitzler E (2005) A tutorial on the performance assessment of stochastic multiobjective optimizers. Comput Eng Netw Lab. TIK-Report 214:1–35

Lin Y, Pfund M, Fowler J (2011) Heuristics for minimizing regular performance measures in unrelated parallel machine scheduling problems. Comput Oper Res 38(6):901–916

Merkle D, Middendorf M (2001) A new approach to solve permutation scheduling problems with ant colony optimization. Appl Evol Comput (EvoWorkshops 2001) 2037:484–494

Nowicki E, Smutnicki C (2005) An advanced Tabu search algorithms for the job-shop problem. J Sched 8(2):145–159

Panwalker S, Iskander W (1977) A survey of scheduling rules. Oper Res 25(1):45–61

Papadimitriou CH (1994) Computational complexity. Addison Wesley, USA

Park BJ, Choi HR, Kim HSA (2003) Hybrid genetic algorithm for the job-shop scheduling problems. Comput Ind Eng 45(4):597–613

Roy B, Sussman B (1964) Les problèmes d'ordonnancement avec contraintes disjonctives. SEMA. Technical Report 9 bis:1–21

Sadeh NM, Fox MS (1995) Variable and value ordering heuristics for the job-shop scheduling constraint satisfaction problem. Technical Report CMU-RI-TR 95-39

Srinivas N (1994) Multi-objective optimization using nondominated sorting in genetic algorithms. Master thesis. Indian Institute of Technology, Kanpur, India

Storer RH, Wu SD, Vaccari R (1992) New search spaces for sequencing instances with application to job-shop scheduling. Manage Sci 38:1495–1509

T'kindt V, Billaut JC (2006) Multicriteria scheduling. Theory, models and algorithms. Springer, Germany

Tsai CF, Lin FC (2003) A new hybrid heuristic technique for solving job-shop scheduling problems. Second IEEE international workshop on intelligent data acquisition and advanced computing systems: technology and applications

Ullman JD (1975) NP-complete scheduling problems. J Comput Syst Sci 10:384–393

Van Laarhoven PJM, Aarts EHL, Lenstra JK (1992) Job-shop scheduling by simulated annealing. Oper Res 40(1):113–125

Varadharajan TK, Rajendran C (2005) A multi-objective simulated-annealing algorithm for scheduling in flowshops to minimize the makespan and total flowtime of jobs. Euro J Oper Res 167:772–795

Wu CG, Xing XL, Lee HP, Zhou CG, Liang YC (2004) Genetic algorithm application on the job-shop scheduling problem. In: Proceedings of the 2004 international conference machine learning and cybernetics, vol 4, pp 2102–2106

Zalzala AMS, Flemming PJ (1997) Genetic algorithms in engineering systems. London Institution of Electrical Engineers, London

Zhang CY, Li P, Rao Y, Guan Z (2008) A very fast TS/SA algorithm for the job-shop scheduling problem. Comput Oper Res 35:282–294

Zitzler E, Thiele L (1999) Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. IEEE Trans Evol Comput 3(4):257–271

Zitzler E, Laumanns M, Thiele L (2002) SPEAII: improving the strength pareto evolutionary algorithm for multiobjective optimization. In Giannakoglou, Tsahalis, Periaux, Papailiou, Fogarty (eds) Evolutionary methods for design. Optimisations and Control, pp 19–26