International Journal of Industrial Engineering Computations 7 (2016) 585–596

Contents lists available at GrowingScience

International Journal of Industrial Engineering Computations

homepage: www.GrowingScience.com/ijiec

An alternative hybrid evolutionary technique focused on allocating machines and sequencing operations

Mariano Frutos[a*], Fernando Tohmé[b], Fernando Delbianco[b] and Fabio Miguel[c]

[a]*Department of Engineering, Universidad Nacional del Sur and IIESS-CONICET. Av. Alem 1253, Bahía Blanca, Argentina*
[b]*Department of Economics, Universidad Nacional del Sur and CONICET. 12 de Octubre 1198, Bahía Blanca, Argentina*
[c]*Sede Alto Valle y Valle Medio, Universidad Nacional de Río Negro. Mitre 305, Villa Regina, Argentina*

C H R O N I C L E

Article history:
Received November 4 2015
Received in Revised Format
April 6 2016
Accepted April 7 2016
Available online
April 7 2016

Keywords:
Flexible job-shop scheduling problem
Optimization
Multi-objective hybrid
Evolutionary algorithm
Production

A B S T R A C T

We present here a hybrid algorithm for the Flexible Job-Shop Scheduling Problem (FJSSP). This problem involves the optimal use of resources in a flexible production environment in which each operation can be carried out by more than a single machine. Our algorithm allocates, in a first step, the machines to operations and in a second stage it sequences them by integrating a Multi-Objective Evolutionary Algorithm (MOEA) 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 FJSSP. An important step in the development of the algorithm was the selection of the right MOEA. Candidates were tested on problems of low, medium and high complexity. Further analyses showed the relevance of the search algorithm in the hybrid structure. Finally, comparisons with other algorithms in the literature indicate that the performance of our alternative is good.

1. Introduction

The design of production plans involves decisions on the allocation of limited resources in order to optimize efficiency-related short-term objectives (Bihlmaier et al., 2009; Nowicki & Smutnicki, 2005; Armentano & Scrich, 2000). The framework of analysis of this kind of problems is the Job-Shop Scheduling Problem (JSSP) (Agnetis et al., 2001; Lin et al., 2011; Heckman & Beck, 2011; Nazarathy & Weiss, 2010), which assumes a class of jobs, consisting of ordered sequences of operations that have to be distributed over several machines. One of the goals is to minimize the makespan, i.e. the total processing time of the jobs (Heinonen & Pettersson, 2007; Chao-Hsien & Han-Chiang, 2009; Della Croce et al., 2014). Flexible JSSP (FJSSP) generalizes this problem. It assumes that operations can be performed on different machines. Thus, it involves the decision of the allocation of operations on machines, a NP-Hard problem (Ullman, 1975; Papadimitriou, 1994). While most of the literature on this problem focuses on its single objective versions, some authors

* Corresponding author.
E-mail: mfrutos@uns.edu.ar (M. Frutos)

doi: 10.5267/j.ijiec.2016.4.002

state that several objectives have to be optimized as to achieve an efficient production process (Chinyao & Yuling, 2009). Based on the latter motivation we present here an algorithm for a multi-objective version of FJSSP. On the grounds of a preliminary analysis of the problem we decided to base our alternative on an evolutionary approach (Goldberg, 1989; Pezzella et al., 2008). One of the main advantages of this strategy is that evolutionary algorithms can be easily adapted to the problem at hand. They are quite efficient in handling single-objective problems, but their high rate of convergence hampers their usefulness on multi-objective versions. This is because their fast convergence leads to a loss of diversity, indicated by poorly distributed Pareto frontiers. A multi-objective algorithm based on an underlying evolutionary component should be, therefore, complemented by an efficient search procedure in order to diversify solutions with a few rounds of evaluation of the fitness functions. This is the approach we followed in this paper, providing a methodological ground for the design of such a hybrid algorithm, as introduced in Frutos et al. (2010) and Frutos and Tohmé (2015). We present here a Multi-Objective Evolutionary Algorithm (MOEA) (Coello et al., 2006) joined by a local search procedure (MOSA, Multi-Objective Simulated Annealing) to solve a FJSSP (Hansmann et al., 2014; Tsai & Lin, 2003; Wu et al., 2004; Nidhiry & Saravanan, 2012). From now on, we will call this hybrid structure a Multi-Objective Hybrid Evolutionary Algorithm or MOHEA. The rest of the paper is organized as follows. Section 1.1 discusses some of the literature on the FJSSP while section 1.2 introduces the formal description of multiple-objective optimization problems. Section 2 presents our formal characterization of FJSSP while in section 3 the MOHEA for this framework is introduced. Results of running the algorithm are shown in section 4, and in section 5 we present the conclusions.

1.1 Approaches to the FJSSP

The FJSSP is particularly hard to solve. It has been analyzed for 1, 2 and 3 machines and some arbitrary number of jobs. Very few developments have been devoted to the case of 4 or more machines for at least 3 jobs, due to the combinatorial explosion of feasible sequences. A brief survey of the literature on the problem shows that, (Brandimarte, 1993) proposed a hierarchical approach, distinguishing the allocation from the sequencing sub-problem, where the former is handled as a routing problem while the latter is seen as a Job-Shop one. (Mesghouni et al., 1997), instead, attacked the problem with genetic algorithms. On the other hand, (Kacem et al., 2002) proposed a localization approach to the control of the genetic algorithm, yielding good solutions and minimizing the makespan and the workload of the machines. Tay and Wibowo (2004) and Ho and Tay (2005) introduced dispatch rules to generate populations and a scheme structure under which each generation explores the search space. They also studied the representation of solutions in order to achieve a more efficient makespan. This approach was generalized in (Ho et al., 2007), in which the evolutionary algorithm is complemented by learning under schemes and composite dispatch rules. (Fattahi et al., 2007) proposed a hierarchical approach to the FJSSP in which the allocation sub-problem is solved by a Taboo Search treatment, while the sequencing one is handled by Simulated Annealing. This approach was tested on twenty instances of the FJSSP, although only the simplest ones got solved. (Zhang & Gen, 2005) presents a genetic algorithm working on multiple scenarios, in which each one corresponds to an operation and each feasible machine to a state. (Pezzella et al., 2008), also used a genetic algorithm aided by Kacem's et al. (2002) localization approach. This allows for intelligent mutations that reassign operations from heavy loaded to less loaded machines. (Yazdani et al., 2010) proposed the minimization of makespan by handling, in parallel, variable neighbourhoods while (Yang et al., 2010) solved the FJSSP with an improved constraint satisfaction adaptive neural network. Recent approaches involve hybridizations with an artificial bee colony algorithm (Li et al., 2011) or a shuffled frog-leaping algorithm (Li et al., 2012). On the other hand, the local search procedure included has been defined on the critical path (Xiong et al., 2012) or performed hierarchically over the different objectives (Yuan & Xu, 2015).

1.2 Multi-Objective Optimization: Basic Concepts

Let us assume that several goals (objectives) have to be minimized. Thus, a vector $\vec{x}^* = [x_1^*, ..., x_n^*]^T$ of decision variables is required, satisfying q inequalities $g_i(\vec{x}) \geq 0, \quad i = 1, ..., q$ as well as p equations $h_i(\vec{x}) = 0, \; i = 1, ..., p$, such that $\vec{f}(\vec{x}) = [f_1(\vec{x}), ..., f_k(\vec{x})]^T$, a vector of k functions, each one corresponding to a goal, attains its minimum. The family of 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 $i = 1,\ldots,k$, $f_i(\vec{x}^*) \leq f_i(\vec{x})$. This means that no $\vec{x}$ can improve a goal without worsening others. We say that a vector $\vec{u} = [u_1,...,u_n]^T$ dominates another, $\vec{v} = [v_1,...,v_n]^T$ (denoted $\vec{u} \prec \vec{v}$) if and only if $\forall i \in \{1,...,k\}$, $u_i \leq v_i \wedge \exists i \in \{1,...,k\} : u_i < v_i$. The set of Pareto optima is $P^* = \{\vec{x} \in \Omega \mid \neg\, \exists\, \vec{x}' \in \Omega,\ \vec{f}(\vec{x}') \prec \vec{f}(\vec{x})\}$ and the associated Pareto frontier is $FP^* = \{\vec{f}(\vec{x}), \vec{x} \in P^*\}$. The main goal of Multi-Objective Optimization is to find the corresponding FP^*. A good approximation should yield a few feasible candidates; close enough to the frontier (Frutos & Tohmé, 2009).

2. The flexible job-shop scheduling problem

The FJSSP is defined in terms of *m* machines, $M = \{M_k\}_{k=1}^m$, and a class of n independent jobs, $J = \{J_j\}_{j=1}^n$. Each job J_j amounts to a set of sequenced operations, $S_j = \{O_{jk}^i\}_{i=1}^h$. Each of these must be processed by a machine in *M*. Operation O_{jk}^i in the sequence S_j requires to use machine M_k during an un-interrupted processing time τ_{jk}^i (assumed constant), with an operational cost υ_{jk}^i. No machine can run two operations at the same time and all jobs and machines are available at time 0. The different operations allocated to M_k constitute the set E_k. Therefore $E = \{E_k\}_{k=1}^m$ will involve the same operations as in $J = \{J_j\}_{j=1}^n$ and each operation will be allocated only once. From the many possible objectives that can be pursued in this setting we choose the minimization of the total processing time, (makespan) Eq. (1), and the minimization of the total operational cost given by Eq. (2).

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

$$f_2 : \sum_j \sum_i \sum_k x_{jk}^i \upsilon_{jk}^i, \tag{2}$$

where $x_{jk}^i = 1$ if $O_{jk}^i \in E_k$ and 0 otherwise. On the other hand $\sum_k x_{jk}^i = 1$. Besides, $t_{jk}^i = \max\,(t_{jh}^{(i-1)} + \tau_{jh}^{(i-1)},\ t_{pk}^s + \tau_{pk}^s,\ 0)$ for each pair $O_{jh}^{i-1}, O_{pk}^s \in E_k$ and all machines M_k, M_h and sequences of operations S_i, S_s. As we will see the two objectives are in conflict, which makes this problem interesting.

3. A multi-objective hybrid evolutionary algorithm

Evolutionary algorithms imitate genetic processes, improving solutions by breeding new solutions up from older ones. The solutions are represented as a fixed number of chromosomes, composed by smaller units called genes. They codify the hereditary features of an individual (solution). In the case of sequencing problems the chromosomes indicate the programming of jobs. It is assumed that among all possible chromosomes one codifies the optimal sequence. To show how this works we will consider the case of instance MF01 analyzed in (Frutos et al., 2010), with three jobs and four machines (3×4). The first and second jobs require three operations each while the third one requires only two. This amounts to eight operations with processing times and operational costs shown in Table 1. Solutions have to be codified in terms of the characteristics of the problem, respecting its constraints. We will use two chromosomes for each solution. The first one determines the solution of the allocation sub-problem while the second the solution of the sequencing of operations sub-problem. The size of the allocation chromosomes is the number of total operations in the problem. The size of the sequencing chromosomes is the number of machines in M. In the allocation chromosome each gene is: 0→M_1, 1→M_2, 2→M_3, 3→M_4 (Third column, rows 3 to 10 in Table 2). For the sequencing chromosome each gene is: 0→1 | 2 | 3, 1→1 | 3 | 2, 2→2 | 1 | 3, 3→2 | 3 | 1, 4→3 | 1 | 2 and 5→3 | 2 | 1 (Third row, columns 4 to 7 in Table 2). This means that, in the first chromosome, gene 0 corresponds to an allocation to machine M_1; if it is 1 to machine M_2 and so on. In the second chromosome, if the gene is 0 the sequence of jobs is 1, 2 and 3; if it is 1 the sequence is 1, 3 y 2, and so on.

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}	υ^i_{j1}	τ^i_{j2}	υ^i_{j2}	τ^i_{j3}	υ^i_{j3}	τ^i_{j4}	υ^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

Table 2
Allocation and Sequencing Chromosomes

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

Allocation Chromosome: 0-3-2-2-0-1-2-3 (Ref. 0→M_1, 1→M_2, 2→M_3, 3→M_4)
Sequencing Chromosome: 4-0-2-1 (Ref. 0→1 | 2 | 3, 1→1 | 3 | 2, 2→2 | 1 | 3, 3→2 | 3 | 1, 4→3 | 1 | 2 and 5→3 | 2 | 1)

A crucial role in the algorithm is played by the chromosome decoding procedure, since it is in charge of the interpretation and representation of the solutions of the FJSSP. This procedure is the most computationally costly in the algorithm. It runs by fetching the information in the genes and looking up in the times and costs table. After this, the objectives are evaluated on the resulting solution candidate (see Figure 1). The initial population is randomly generated: the allocation chromosome is obtained by assigning at random values between 0 and m-1 (between 0 and 3 in our example) to the genes. On the other hand, the sequencing chromosome is generated by the random assignation of values between 0 and n!-1 (between 0 and 5 in the example) to its genes. Then, the crossover and mutation genetic operators are applied over the candidates. Crossover exchanges segments of chromosomes between pairs of candidates, making the new candidates carry a mixture of the information of its 'parents'. Mutation introduces random feasible changes in the chromosomes, as to yield new solution candidates and allowing the exploration of different areas of the search space. The crossover operator is applied on chromosomes of the same type, exchanging segments of the allocation chromosome and segments of the sequencing chromosome among candidates. In our algorithm we use uniform crossover, which exchanges the genes in the same position in the chromosomes of two candidates, breeding two new solution candidates. The version of the mutation operator used here swaps two genes in the same chromosome (Two-swap). On the allocation chromosome it operates on a 20 % of the genes, while on the sequencing chromosome it acts only over the 10 %. After a careful examination of local search algorithms we selected Simulated Annealing as our 'improves' operator. The main components of this algorithm are the way in which the neighborhood of solutions is generated and the probability function $e^{-(\delta/T)}$, in particular the value of δ. The general structure of our Multi-Objective Simulated Annealing is the same as in (Frutos et al., 2010).

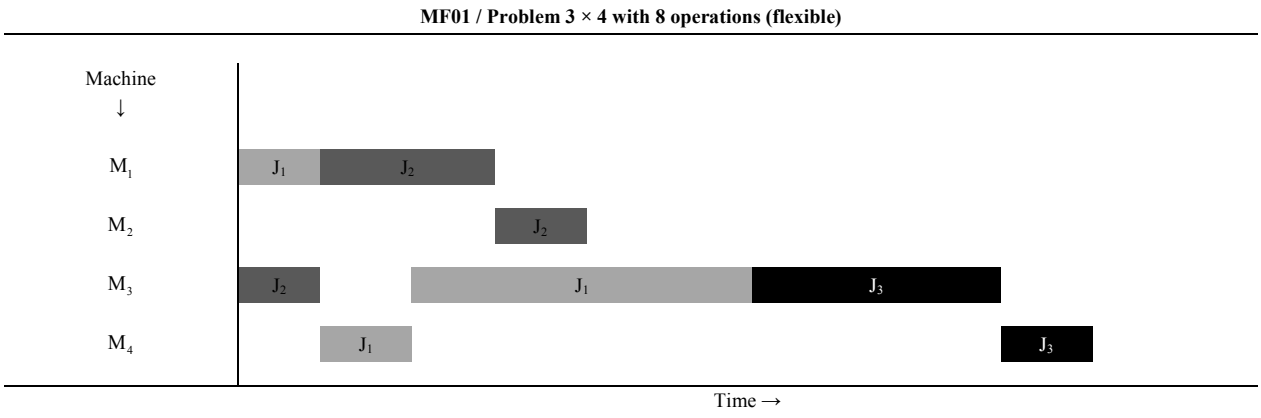

Fig. 1. Decoding of both chromosomes and evaluation of the resulting solution, f_1: 10 and f_2: 97

The neighborhood is generated by taking both chromosomes in a candidate and altering at random the value of some genes (with the same percentages as the mutation operator). This induces a change of the machines on which operations are run and at the same time a change in their ordering. The determination of δ, somehow lacking in the multi-objective literature is as follows: $\delta = \max\left[\left(f_i(x') - f_i(x)\right)/f_i(x)\right]$. (Yazdani et al., 2010) have shown that in appropriate settings this specification yields very good solutions. As for those settings, they are given by the following parameters: T_i (initial temperature), T_f (final temperature), the cooling function ($T_{k+1} = \alpha T_k$, where α is the rate of cooling and k the iteration step), and the number of iterations M ($M = \lfloor 1/T_k \rfloor + \omega$, where ω is a control parameter). The hybrid structure was designed in the PISA (A Platform and Programming Language Independent Interface for Search Algorithms) environment (Frutos & Tohmé, 2009) (see Fig. 2).

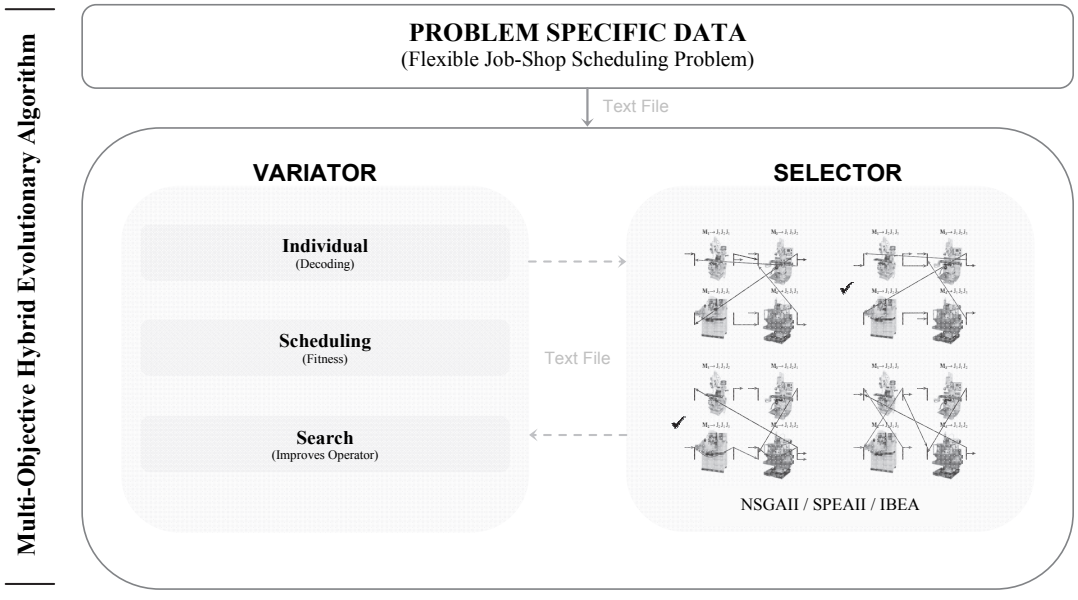

Fig. 2. Hybrid Structure in PISA

4. Implementation and design of experiments

Any instance of the FJSSP is specified by the number of jobs, the number of machines and the total number of operations to be performed. For our experiments we use the problems reported in (Frutos et al., 2010): MF01 (3 jobs × 4 machines with 8 operations), MF02 (4 × 5 with 12 operations), MF03 (10 × 7 with 29 operations), MF04 (10 × 10 with 30 operations) and MF05 (15 × 10 with 56 operations). We compared the

global performance of the evolutionary stage by exchanging three selectors (see Fig. 2): Nondominated Sorting Genetic Algorithm II (NSGAII) (Deb et al., 2002) (with a $O(g \times N^2)$ complexity, for a population of size 2N), Strength Pareto Evolutionary Algorithm II (SPEAII) (Zitzler et al., 2002) (its complexity is O((N+N')2*log(N+N')), where N is the size of the population and N' the size of the file that stores the results) and Indicator-Based Evolutionary Algorithm (IBEA) (Zitzler & Künzli, 2004) (of $O(N^2)$ complexity). 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. In the case of the IBEA selector we chose the additive epsilon index. We run each algorithm 30 times and the undominated solutions were picked up.

4.1 Selection of MOEA

As a first instance, we compare the performance of the MOEAs using the following metrics: the multiplicative version of the Unary Epsilon Index (I_e), the Hypervolume Index (I_H) and the R2 index (I_{R2}). These indexes allow distinguishing differences among the approximations when the dominance rankings yield very similar results (Knowles et al., 2005). For the parameters of the indexes we kept the initial specifications of PISA. Unary indexes were obtained using normalized approximation sets (Zitzler & Künzli, 2004) and the reference class generated by PISA. A non-parametric test was run on the approximation sets in order to obtain valid conclusions on the quality of the optimization methods. In our analysis we used Fisher' test (Knowles et al., 2005) with a confidence level $\alpha = 0.05$. Tables 3, 4 and 6 indicate that problems MF01, MF02 and MF04 yield no significantly different results. Tables 5 and 7, instead, show significant differences in the results of MF03 and MF05 between IBEA and the other two algorithms, SPEAII and NSGAII.

Table 3

I_e, I_H, and I_{R2} (MF01) for IBEA, NSGAII and SPEAII

MF01 / Problem 3 × 4 with 8 operations (flexible)									
	I_e			I_H			I_{R2}		
	IBEA	NSGAII	SPEAII	IBEA	NSGAII	SPEAII	IBEA	NSGAII	SPEAII
IBEA	-	0,31466	0,27123	-	0,32095	0,27665	-	0,30207	0,26038
NSGAII	0,68534	-	0,38665	0,67905	-	0,38193	0,69793	-	0,37438
SPEAII	0,72877	0,61335	-	0,72335	0,61807	-	0,73962	0,62562	-

Table 4

I_e, I_H, and I_{R2} (MF02) for IBEA, NSGAII and SPEAII

MF02 / Problem 4 × 5 with 12 operations (flexible)									
	I_e			I_H			I_{R2}		
	IBEA	NSGAII	SPEAII	IBEA	NSGAII	SPEAII	IBEA	NSGAII	SPEAII
IBEA	-	0,25667	0,15743	-	0,26180	0,16058	-	0,24640	0,15113
NSGAII	0,74333	-	0,47696	0,73820	-	0,48697	0,75360	-	0,48387
SPEAII	0,84257	0,52304	-	0,83942	0,51303	-	0,84887	0,51613	-

Table 5

I_e, I_H, and I_{R2} (MF03) for IBEA, NSGAII and SPEAII

MF03 / Problem 10 × 7 with 29 operations (flexible)									
	I_e			I_H			I_{R2}		
	IBEA	NSGAII	SPEAII	IBEA	NSGAII	SPEAII	IBEA	NSGAII	SPEAII
IBEA	-	0,03175	0,01738	-	0,03239	0,01773	-	0,03048	0,01668
NSGAII	0,96825	-	0,51132	0,96762	-	0,49577	0,96952	-	0,48233
SPEAII	0,98262	0,48868	-	0,98227	0,50423	-	0,98332	0,51767	-

Table 6

I_e, I_H, and I_{R2} (MF04) for IBEA, NSGAII and SPEAII

MF04 / Problem 10 × 10 with 30 operations (flexible)									
	I_e			I_H			I_{R2}		
	IBEA	NSGAII	SPEAII	IBEA	NSGAII	SPEAII	IBEA	NSGAII	SPEAII
IBEA	-	0,45378	0,42336	-	0,44017	0,41066	-	0,46739	0,43606
NSGAII	0,54622	-	0,49118	0,55983	-	0,47939	0,53261	-	0,49118
SPEAII	0,57664	0,50882	-	0,58934	0,52061	-	0,56394	0,50882	-

Table 7

I_e, I_H, and I_{R2} (MF05) for IBEA, NSGAII and SPEAII

MF05 / Problem 15 × 10 with 56 operations (flexible)									
	I_e^1			I_H			I_{R2}^1		
	IBEA	NSGAII	SPEAII	IBEA	NSGAII	SPEAII	IBEA	NSGAII	SPEAII
IBEA	-	0,02654	0,04166	-	0,02574	0,04041	-	0,02734	0,04291
NSGAII	0,97346	-	0,39616	0,97426	-	0,38691	0,97266	-	0,40776
SPEAII	0,95834	0,60384	-	0,95959	0,61309	-	0,95709	0,59224	-

Given these results, we have to note that an assessment based on a relatively small number of runs requires further analysis in order to ensure its robustness. We have to see that the results reported in the previous subsections are statistically significant. We proceed as follows. For each algorithm we take the final outcome on each problem. This is written as a vector. Then we take a component by component distance to the vector of solutions (of the same dimensionality). Then we postulate different hypotheses, one the null hypothesis (that the algorithms do not yield differences) and alternative ones, indicating differences among the algorithms. To see this, we start introducing a *distance to the frontier* variable, which is basically a variant of a *taxicab* metric. Let us remark that the results we present below are robust under changes in the underlying metric: the same analysis based on the *Euclidean* and the *supremum* metric yield analogous results. The distance of the frontier is obtained as the addition of the distances to their corresponding values in the frontier of the actually obtained values f_1 and f_2: $d_{x,i} = \left(f_{1,i} - f_{1,i}^*\right) + \left(f_{2,i} - f_{2,i}^*\right)$, where $d_{x,i}$ is the distance yield by algorithm *x* on observation *i*. The values of *i* refer to the observations (i=1,…,n), $f_{1,i}$ and $f_{2,i}$ to values on the frontier and $f_{1,i}^*$, $f_{2,i}^*$ to the actual output of the algorithm on i. Notice that this distance (unlike the taxicab one) is negative. In case an algorithm does not reach a solution, we assign the maximal distance found for the other algorithms on that observation. Table 8 shows the P-values of differences in means test for the different algorithms. As indicated, the null hypothesis is the means are equal. All the results are significant, meaning, in particular, that the number of cases considered were enough to make the assessment.

Table 8

P-values in the difference of means test between IBEA, NSGAII and SPEAII.

P-values (IBEA, NSGAII and SPEAII)			
Test	Pr (|T| > |t|)	Pr(T > t)	Pr(T < t)
IBEA=SPEAII	0.0000	0.0000	1.0000
IBEA=NSGAII	0.0000	0.0000	1.0000
SPEAII=NSGAII	0.4895	0.2448	0.7552
H0	mean(diff) = 0	mean(diff) = 0	mean(diff) = 0
H1	mean(diff) ≠ 0	mean(diff) > 0	mean(diff) < 0

It can be seen that IBEA is more efficient (in the sense that the distance to the frontier is much shorter, indicated by Pr(T>t)). On the other hand, there are no significant differences between SPEAII and NSGAII.

4.2 MOHEA vs. MOEA: Why Hybrid?

Now we report the results of experiments comparing the MOEA (IBEA in our case) and the MOEA complemented with a search process (IBEA + Simulated Annealing). In Figures 3, 4, 5, 6 and 7 (Left) we show the Pareto frontiers for both algorithms. The MOEA yields an incomplete frontier, which, moreover is sometimes dominated by the frontier obtained by the MOHEA. Furthermore, the latter

exhibits a better distribution of solutions. In Figs. 3, 4, 5, 6 and Fig. 7 (Right) show the mean number of undominated solutions (S) found by the MOHEA and the MOEA, for different generation numbers (G). It can be seen that for the 250 generations run by the two algorithms, there exists a clear difference between them. Other experiments, not reported here, indicated that the MOEA reached the undominated solutions for these problems around generation 500. Putting this together with the number of evaluations in the search process we conclude that for similar results, the MOHEA on average required 35,2 % less evaluations than MOEA.

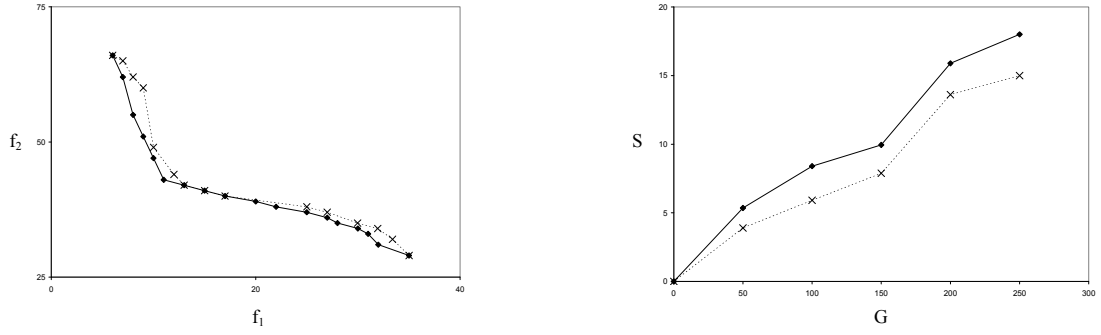

Fig. 3. f_1 vs. f_2 (Left) and G vs. S (Right) for MOHEA (–◆–) and MOEA (··×··) (MF01)

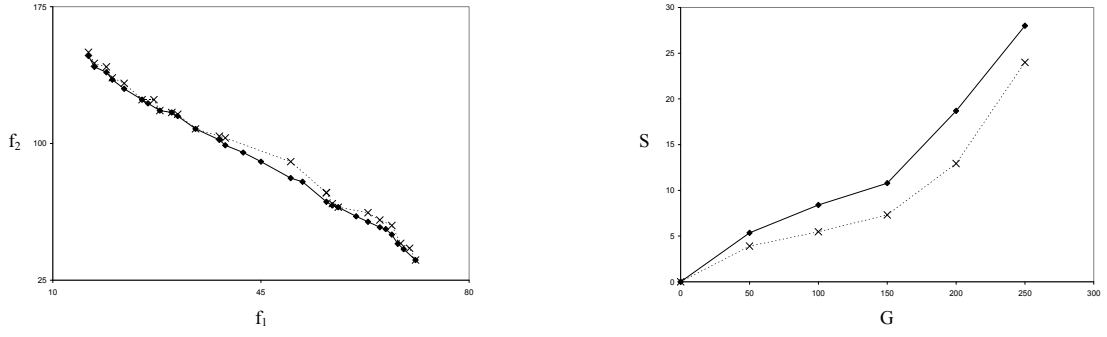

Fig. 4. f_1 vs. f_2 (Left) and G vs. S (Right) for MOHEA (–◆–) and MOEA (··×··) (MF02)

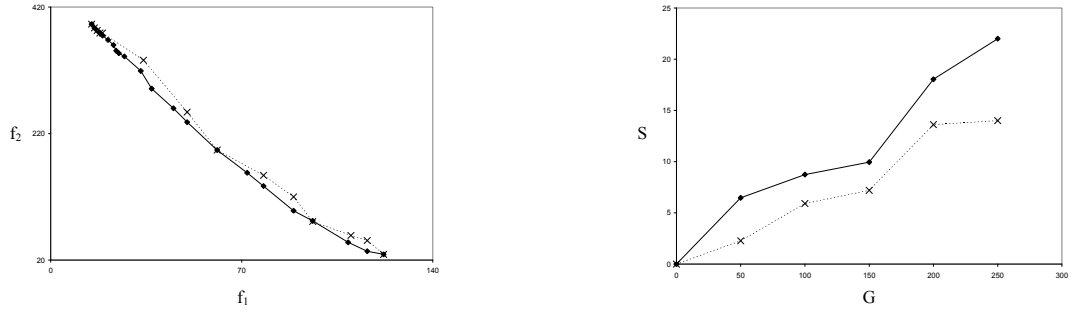

Fig. 5. f_1 vs. f_2 (Left) and G vs. S (Right) for MOHEA (–◆–) and MOEA (··×··) (MF03)

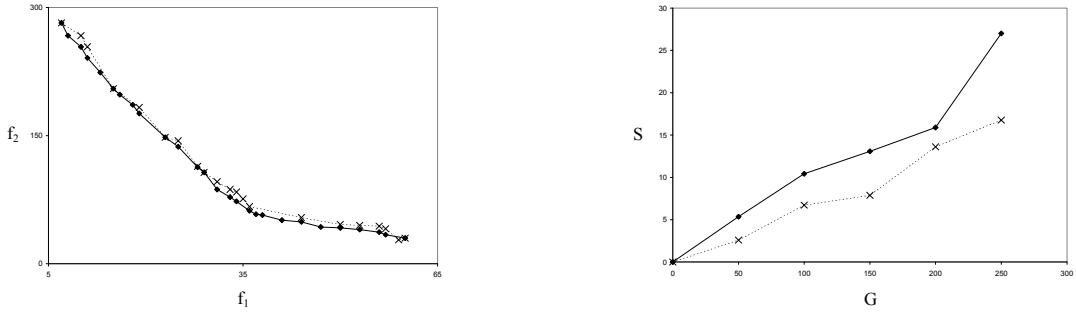

Fig. 6. f_1 vs. f_2 (Left) and G vs. S (Right) for MOHEA (–◆–) and MOEA (··×··) (MF04)

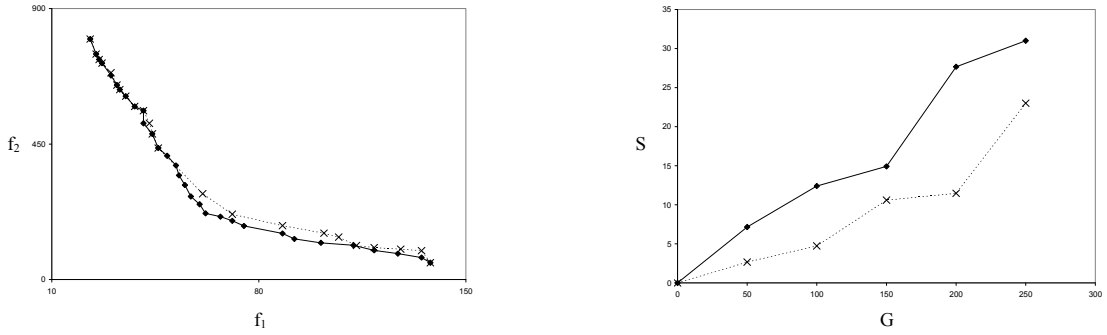

Fig. 7. f_1 vs. f_2 (Left) and G vs. S (Right) for MOHEA (–◆–) and MOEA (··×··) (MF05)

4.3 Comparison of the MOHEA with HABC and MPICA

We compared also the results under our MOHEA with those obtained by the Hybrid Artificial Bee Colony Algorithm (HABC) introduced by Li et al., (2011) and by the Multi-Population Interactive Coevolutionary Algorithm (MPICA) presented in (Xing et al., 2011). These two algorithms were implemented in C++. The parameters for them were taken from the publications in which they have been presented. They were also run 30 times each and the outcomes were evaluated according to the same metrics used in the choice of the selector. Fisher's test was used again with a confidence level $\alpha = 0.05$. Tables 9, 10 and 11 show no significant differences for MF01, MF02 and MF03 under the different algorithms and indexes. Tables 12 and 13 show significant differences in problems MF04 and MF05, between MOHEA and MPICA over HABC.

Table 9
I_e, I_H, and I_{R2} (MF01) for HABC, MPICA and MOHEA

MF01 / Problem 3 × 4 with 8 operations (flexible)									
	I_e			I_H			I_{R2}		
	MOHEA	HABC	MPICA	MOHEA	HABC	MPICA	MOHEA	HABC	MPICA
MOHEA	-	0,36593	0,41144	-	0,35495	0,39910	-	0,38423	0,43201
HABC	0,63407	-	0,52850	0,64505	-	0,51265	0,61577	-	0,55493
MPICA	0,58856	0,47150	-	0,60090	0,48736	-	0,56799	0,44508	-

Table 10
I_e, I_H, and I_{R2} (MF02) for HABC, MPICA and MOHEA

MF02 / Problem 4 × 5 with 12 operations (flexible)									
	I_e			I_H			I_{R2}		
	MOHEA	HABC	MPICA	MOHEA	HABC	MPICA	MOHEA	HABC	MPICA
MOHEA	-	0,18634	0,23476	-	0,17702	0,22302	-	0,19938	0,25119
HABC	0,81366	-	0,51885	0,82298	-	0,49291	0,80062	-	0,55517
MPICA	0,76524	0,48115	-	0,77698	0,50709	-	0,74881	0,44483	-

Table 11
I_e, I_H, and I_{R2} (MF03) for HABC, MPICA and MOHEA

MF03 / Problem 10 × 7 with 29 operations (flexible)									
	I_e			I_H			I_{R2}		
	MOHEA	HABC	MPICA	MOHEA	HABC	MPICA	MOHEA	HABC	MPICA
MOHEA	-	0,37590	0,43288	-	0,35335	0,40691	-	0,41349	0,47617
HABC	0,62410	-	0,56399	0,64665	-	0,53015	0,58651	-	0,62039
MPICA	0,56712	0,43601	-	0,59309	0,46985	-	0,52383	0,37961	-

As in the previous section we run a robustness analysis, again by the same procedure. The results are reported in Table 14, which shows that MOHEA yields better results than MPICA and HABC. In turn, these last two algorithms do not exhibit significant differences.

Table 12

I_e, I_H, and I_{R2} (MF04) for HABC, MPICA and MOHEA

MF04 / Problem 10 × 10 with 30 operations (flexible)									
	I_e			I_H			I_{R2}		
	MOHEA	HABC	MPICA	MOHEA	HABC	MPICA	MOHEA	HABC	MPICA
MOHEA	-	0,03178	0,60145	-	0,03019	0,57138	-	0,03448	0,65257
HABC	0,96822	-	0,96711	0,96981	-	0,95260	0,96552	-	0,98162
MPICA	0,39855	0,03289	-	0,42862	0,04740	-	0,34743	0,01838	-

Table 13

I_e, I_H, and I_{R2} (MF05) for HABC, MPICA and MOHEA

MF05 / Problem 15 × 10 with 56 operations (flexible)									
	I_e			I_H			I_{R2}		
	MOHEA	HABC	MPICA	MOHEA	HABC	MPICA	MOHEA	HABC	MPICA
MOHEA	-	0,02749	0,45332	-	0,02653	0,43745	-	0,02776	0,47825
HABC	0,97251	-	0,97112	0,97347	-	0,95170	0,97224	-	0,97209
MPICA	0,54668	0,02888	-	0,56255	0,04830	-	0,52175	0,02791	-

Table 14

P-values in the difference of means test between MOHEA, HABC and MPICA

P-values (MOHEA, HABC and MPICA)			
Test	Pr (|T| > |t|)	Pr(T > t)	Pr(T < t)
MOHEA=MPICA	0.0001	0.0000	1.0000
MOHEA=HABC	0.0000	0.0000	1.0000
MPICA=HABC	0.9250	0.5375	0.4625
H0	mean(diff) = 0	mean(diff) = 0	mean(diff) = 0
H1	mean(diff) ≠ 0	mean(diff) > 0	mean(diff) < 0

Finally, we also compare the mean running times of the three algorithms (see Table 15). We see that HABC runs faster that MOHEA and MPICA, but as seen above, its solutions are not as good as those of MOHEA. On the other hand MOHEA ran faster than MPICA in four out of five cases.

Table 15

Mean Running Time for MOHEA, HABC and MPICA

Mean Running Time			
	MOHEA (in seconds)	HABC (Δ% MOHEA)	MPICA (Δ% MOHEA)
MF01	84,28	-15,66	4,32
MF02	146,91	-10,41	7,89
MF03	165,98	-9,32	2,31
MF04	302,20	-17,97	-5,20
MF05	398,90	-20,75	10,78

5. Conclusion

We presented a Multi-Objective Hybrid Evolutionary Algorithm (MOHEA) for the Flexible Job-Shop Scheduling Problem (FJSSP). 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 the first component of the MOHEA. IBEA showed to perform better than NSGAII and SPEAII. We also compared MOHEA with the Hybrid Artificial Bee Colony Algorithm (HABC) and the Multi-Population Interactive Coevolutionary Algorithm (MPICA). The running time performances of MOHEA and MPICA were better than that of HABC. Only in one case MPICA improved over MOHEA. We can conclude that our hybrid structure intended to provide solutions to the FJSSP obtains good solutions at a reasonable time.

Acknowledgement

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/ZJ34. We want

also thank Dr. Ana C. Olivera for her constant support and help during this research.

References

Agnetis, A., Flamini, M., Nicosia, G. & Pacifici, A. (2001). A job-shop problem with one additional resource type. *Journal of Scheduling*, *14*(3), 225-237.

Armentano, V. A. & Scrich, C. R. (2000). Tabu search for minimizing total tardiness in a job-Shop. *International Journal Production Economics*, *63*(2), 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. *Evolutionary Multi-Criterion Optimization*, *2632*, 494-508.

Brandimarte P. (1993). Routing and scheduling in a flexible job-shop by tabu search. *Annals of Operations Research*, *41*(1), 157-183.

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.

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*. New York, Springer-Verlag.

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 matheuristic approach for the two-machine total completion time flow-shop problem. *Annals of Operations Research*, *213*(1), 67-78.

Fattahi, P., Saidi, M. & Jolai, F. (2007). Mathematical modeling and heuristic approaches to flexible job-shop scheduling problems. *Jounal of Intelligent Manufacturing*, *8*(3), 331-342.

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. & Tohmé, F. (2015). Choice of a PISA selector in a hybrid algorithmic structure for the FJSSP. *Decision Science Letters*, *4*(1), 247-260.

Frutos, M. & Tohmé, F. (2009). Desarrollo de un procedimiento genético diseñado para programar la producción en un sistema de manufactura tipo job-shop. *Proc. VI Congreso Español sobre Meta-heurísticas, Algoritmos Evolutivos y Bioinspirados*, Málaga, Spain, 23-30.

Goldberg, D. E. (1989). *Genetic Algorithms In Search. Optimization and Machine Learning*. Massachusetts, Addison Wesley.

Hansmann, R. S., Rieger, T. & Zimmermann, U. T. (2014). Flexible job shop scheduling with blockages. *Mathematical Methods of Operations Research*, *79*(2), 135-161.

Heckman, I. & Beck, J. C. (2011). Understanding the behavior of solution-guided search for job-shop scheduling. *Journal of Scheduling*, *14* (2), 121-140.

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.

Ho, N. B., Tay, J. C. & Lai, E. M. (2007). An effective architecture for learning and evolving flexible job-shop schedules. *European Journal of Operational Research*, *179*(2), 316-333.

Ho, N. B. & Tay, J. C. (2005). Evolving dispatching rules for solving the flexible job-shop problem. *Proc. IEEE Congress on Evolutionary Computation*, *3*, 2848-2855.

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

Li, J., Pan, Q. & Chen, J. (2012). An effective shuffled frog-leaping algorithm for multi-objective flexible job shop scheduling problems. *Journal of Production Research*, *50*(4), 1063-1078.

Li, J., Pan, Q. & Gao, K. (2011). Pareto-based discrete artificial bee colony algorithm for multi-objective flexible job shop scheduling pProblems. *International Journal of Advanced Manufacturing Technology*, *55*(9), 1159-1169.

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.

Mesghouni, K., Hammadi, S. & Borne, P. (1997). Evolution programs for job-shop scheduling. *Proc. IEEE International Conference on Systems, Man, and Cybernetics*, *1*, 720-725.

Nazarathy, Y. & Weiss, G. (2010). A fluid approach to large volume job shop scheduling. *Journal of Scheduling*, *13*(5), 509-529.

Nidhiry, N. M. & Saravanan, R. (2012). Evaluation of genetic algorithm approach for scheduling optimization of flexible manufacturing systems. *International Journal of Engineering Research and Applications*, *2*(4), 437-446.

Nowicki, E. & Smutnicki, C. (2005). An advanced tabu search algorithm for the job shop problem. *Journal of Scheduling*, *8*(2), 145-159.

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

Pezzella, F., Morganti, G. & Ciaschetti, G. (2008). A genetic algorithm for the flexible job-shop scheduling problem. *Journal Computers and Operations Research*, *35*(10), 3202-3212.

Tay, J. C. & Wibowo, D. (2004). An effective chromosome representation for evolving flexible job-shop schedules. *Proc. GECCO 2004*, LNCS 3103, 210-221.

Tsai, C. F. & Lin, F. C. (2003). A new hybrid heuristic technique for solving job-shop scheduling problems. *Proc. Second IEEE International Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications*.

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.

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. *Proc. 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.

Xiong, J., Tan, X., Yang, K., Xing, L. & Chen, Y. (2012). A hybrid multiobjective evolutionary approach for flexible job shop scheduling problems. *Mathematical Problems in Engineering*, *1*, 1-27.

Yang, S., Wang, D., Chai, T. & Kendall, G. (2010). An improved constraint satisfaction adaptive neural network for job-shop scheduling. *Journal of Scheduling*, *13*(1), 17-38.

Yazdani, M., Zandieh, M. & Amiri, M. (2010). Flexible job-shop scheduling with parallel variable neighborhood search algorithm. *Expert Systems with Applications: An International Journal*, *37*(1), 678-687.

Yuan, Y. & Xu, H. (2015). Multiobjective flexible job shop scheduling using memetic algorithms. *IEEE Transactions on Automation Science and Engineering*, *12* (1), 336-353.

Zhang, G. & Gen, M. (2005). Multistaged-based genetic algorithm for flexible job-shop scheduling problem. *Complexity International*, *11*, 223-232.

Zitzler, E. & Künzli, S. (2004). Indicator-based selection in multiobjective search. *Proc. Conference on Parallel Problem Solving from Nature (PPSN VIII)*, LNCS 3242, 832-842.

Zitzler, E., Laumanns, M. & Thiele, L. (2002). SPEAII: Improving the strength pareto evolutionary algorithm for multi-objective optimization. *Evolutionary Methods for Design, Optimisations and Control*, 19-26.