Using Genetic Algorithms for Dynamic Scheduling

Ana Madureira [*] *Carlos Ramos* [*] *Sílvio do Carmo Silva* [†]

anamadur@dei.isep.ipp.pt,, csr@dei.isep.ipp.pt, scarmo@dps.uminho.pt

[1]*Institute of Engineering Polytechnic of Porto, GECAD - Knowledge Engineering and Decision Support Research Group, Dept. of Computer Science*
Rua de São Tomé, 4200 Porto-Portugal
Phone: +351 228340500 Fax: +351 228321159

[2] *Minho University, Dept. of Production and Systems*
4710-057, Braga -– Portugal, Phone: +351 253604745

Abstract

In most practical environments, scheduling is an ongoing reactive process where the presence of real time information continually forces reconsideration and revision of pre-established schedules. Scheduling algorithms that achieve good or near optimal solutions and can efficiently adapt them to perturbations are, in most cases, preferable to those that achieve optimal ones but that cannot implement such an adaptation. This reality, motivated us to concentrate on tools, which could deal with such dynamic, disturbed scheduling problems, both for single and multi-machine manufacturing settings, even though, due to the complexity of these problems, optimal solutions may not be possible to find. We decided to address the problem drawing upon the potential of Genetic Algorithms to deal with such complex situations.

We decided to address the problem drawing upon the potential of Genetic Algorithms to deal with such complex situations. Since in a sense natural evolution is a process of continuous adaptation, it seems appropriate to consider Genetic Algorithms as good candidates for dynamic scheduling problems.

This paper is concerned with vertical oriented detailed scheduling of Extended Job-Shop on dynamic environments. It addresses the scheduling of tasks, either simple or complex products, comprehending the parts fabrication and their multistage assembly into complex products.

Key Words: Dynamic Scheduling, Population Dynamic Adaptation, Regenerating Mechanism, Genetic Algorithms.

1. INTRODUCTION

Research on the theory and practice of scheduling has been pursued for many years. Theoretical scheduling problems concerned with searching for optimal schedules subject to a limited number of constraints have adopted a variety of techniques including branch-and-bound and dynamic programming. From the point of view of combinatorial optimization the question of how to sequence and schedule jobs in a dynamic environment looks rather complex and is known to be NP-hard. For literature on this subject, see for example, Baker (1974), French (1982), Blazewicz et al. (2001), Pinedo (2001) and Brucker (2001).

In generic terms, the scheduling process can be defined as the assignment of time-constrained jobs to time-constrained resources within a pre-defined time framework, which represents the complete time horizon of the schedule. An admissible schedule will have to satisfy a set of hard and soft constraints imposed on jobs and resources. So, a scheduling problems can be seen as a decision making process for operations starting and resources to be used. A variety of characteristics and constraints related with jobs and production system, such as operation processing times, release and due dates, precedence constraints and resource availability, can affect scheduling decisions.

If all jobs are known before processing starts a scheduling problem is said to be **static**, while, to classify a problem as **dynamic** it is sufficient that job release times are not fixed at a single point in time, i.e. jobs arrive to the system at different times. Scheduling problems can also be classified as either **deterministic**, when processing times and all other parameters are known and fixed, or as **non-deterministic**, when some or all parameters are uncertain (French, 1982).

Most of the known work on scheduling deals with optimisation of scheduling problems in static environments, whereas, due to several sorts of random occurrences and perturbations, real world scheduling problems are usually of dynamic nature. Due to their dynamic nature, real scheduling problems have additional complexity in relation to static ones. However, in many situations, both static and dynamic problems, even for apparently simple cases, are hard to

solve, i.e. the time required for computing an optimal solution increases exponentially with the size of the problem (Blazewicz, 2001). From the point of view of combinatorial optimisation the question of how to sequence and schedule jobs in a dynamic environment looks rather complex and is known to be NP-hard to almost every state (Parunak, 1992).

In this work, a job is defined as a manufacturing order for a product, which may be simple, i.e. having a set of operations to be carried out in a given sequence, called **simple products**. The products may also be **complex products**, having a set of operations, carried out in a given sequence, each one made of several different parts.

We adopt a strategy to scheduling known as **Resource-Oriented or Vertical Scheduling** (Vollmann, 1996), where the objective is to sequence all the operations on each machine, in order to optimise some objective function. Resource-oriented scheduling should be applied whenever there are limited resources available and the competition for these resources among the operations is keen. In effect, delays are liable to occur in such cases, as operations must wait until common resources become available. To the extent that resources are limited and demand for the resource is high, this waiting may be considerable. In turn, the congestion associated with these waits represents increased costs, poor productivity and, in the end, job delays. Schedules made without consideration for such bottlenecks can be completely unrealistic.

Since Davis (Davis, 1991) proposed the first Genetic Algorithm (GA) to address scheduling problems in 1985, GA have been widely used in manufacturing scheduling applications. However, most of the work deals with the scheduling problems in the static environment. The approach proposed in this paper aims at dealing with dynamic problems, both deterministic and non-deterministic.

Many real world problems happen in dynamic environments frequently subject to several kinds of random occurrences and perturbations. These include new job arrivals, machine breakdowns, employees' sickness, jobs cancellation and due date and time processing changes. Such perturbations soon cause original schedules becoming poor and sometimes unfeasible. For such disturbed environments, the goal is no longer to find a single optimum to a scheduling problem, but rather to continuously adapt the solution to the changing environment. When changes in the environment occur, rescheduling is needed. Therefore, the existence of good or near-optimal schedules that are easily modifiable will be, in most situations, preferable to optimal ones that are not. So, the purpose of scheduling applications should be not so much to find optimal solutions, which we know that soon deteriorate, but instead, to be able to efficient and effectively adapt, on a continuous basis, existing solutions according to disturbances, keeping performance levels high. Since, in a sense, natural evolution is a process of continuous adaptation, it seems straightforward to consider Genetic Algorithms as appropriate candidates for dynamic scheduling problems.

The paper is structured as follows: section 2 provides a description of the Extended Job-Shop Scheduling Problems (EJSSP) to be solved. Section 3 presents a review of some genetic algorithm essential concepts and some literature review with emphasis on Genetic Algorithms applications in dynamic environments. In section 4 we describe a method based on GA for the resolution of the dynamic deterministic version of the same problem. Our approach for solving Dynamic non-deterministic Extended Job-Shop Scheduling Problems is presented in section 5. Finally, some conclusions are drawn and some ideas for future work are presented.

2. SCHEDULING PROBLEMS

2.1 Real-world and academic scheduling problems

Most real-world multi-operation scheduling problems can be described as dynamic and extended versions of the classic Job-Shop scheduling combinatorial optimization problem (Pinedo, 2001). In a Job-Shop each job has a specified processing order through the machines, i.e. a job is composed of an ordered set of operations each of which is to be processed, during a given time, i.e. the operation processing time, in a machine of the system. In classic JSSP several constraints on jobs and machines are considered: 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 a machine at a time; setup times are independent of the schedules and are included in processing times; technological constraints are deterministic and known in advance.

In practice, many scheduling problems include further constraints. This means that problems can become more complex and more general, i.e. non-basic (Portmann, 1997). Thus, for example, precedence constraints among operations of the different jobs are common because, most of the time, mainly in discrete manufacturing, products are made of several components that can be seen as different jobs whose manufacturing must be coordinated. Additionally, since a job can be the result of fabrication and assembly of parts at several stages, different parts of the same job may be processed simultaneously on different machines. Moreover, in practice, scheduling environment tends to be dynamic, i.e. new jobs arrive at unpredictable intervals, machines faults, job are cancelled and due date and time processing

changes happen frequently. This non-basic JSSP, focused in our work, which we called **Extended Job-Shop Scheduling Problem** (EJSSP), has major extensions and differences in relation to the classic or basic JSSP. We must emphasize the existence of operations on the same job, on different components, processed simultaneously on different machines, followed by components assembly operations. This is not typical of scheduling problems addressed in the literature, in spite of being very typical of real world manufacturing. Moreover, we can observe that this approach to job definition, emphasizing the importance of considering complex jobs, which mimic customer orders of products, is in accordance with real world scheduling in manufacturing.

2.2 Single Machine Scheduling Problems

The Single Machine Scheduling Problem (SMSP) is a class of scheduling problems that deals with sequencing a set of jobs on a single processor or machine.

The study of the SMSP is important by itself and also, among other reasons, because it can provide help and insight into the resolution, understanding, managing and modelling of more complex multi-processor problems such as job-shop problems. In fact, quite often, it appears as a component in larger scheduling problems (Baker, 1974). Sometimes SMSP are independently solved and results incorporated into larger and more complex problems. For example, in multistage multiple machine problems there are often critical machines, i.e. bottlenecks, whose processing capacity is lower than the necessary. The analysis and treatment of bottlenecks as single-machines may determine the properties of the entire schedules for more complex multi-machine systems, like Job-Shop. A well-known method based on this thinking is the Shifitting Bottleneck Algorithm developed by Adams *et al* (1988). Identical strategy can be applied to manufacturing systems comprehending a network of distributed manufacturing units as is typical of what recently have been defined as Virtual Enterprises (Camarinha-Matos, 1999). Here too, a critical unit may be identified for which the schedule can restrict the properties of the schedule in the entire network of units.

In our work, we use also the strategy of using the solutions of SMSP, for the machines in a Job-Shop, as a basis for solving both, deterministic and non-deterministic Extended Job-Shop Scheduling Problems, in manufacturing. Some previouys work on the resolution of dynamic single machine scheduling problem can be seen on (Madureira *et al*., 2000) and on (Madureira *et al*., 2001b).

3. GENETIC ALGORITHMS FOR SCHEDULING

Genetic Algorithms (GA) were originally proposed by Holland in 1975 (Holland, 1992). In discovering good solutions to difficult problems, these algorithms mimic the biological evolution process.

A Genetic Algorithm is based on populations of solutions. Initially a population is created by some mechanism. Then, the GA generates other solutions, which tend to be better, by combining chromosomes, i.e. solutions, using genetic operators for selection, crossover and mutation.

Genetic Algorithms are based on the same fundamental algorithm structure as shown in Figure 1. First, an initial population of N individuals, which evolves at each generation, is created. Generally, we can say that a generation of solutions is obtained from the previous generation through the following procedure: solutions are randomly selected from the current population; pairs of selected individuals are then submitted to the crossover operation with a given crossover probability P_c; each descendant is then submitted to a mutation operation with a mutation probability P_m usually very small; the chromosome ability to solve the problem is determined by its fitness function; the final step in the generation process is the substitution of "bad" individuals of the current population by the new descendants; the algorithm stops after a predefined number, *Nger*, of generations has been created.An alternative stopping mechanism is a limit on computing time.

The interest in GA is that although guarantees of optimal solutions to problems cannot be given, good solutions are likely to be obtained, within the time available to get one, when using GA (Congram, 1998).

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.

```
NGer=0
Initialise a population of N chromosomes
Let M=N/2
While not (stop condition) do
   Nger= Nger + 1
   Create new chromosomes by mating current chromosomes
   For M pairs of solutions do
         Select randomly a pair of chromosomes
         Crossover operation with Pc
        Mutation operation with Pm
  EndFor
  Evaluate each chromosome in the population and take the best current individual.
  Substitute the "bad" individuals of the population by the new generation.
EndWhile
End
```

Figure 1 - Genetic Algorithm

Recently, scheduling in dynamic environments has been studied by a number of authors in the evolutionary community, see for example, (Fang et al, 1993), (Jain and Meeran, 19997), and (Dimopoulos, 2000). In these studies, dynamism is usually treated considering the approach "Rolling Time Horizon" (Raman and Talbot, 1993), where new jobs can arrive at any time. Initially, the scheduling problem with all known jobs is solved. Then, whenever new jobs arrive, the part of the solution consisting of operations that started before the changing occurrence is fixed and a new problem is created. This includes previously scheduled operations that have not yet been started and also the new operations of the new jobs. Thus, the dynamic problem can be decomposed into a set of static sub-problems that can be solved independently by a standard GA. Some works that use this strategy are reported in Bierwirth (1999) and Branke (1999).

Branke (1999) surveys the strategies for making evolutionary algorithms, which include GA, suitable for dynamic 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.

4. DYNAMIC DETERMINISTIC EJSSP

The scheduling approach presented in this section is applicable to both static and dynamic manufacturing environments for any optimizing criteria that are possible to establish. It is based on the decomposition of the problems into SMSP, one for each machine involved in processing, and later integration for obtaining a solution to the original problem, i.e. to the Extended JSSP. Therefore, the deterministic EJSSP problem is decomposed into a series of deterministic SMSP. We assume the existence of different and known job release times r_j, prior to which no processing of the job can be done and job due dates d_j. Based on these, release dates and due dates are determined for each SMSP and, subsequently, each such problem is solved independently by a Genetic Algorithm. Finally, the solutions obtained for each SMSP are integrated into the main problem, i.e. the non-basic EJSSP. The steps of the method used are described in table 1.

Table 1 – GA-based scheduling method for Deterministic Problems

Step 1:	Determine the completion due times (due dates) for all operations of each job;
Step 2:	Determine starting due time intervals (release times) for all operations of each job;
Step 3:	Define all SMSP based on information defined in Step1 and Step 2;
Step 4:	Solve all with those release and due dates, using a GA;
Step 5:	Integrate all the optimal or near-optimal solutions into the main problem;
Step 6:	Verify if a feasible solution has been found; otherwise apply a repairing mechanism.

In our work solutions are encoded by the direct representation, where the schedule is described as a sequence of operations, i.e. each position represents an operation index with initial and final processing times. Each operation is

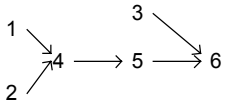
characterized by the index (i, j, l), where *i* defines the machine where the operation is processed, *j* the job that belongs, and *l* the graph precedence operation level (level 1 correspond to initial operations, without precedents).

The integration of the SMSP solutions may give an unfeasible solution to the EJSSP. In this case, step 6 must be applied. This is carried out trough a repairing mechanism named Inter-Machine Activity Coordination Mechanism (IMACM). 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 (Madureira et al., 2001d) (Madureira, 2002). Essential to the coordination mechanism is to establish the starting and the completion times for each operation. These times are related. The starting time for each operation must be equal to the higher of the two following values: the highest completion time of the immediately precedent operation (s) in the job or the completion time of the immediately precedent operation on the machine.

4.1 Ilustration example

Let us consider the following example with 6 machines and 4 jobs. A precedence graph representing the ordered allocation of machines for each job operation, i.e. the machine sequence, as well the processing times, release times and due dates are shown in table 2:

Table 2 – Illustration Example

Jobs	Machine sequence of operations	Processing times	Release Dates	Due dates
1	$1 \rightarrow 2 \rightarrow 3$	$p_{111}=10\ p_{212}=8\ p_{313}=4$	0	27
2	$2 \rightarrow 1 \rightarrow 4 \rightarrow 3$	$p_{221}=8\ p_{122}=3\ p_{423}=5\ p_{324}=6$	0	33
3	$1 \rightarrow 2 \rightarrow 4$	$P_{131}=4\ p_{232}=7\ p_{433}=3$	0	18
4		$p_{141}=3\ \ p_{241}=2\ \ p_{341}=2\ p_{442}=4$ $p_{543}=2\ p_{644}=3$	0	18

The application of the scheduling method referred in table 2, to the example is described below.

Step 1: Define completion due time for each operation of each job.

The final operation completion due time, for example, for the job 1, is $C_{313}=27$ and corresponds to job due date, as can be seen in the processing precedence graph in Figure 2. The operation completion due time C_{ijk} can be easily calculated with basis on Critical Path Analysis (CPA), as referred above. For the 2nd operation that is processed in machine 2, C_{212} is:

$$C_{212} = C_{313} - p_{313} \quad \Rightarrow \quad C_{212}=23$$

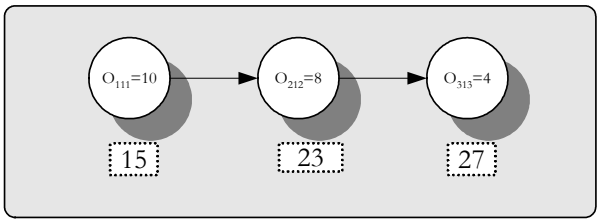


Figure 2 – Processing Precedence Graph with operation completion due times for operations of job 1

Step 2: Determine starting due time intervals for all operations of each job

The starting time interval for an operation corresponds to the time interval during which the processing of the operation must be start, i.e. the interval during which job release must take place in order to guarantee that job due date can be achieved. The completion due time C_{111} is 15, the release time r_{111} is zero and the processing time p_{111} is 10. Thus, the processing of operation O_{111} must be initiated within the time interval [0, 5], for operation O_{212}, the starting time interval is [10, 15] and for O_{313} is [18, 23]. time interval of an operation with more than one precedent operation is the intersection interval of the starting time intervals from all precedent operations correlated by the respective processing times. Thus, for example, for operation O_{442} the starting time interval is:

$$STI_{442} = [t_{141}+p_{141}, T_{141}+p_{141}] \cap [t_{241}+p_{241}, T_{241}+p_{241}]$$

$$STI_{442} = [0+3, 6+3] \cap [0+2, 7+2] = [3,9]$$

Step 3: Define all SMSP based on information defined on Step1 and Step 2.

Now, we can establish, as referred in the previous steps, the necessary data for each of the six SMSP, corresponding to the six machines available and required for processing the jobs (Table 3). The release times r_j are the earliest starting due times for each operation. The due dates d_j correspond to the operation completion due times. The notation r_j and d_j used at this stage, considers that we are dealing with single machine cases.

Table 3 – Job attributes of the SMSP

	M1				M2				M3			M4			M5	M6
Jobs	**111**	**122**	**131**	**141**	**212**	**221**	**232**	**241**	**313**	**324**	**341**	**423**	**433**	**442**	**543**	**644**
p_i	10	3	4	3	8	8	7	2	4	6	2	5	3	4	2	3
r_i	0	8	0	0	10	0	4	0	18	16	0	11	11	3	7	9
d_i	15	22	8	9	23	19	15	9	27	33	15	27	18	13	15	18

Step 4: Solve all SMSP using the GA

Applying the Genetic Algorithm previously described, to each problem, with the objective for example, of minimise the completion time C_{max}, we can obtain the following solutions for the machines 1, 2, 3 and 4, respectively: (131, 141, 111, 122), (241, 232, 221, 212), (341, 313, 324), (442, 433, 423). Machines 5 and 6 do not need an optimization process (one single solution).

Step 5: Integrate all obtained solutions into the main problem

Interpreting the Gantt diagram of figure 3 we mean, for example by the schedule (O_{442}, O_{433}, O_{423}) on machine 4, that this machine first process the operation of job 4 than the one of job 3 and finally that of job 2. Job one does not require machine 4 as can be seen from table 3.

At this point, it is necessary to apply a decoder to each solution, in such a way that starting times are defined for all operations. The obtained schedules are characterized by the following starting times (Ti) and completion times (Tc) as it is shown in Table 4.

Table 4 – Obtained results by the decoder

	M1				M2				M3			M4			M5	M6
	111	**122**	**131**	**141**	**212**	**221**	**232**	**241**	**313**	**324**	**341**	**423**	**433**	**442**	**543**	**644**
Ti	7	17	0	4	19	11	4	0	27	31	0	22	11	7	11	13
Tc	17	20	4	7	27	19	11	2	31	37	2	27	14	11	13	16

Step 6: Verify if feasible solution has been found, otherwise apply the repairing mechanism.

In this example, the union of the several local optimum solutions, one for each $\mathbf{1|r_j|C_{max}}$ problem, could not produce a feasible solution to the $\mathbf{J|rj|C_{max}}$ problem due to processing overlap operations belonging to the same job, as we can see from Figure 3. For example, processing of operation O_{442} of job 4 is scheduled to start even before the immediately precedent operation O_{141} has been finished, violating, therefore, technological precedence constraints. Identical violation occurs with job 1. This happens because we only consider the precedence relationships and job due dates on individual machines. The interrelated activity of all the machines, i.e. the inter-machine activity coordination has not been taken care of. So, due to such violation the IMACM mechanism for repairing the schedule must be applied. The results from such application are shown in Table 5, where Ti' is the new starting time, Tc' the completion time and Level is the processing order established by the IMACM mechanism. This led to a feasible schedule for the EJSSP of the example where all due dates are satisfied, figure 3.

Table 5 – Obtained results

	M1				M2				M3			M4			M5	M6
	111	**122**	**131**	**141**	**212**	**221**	**232**	**241**	**313**	**324**	**341**	**423**	**433**	**442**	**543**	**644**
Ti'	7	19	0	4	19	11	4	0	27	31	0	22	11	7	11	13
Tc'	17	22	4	7	27	19	11	2	31	37	2	27	14	11	13	16
Level	2	3	0	1	3	2	1	0	4	5	0	4	3	2	6	7

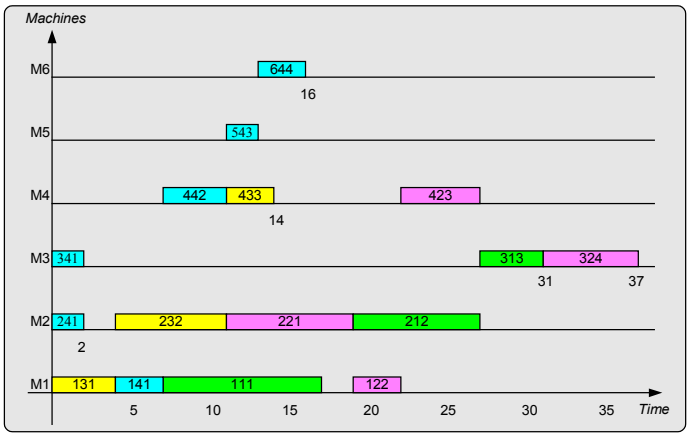


Figure 3 –Feasible solution for the EJSSP obtained through joint application of the SMSP GA algorithm and the IMACM mechanism

Most of the research on JSSP focuses on basic problems as described above. The method developed and just described is in line with reality and away from the approaches that deal solely with static and classic or basic job-shop scheduling problems. Thus, the method is likely to perform worse than the best available algorithms found for such problems. However, it is not our purpose, neither it would be reasonable, to rate our method against such good performing algorithms for academic and basic JSSP. Our aim is to provide an efficient tool, which we think we managed with our method, for obtaining good solutions, for a variety of criteria, for many real world scheduling problems, i.e. complex non-basic JSSP as described above, which we named Extended JSSP. For these problems, the referred best performing algorithms are unable to give solutions. Further, through the survey we made to the literature we were unnable to find methods to solve the EJSSP as here described.

5. DYNAMIC NON-DETERMINISTIC EJSSP

For non-deterministic problems some or all parameters are uncertain, i.e. are not fixed as we assumed in the deterministic problem. Non-determinism of variables has to be taken into account in real world problems. For generating acceptable solutions in such circumstances our approach starts by generating a predictive schedule, as defined by Smith (1994), using the available information and then, if perturbations occur in the system during execution, the schedule may have to be modified or revised accordingly, i.e. rescheduling is performed. Therefore, in this process, an important decision must be taken, namely that of deciding if and when should rescheduling happen. The decision strategies for rescheduling may be grouped into three categories (Sabuncuoglu et al., 2000): continuous, periodic and hybrid rescheduling. In the continuous one rescheduling is done whenever an event modifying the state of the system occurs. In periodic rescheduling, the current schedule is modified at regular time intervals, taking into account the schedule perturbations that have occurred. Finally, for the hybrid rescheduling the current schedule is modified at regular time intervals if some perturbation occurs.

In the scheduling system for Extended JSSP, implementing our approach, rescheduling is necessary due to two classes (Madureira *et al.*, 2001a) of events: **partial events** and **total events**. **Partial events** imply changes in jobs or operations attributes such as processing times, due dates and release times. **Total events**, imply changes in population structure, resulting from either new job arrivals or job cancellations. While, on one hand, partial events only require redefining job attributes and re-evaluation of the objective function of solutions, total events, on the other hand, require a change on chromossome structure and size, carried out by inserting or deleting operations, and also re-evaluation of the objective function.

Therefore, under a total event, the modification of the current chromossome is imperative. In this work, this is carried out by a mechanisms described in Madureira et al. (2000, 2001b) for SMSP. Considering the processing times involved and the high frequency of perturbations, rescheduling all jobs from the beginning should be avoided. However, if work has not yet started and time is available, then an obvious and simple approach to rescheduling would be to restart the scheduling from scratch with a new modified solution on which takes into account the perturbation, for example a new job arrival. When there is not enough time to reschedule from scratch or job processing has already started, a strategy must be used which adapts the current schedule having in consideration the kind of perturbation occurred.

A non-deterministic environment is characterized by several variations and uncertainties on working conditions and requirements over time. The, here proposed, GA-based scheduling system for non-deterministic EJSSP is structured, as

shown in figure 4, in three modules, namely the modules for pre-processing, scheduling and dynamic schedule adaptation below described.

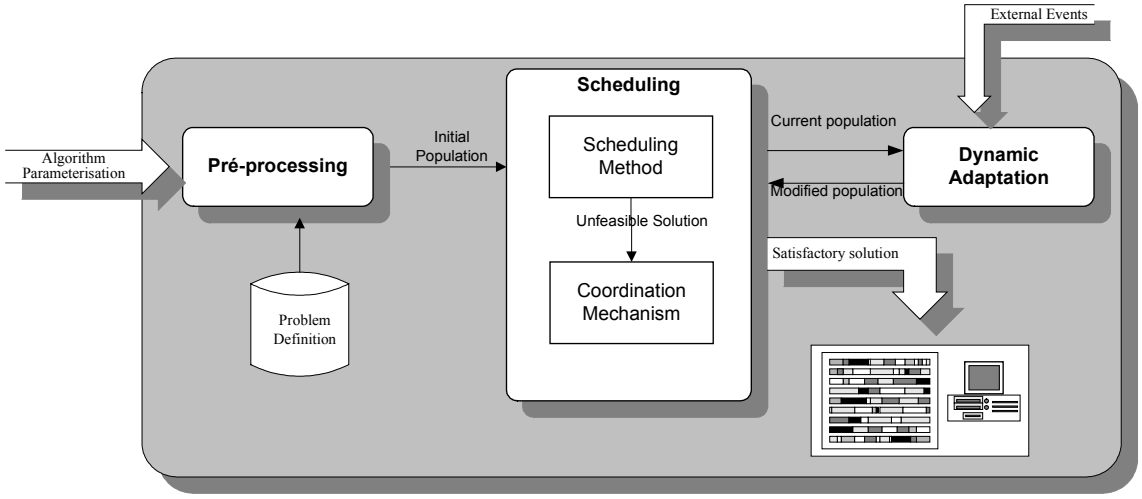


Figure 4 - GA-based Scheduling System for Non-Deterministic Scheduling Problems

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.

5.2 Scheduling Module

The scheduling module is concerned with the application of the GA-based scheduling method for deterministic EJSSP presented above, considering that all release dates, processing times and due dates are known in advance.

Whenever a new event occurs which disturbs the schedule, in such a way that rescheduling is to be done, the dynamic adaptation module generates a new deterministic problem. Then, the scheduling module solves this new problem.

5.3 Dynamic Adaptation Module

The occurrence of a *partial event* requires redefining job attributes and a re-evaluation of the schedule fitness function.

A change in job due date requires the re-calculation of the operation starting and completion due times of all respective operations. However, changes in the operation processing times only requires re-calculation of the operation starting and completion due times of the succeeding operations.

A new job arrival requires definition of the correspondent operation starting and completion times and a regenerating mechanism to integrate all operations on the respective single machine problems. In the presence of a job cancellation, the application of a regenerating mechanism eliminates the job operations from the SMSP where they appear. After the insertion or deletion of genes, population regeneration is done by updating the size of the population and ensuring a structure identical to the existing one (Madureira, 2000).

Then the scheduling module can apply the search process for better solutions with the new regenerated population.

Job arrival integration mechanism

When a new job arrives to be processed an integration mechanism is needed. This analyses 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. Two alternative procedures could be used for each operation: either 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.

Job elimination mechanism

When a job is cancelled, an eliminating mechanism must be implemented so the correspondent gene will be deleted from all chromosomes.

Regeneration mechanisms

After integration/elimination of operations is carried out, by inserting/deleting genes in all chromosomes, population regeneration is done by updating the size of the population, maintaining an identical population structure. Thus, either some further chromosomes have to be generated by some procedure, for example by REDD – Random Earliest Due Date rule (Madureira, 2000), or some existing chromosomes have to be deleted. In this case, the choice could either be random or fall on the less adapted chromosomes. After this, the scheduling module can apply the search process for better solutions with the regenerated population.

6. CONCLUDING REMARKS

In most practical environments, scheduling is an ongoing reactive process where the presence of real time information continually forces reconsideration and revision of pre-established schedules.

The handling of uncertainty is an important issue in real-world scheduling problems. Uncertainty can arise through incomplete knowledge of the problem, through incomplete knowledge of how the problem will change over time or may be inherent to the problem itself. Although the treatment of uncertainty has been already a focus of investigation from academic community, the main efforts on that way arise from the modeling of uncertainty in static deterministic scheduling problems and the development of algorithms to solve it (Raman and Talbot, 1993).

This work is concerned with the resolution of realistic Job-Shop Scheduling Problems (JSSP). Thus, not only it focuses on the solution of the dynamic JSSP problem but also on both the deterministic and non-deterministic versions of it. Moreover, it is concerned with integrated scheduling of jobs which are products composed by several parts or components which may be submitted to a number of manufacturing and multi level assembly operations, having as the main criterion meeting due dates. We call these problems Extended JSSP. For solving these scheduling problems we developed a framework embedded in a dynamic scheduling system whose main pieces of intelligence are a genetic algorithm and a mechanism for inter machine activity coordination. The Genetic Algorithm for the realistic JSSP problem draws upon a genetic algorithm developed initially for the minimisation of the static weighted tardiness Single Machine Scheduling Problem (SMSP), but adaptable to a variety of other performance measures, such as achieving due dates. Having in mind that finishing a job early, i.e. when tardiness is zero, does not necessarily means good performance, it seemed to us to be more important the accuracy of achieving due dates for the dynamic EJSSP, reason why we centre the scheduling system on achieving job due dates.

Considering that natural evolution is a process of continuous adaptation, it seemed us appropiate to consider Genetic Algorithms for tackling real Non-Deterministic Scheduling Problems. Thus, the GA based scheduling system developed adapts the resolution of the static and deterministic problem to the dynamic one in which changes may occur continually. A population regenerating mechanism is put forward, for adapting the population of solutions, according to disturbances, to a new population, which increases or decreases according to new job arrivals or cancellations.

If an unfeasible schedule is obtained through the genetic algorithm, the IMACM mechanism is applied for ensuring that work at each machine does not violate constraints and a feasible solution is obtained. The IMACM mechanism is coordination mechanism of manufacturing activity, which coordinates the machine working times with job operations precedence relationships.

We could not make a comparative performance study with other methods for two reasons: first, because no benchmark problems were found, in the literature surveyed, for the EJSSP we address. Second, because the same was true for methods, i.e. none was found addressing the EJSSP. Work still to be done, includes more testing of the proposed algorithms and mechanisms under dynamic Job-Shop environments subject to several random perturbations. We realize, however, that this is not an easy task because it is difficult to find test problems and computational results for the dynamic environment considered, where the jobs to be processed have release dates, due dates and different job assembly levels (parallel operations).

7. REFERENCES

Adams, Joseph, Balas, Egon and Zawack, Daniel (1988). The Shifting Bottleneck Procedure for Job Shop Scheduling, *Management Science*, Vol. 34, nº 3, USA.

Baker, K.R. (1974). *Introduction to sequencing and scheduling*, Wiley, New York.

Bierwirth, C. and Matfeld, D.C. (1999). Production scheduling and rescheduling with genetic algorithms, *Evolutionary Computation*, 7, 1-17.

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

Branke, J. (1999). Evolutionary Approaches to Dynamic Optimization Problems - A Survey-, *GECCO Workshop on Evolutionary Algorithms for Dynamic Optimization Problems*, 134-137.

Brucker, P. (2001). Scheduling Algorithms, Springer, 3rd edition, New York.

Camarinha-Matos, L. M. and Afarmanesh, A. (1999). The Virtual Enterprise Concept, *In IFIC TC5/PRODNET Working Conf. Infraestructures for Virtual Enterprises*, Kluwer Academic Publishers.

Congram, Richard, Potts, C. and Velde, S.L.V. (1998). An iterated dynasearch algorithm for the single machine total weighted tardiness scheduling problem, *Preprint Series nºOR95*, University of Southampton.

Davis, Lawrence (1991). *Handbook of genetic algorithms*, Van Nostrand Reinhold, New York.

Dimopoulos, C. and Zalzala, Ali M. S. (2000). Recent Developments in Evolutionary Computation for Manufacturing Optimization: Problems, Solutions and Comparisons, *IEEE Trans. on Evol. Computation*, nº4, vol.2, 93-113.

Fang, Hsiao-Lan, Ross, Peter and Corne, Dave (1993). A promising Genetic algorithm approach to Job Shop scheduling, Rescheduling and Open Shop Scheduling Problems, International Conf. Genetic Algorithms, 375-382.

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

Holland, J. H., (1992), *Adaptation in Natural and Artificial Systems*, 1ª edição, MIT Press.

Jain, A. S. e Meeran, S., (1999), Deterministic Job Shop scheduling: past, present and future, *European Journal of Operational Research* 113, 390-434.

Madureira, Ana M., Ramos, Carlos and Silva, Sílvio do Carmo, (2000), A Genetic Algorithm for The Dynamic Single Machine Scheduling Problem, *BASYS'2000* , Berlim, Alemanha.

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

Madureira, Ana M., Ramos, Carlos and Silva, Sílvio do Carmo, (2001b), A GA Based Scheduling System for The Dynamic Single Machine Scheduling Problem, ISATP'2001 (*IEEE International Symposium Assembly and Task Planning*), Fukuoka (Japão).

Madureira, Ana M., Ramos, Carlos and Silva, Sílvio do Carmo, (2001d), An Inter-Machine Activity Coordination based Approach for Dynamic Job Shop Scheduling, *International Journal for Manufacturing Science and Production*, Freund Publishing House Ltd.vol 4, nº2.

Madureira, Ana M., Ramos, Carlos and Silva, Sílvio do Carmo, (2002), A Coordination Mechanism for Real World Scheduling Problems Using Genetic Algorithms, *IEEE 2002 IEEE World Congress on Computational Intelligence - Congress on Evolutionary Computation* (CEC'2002), Honolulu - Hawaii (EUA).

Parunak, H. Van Dyke, (1992), Characterizing the Manufacturing Scheduling Problem, *Journal of Manufacturing Systems*, nº 10, 241-259.

Pinedo, M., (2001), *Scheduling – Theory, Algorithms and Systems*, 2ª edição, Prentice-Hall.

Portmann, M. C. (1997). Scheduling Methodology: optimization and compu-search approaches, in *The planning and scheduling of production systems*, Chapman &Hall.

Raman, N, and Talbot, F.B., (1993), The Job-Shop tardiness problem: a decomposition approach, *European Journal Oper. Res.*, 69, 187-199.

Sabuncuoglu, I. and Bayiz, M. (2000). Analysis of reactive scheduling problem in a job shop environment, *European Journal of Operational Research*, 567-586.

Smith, S.F., (1994), in *Intelligent Scheduling*, Morgan Kaufman Publishers, San Francisco.

Vollmann, Thomas E., Berry, William L. e Whybark, D. Clay, (1996) Manufacturing Planning and Control Systems, McGraw-Hill, New York.