# A HYBRID TECHNIQUE FOR HANDLING THE FLEXIBLE JOB-SHOP SCHEDULING PROBLEM

**Frutos Mariano**
Universidad Nacional del Sur
Departamento de Ingeniería
Bahía Blanca (B8000CPB), Argentina
mfrutos@uns.edu.ar

**Olivera Ana Carolina**
Universidad Nacional del Sur
Departamento de Ciencias e Ingeniería de la Computación
Bahía Blanca (B8000CPB), Argentina
aco@cs.uns.edu.ar

**Tohmé Fernando**
Universidad Nacional del Sur
Departamento de Economía
Bahía Blanca (B8000CPB), Argentina
ftohme@criba.edu.ar

## I. INTRODUCTION

To schedule production in a Job-Shop environment means to allocate adequately the available resources. To do that it is necessary to rely on efficient optimization procedures. In fact, the Job-Shop Scheduling Problem is a NP-Hard problem [Ullman, 1975], so ad-hoc algorithms have to be applied to its solution [Frutos et al., 2008]. This is similar to other combinatorial programming problems [Larrazábal, 2003], [Olivera et al., 2006], [Cortés et al., 2004].

Most instances of the Job-Shop Scheduling Problem involve the simultaneous optimization of two usually conflicting goals. This one, like most multi-objective problems, tends to have many solutions. The Pareto frontier reached by an optimization procedure has to contain a uniformly distributed number of solutions close to the ones in the true Pareto frontier. This feature facilitates the task of the expert who interprets the solutions [Kacem et al., 2002].

In this paper we present a Genetic Algorithm linked to a Simulated Annealing procedure able to schedule the production in a Job-Shop manufacturing system [Cortés et al., 2004], [Tsai and Lin, 2003], [Wu et al., 2004].

## II. FLEXIBLE JOB-SHOP SCHEDULING PROBLEM

The Job-Shop Scheduling Problem (JSSP) can be described as that of organizing the execution of n jobs on m machines. We assume a finite number of tasks, $\{J_j\}_{j=1}^{n}$. These tasks must be processed by a finite number of machines $\{M_k\}_{k=1}^{m}$. To process a task $J_j$ in a machine $M_k$ is denoted by $O^i_{jk}$, where $i$ indicates the order in which a class of operations $\{S_j\}_{j=1}^{n}$ is applied on a task $J_j$. $O^i_{jk}$ requires the uninterrupted use of a machine $M_k$ for a period $\tau^i_{jk}$ (the processing time) at a cost $\upsilon^i_{jk}$. In the particular case of Flexible JSSP (FLEX-JSSP), the allocation of $O^i_{jk}$ on $M_k$ is undifferentiated, which means that each $O^i_{jk}$ can be processed by any of the machines in $\{M_k\}_{k=1}^{m}$. After allocating the operations, we obtain a finite class $E$ of groupings of the $O^i_{jk}$s on the same machine. We denote each of these groupings as $E_k$, for $k = 1,...,m$. A key issue here is the scheduling of activities, i.e. the determination of the starting time $t^i_{jk}$ of each $O^i_{jk}$.

The FLEX-JSSP demands a procedure to handle its two sub-problems: the allocation of the $O^i_{jk}$s on the different $M_k$s and their sequencing, guided by the goals to reach. That is, to find optimal levels of Processing Time (PT*) (see EQ. 1) and Operation Costs (OC*) (see EQ. 2).

$$PT^* \rightarrow \text{Min}: \ \text{máx} \ (t^i_{jk} + \tau^i_{jk}) \quad \textbf{(EQ. 1)}$$

$$OC^* \rightarrow \text{Min}: \ \sum_j \sum_i \sum_k x^i_{jk} \upsilon^i_{jk} \quad \textbf{(EQ. 2)}$$

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, $t^i_{jk} = \text{máx} \ (t^{(i-1)}_{jh} + \tau^{(i-1)}_{jh}, \ t^s_{pk} + \tau^s_{pk}, \ 0)$ for each pair $O^{i-1}_{jh}, O^s_{pk} \in E_k$ and all machines $M_k$, $M_h$ and operations $S_i$, $S_s$.

## III. HYBRID GENETIC ALGORITHM

Due to its many advantages, evolutionary algorithms have become very popular for solving multi-objective optimization problems [Ztzler et al., 2001], [Coello Coello et al., 2002]. Among the evolutionary algorithms used, some of the most interesting are Genetic Algorithms (GA) [Goldberg, 1989].

To represent the individuals, we use a variant of [Wu et al., 2004]. Since the FLEX-JSSP has two subproblems, the Hybrid Genetic Algorithm (HGA) presented here operates over two chromosomes. The first one represents the allocation $A^i_{jk}$ of each $O^i_{jk}$ to every $M_k$. We denote with values between 0 and (k-1) the allocation of each $M_k$, that is, for k = 4, we might have something like $0 \rightarrow M_1$, $1 \rightarrow M_2$, $2 \rightarrow M_3$ and $3 \rightarrow M_4$ (see TABLE 1A). The second chromosome represents the sequencing of the $O^i_{jk}$ already assigned to each of the $M_k (\forall O^i_{jk} \in E_k)$. We denote with values between 0 and (n!-1) the sequence of $J_j$ in each $M_k$. That is, for n = 3, we may have $0 \rightarrow J_1J_2J_3$, $1 \rightarrow J_1J_3J_2$, $2 \rightarrow J_2J_1J_3$, $3 \rightarrow J_2J_3J_1$, $4 \rightarrow J_3J_1J_2$ y $5 \rightarrow J_3J_2J_1$ (see TABLE 1B).

| Allocations | | | | Sequences | | |
|---|---|---|---|---|---|---|
| $J_j$ | $O_j^i{}_k$ | $A_j^i{}_k$ | Chr | $M_k$ | Order | Chr |
| $J_1$ | $O_1^1{}_k$ | $M_4$ | 3 | $M_1$ | $J_2\ J_3\ J_1$ | 3 |
| | $O_1^2{}_k$ | $M_4$ | 3 | $M_2$ | $J_2\ J_3\ J_1$ | 3 |
| | $O_1^3{}_k$ | $M_1$ | 0 | $M_3$ | $J_1\ J_2\ J_3$ | 0 |
| $J_2$ | $O_2^1{}_k$ | $M_2$ | 1 | $M_4$ | $J_3\ J_2\ J_1$ | 5 |
| | $O_2^2{}_k$ | $M_2$ | 1 | | | |
| | $O_2^3{}_k$ | $M_4$ | 3 | | | |
| $J_3$ | $O_3^1{}_k$ | $M_3$ | 2 | | | |
| | $O_3^2{}_k$ | $M_4$ | 3 | | | |
| (A) | | | | (B) | | |

**TABLE 1** CHROMOSOMES OF ALLOCATIONS AND THE SEQUENCES

The algorithm NSGA-II (Non-Dominated Sorting Genetic Algorithm II) [Deb et al., 2002], creates an initial population, be it random or otherwise. NSGA-II uses an elitist strategy joint with an explicit diversity mechanism. Each individual candidate solution i is assumed to have an associated rank of non-dominance $r_i$ and a distance $d_i$ which indicates the radius of the area in the search space around i not occupied by another solution. A solution $i$ is preferred over j if $r_i < r_j$. When $i$ and $j$ have the same rank, i is preferred if $d_i > d_j$. Let $Y_i$ be an ordered class of individuals with same rank as $i$ and $f^{i+1}_j$ the value for objective j for the individual after i, while $f^{i-1}_j$ is the value for the individual before i. $f_j^{máx}$ is the maximal value for j among $Y_i$ while $f_j^{min}$ is the minimal value among $Y_i$ (see FIG. 1). The distances consider all the objective functions and attach an infinite value to the extreme solutions in $Y_i$. Since these yield the best values for one of the objective functions on the frontier, the resulting distance is the sum of the distances for the objective functions.

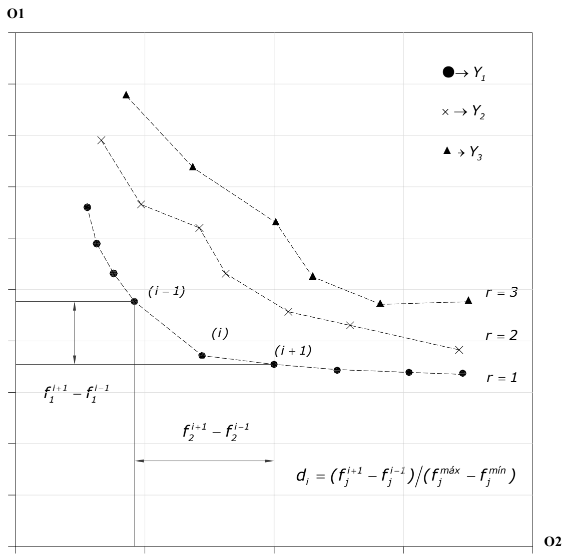


**FIG. 1** NON-DOMINATED SORTING GENETIC ALGORITHM II (NON-DOMINANCE RANK $R_I$ AND DISTANCE $D_I$)

Starting with a population $P_t$ a new population of descendents $Q_t$ obtains. These two populations mix to yield a new one, $R_t$ of size $2N$ (N is the original size of $P_t$). The individuals in $R_t$ are ranked with respect the frontier and a new population $P_{t+1}$ obtains applying a tournament selection to $R_t$ (see FIG. 2).

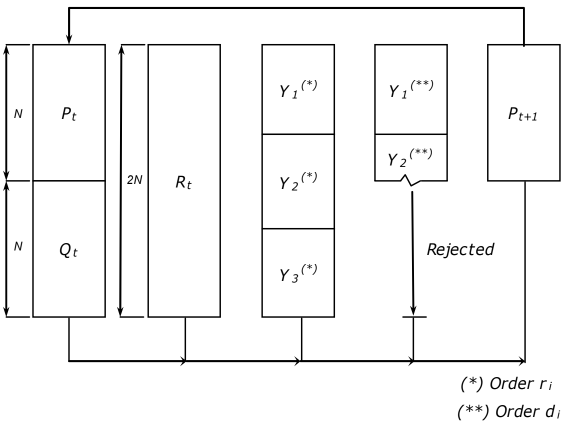


**FIG. 2** NON-DOMINATED SORTING GENETIC ALGORITHM II (OUTLINE OF THE MECHANISM OF PROMOTION OF INDIVIDUALS)

After experimenting with several genetic operators we have chosen the Partially Mapped Crossover (PMX) for the crossover and 2-SWAP for mutation. After the individuals have been affected by these operators and before allowing them to become part of a new population we apply an improvement operator. This operator has been designed following the meta-heuristic Simulated Annealing (SA) [Dowsland, 1993]. For the change of structure of both chromosomes we select a gen randomly to change its value. This process, which complements the genetic procedure follows the guidelines of SA.

## IV. CONCLUSIONS

We presented a Hybrid Genetic Algorithm (HGA) intended to solve the Flexible Job-Shop Scheduling Problem (FLEX-JSSP). The application of HGA required the calibration of parameters, in order to yield valid values for the problem at hand, which are also a reference for similar problems. We are currently experimenting with other techniques of local search in order to achieve a more aggressive exploration. We are also interested in evaluating the performance of the procedure for other kinds of problems to see whether it saves resources without sacrificing precision in the convergence.

## REFERENCE


[1] Coello Coello C. A., Van Veldhuizen D. A. and Lamont G. B., Evolutionary Algorithms for Solving Multi-Objective Problems, Kluwer Academic Publishers, New York, 2002.

[2] Cortés Rivera D., Coello Coello C. A. and Cortés N. C., Job shop scheduling using the clonal selection principle, ACDM'2004, UK, 2004.

[3] Deb K., Pratap A., Agarwal S. and Meyarivan T., A Fast and Elitist Multi-objective Genetic Algorithm: NSGAII, IEEE Transactions on Evolutionary Computation, Vol. 6 (2), pp 182-197, 2002.

[4] Dowsland K. A., Simulated Annealing, Modern Heuristic Techniques for Combinatorial Problems, Ed. C. R. Reeves, Blackwell Scientific Pub, Oxford, 1993.

[5] Frutos M. y Olivera A. C., Job shop scheduling problem: desarrollo de un procedimiento eficiente, In Proceeding XIV Congreso Latino-Ibero-Americano en Investigación de Operaciones, Colombia, 2008.

[6] Goldberg, D., Genetic Algorithms in Search, Optimization, and Machine Learning, Addison Wesley, 1989.

[7] Kacem I., Hammadi S. and Borne P., Approach by Localization and Multi-Objective Evolutionary Optimization for Flexible Job-Shop Scheduling Problems, IEEE Trans. Syst. Man Cybernetics, Vol. 32, 2002.

[8] Larrazábal M. C., Configuración de Sistemas de Producción, Optimización basada en Simulación, Memoria de Título, Universidad de Concepción. Chile, 2003.

[9] Olivera A. C., Frutos M., y Casal R., Métodos para determinar secuencias de producción en un ambiente productivo complejo, In Proceeding XIII Congreso Latino-Iberoamericano de Investigación Operativa, Uruguay, 2006.

[10] Tsai C. F. and Lin F. C., A new hybrid heuristic technique for solving job-shop scheduling problem, Intelligent Data Acquisition and Advanced Computing Systems, Second IEEE International Workshop, 2003.

[11] Ullman J. D., NP-complete scheduling problems". Journal of Computer System sciences, Vol. 10, pp 384-393, 1975.

[12] Wu C. G., Xing X. L., Lee H. P., Zhou C. G. and Liang Y. C., Genetic Algorithm Application on the Job Shop Scheduling Problem, Machine Learning and Cybernetics, International Conference, Vol. 4, pp 2102- 2106, 2004.

[13] Zitzler E., Deb K., Thiele L., Coello Coello C. A. and Corne D., Evolutionary multi-criterion optimization, In Lecture Notes in Computer Science, Vol. 1993, Springer, Berlin, 2001.