GENETIC OPERATORS FOR THE MULTIOBJECTIVE FLOWSHOW PROBLEM

Magdalena Bandala*, María A. Osorio-Lama**
School of Computer Sciences, Universidad Autónoma de Puebla
Ciudad Universitaria, Puebla, Pue. 72560, México
*

RESUMEN:
Uno de los problemas más importantes en los Algoritmos Genéticos, es la selección correcta de los operadores de cruza y mutación. Los operadores genéticos son más importantes para los cromosomas no binarios debido a su impacto en los resultados. Este trabajo presenta un análisis comparativo de diferentes operadores de cruza y mutación aplicados a un algoritmo genético para el problema multiobjetivo de calendarización de procesos con transferencia cero. El algoritmo utilizado está adaptado de un método de partición propuesto por Tagami et al [7] y construye una frontera de Pareto, minimizando la duración y el tiempo promedio de proceso**.**

ABSTRACT: One of the must important issues in Genetic Algorithms is the right selection of crossover and mutation operators. Genetic Operators are even more important for non binary chromosomes due to their high impact on the results. This work presents a comparative analysis of different crossover and mutation operators applied to a genetic algorithm for the multiobjective flowshop problem. The algorithm used is adapted from the partition method proposed by Tagami et al[8] and builds a Pareto's frontier. We minimize the makespan and the mean flowtime.

Key Words: mutation operators, partition methods, Parteto's frontier

MSC: 90C29

1. INTRODUCTION

In the flowshop problem, we have a set of tasks that must be processed in several machines. Not all the tasks have to use all the machines, each task is processed in a subset of stages. It can be assumed transference with zero time or a continuous flow between the stages in the process. The flowshop sequencing is characterized by unidirectional flow and is NP-hard.

The flowshop consist of *m* machines and *n* different jobs to be optimally sequenced through these machines. The common assumptions used in modeling the flowshop problem are:

- All *n* jobs are available for processing at time zero and each job follows identical routing through the machines.
- Jobs are independent and available in time 0.
- Unlimited storage exists between the machines. Each job requires m operations and each operation requires a different machine.
- Every machine processes only one job at one time and every job is processed on one machine at one time.
- Setup times for the operations are sequence-independent and are included in processing times.
- The machines are continuously available.
- Individual operations cannot be pre-empted.

* *arflor@yahoo.com.mx, **aosorio@cs.buap.mx

In theory, integer programming and the branch and bound technique can be used to solve the flowshop problem optimally. However, these methods are not viable on large problems. Most scheduling problems including flowshop problems belong to the NP-hard class for which the computational effort increases exponentially with problem size. To remedy this, researchers have continually focused on developing heuristic and other methods. Heuristics methods typically do not guarantee optimality of the final solution, but a final solution is reached quickly ad is acceptable for practical use.

Several good heuristic methods have appeared in the past three decades to help minimize makespans for flowshops. Recent advances in meta-heuristic search methods that help conduct directed "intelligent" search of the solution space have brought yet new possibilities to rapidly find good schedules, even if they are not optimal.

2. MIP FORMULATION

The Mixed Integer Formulation (MIP) for the flowshop problem considers a set of constraints that avoid collisions in every stage in the process. The mono objective problem only addresses the minimization of total time in the process, the makespan while the multiobjective may consider the flowtime and the tardiness.

2.1. Constraints

The model has two types of constraints. The first set guarantees that the total time, named makespan (T) exceeds the total time of every job in the system. The second type, avoid collisions between each pair of the jobs in the first stage, that coincides in the system.

$$T \geq t_i + \sum_{j \in J} t_{ij} \qquad \forall i \in I$$

$$t_k + \sum_{\substack{m \in J(k) \\ m \leq j}} t_{km} \leq t_i + \sum_{\substack{m \in J(i) \\ m < j}} t_{im} + M y_{ik}$$

$$t_k + \sum_{\substack{m \in J(k) \\ m \leq j}} t_{km} \leq t_i + \sum_{\substack{m \in J(i) \\ m < j}} t_{im} + M y_{ik}$$

$$\forall j \in C_{ik}, \forall i,k \in I, i \leq k$$

$$t_i \geq 0, \quad i \in I, \quad y_{ik} = 0,1 \qquad \forall i,k \in I, i \leq k$$

Where t_{ij} is the time that job *i* stays in stage *j* and t_i is the initial time for job *i*.

2.2. Objectives

The objective is to find a stages' sequence according to an optimal criteria. The classical three scheduling objectives are (1) makespan, (2) mean flowtime and (3) mean tardiness. In this research, we only worked with makespan and flowtime.

2.1.1. Makespan.

In scheduling literature, makespan is defined as the maximum completion time of all jobs, or the time taken to complete the last job on the last machine in the schedule –assuming that the processing of the first job began at time 0. Makespan is denoted by *T* and computed as:

$$T = max \{F_j\}$$

$i \leq j \leq n$

Where F_j is the flowtime for job j, i.e., the total time taken by job j from the instant of its release to the shop to the time it is processed by the last machine.

2.1.2 Mean Flowtime.

The mean flowtime measures the average response of the schedule to individual demands of jobs for service. Mathematically, mean flow time is the average of the flow times of all jobs. It is usually represented by F and expressed as:

$$F_2(x) = \sum_{j=1}^{n} f_j \text{ /JOBS}$$

3. MULTIOBJECTIVE FLOWSHOP

In sequencing jobs in a flowshop we are likely to confront several different (and often conflicting) management objectives. Consequently, a schedule may have to be evaluated by different types of performance measures. Some of these measures may give importance to completion time (e.g. makespan), some to due date (e.g. mean tardiness, maximum tardiness), and some others to speed with which the jobs flow (e.g. mean flow time). The simultaneous consideration of these objectives is a multiobjective optimization problem. But, even for a single objective, flowshop sequencing is NP-hard.

3.1 Pareto Optimality

A common difficulty with multiobjective optimization is the appearance of "objective conflict" (Hans, 1988): none of the feasible solutions achieves simultaneous optimization of makespan, mean flow time, mean tardiness, machine utilization, etc. in a flowshop, for example. In other words, the individual optimal solutions of each objective are usually different. Thus a "most favored" solution would be one that would cause minimum objective conflict. Such solutions may be viewed as points in a solution space.

The general multiobjective optimization problem contains a number of objectives while the solution (2) must also satisfy a number of inequality and equality constraints. Mathematically, the problem may be stated as follows.

$$\text{Minimize} \quad F(x) = (F_1(x), F_2(x))$$

$$F_1(x) = T \text{ (Makespan)}$$

$$F_2(x) = \sum_{j=1}^{n} f_j \text{ / } JOBS \text{ (Flowtime)}$$

Here the decision parameter $\boldsymbol{x}$ is a p-dimensional vector made up of p decision variables. Solutions to a multiobjective optimization problem are mathematically expressed in terms of nondominated points.

The nondomination property of solutions may be explained as follows. In a minimization problem, a solution vector $x(1)$ is partially dominated by another vector $x(2)$ (written as $x(1) \prec x(2)$), when no component value of $x(2)$ is less than $x(1)$ and at least one component of $x(2)$ is strictly greater than $x(1)$. In a minimization problem if $x(1)$ is partially less than $x(2)$, we say that the solution $x(1)$ dominates $x(2)$ or the solution $x(2)$ is inferior to $x(1)$. Any ember of such vectors that is not dominated by any other member is said to be nondominated or noninferior. This research gets the Pareto's frontier for solutions that minimize the makespan and the mean flowtime.

3.3 Multiobjective Genetic Algorithms

There are two types of evolutive algorithms for mulitobjective optimization. Algorithms that do not incorporate the Pareto's optimality concepts for the selection process and algorithms that hierarchize the population according to Pareto's domination.

For the algorithms that consider Pareto optimization there are two generations. The first generation of algorithms utilizes a ranking for the best individuals. The second generation uses the concept of elitism for the individuals' selection and for the selection of secondary populations of individuals.

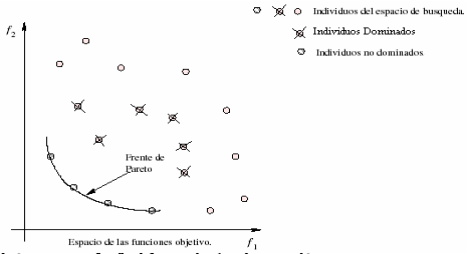

Figure. 1 A Pareto's Frontier

4. GENETIC OPERATORS

4.1 Crossover

The genetic algorithm was implemented in C. The individuals use a non binary representation for the chromosomes using the jobs sequence as in Kobayashi et al [1995].

The selection of chromosomes for crossover is made by the creation of an elite subset. From the original population, the algorithm selects the individuals with better qualifications ($f_i \geq 8$) and then randomly selects the individuals that will perform the crossover. The selected individuals can have different f_i .This type of selection does not use the crossover probability because the elements in the elite subset will perform the crossover with the other elements in the subset until the population generated reaches the same size than the original population.

Due to the nonbinary chromosomes, we used two crossover: 1) a one point crossover with a reparation algorithm proposed by Jayaram [16], that forces solutions to be feasible, and 2) the subsequence exchange crossover presented by Kobayashi et al [1995].

4.1.1 One Point Crossover

It randomly choices a point in the chromosomes chosen to be the parents and from this point, it interchanges the information from both chromosomes to generate new individuals. This crossover method can generate invalid chromosomes and requires a reparation method that restores feasibility in the chromosome. The algorithm has the following steps:

1. It randomly selects a cutting point in the parent chromosomes.
2. It copies in the new individuals, the chain before the cutting point, in the chromosomes parents
3. In the first individual generated, the empty positions will be filled with elements from the second parent that do not exist in the first individual, in the same order that they appear in the second father.

4. In the second individual generated, the empty positions will be filled with elements from the first parent that do not exist in the second individual, in the same order that they appear in the first father.
5.

FIRST GENERATION	Author	Characteristics	Year
VEGA	Schaffer	Divides population according to the number of objectives, gives priority to only one.	1984
MOGA	Fonseca and Fleming,	Hierarchize the population according to Pareto dominance with niches.	1993
NSGA	Srinivas y Deb	Hierarchize the population by slides. Keeps diversity.	1994
NPGA	Horn, Nafpliotis and Goldberg	Make tournaments between individuals. Hierarchize the population according to the tournament.	1994
SECOND GENERATION Elitism + Niches	**Author**	**Characteristics**	**Year**
SPEA	Zitzler and Thiele	Uses a population to help nondominated individuals. Hierarchize the external population.	1999
PAES	Knowlesand and Corne	Uses an auxiliar population for the nondominated individuals. Divides the objective spaces in grids. Make selection by elitism and Pareto.	1999
NSGA II	Deb, Agrawal, Pratap and Meyarivan	Hierarchize population according to NSGA:.	2000
PESA	Corne, Knowles and Oates	Uses auxiliary population for the nondominated individuals.. Divides the objective space in grids and uses a selection criterion based on elitism and Pareto.	2000
NSGA-II	Ded and Goel	Controls elitism.	2001
SPEA2	Zitzler, Laumanns and Thiele	Based on Pareto's dominance. Keeps diversity.	2001

Figure. 2 Multiobjective algorithms' evolution

4.1.2 Subsequence Exchange Crossover

This method interchanges subsequences that have the same elements in each chromosome parent an interchange the chains. It originates valid chromosomes and does not need a reparation algorithm.

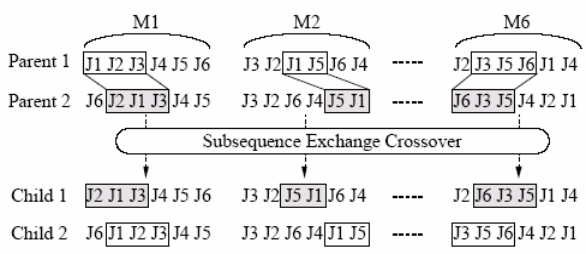

Figure. 3 Subsequence Exchange Crossover

4.2 Mutation

Mutation is made by interchanging adjacent jobs in the chromosome. The jobs used for mutation are chosen randomly.

During all experiments, the population size is set is set as 25, the mutation rate as 0.1 with only one interchange in the chromosomes. In the Pareto partitioning method, the number of partitioning region is set as the 50^2.

5. GENETIC ALGORITHM UTILIZED

The genetic algorithm proposed is based in the Pareto partitioning algorithm proposed by Takanori Tagami and Tohru Kawabe [1998].

The Pareto partitioning algorithm consists in dividing the solution space in specific regions, as a grid where every objective function has a certain number of specific regions and the objective is to put solutions in the Pareto's frontier. The evaluation function (fitness) for each individual p_i is defined as $f_i = 1/n_i$. The values of n_i denote the number of solutions nondominated in the region p_i.

There are algorithms that work with a very similar method and divide the solution space in cubic subspaces that constitutes a grid where the objective is to locate individuals that are nondominated in an uniform and distributed way in the Pareto's frontier. Those algorithms are: **PAES** (Pareto Archived Evolution Strategy), Knowles and Corne, (1999), **PESA** (Pareto Envelope-based Selection Algorithm), Knowles and Corne, (2000), **PESA II** Corne, Jerram, Knowles and Oates (2001).

The Pareto partitioning algorithm partition can be stated as follows:

Algorithm

Step 1 Generate a random initial population with M individuals.

Step 2 Calculate the makespan and the flowtime for each individual in the initial population.

Step 3 Calculate the fitness of each solution in the current population according to the multobjective ranking.

i) Assign a rank R_i to every individual in the population. The rank will correspond to the number of members in the population that dominate the individual.

ii) Assign the fitness f_i to all the nondominated individuals. The evaluation function f_i of every individual p_i is defined as follows:

$$f_i = (f_{\max} - f_s * (R_i - 1))$$

$$f_s = \frac{1}{M-1}(f_{\max} - f_{\min})$$

where M denotes the population size, f_{max} and f_{min} denote the maximun and the minimum fitness values.

Step 4 Generate a new population from the initial population, using the crossover operator

Step 5 Apply mutation to the new population.

Step 6 Apply the evaluation function (Fitness) to both populations.

Step 7 Select *M* individuals from all population on the basis of their fitness.

Step 8 If a terminal condition is satisfied, stop and return the best individuals. Otherwise go to Step 2.

6. COMPUTATIONAL RESULTS

We tested several sizes for the numerical examples presented by Bagchi in [1999]. The algorithm is implemented in C (gcc Linux/3.2.2) V 5.1 under Linux. We tested problems that range from 15 to 25 jobs and from 10 to 15 machines. The parameters considered are: population size (p_s), mutation probability (p_m), number of regions in the Pareto's partition $(p_s)^2$ and maximum number of generations ($t_{\max}$). The values for the parameters are reported in Table 1.

Parameter	Value
p_s	50, 70, 100
$(p_s)^2$	$(50)^2$, $(70)^2$, $(100)^2$
p_m	0.1
$t_{\max}$	100, 300,400,500

Table 1

Figure 4 shows the graph obtained with the settings that yield the best results for an example with 20 jobs and 10 machines. Figure 5 corresponds to an example with 25 jobs and 15 machines. The Pareto's frontier can be seen in both graphs.

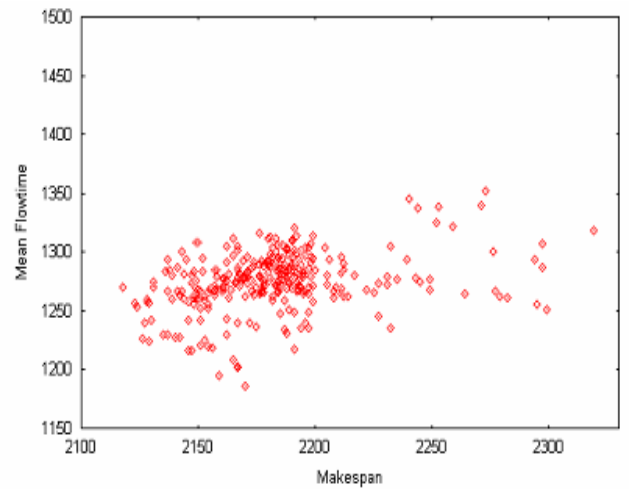

Figure. 4 Example with 20 jobs and 10 machines

Execution times in seconds of CPU, were obtained with the clock functions in language C for unix. We tested instances with 25 jobs and 15 machines and with 20 jobs and 10 machines and reported the execution times.

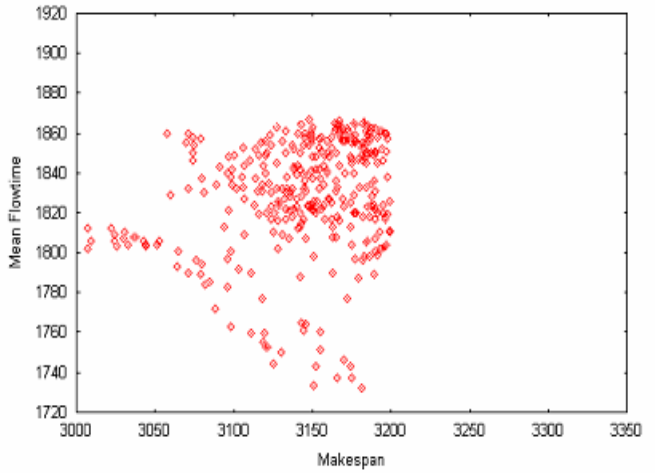

Figure. 6 Example with 25 jobs and 15 machines

Tables 2 and 3 show the execution average time in seconds for different population sizes (p_s), and with generations ranging from 100 to 1000. Table 4 shows that with a bigger of jobs and machines, the average execution time increases with the size of the population and the number of generations, although the time is less for p_s =50 and $t_{\max}$ =100. For our tests the best results were obtained with a size of p_s =100 and $t_{\max}$ =500 with the same execution time for both examples.

Figure 7 shows the average computational time and the population size for different number of generations. It can be seen that the computational time increases for bigger instance

p_s	t_{max} (100 ,1000)	Execution Average Time
50	(0.01 , 0.02)	0.018
70	(0.02,0.04)	0.029
100	(0.04,0.05)	0.046

Table 2 Execution Time (secs) for *n* =20 and *m* =10

5. CONCLUSIONS

The implemented selection produces diversity in the population, because it does not reproduce children equal to their parents. In addition, 65% to 70% of the individuals are apt for crossover. From both crossover methods, the 'subsequences interchange' has more computational work, because the number of comparisons necessary to find the subsequences with equal elements. When the chains contain greater number of elements (20 to 25), the resultant chains do not present a great change among them and its corresponding parents. This situation limits the exploration, since it generates very similar or equal chains sometimes.

p_s	t_{max} (100 , 1000)	Execution Average Time
50	(0.02 , 0.03)	0.024
70	(0.03, 0.04)	0.035
100	(0.04 , 0.06)	0.052

Table 3 Execution Time (secs) for *n* =25 and *m* =15

It is difficult to find subsequences of more than 5 characters with significant changes. If the chains correspond to small subsequences, they may be different between them but very similar to their parents, and a mutation procedure is necessary. For this reason, the method of crossover with chromosome reparation is used. For more than 10 works, the method of crossover with chromosome reparation offers a good exploration in a smaller computational time.

N	*m*	p_s =50 p_m =0.1 t_{max} (100,1000)	p_s =70 p_m =0.1 t_{max} (100,1000)
20	10	0.018	0.029
25	15	0.024	0.035

Table 4 Execution Time (secs)

In the mutation case, tests were done interchanging adjacent and random works. Better chains are obtained with a randomly mutation. Best results are obtained with a population of 100 individuals and a probability of 0.1 in 500 generations, as shown in table 5.

Parámeters	Best Value
p_s	100
$(p_s)^2$	$(100)^2$
p_m	0.1
$t_{\max}$	500

Table 5 Best Parameters

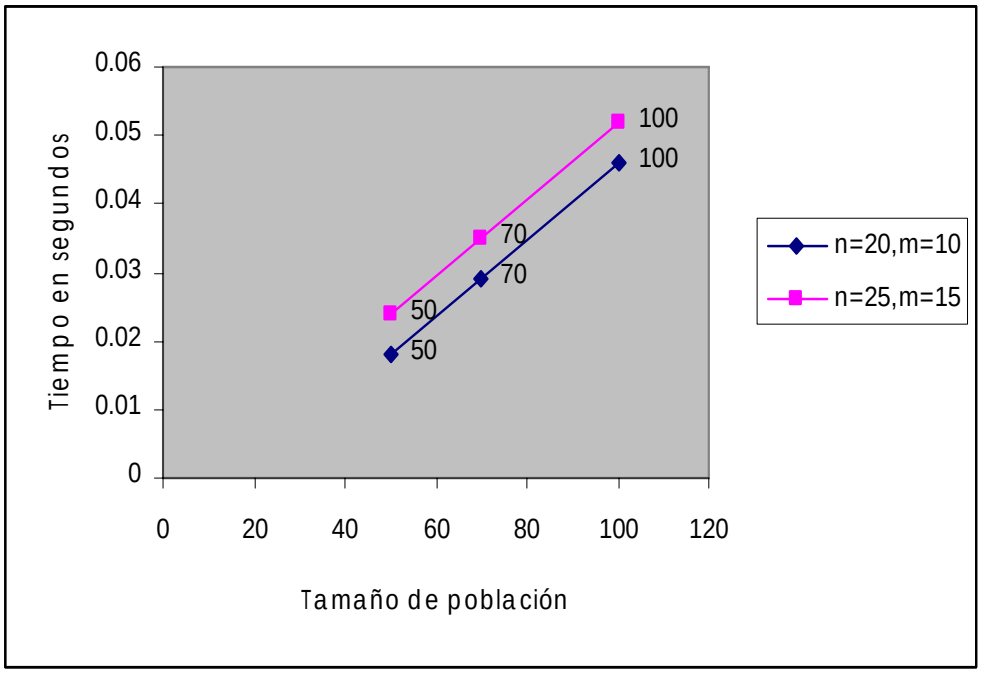

Figure 7 Time vs. Population size

Received April 2006
Revised March 2007

REFERENCES

BAGCHI, T. (1999) **Multiobjetive Scheduling by Genetic Algorithms,** Kluwer Academic Publishers, Boston/Dordrecht/London.

COELLO, C.A.; D. A. VELDHUIZEN, and G.B. LAMONT, (2002) **Evolutionary Algorithms for Solving Multiobjective Problems**. Kluwer Academic Publishers, Boston/Dordrecht/London .

CORNE, D.W; J. D. KNOWLES, and M. J. OATEs(2000), "The Pareto-Envelope based Selection Algorithm for Multiobjetive Optimization". Parallel Problem Solving from Nature-PPSN VI, **Springer Lecture Notes in Computer Science, Springer**, Berlin.

CORNE, D.W. N. R. JERRAM, J. D. KNOWLES and M. J. OATES (2001)"PESA-11: Region-based Selection in Evolutionary Multiobjetive Optimization". **Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001),** Morgan Kaufmann Publishers, Illinois

FINK, A. and V.B.STEFAN, (2001) Solving the Continuos Flow-Shop Scheduling Problem by Metaheuristics". **Technical University of Braunschweig, Germany.**

FONSECA C. m. AND P. J. FLEMING. (1993) Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization. In Stephanie Forrest, editor, **Proceedings of the Fifth International Conference on Genetic Algorithms**, 416-432, San Mateo, California , 1993. University of Illinois at Urbana –Champaign, Morgan Kauffman Publishers, Illinois.

GREINER, D. (2000) A summarized overview of evolutionary multiobjetive algorithms and new approaches" **CEANI, USA,.**

ISHIBUCHI, H., T. YOSHIDA, and T. MURATA, (2000) Balance between Genetic Search and Local Search in Memetic Algorithms for Multiobjective Permutation Flowshop Scheduling ". **Deparment of Industrial Engineering, Osaka Prefecture University, Japan; Deparment of Informatics,** Faculty of Informatics, Kansai University, Japan.

JAYARAM,K. (1998) Multiobjetive Production Scheduling, M Tech Thesis, **Deparment of industrial and Management Engineering, Indian Institute of Technology** Kanpur.

KNOWLES, J. D. and D. W. CORNE "Approximating the Nondominated Front using the Pareto Archived Evolution Strategy " . **Evolutionary Computation**, 8(2) pp149-172 Massachusetts Institute of Technology, 2000. http://iridia.ulb.ac.be/~jknowles/pubs.html

KOBAYASHI, S; I. ONO, and M. YAMAMURA, (1995), An Efficient Genetic Algorithm for Job Shop Scheduling Problems, **Proceedings of the Sixth International Conference on Genetic Algorithms University of Pittsburgh**, . 506-522.

KURI A and, J. GALAVIZ (2002), **Algoritmos Genéticos**, Instituto Politécnico Nacional, Universidad Nacional Autónoma de México, Fondo de Cultura Económica, México

PÉREZ, B y M.A. OSORIO,(2003) Análisis Comparativo de Heurísticas para Problemas de Calendarización de Trabajos con Transferencia Cero**" Memorias del Primer Congreso Mexicano de Computación Evolutiva, CI**MAT, 43-54.

RÍOS, R. Z. y J. F. BARD (1999) Heurísticas para Secuenciamiento de Tareas en Líneas de Flujo". **Reporte Técnico, Universidad Autónoma de Nuevo León**.

RUIZ, R. y C. MORATO (2003) Evaluación de Heurísticas para el problema del taller de flujo". **Reporte Técnico, Departamento de estadística e Investigación Operativa Aplicadas y Calidad. Universidad Politécnica de Valencia**, España/

.

TAGAMI, T AND T. KAWABE (1998) Genetic Algorithm with a Pareto Partitioning Method for Multiobjetive Flowshop Scheduling", **Technical Report, Kagawa Junior College, University of Tsukuba, Japan**, .

.