A New Contribution for Solving Dynamic Scheduling Problems Using a Tabu Search

[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
Email: scarmo@dps.uminho.pt

Abstract: Many real-world optimisation problems are eventually dynamic. New jobs are to be added to the schedule, the quality of the raw material may be changing, new orders have to be included into the problem etc. In such 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 track the movement of the optimum through time.

This paper starts by presenting a new scheduling method based on Tabu Search for the resolution of the dynamic Job-Shop Scheduling Problem, which considers job release times, job due dates and different assembly levels (parallel operations). This framework is based on a decomposition of the Job-Shop Scheduling Problem into a series of deterministic Single Machine Scheduling Problem (SMSP) and on a Tabu Search Algorithm, which solves each SMSP whose solutions are, then, integrated. An inter-machine activity coordination mechanism is described. Finally, the used approach 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 neighbourhood to a new regenerated neighbourhood.

1. Introduction

Scheduling Problems can be seen as a decision making process, which decides when an operation should start and which resources should be used for each operation. A significant variety of constraints such as operation processing times, release and due dates, precedence constraints and resource availability can affect the scheduling decision.

Several attempts have been made to modify algorithms, to tune them for optimisation in a changing environment. It was observed in all these studies, that the dynamic environment requires an algorithm to maintain sufficient diversity for a continuous adaptation to the changes of the landscape. Although the interest in optimisation 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.

There are many techniques that have been reported in the literature that have produced excellent results when applied to scheduling problems. For example the use of meta-heuristic techniques (such as tabu search and simulated annealing) and evolutionary techniques (such as genetic and memetic algorithms). One emerging research area is to develop heuristics that operate at a higher level of generality than current technology can support. This will involve advances in heuristics, meta-heuristics and an emerging technique tentatively called a hyper-heuristic. Another interesting idea is to use an "adaptive" heuristic. This uses the idea that a scheduling problem can be solved using a heuristic but, for many reasons, this heuristic can lead to solutions which, although, better than previous efforts, can be even better if the heuristic is allowed to adapt as the search progresses

Local Search Meta-heuristics form a class of powerful and practical solution techniques for tackling complex, large-scale combinatorial problems. During the past decade, there has been an increasing interest in local search optimisation methods, in which a given solution is iteratively changed by making local modifications. The most well known Local Search Meta-heuristics are Tabu search and Simulated Annealing.

Most of the heuristic techniques or approximation methods proposed for the Job-Shop problem are tailored techniques, i.e., developed specifically for the problem in consideration, some examples are the priority rules and the Shifting Bottleneck procedure.

Many real world applications operate in dynamic environments frequently subject to several kinds of random occurrences and perturbations, such as new job arrivals,

Proceedings of the International Workshop on Heuristics, Beijing, China, July 24-27, 2002, pp. 69-78.

machine breakdowns, employees sickness, jobs cancellation and due date and time processing changes, causing that the original schedule becomes unfeasible. In such cases, when the problem changes over the course of the process, the purpose of the optimisation approach is no more to find an optimal solution but to be able to continuously adapt the solution.

The problem of finding good solutions to scheduling problems is very important to real manufacturing systems because the production rate and production costs are very dependent on the schedules used for controlling the flow of work through the system. Most research in scheduling focuses on optimisation of static and dynamic deterministic problems where all problem data are known before scheduling starts. However many real world optimisation problems are non-deterministic, in which changes may occur continually.

Dynamic changes of a problem come from new user requirements and the evolution of the external environment. These changes are received by the system as user events, fault alarms, sensor information updates, etc. For example, in manufacturing systems, the user events can be a client request for a new task, a technician modifying his time for doing a task, or a manager modifying the current schedule; and external events can be a tool breakdown, deliveries get delayed and workers become sick. In a more general view, dynamic problem changes can be seen as a set of constraint addings and removals. More specifically, for scheduling problems, this can be expressed as the addition or deletion of a set of tasks, or the restriction or relaxation of the time windows on tasks.

Due to their dynamic nature, real scheduling problems have an additional complexity in relation to static ones. In many situations these problems, even for apparently simple situations, are hard to solve, i.e. the time required to compute an optimal solution increases exponentially with the size of the problem[17].

Recently the scheduling problem in dynamic environments have been investigated by a number of authors in the evolutionary community, see for example [9], [14], [15] e [16].

If all jobs are known and ready to process before processing starts, the scheduling problem is called *static,* while if job release times are not fixed at a single point in time, i.e. jobs arrive to the system at different times and its parameters can change, the problem is called *dynamic.* Dynamic scheduling problems can also be classified as *deterministic,* when release times and all other parameters are known and fixed, and *non-deterministic,* when jobs can arrive over time and some or all parameters are uncertain.

The purpose of this paper is to describe an approach based on Tabu Search for solving a class of real world scheduling problems, where the products (jobs) to be processed have due dates, release times, different assembly levels (parallel operations) and where random disturbances may occur over time. As the approach starts by solving a surrogated set of single machine scheduling problems their solutions are repaired, through a proposed inter-machine activity coordination mechanism, for obtaining a feasible solution to the original problem.

Finally, the 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 neighbourhood.

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

2. Problem Definition

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 $m=\{0, 1,, ...,m\}$ resources over time to perform a set of $n = \{0, 1, ..., n\}$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 machine 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 p_j, due dates d_j 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 optimizes some performance measure.

Several Branch-and Bound methods have been developed for solving Job-shop scheduling to optimality. These class of methods require a large amount of computation time which increase with problem size. For literature on this subject see for example [4], [5], [6], [7], [8] and [18].

Most of the approximation methods proposed for the JSSP are oriented methods, i.e. developed specifically for the problem in consideration. Some examples of this class of methods are the priority rules [4][7][8] and the Shifting Bottleneck proposed by Adams, Balas and Zawack [2].

Local Search Meta-heuristics are a class of approximation algorithms that are considered to be efficient tools for solving hard combinatorial optimisation problems. A *MetaHeuristic* is an iterative generation process which guides a subordinate heuristic by combining intelligently different concepts for exploring and exploiting the search space using learning strategies to structure information (initial solution and the neighbourhood generation mechanisms) in order to find efficiently near-optimal solutions. There has been an increasing interest on methods based on Local Search techniques (such as Tabu Search, Simulated Annealing or Genetic Algorithms), in which a given solution is iteratively improved by local modifications. For a review on this subject see for example [1],[3],[9] and [10].

Recently the scheduling problem in dynamic environments have been investigated by some of authors using Metaheuristics, see for example, [9] and [20].

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 r_j, 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 (parallel operations).

3. Previous Work on Static Scheduling

A vast amount of literature about Tabu Search applications to the scheduling problem has been published in the last few decades.

In a previous work [11] the adequacy and efficiency of the Tabu Search (TS) with other Local Search MetaHeuristics, for the static Single Machine Scheduling Problem (SMSP) was studied. In order to define the attributes that characterise a tabu move, and at the same time the length of the tabu list, some computational tests have been performed on instances of the Weighted Tardiness problem using Tabu Search (TS) method. Due to the remarkable degree of randomisation of the algorithm it has become difficult to take absolute and definitive conclusions about the values of those parameters, considering that the obtained computational results depends from the random component introduced either on generation of the initial solution (the solution is constructed by REDD rule) or on the neighbourhood generation (the neighbourhood elements are randomly chosen). It is possible to conclude that better solutions were generally obtained with a tabu list that stores the pairs of jobs involved on exchanging of positions (on the neighbourhood generation process), and there is some advantage on using tabu lists with length 7 or near to 7 [11].

Finally, the performance of the RNDLS and Tabu Search algorithms has been compared on resolution of the instances above mentioned for the minimisation Weighted Tardiness problem. In order to analyse the obtained results, performance measures were computed: the best value and the deviation from the best value to the optimal. The optimal values from instances of problems with 50 e 60 jobs (that were randomly generated using a software tool described in [11] are not known, so in these cases only the best value is considered. The definition of parameters of both algorithms has been made, in order to obtain identical computational efforts to permit the comparison in effectiveness (quality of solutions).

Both algorithms obtained a good performance on resolution of the mentioned instances, however, the Tabu Search procedure has presented better solutions in most of the instances, and in some cases, the Tabu Search found the optimal solution. The obtained results show that these TS perform well for the studied cases, being possible to find good solutions in a short period of time, i.e. a few minutes of CPU time.

4. The proposed approach

The proposed approach consists on a centralised TS-based scheduling system for the machines 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 (JSSP) is decomposed into a series of deterministic Single Machine Scheduling Problems solved one at a time, consecutively. Each machine is considered as $1|r_j|C_{max}$ problem with release dates and due dates and solved by a

Tabu Search Algorithm. The obtained solutions are then incorporated into the main problem[14].

To solve the dynamic Job-Shop Scheduling Problem (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 consider 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 (parallel operations), 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
n – Number of jobs
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 Scheduling System for Deterministic Scheduling Problems based on Tabu Search

The problem $J|r_j|C_{max}$ is decomposed into a series of deterministic SMSP, solved one at a time, consecutively. Each machine is considered as problem with release dates and due dates and solved by a Tabu Search, the obtained solutions are then incorporated into the main problem, using the procedure described on table 2.

Table 2 – Scheduling method for deterministic JSSP

1st PHASE	**Finding a first job shop schedule based on single machine scheduling problems**
Step 1	Define completion time estimates (due dates) for each operation of each job.
Step 2	Define the interval of the starting time estimates (release times) for all operations of each job.
Step 3	Define all SMSP $1\|r_j\|C_{max}$ based on information defined on Step1 and Step 2.
Step 4	Solve all SMSP $1\|r_j\|C_{max}$ with those release times and due dates using a Tabu Search.
Step 5	Integrate all the obtained near-optimal solutions into the main problem.
2nd PHASE	**Check feasibility of the schedule and, if necessary apply the coordination mechanism**
Step 6	Verify if they constitute a feasible solution and terminate with a local optimum; If not, apply a repairing 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:

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

The proposed approach (Table 2) is implemented in a two-phase procedure: in a first phase we find a first Job-Shop schedule based on single machine scheduling problems then in a second phase the schedule feasibility is checked and applied a coordination mechanism if necessary.

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.

At this stage, it will be considered only technological precedence constraints of operations and job due dates, for defining completion and starting times.

4.2.1 Tabu Search

Frequently classical optimization methods are not efficient enough for the resolution of Job-Shop Scheduling problems. In most cases they are good for solving only some specific and small size ones. The interest of new approaches, namely *MetaHeuristics* such as Tabu Search, Simulated Annealing, Genetic Algorithms and Neural Networks, based on *local search*, is that they lead, in general, to satisfactory solutions in an effective and efficient way, i.e. short computing time and small implementation effort.

Tabu Search is a Local Search procedure introduced by Glover and designed for escaping from local optimum. It is based on the general tenets of intelligent problem solving. The process in which the Tabu Search algorithm seeks to transcend local optimality is based on an aggressive evaluation that chooses the best available solution at each iteration even when this solution may result in a degradation of objective function value. The recent moves are penalised [19].

In developing a Tabu Search 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 representation of solutions, initial generation of solutions, evaluation of the fitness of solutions, such as neighbourhood size, tabu list length, tabu list attributes and stop criteria.

Details of the algorithm parameterisation are described as follows:

Solution Representation

The solutions are encoded by the natural (indirect) representation, where the schedule is described as a sequence of operations, i.e., each position represents an operation index. The position in the sequence represents the operation position in a scheduling solution, defining, therefore, the operation processing order or priority. The number of positions in the sequence represents the number of operations in a solution.

Initial Neighbourhood Generation

An initial solution 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. Thus, we expect to generate a good initial solution from which an initial neighbourhood will be obtained. This is done using a neighbourhood generating mechanism, which exchanges jobs not apart more than *af* positions (Madureira 1999). Here, the initial neighbourhood is obtained by a neighbourhood search procedure named "*Exchange jobs not apart more than af positions*" mechanism. The distance *af* is dependent on problem size, i.e. number of jobs. Let $af = p*n$, where p is a percentage and n the number of jobs. This generating mechanism produces a neighbourhood size of N different solutions, with:

$$N = (n\text{-}1) + \sum_{i=n-af}^{n-2} i \qquad (2)$$

Let us consider a initial sequence composed by n jobs and the exchanging of the *first* job with the next *af* jobs, then the *second* with the next *af* jobs, and so on, until the last solution has been obtained by exchanging the job *n-1* with job *n*.

Some of the parameter definition presented on this section is based on previous work (Madureira 1999).

Objective Function Evaluation

The quality of a solution is measured by means of the makespan C_{max} function, which attends to consider release dates and due dates for the jobs. Although other criteria could be calculated, such as: 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 $\sum w_j T_j$.

Tabu lists attributes and length

	Tabu Search	1	13	7	3	55	10	178	41
	EDD	1	30	12	2	71	13	118	26
3x3	General SB(Cmax)	1	28	10	2	74	16	124	32
	Tabu Search	1	34	10	3	75	17	116	24
	EDD	1	122	32	10	830	195	1642	327
10x10	General SB(Cmax)	1	94	40	9	817	186	1641	338
	Tabu Search	1	96	51	10	856	221	1658	343
	EDD	1	1263	163	6	13807	540	13807	540
18x5	General SB(Cmax)	30	1220	347	12	17026	1856	17026	1856
	Tabu Search	3	1523	780	17	23284	7554	23284	7554

Parameters Tabu Search 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 iterations.

With a simple implementation of the Tabu Search 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 [2].

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.

4.5 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 [14]:

- **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 neighbourhood 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 neighbourhood is imperative.

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

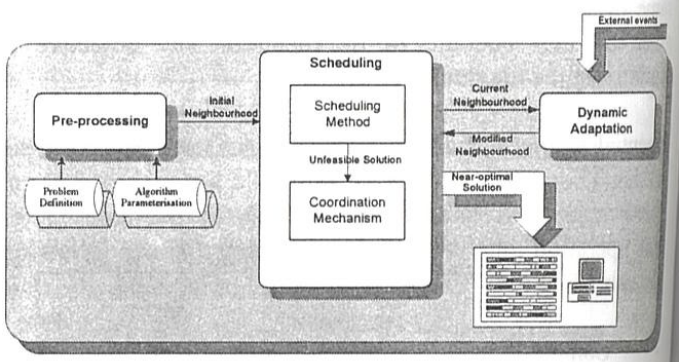

Figure 3 - Scheduling System for Dynamic Scheduling Problems

As previously referred, a dynamic environment is characterized by several variations on working conditions and requirements over time. The TS-based scheduling system for dynamic JSSP under such an environment, here proposed, can also be structured in three modules as in the case of SMSP, namely the modules for pre-processing, scheduling and dynamic schedule adaptation (Figure3):

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 solution, neighbourhood generation mechanisms, size of neighbourhood, tabu list attributes and respective length, etc.

4.5.2 Scheduling Module

The scheduling module is concerned with the application of the TS-based scheduling algorithm for deterministic JSSP presented on section 4.2, considering that all release dates, processing times and due dates are known in advance.

4.5.3 Dynamic Adaptation Module

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

Partial events occurence

The occurrence of a job due date, job release date or a operation processing times changing requires a modification procedure to redefine job attributes and a re-evaluation of the objective function.

A job due date modification also requires the re-calculation of the operation starting and completion times of all respective operations. A job release date changing implies the re-calculation of the operation starting 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. Those procedures must be executed before the fitness re-evaluation process.

Total events occurrence

Total events occurrence require a modification on chromosome structure and size, by inserting or deleting operations, as well as the re-evaluation the fitness function. Therefore, under a total event an regeneration of the neighbourhood is necessary as well, in order to insert or delete new solutions.

A new job arrival requires the definition of the correspondent operation starting and completion times and an **integration mechanism** to insert all operations belonging to the new job on the respective single machine problem. For that, the integration mechanism consists basically in analysing the job precedence graph that represents the ordered allocation of machines to each job operation, and integrates each operation into the respective single machine problem. The following integration mechanisms could be implemented for each operation: randomly select one position to insert the new operation into all current solutions (sequences) or use some intelligent mechanism to insert this operation in the schedules, based on high job priority, for example.

While a job cancellation requires an **elimination mechanism** to delete the operations from respective single machines problems, i.e., the correspondent operations sequence position will be deleted in all solutions.

After integration/elimination of operations is carried out (which consists on inserting/deleting positions into all the sequences), **neighbourhood regenerating** is done by updating the size of the neighbourhood and ensuring a structure identical to the existing one. Thus, either some further solutions have to be generated through some procedure, such as REED rule [12], or some existing solutions have to be deleted. In this case either the choice could be random or fall on the worst-case solutions. After this is done, the scheduling module can apply the search process with the new regenerating neighbourhood.

5. Conclusions and Further Work

More than developing algorithms with unquestionable practice utility, the main purpose of this work was to present a simple framework to solve difficult problems using the potentiality of the Metaheuristic approaches.

The paper presents a new scheduling system, based on Tabu Search to solve the dynamic version of the JSSP, where the products (jobs) to be processed have due dates, release dates and different assembly levels (parallel operations). 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 manufacturing activity coordination mechanism, is presented 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 neighbourhood regenerating mechanism is put forward. This adapts the current solution to a new solution, 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. However we have found some difficulties in to find test problems with specific constraints: where the products (jobs) to be processed have due dates, release dates and different assembly levels (parallel operations).

References

[1] Aarts, Emile e Lenstra, Jan K., Local Search in combinatorial optimization, Wiley- Interscience Publication, 1997.

[2] Adams, Joseph, Balas, Egon e Zawack, Daniel, The Shifting Bottleneck Procedure for Job Shop Scheduling, Management Science, Vol. 34, nº 3 Março, USA, 1988.

[3] Artiba, A. e S. E. Elmaghraby, The planning and scheduling of production systems, Chapman &Hall, 1997.

[4] Baker, K.R., Introduction to sequencing and scheduling, Wiley, New York, 1974.

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

[6] Brucker, Peter, "Scheduling Algorithms", Springer, 3rd edition, New York, 2001.

[7] Conway, R. W., Maxwell, W. L., e Miller, L. W., Theory of scheduling, Addison-Wesley Publishing Company, 1967.

[8] French, S., Sequencing and Scheduling: An introduction to the Mathematics of the Job Shop, Ellis Horwood, Chichester, 1982.

[9] Jain, A. S. and Meeran S., Deterministic Job Shop scheduling: past, present and future, European Journal of Operational Research, nº 113, 390-434, 1999.

[10] Lourenço, Helena R., Job Shop scheduling problem: computational study on local search and large-step Optimization methods, European Journal of Operational Research, 347-364, 1995.

[11] Madureira, Ana M., Meta-heuristics for Single-Machine Scheduling Problems, 4th IMS-WG Workshop, Nancy (France), 1998.

[12] Madureira, Ana M., Meta-heuristics for the Single-Machine Scheduling Total Weighted Tardiness Problem, Proc. IEEE Int. Symposium Assembly and Task Planning (ISATP'1999), Portugal, 1999.

[13] Madureira, Ana M., Carlos Ramos e Sílvio C. Silva, A Genetic Algorithm for The Dynamic Single Machine Scheduling Problem, 4th IEEE/IFIP Intl. Conf. on Information Technology for Balanced Automation Systems in Production and Transportation, Berlim (Germany), 2000.

[14] Madureira, Ana M., Carlos Ramos e Sílvio do Carmo Silva., A Genetic Approach for Dynamic Job-Shop Scheduling Problems, 4th MetaHeuristics International Conference (MIC'2001), Porto, 2001.

[15] Madureira, Ana M., Carlos Ramos e Sílvio do Carmo Silva,, A Coordination Mechanism for Real World Scheduling Problems Using Genetic Algorithms, to be presented on 2002 Congress on Evolutionary Computation (CEC'2002), Honolulu- Hawaii (EUA),.2002.

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

[17] Morton, E. Thomas, and David W. Pentico, 1993, Heuristic Scheduling Systems, John Wiley & Sons, 1993.

[18] Pinedo, M., Scheduling – Theory, algorithms and systems, The MIT Press, 1995.

[19] Pirlot, M., General Local Search heuristics in combinatorial optimization: a tutorial, JORBEL, vol. 32, Bruxelas, 7-68, 1992.

[20] Report from EVONET Working Group on Evolutionary Approaches to timetabling and Scheduling, The state of the art in evolutionary Approaches to timetabling and scheduling.