A New Contribution for Solving Dynamic Scheduling Problems Using Genetic Algorithms

[a] Ana Madureira, [a] Carlos Ramos, [b] Sílvio do Carmo Silva
[a] Instituto Superior de Engenharia do Porto, Dept. de Engenharia Informática Rua de São Tomé, 4200 Porto-Portugal
Phone: +351 - 228340500; Fax: +351 – 228321159 Email: {anamadur, csr}@dei.isep.ipp.pt
[b] Universidade do Minho, Dept. de Produção e Sistemas 4710-057, Braga – Portugal
Phone: +351 - 253604100 Fax:+351- 253604456 Email: scarmo@dps.uminho.pt

Abstract: Scheduling is an important element of manufacturing systems because it allows to improve the system performance and serves as an overall plan on which system activities are based. The main purpose of this paper is to explore the use of evolutionary computation techniques for solving real world optimisation problems. These classes of problems have additional difficulties for the traditional optimisation techniques.

This paper presents a simple and general framework based on Genetic Algorithms to solve dynamic Job-Shop scheduling problems. A new generation of initial individual and population is proposed. The proposed framework adapts the resolution of the deterministic problem to the non-deterministic one in which changes may occur continually. This takes into account dynamic occurrences in a manufacturing system and adapts the current population.

1. Introduction

Several techniques have been proposed to generate, for a given problem, a single schedule satisfying the problem constraints. However, when this schedule is released for processing, continual adapting is required to take into account the perturbations occurrences. At the beginning, the schedule is slightly modified and the performance is a little a bit affected. But when there are more significant perturbations, the performance of the current schedule is generally much worse than the initial one.

Real world scheduling applications operate in dynamic environments frequently subject to several kinds of random occurrences and perturbations, such as new job arrivals, machine breakdowns, employees sickness, jobs cancellation and due date and time processing changes, causing that the original schedule becomes unfeasible. In such class of problems cases, when the problem changes over the course of the optimisation, the purpose of the optimisation algorithm changes from finding an optimal solution to being able to continuously adapt the current solution through time. Since natural evolution is a process of continuous adaptation, it seems straightforward to consider Genetic Algorithms as appropriate candidates for dynamic optimisation problems.

To handle such class of problems many approaches of scheduling and rescheduling have been proposed in literature to take into account the several kinds of random occurrences [3][6][19].

The remaining sections are organised as follows: section 2 summarises related work. Section 3 provides a description of the scheduling problem under consideration. Section 4 summarises previous work on static scheduling problems. The proposed approach for dynamic scheduling is presented on section 4. Finally, the paper presents conclusions and some ideas for future work.

2. Literature review

A vast amount of literature about GA applications to the scheduling problem has been published in the last few decades. One of the earliest published work on the application of GA to scheduling is that by Davis [10]. Some GA studies for the static case of the JSSP scheduling are referred in [12], [13] and [14].

The Shifting Bottleneck procedure [1] is one of the most powerful procedures among approximation heuristics for the Job-Shop scheduling problem. The main idea is to solve for each machine a one machine-scheduling problem to optimality. As its name suggests, the Shifting Bottleneck heuristic always schedules bottleneck machines first.

Recently the scheduling problem in dynamic environments have been investigated by a number of authors in the evolutionary community, see for example [11] [15][22].

Proceedings of the 9th Bellman Continuum: International Workshop on Uncertain Systems and Soft Computing, Beijing, China, July 24-27, 2002, pp. 171-178.

Several attempts have been made to modify evolutionary algorithms, to tune them for optimisation in a changing environment. It was observed in all these studies, that the dynamic environment requires the evolutionary algorithm to maintain sufficient diversity for a continuous adaptation to the changes of the landscape. Although the interest in evolutionary algorithms for dynamic optimisation problems is growing and a number of authors have proposed an even greater number of new approaches, the field lacks a general understanding as to suitable benchmark problems, fair comparisons and measurement of algorithm quality.

The dynamism of the problem is usually treated considering the approach of a "Rolling Time Horizon" (see Raman and Talbot [21]), where new jobs can arrive all the time. Initially, the scheduling problem with all known jobs is solved. Then, whenever a new job arrives, the part of the solution consisting of operations already started before the changing occurrence is fixed and a new problem is created, consisting of the operations not yet processed and the operations from the new jobs. Thus, the dynamic problem is decomposed into a set of static sub-problem that can be solved independently by a standard GA. Some works that use that approach can be found in [3][4][6][7].

Branke [8] surveys the techniques that have been published in the literature to make evolutionary algorithms suitable for dynamic optimisation problems. The author grouped the different techniques into three categories: react on changes, where as soon as a change in the environment has been detected explicit actions are taken; maintaining diversity throughout the run, where convergence is avoided all the time and it is hoped that a spread-out population can adapt to modifications more easily; and the memory-based approaches, where the evolutionary approach is supplied with memory to be able to recall useful information from past generations.

3. Problem Definition

Almost all practical scheduling problems can be described in terms of the Job-Shop Scheduling Problem. Usually as restricted or relaxed versions of this classic combinatorial optimisation problem. The general Job-Shop Scheduling Problem (JSSP) of size n x m can generally be described as a decision-making process concerning about the allocation of a limited set of m resources over time to perform a set of n tasks or jobs.

In manufacturing systems, scheduling typically concerns allocating a set of machines to perform a set of jobs within a certain time period. Each job has a specified processing order through the machines, i.e. a job is composed of an ordered list of operations, which are characterised by the ma-chine required, and the processing time. Several constraints on jobs and machines can be defined:

- machines are always available and never break down;
- there are no precedence constraints among operations of the different jobs;
- the operations processing can not be interrupted and each machine can process only one job at a time;
- each job can be processed only on one machine at a time;
- setup times are independent of the schedules and are included in processing times
- processing times, due dates and technological constraints are deterministic and known in advance.

A schedule is an assignment of jobs over time in the respective machines. The objective is to find a schedule, which optimises some performance measure. For literature on this subject see for example [5], [9] and [20].

Real-world scheduling problems tend to be very different from the bulk of those dealt with by mathematical models developed by researchers in academy. In reality many scheduling problems are not so well defined as they are in the models. Most of the times, the environment is dynamic with new jobs arriving at unpredictable intervals, machines breaking down, jobs cancellation and due date and time processing changes happening frequently.

In addition to these and others disturbances, the following settings are considered in our approach for solving JSSP: the existence of different job release times, prior to which no processing of the job can be done; the existence of job due dates, a date by which the processing of the job is supposed to be finished, and different assembly levels for the jobs.

4. The proposed approach

The proposed approach[18] consists on a centralised GA-based scheduling system for each machine involved in the process. One advantage of this is that the solutions (schedules) planned for single machines guarantees some consistency of manufacturing. The original Job-Shop Scheduling Problem is decomposed into a series of deterministic Single Machine Scheduling Problems solved one at a time, consecutively. Each machine is denoted by $1|r_j|L_{max}$ in the Graham classification [9]. A set of n jobs (operations) has to be scheduled on a single machine without pre-emption. For each job, we know the release dates r_j, the processing times p_j, and the due dates d_j. the objective is to minimise the maximum lateness. A Genetic Algorithm solves each single machine problem. The obtained solutions are then incorporated into the main problem.

To solve the dynamic JSSP a predictive schedule is initially generated using the information available. As relevant disruptions occur in the system during the job execution, the predictive schedule is modified or revised taking them into account. The proposed approach considers additional constraints, which characterise real manufacturing systems: the existence of different job release dates; the existence of job due dates and different assembly levels for the jobs, and random perturbations can occur.

4.1 Notation

An operation O_{ijk} is described by the triplet (i, j, k), where i defines the machine where the operation is processed, j the job which belongs, and k the graph precedence operation level (level 1 correspond to initial operations, without precedents). In Table 1 the used notation will be presented:

Table 1 - Notation

m – Number of machines
$M = \{M_1, .., M_m\}$ – Set of machines
n – Number of jobs
$J=\{J_1, .., J_n\}$ – Set of jobs
T – Time horizon
k – Operation level defined on precedence graph
O_{ijk} – Operation from job j, to be processed on machine i with level k (defined on precedence graph)
$I\,O_{ijk}$ – Time interval for starting operation O_{ijk}
d_j – Due date of job j
t_j – Initial processing time of job j
r_j – Release time of job j
r_{ijk} – Release time of operation O_{ijk}
t_{ijk} – The earliest time at which O_{ijk} can start
T_{ijk} – The latest time at which O_{ijk} can start
p_{ijk} – Processing time of the operation O_{ijk}, from job j, level k on the machine i
C_{ijk} - Operation completion time from job j, level k on the machine I
L_j – Lateness ($L_j = C_j - d_j$)
T_j – Tardiness ($T_j = \max\{ L_j , 0 \}$)

4.2 GA-based Scheduling System for Deterministic Scheduling Problems

The Job-Shop Scheduling Problem (JSSP) is decomposed into a series of deterministic Single Machine Scheduling Problem SMSP, solved one at a time, consecutively. Each machine is considered as a SMSP $1|r_j|L_{max}$ problem with release dates and due dates and solved by a Genetic Algorithm, the obtained solutions are then incorporated into the main problem, using the following procedure:

Table 2 – Scheduling algorithm

1st PHASE	Finding a 1st job shop schedule based on single machine scheduling problems
Step 1	**Define completion time estimates (due dates) for eac operation of each job.**

Step 2	Define the interval between starting time estimates (release times) for all operations of each job.
Step 3	Define all SMSP $1\|r_j\|L_{max}$ based on information defined o Step1 and Step 2.
Step 4	Solve all SMSP $1\|r_j\|L_{max}$ with those release times and due dates using the genetic algorithm (GA).
Step 5	Integrate all the obtained near-optimal solutions into th main problem.
2nd PHASE	Check feasibility of the schedule and, if necessary apply th IMACM coordination mechanism
Step 6	**Verify if they constitute a feasible solution and terminate with a local optimum; If not, apply a repairin mechanism.**

The completion times C_{ijk} for each operation of a job are derived from job due dates and processing times by subtracting the operation processing time from the operation completion due time of the immediately succeeding job operation. This procedure begins with the last job operation and ends with the first.

$$C_{ijk-1} = C_{ijk} - p_{ijk} \qquad (1)$$

where k is the operation level on the precedence graph, i is the machine where the operation is processed and j the job which operation belongs.

The estimate of operation starting times interval $[t_{ijk},T_{ijk}]$ is also defined considering the job due date and the operation processing times. The earliest starting time t_{ijk} corresponds to the time from which the operation processing can be started. The latest starting time T_{ijk} correspond to the time at which the processing of the operation must be started in order to meet its estimate completion time (due date). This means that no further delay is allowed. When an operation has more than one precedent operation (multi-level structure) the $[t_{ijk},T_{ijk}]$ corresponds to the interval intersection from precedent operations correlated by the respective processing times.

The union of the several local optimal, one for each SMSP problem could not produce a feasible solution (processing overlapping operations belonging to the same job). This is due because it will be considered only technological precedence constraints of operations and job due dates, for defining completion and starting times. The machine occupation coordination with precedence relationships is not considered at this stage.

In order to obtain a feasible solution it is necessary to apply a coordination mechanism, which will be later described on section 4.3.

4.2.1 The Genetic Algorithm

In developing a genetic algorithm we must have in mind that its performance depends largely on the careful design and set-up of the algorithm components, mechanisms and parameters. This includes genetic encoding of solutions, initial population of solutions, evaluation of the fitness of solutions, genetic operators for the generation of new solutions and parameters such as population size, probabilities of crossover and mutation, replacement scheme and number of generations.

Details of the algorithm parameterisation are described as follows:

Solution Encoding

For the development of a genetic algorithm it is important to decide how to represent a solution and how this solution is to be manipulated during the search process. The success of GA is related with an appropriate representation of solutions. An overview of these subject and some applications are given in [10] [12].

The solutions are encoded by the direct representation [2], where the schedule is described as a sequence of operations, i.e., each gene is characterised by the triplet (i, j, k), where i defines the machine where the operation is processed, j the job that belongs, and k the graph precedence operation level (level 1 correspond to initial operations, without precedents). The gene position in a chromosome represents the operation position in a scheduling solution, defining, therefore, the operation processing order or priority. The number of genes in the chromosome represents the number of operations in a solution.

Initial Individual Generation

An initial solution (individual) is generated by a procedure, where the operations are sequenced in order of non-decreasing processing level (defined on precedence graph),

giving priority to operations that are processed earlier, as following exemplified:

(1,9,1)	(1,10,2)	(1,3,3)	(1,7,5)	(1,5,5)	1,2,7)	(1,8,8	(1,4,8)	(1,1,9)	(1,6,10)

Figure 1 – Initial individual with 10 genes

Thus, we expect to generate a good initial individual from which an initial population will be obtained.

Initial Population Generation

This is done using a generating mechanism that exchanges operations belonging to the same level defined on the precedence graph. From the initial chromosome presented on the figure 1, we can obtain the population of feasible individuals presented on Figure 2. Some computational tests have proven that this population generation mechanism is too restrictive, because generate smaller populations. In GA the quality of results tends to increase with population size. However, the time required for the search procedure also increases. Therefore, when using GA the size of the population can inhibit obtaining satisfactory solutions within an acceptable time frame.

Machine 1
(1,9,1)□(1,10,2)□(1,3,3)□(1,7,5)□(1,5,5)□(1,2,7)□(1,8,8)□(1,4,8)□(1,1,9)□(1,6,10)
(1,9,1)□(1,10,2)□(1,3,3)□(1,5,5)□(1,7,5)□(1,2,7)□(1,8,8)□(1,4,8)□(1,1,9)□(1,6,10)
(1,9,1)□(1,10,2)□(1,3,3)□(1,7,5)□(1,5,5)□(1,2,7)□(1,4,8)□(1,8,8)□(1,1,9)□(1,6,10)
(1,9,1)□(1,10,2)□(1,3,3)□(1,5,5)□(1,7,5)□(1,2,7)□(1,4,8)□(1,8,8)□(1,1,9)□(1,6,10)

Figure 2 – Initial population

So, it is proposed a new generation population mechanism, that unsure the operations precedence relationships and permit to introduce some diversity. This is done by exchanging genes (operations) belonging to adjacent levels and/or exchanging genes not apart more than one or two levels. Thus, we can obtain a greater population and consequently more diversity.

Selection

Individuals, i.e. solutions, are randomly selected from the population and combined to produce descendants.

Crossover

The order crossover operator [10] is used. The order crossover operator will be applied to M pairs of solutions randomly chosen, with M=N/2, where N is the size of the population.

Mutation

As mutation operator we use the inversion mechanism to prevent the lost of diversity. Thus, two points in a chromosome are randomly selected between which the order of genes, i.e. the processing order of the jobs, is reversed.

Replacement Scheme

The replacement of the less fit individuals of the current population by offspring is based on elitism [10]. Thus, the best individuals, i.e. solutions, will survive into the next generation. However, duplicated solutions may occur. To minimise this, the inversion operator is applied to all duplicated solutions.

Some of the parameter definition presented on this section is based on previous work [16][17].

4.3 Coordination Mechanism

The purpose of the coordination mechanism (Table 3) is to repair the schedule obtained up to the step 5 of the scheduling algorithm, described above. The repairing is done in order to obtain a feasible schedule through coordination of machine activity, having into account job operation precedence and all other problem constraints and, at the same time, keeping job allocation order in each machine as given by the schedule to be repaired. The objective is minimizing the maximum lateness.

Table 3 - Coordination Mechanism

Step 1	Start from initial operations, without precedents, whic correspond the level 1 on the sequence for processing on the machine. At this level, the starting and the completion time are the same, i. e., they are equal to those defined by th scheduling algorithm on the previous phase.
Step 2	At level 2, we will have all the operations which immediately precedents (defined on the precedence graph) and on the machine sequence has been already scheduled.

Step 3	The process will be repeated until all the operations have been scheduled.

Essential to the coordination mechanism is to establish the starting Ti and the completion times Tc for each operation.

These times are related being that the starting time Ti for each operation must be equal to the larger of the following two values:

- Completion time of the immediately precedent operation in the job;
- Completion time of the immediately precedent operation in the machine.

4.4 Computational Results

This section presents the results of the Genetic Algorithm on the resolution of a set of instances of the Job-Shop problem.

A software tool was developed to perform a computational study to analyse and evaluate the performance of a Genetic Algorithm on the resolution of JSSP for minimization of the *makespan* (C_{max}). The computational tests were carried out on a PC Pentium III 180Mhz, with the TS coded in C language.

LEKIN is an academic scheduling system developed at the Stern School of Business and Professor Michael Pinedo has directed this project.

For our experiments we consider some instances presented on LEKIN (version 2.4). The obtained results with our framework based on Tabu Search are compared with the results obtained with the LEKIN tool for EDD rule and Shifting Bottleneck method, some measures performance were computed: maximum tardiness T_{max}, total number of late jobs $\sum U_j$, total flow time $\sum C_j$, total tardiness $\sum T_j$, total weighted flow time $\sum w_j C_j$ and the total weighted tardiness $\sum w_j T_j$;

Genetic algorithms parameters definition has been made, in order to obtain identical computational efforts to permit the comparison in effectiveness (quality of solutions) and efficiency. As stopping criteria we use the 20 maximum number of generations.

With a simple implementation of the Genetic Algorithm and a small parameterisation effort it is possible to see, from Table 8, that the algorithm achieved good performance for most instances of the problem when comparing with the other methods.

It is important to refer that our framework is flexible in several ways. It is prepared to use other Local Search Metaheuristics and to evaluate any performance measure. It is not tailored, i.e., it is not developed specifically for the problem in consideration, such as Shifting Bottleneck [1].

More important is that our approach considers additional constraints when comparing the LEKIN tool, namely: the resolution of problems with different job assembly levels (parallel operations), and a given job may pass on a given machine more than once, i.e., a recirculation is permitted.

Table 8 – Computational results

Instance	Heuristic	Time(sec)	C_{max}	T_{max}	$\sum U_j$	$\sum C_j$	$\sum T_j$	$\sum w_j C_j$	$\sum w_j T_j$
6x4	EDD	1	16	2	3	55	4	166	17
	General SB(Cmax)	1	13	7	3	59	11	175	38
	Genetic Alg.	1	13	7	3	55	10	178	41
3x3	EDD	1	30	12	2	71	13	118	26
	General SB(Cmax)	1	28	10	2	74	16	124	32
	Genetic Alg.	1	34	10	3	75	17	116	24
10x10	EDD	1	122	32	10	830	195	1642	327
	General SB(Cmax)	1	94	40	9	817	186	1641	338
	Genetic Alg.	1	96	51	10	856	221	1658	343
18x5	EDD	1	1263	163	6	13807	540	13807	540
	General SB(Cmax)	30	1220	347	12	17026	1856	17026	1856
	Genetic Alg.	3	1523	780	17	23284	7554	23284	7554

4.5 GA-based Scheduling System for Non-Deterministic Scheduling Problems

In a dynamic environment frequent rescheduling of work is necessary due to variations on working conditions and requirements over time. This is due to many random events or disturbances as previously referred. These could be classified in two categories:

- **Partial events**, which imply changes in job (or operations) attributes, such as processing times, due dates and release times;

- **Total events**, which imply changes in population structure, such as new job arrivals and jobs cancellation.

While *partial* events only require a modification procedure to redefine job attributes and a re-evaluation of the fitness function of solutions, *total* events require a modification on chromosome structure and size, by inserting or deleting genes, as well as the re-evaluation the fitness function. Therefore, under a *total* event the modification of the solutions population is imperative. In this work, this is carried out by the mechanisms described in [16][17].

In this class of problems rescheduling from the beginning must be avoided, considering the processing times involved and the frequency of the dynamic perturbations.

As previously referred, a dynamic environment is characterized by several variations on working conditions and requirements over time. The GA-based scheduling system for dynamic JSSP under such an environment, here proposed, can be structured in three modules, namely the modules for pre-processing, scheduling and dynamic schedule adaptation.

4.5.1 Pre-processing Module

The pre-processing module deals with processing input information, namely problem definition and instantiation of algorithm components and parameters, such as, the initial individual and population generation mechanisms, size of population, genetic operators and respective probabilities, etc.

4.5.2 Scheduling Module

The scheduling module is concerned with the application of the GA-based scheduling algorithm for deterministic JSSP presented on section 5.2, considering that all release dates, processing times and due dates are known in advance.

Whenever new events occur, deterministic problems are generated by the dynamic adaptation module and then solved by the scheduling module.

4.5.3 Dynamic Adaptation

The occurrence of a *partial event* requires a modification procedure to redefine job attributes and a re-evaluation of the fitness function. A job due date modification also requires the re-calculation of the operation starting and completion times of all respective operations. While operation processing times modification only requires re-calculation of the operation starting and completion times of the succeeding operations.

A new job arrival requires definition of the correspondent operation starting and completion times and a regenerating mechanism [17] to integrate all operations on the respective single machine problem. While a job cancellation requires a regenerating mechanism [17] to eliminate the operations from respective single machines problems. After the insertion or deletion of genes is carried out, population regenerating is done by updating the size of the population and ensuring a structure identical to the existing one [16][17].

Then, scheduling module can apply the search process with the new regenerated population.

Integration mechanism

When a new job arrives, an integration procedure will be needed. For that, the integration mechanism consists basically in analysing the job precedence graph that represents the ordered allocation of machines for each job operation, and integrates each operation into the respective single machine problem. The following integration mechanisms [16][17] could be implemented for each operation: randomly select one position to insert the new operation into all chromosomes or use some intelligent mechanism to insert this operation in the schedules, (based on job priority, for example).

Eliminating mechanism

When a job is cancelled, an eliminating mechanism [16] [17] must be implemented so the correspondent gene will be deleted in all chromosomes.

Regeneration mechanisms

After integration/elimination of operations is carried out (which consists in inserting/deleting genes in all chromosomes), population regenerating is done by updating the size of the population and ensuring a structure identical to the existing one. Thus, either some further chromosomes have to be generated through some procedure, i.e. REDD (Randomised Earliest Due Date) rule [20], or some existing chromosomes have to be deleted. In this case either the choice could be random or fall on the less adapted chromosomes. After this is done, the scheduling module can apply the search process with the new regenerating population.

5. Conclusions

The paper presents a scheduling system, based on Genetic Algorithms to solve the dynamic version of the JSSP, where the products (jobs) to be processed have due dates. The approach adapts the resolution of the static problem to the dynamic one in which changes may occur continually. In order to repair the unfeasible schedule, a inter manufacturing activity coordination mechanism, called IMACM, is described for ensuring that work at each machine does not violate constraints and therefore ensuring a feasible solution. The objective is to coordinate the machine working times with job operations precedence relationships.

A population regenerating mechanism is put forward. This adapts the population to a new population, which increases or decreases according to new job arrivals or cancellations.

More research is needed in testing the performance of the proposed mechanisms under dynamic Job-Shop environments subject to several random perturbations.

References

[1]. Adams, Joseph, Balas, Egon e Zawack, Daniel, The Shiftin Bottleneck Procedure for Job Shop Scheduling, *Managemen Science*, Vol. 34, nº 3 Março, USA, 1988.

[2]. Bagchi, Sugato et al., Exploring Problem-Specific Recombinatio Operators for Job-Shop Scheduling, *Proc. 4th Int. Conf. O Genetic Algorithms*, Morgan Kaufman, 10-17, 1991.

[3]. Bierwirth, C. and Matfeld, D.C., Production scheduling an rescheduling with genetic algorithms, *Evolutionary Computation* 7, 1-17, 1999.

[4]. Bierwirth, C., Kopfer, H., Matfeld, D. C. and Rixen, I., Geneti algorithm based scheduling in a dynamic manufacturing environment, *Proc. of the second conference on evolutionar computation*, IEEE Press, New York, 439-433, 1995.

[5]. Blazewicz, J., Ecker, K.H., Pesch, E., Smith, G. and Weglarz, J. *Scheduling Computer and Manufacturing Processes*, Springer, 2n edition, New York, 2001.

[6]. Branke, J. and Mattfeld, D., Anticipation in Dynami Optimization: The Scheduling Case, *Parallel Problem Solvin from Nature*, Schoenauer et al. (eds.), Springer, 253-262, 2000

[7]. Branke, J., Efficient Evolutionary Algorithms for Searching Robus Solutions, *Fourth Intl. Conf. on Adaptive Computing in Design an Manufacture*, Springer, 275-286, 2000.

[8]. Branke, J., Evolutionary Approaches to Dynamic Optimizatio Problems – A Survey, *GECCO Workshop on Evolutionar Algorithms for Dynamic Optimization Problems*, A. Wu (ed.), pp 134-137, 1999.

[9]. Brucker, P., *Scheduling Algorithms*, Springer, 3rd edition, New York, 2001.

[10]. Davis, Lawrence, *Handbook of Genetic Algorithms*, Van Nostran Reinhold, New York, 1991.

[11]. Fang, Hsiao-Lan, Ross ,Peter and Corne, D., A promising geneti algorithm approach to Job-Shop scheduling, rescheduling and ope shop scheduling problems, *Proc. of the 5th Int. Conf. on Geneti Algorithms*, Morgan Kaufmann, 375-382, 1993.

[12]. Jain, A. S. and Meeran, S., Deterministic Job Shop scheduling past, present and future, *European Journal of Operationa Research*, nº113, 390-434, 1999.

[13]. Kumar, N. S. Hemant and Srinavasan, G., A genetic algorithm fo Job Shop scheduling-a case study, *Computers Industry*, 155-160 1996.

[14]. Lee, C.Y., Piramuthu, S. and Tsai, Y.K., Job Shop Schedulin With a Genetic Algorithm and Machine Learning, *Int. J. Prod Res.*, vol.35, nº4, 1171-1191, 1997.

[15]. Lin, Shyh-Chang, Goodman, E. D. and Punch , William F., A genetic algorithm approach to dynamic scheduling Job Sho Problem, *Proc. of the 7th Int. Conf. on Genetic Algorithms*, Morga Kaufmann, 481-489, 1997.

[16]. Madureira, Ana M., Ramos, Carlos and Silva, Sílvio C., Genetic Approach to Dynamic Scheduling for Total Weighte Tardiness Problem, *PLANSIG'99 (18th Workshop of the U Planning and Scheduling Special Interest Group)*, Dezembro 1999.

[17]. Madureira, Ana M., Ramos, Carlos and Silva, Sílvio C., A Geneti Algorithm for The Dynamic Single Machine Scheduling Problem, *4th IEEE/IFIP Int. Conf. on Information Technology for Balance Automation Systems in Production and Transportation* , 2000.

[18]. Madureira, Ana M., A Genetic Approach for Dynamic Job-Sho Scheduling Problems, *4th MetaHeuristics International Conferenc (MIC'2001)*, Porto (Portugal), 2001

[19]. Mehta, S.V., and Uzsoy, R., Predictable scheduling of a single machine subject to breakdows, *International Journal of Compute Integrating Manufacturing*, Vol 12, nº1, 15-38. 1999.

[20]. Morton, E. Thomas, and Pentico, David W., *Heuristic Schedulin Systems*, John Wiley & Sons, 1993.

[21]. Raman, N. and Talbot, F.B., The Job Shop Tardiness Problem: a decomposition approach, *European Journal of Operation Research*, nº69, 187-199, 1993.

[22]. *Report from EVONET Working Group on Evolutionar Approaches to Timetabling and Scheduling*, The state of the art i evolutionary Approaches to timetabling and scheduling.