TOWARD DYNAMIC SCHEDULING THROUGH EVOLUTIONARY COMPUTING

ANA MADUREIRA[*] CARLOS RAMOS[*] SÍLVIO CARMO SILVA[†]

[*]Institute of Engineering - Polytechnic of Porto
GECAD – Knowledge Engineering and Decision Support Research Group
Porto, PORTUGAL
{anamadur, csr}@dei.isep.ipp.pt http://www.dei.isep.ipp.pt/~anamadur/

[†] Minho University, Dept. of Production and Systems
Braga – PORTUGAL
scarmo@dps.uminho

Abstract: - Real world scheduling requirements are related with complex systems operating in dynamic environments. This means that they are frequently subject to several kinds of random occurrences and perturbations, such as new job arrivals, machine breakdowns, employee's sickness and jobs cancellation causing prepared schedules becoming easily outdated and unsuitable. Scheduling under this environment is known as dynamic. These problems pose additional challenges for optimization techniques. This paper outlines the limitations of static approaches to scheduling in the presence of dynamic environments and gives a review of currently developing research on real world scheduling problems, which are often complex, constrained and dynamic. We decided to explore the use of Evolutionary computing techniques for solving real-world optimization problems. Therefore, in this paper, we present a Genetic Algorithm based scheduling method, which is of practical utility, embedded in a simple framework to solve difficult problems in manufacturing environments.

Key-Words: - Dynamic Scheduling, Job-Shop Scheduling, Evolutionary Algorithms, Genetic Algorithms, Resource-Oriented Scheduling, Manufacturing.

1 Introduction

A major challenge in the area of global market economy is to develop new techniques for solving real-world scheduling problems. Indeed, any industrial organization can only be economically visible by maximizing customer services, maintaining efficient, low cost operations and minimizing total investment.
With rising global competition, it is becoming increasingly more important for industry to optimize its activities. However, the complexity of real world optimization problems has prevented the industry from exploiting the potential of optimization algorithms. The industry has, therefore, relied on either trial-and-error or over-simplification for dealing with its optimization problems. This has led to a loss of opportunity for saving costs and time. The growth of research in the field of Evolutionary computing has been encouraged by a desire to harness this opportunity.
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. But these approaches are NP-complete and have shown some limitations. One of these limitations is the inability to express domain knowledge and to exploit it. For literature on this subject, see for example[8][13][17][22][26][27][31].
In spite of all the previous trials the scheduling problem is well known as one of the most difficult NP-hard combinatorial optimization problems[26][31] and still known to be NP-complete[22][24][25]. This fact incites researchers to explore new directions.

Since its entry into the mainstream of academic research in the late 1950's, the field of scheduling has been addressed by a diversity of authors from different perspectives. Research in combinatorial optimization has evolved from basic formulations of single machine and Job-Shop scheduling problems, while work in control theory alternatively has emphasized dynamics and stochastic scheduling models, and the field of artificial intelligence has promoted constraint satisfaction and heuristic search paradigms.

Over the years, a vast and diverse body of literature has arisen from these various roots, and a large body of knowledge and algorithms now exists for tackling scheduling applications. At the same time, the complexities and idiosyncrasies of practical scheduling environments continue to pose significant challenges for current techniques and there is a continuing need to expand the scope of scheduling research to address practical problems.

As many real-world optimization problems take place in environments constantly changing. Because of this, optimization algorithms instead of looking for single optimal solutions should continuously track the optimum through time. Real world scheduling requirements are related with complex systems operated in dynamic environments. This means that they are 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 prepared schedules becoming easily outdated and unsuitable. Scheduling under this environment is known as dynamic.

Many real-world optimization problems are changing non-deterministically over time. In those cases, the purpose of the optimization algorithm changes from finding an optimal solution to being able to continuously track the optimum through time.

In recent years, there has been a significant level of research interest in evolutionary approaches for solving large real world scheduling problems which are often complex, constrained and multicriteria in nature. Typical examples of such problems include production scheduling, vehicle routing, personnel rostering, educational timetabling etc.

Evolutionary Algorithms are Metaheuristics, inspired by natural evolution in a broader sense, that have often been shown to be effective for difficult combinatorial optimization problems appearing in various industrial, economical, and scientific domains. These techniques seem to be very well suited to that kind of problem, since basically, nature is a continuously changing challenge.

This paper outlines the limitations of static approaches to scheduling in the presence of dynamic environments and gives a review of currently developing research on real-world scheduling problems, which are often complex, constrained and dynamic.

The main purpose of this paper is to describe a framework based on Genetic Algorithms for solving a class of real-world scheduling problems, where the products (jobs) to be processed have due dates, release times and different assembly levels. This means that parts to be assembled may be manufactured in parallel, i.e. simultaneously. Therefore, in this work, we define a job as a manufacturing order for a final item, simple or complex. It may be simple, like a part, requiring a set of operations to be processed. We call it a **Simple Product** or **Simple Final Item**. **Complex Final** items, requiring processing of several operations on a number of parts followed by assembly operations at several stages, are also dealt with.

We adopt a strategy to scheduling known as Resource-Oriented Scheduling (ROS). This is related with vertical loading [32], but much more detailed in order to define start and completion times of jobs in the available processors. In ROS each job is taken in turn for scheduling on a resource or machine at a time. Although any objective function may be considered, resource-oriented scheduling has an underlined aim of getting good use of resources[4][5].

Our approach, implementing ROS, starts by solving a surrogated set of a Single Machine Scheduling Problems (SMSP), whose solutions are repaired, if necessary, by means of a suitable mechanism, in order to arrive to feasible solutions for the original problem.

The remaining sections are organized as follows: Section 2 characterizes real world scheduling problems. Some concepts and classifications related to dynamic scheduling are presented. Section 3 summarizes related work. In section 4 the scheduling problem under consideration is described. The proposed GA based scheduling system is described on section 5.Finally, the paper presents some conclusions and puts forward some ideas for future work.

2 Real World Scheduling Problems

Scheduling may be defined as the allocation of resources over time to perform tasks. It is a decision process whose goal is the optimization of one or more objectives [22]. The objective is to find a schedule, which optimizes some performance measure. The most extensively investigated one is the *makespan*, i.e. the time elapsed from the beginning of processing until the last operation in the last job has been finished. In our approach, the *makespan* objective is not realistic, since it does not consider due dates, and it is not well suited for dynamic environments where jobs can arrive continuously over time.

We consider more realistic performance measures such as those that reflect schedule implementation costs and are to be minimized, meaning that a low value for the objective function corresponds to a good scheduling solution. We are talking about some regular performance, such as, maximum lateness L_{max}, summed lateness $\sum L$, maximum tardiness T_{max} and weighted summed tardiness $\sum WT$. For literature on this subject see for example[8][13][17][22][26][27][31].

In generic terms, the scheduling process can be also 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 problem can be seen as a decision making process for operations starting and resources assignment to be used. A variety of characteristics and constraints related with jobs and production system, such as operation processing time, release and due dates, precedence constraints and resource availability, can affect scheduling decisions.

In practice, many scheduling problems include further constraints and relaxation of others. This means that problems can become more complex and more general. Thus, for example, precedence constraints among operations of the different jobs are common because, most of the times, 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 manufacturing 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 have faults, jobs are cancelled and due dates and time processing changes frequently. This non-basic Job Shop Scheduling Problem (JSSP)[21], focused in our work, which we name Extended JSSP (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 in 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.

The static scheduling problem refers to the situation in which all jobs are simultaneously available for processing. The complexity increases for dynamic problems. Whenever an unexpected event happens in a manufacturing environment, a scheduling decision must be made in real time about the possible reordering of jobs.

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 to almost every state [9].

Graves [30] referred that "*It may be quite easy to construct a schedule; what is difficult is the constant revision required by the dynamic environment.*"

The process of finding a feasible sequence and assigning starting times to respective operations before the production process takes place may be defined as predictive scheduling.

Dynamic scheduling, rescheduling or reactive scheduling is the process of revising a given schedule due to unexpected perturbations on manufacturing system operation. Thus, scheduling can be seen as a continuous process, which starts by constructing a predictive schedule for subsequently and iteratively to be revised and changed as time passes and disturbances occur.

The robustness and flexibility are important characteristics of a solution. A solution is robust if it resists to random external modifications. A solution is flexible if it is easy to repair in case of environment changes. Some important related work on this subject can be seen in [12].

Dynamic scheduling must be able to perform scheduling in highly dynamic, uncertain environments where there is incomplete information and changes often occur; modify previously formed schedules in compliance with the most recent

dynamic information, minimizing the disruption of earlier schedules and still aiming for the most effective possible use of resources and achievement of goals and provide enough flexibility to react robustly to any disruption in an efficient and timely manner.

The dynamic events commonly considered in the research can be divided into two main categories[24]:

- Events based resources: machine breakdown, operator illness, unavailability of specific tools, loading limits, unavailability of materials, material arrival times, defective material, etc.
- Events based jobs: jobs that arrive early/late, new jobs, dynamic priorities, updated the deadlines, rush jobs, etc.
-

Many research works have surveyed the limitations of predictive scheduling, and have been investigating the importance and impact of real-time information, see for example [16][24].

The problem of repairing schedules in the most effective way in the presence of real-time information is one that is receiving increasing attention. There are two alternatives, which have been discussed in literature, see for example [3]. The first one which is the simplest one is to regenerate a new schedule from scratch for every event (reschedule), but this solution is not efficient because disturbances tend to appear very often, and creating a completely new schedule every time an event occurs would be time consuming and costly. Furthermore, frequent changes to the schedule can cause instability of the system that can be costly. The second alternative is to repair the existing schedule (schedule/revise) with minor changes to maintain stability.

Some different approaches to schedule repair/rescheduling have been proposed. Leon et al.[33] developed robustness measures and robust scheduling repair methods for job shops to deal with machine breakdowns and processing time variability using a right-shift control policy in order to minimize the expected *makespan*. The policy always maintains the original sequence and is very efficient for situations in which sequence changes are costly. Cowling [24] has surveyed rescheduling and schedule-repair techniques. He has proposed two measures, utility and stability, to decide whether to reschedule or schedule repair. For Lee et al. [10] the choice between rescheduling and reschedule repair depends on the severity of the disruptions.

3 Previous Work

A vast amount of literature about Genetic Algorithm applications to the scheduling problem has been published in the last few decades. One of the earliest published works on the application of GA to scheduling is that by Davis[18]. Many GAs have been proposed and analysed for the static case of the JSSP, referred on [19][29]. Other Metaheuristics such as Tabu Search and Simulated Annealing have also been developed for solving the JSSP [5].

Recently the scheduling problems in dynamic environments have been investigated by a number of authors in the evolutionary community, see for example [11][20][28]. The dynamism of the problem is usually treated considering the approach of a "Rolling Time Horizon"[23], 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 [7][15].

In [14] the author 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.

Madureira et al.[6] present several local search meta-heuristics for the problem of scheduling a single machine to minimize total weighted tardiness. A genetic algorithm for the static single machine total weighted tardiness problem is presented, and a multi-start version named metaGA is proposed. The obtained computational results permit to conclude about their efficiency and effectiveness. For that reason, the resolution of the dynamic single machine total weighted tardiness problem using a scheduling

system based on Genetic Algorithms (GA) is questioned. This approach extends the resolution of static Single Machine Scheduling Problems (SMSP) to dynamic SMSP in which changes can occur continually. In Madureira et al. [2] is presented a new strategy of dealing with dynamism of the problem, i.e. a scheduling system based on genetic algorithms to solve the dynamic version of the static weighted tardiness Single Machine Scheduling Problem. A population regenerating mechanism is put forward.
Madureira et al. [1] describes initially a scheduling system, based on Genetic Algorithms to solve the dynamic Job-Shop Scheduling Problem,

4 Problem Definition

Most real-world multi-operation scheduling problems can be described as dynamic and extended versions of the classic or basic Job-Shop scheduling[21] combinatorial optimization problem. 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, for certain duration, on a machines. In the basic Job-Shop scheduling problem (JSSP) several constraints on jobs and machines are considered: machines are always available and never breakdown, there are no precedence constraints among operations of the different jobs; each machine can process only one job at a time; the processing of each operation cannot be interrupted; 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; processing, release and due times of jobs are deterministic and known in advance. These are the most common assumptions and restrictions in academic JSSP.
In practice, many scheduling problems include further restrictions and relaxation of others [21]. Thus, for example, precedence constraints among operations of the different jobs are common because, most of the times, 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 tend to be dynamic, i.e. new jobs arrive at unpredictable intervals, machines breakdown, jobs are cancelled and due dates and processing times change frequently. This non-basic JSSP, focused in our work, which we call **Extended Job-Shop Scheduling Problem** (EJSSP), has major extensions and differences in relation to the classic or basic JSSP. The existence of operations on the same job, on different parts and components, processed simultaneously on different machines, followed by components assembly operations, which characterizes EJSSP, is not typical of scheduling problems addressed in the literature. However, such is very common in practice. 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.

5 GA based Scheduling System

This work is concerned with the solution of problem instances of the EJSSP. It starts focusing on the solution of the dynamic deterministic JSSP problems. For solving these, we developed a framework, leading to a dynamic scheduling system having as a fundamental scheduling tool a GA-based scheduling method with two main pieces of intelligence. One such piece is a GA-based scheduling method for deterministic scheduling problems. This includes a Genetic Algorithm for single machine problems and an inter-machine activity coordination mechanism that attempts to ensure a good feasible solution for the deterministic EJSSP. The other piece is a dynamic adaptation mechanism that includes a method for neighbourhood regeneration under dynamic environments, increasing or decreasing it according new job arrivals or cancellations.
The proposed approach consists on a centralised GA-based scheduling system. One advantage of this is that the solutions (schedules) planned for single machines guarantees some consistency of manufacturing. The original Extended Job-Shop Scheduling Problem is decomposed into a series of deterministic Single Machine Scheduling Problems solved one at a time, consecutively. The obtained solutions are then incorporated into the main problem.
The proposed approach to solve the dynamic EJSSP consists on generating a predictive schedule in advance using the information available. When disruptions occur in the system during the execution, the predictive schedule is modified or revised in order to consider the recent modifications.

5.1 GA based scheduling method

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).
Initially, we start by decomposing the deterministic EJSSP problem into a series of deterministic Single Machine Scheduling Problems. 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, also, 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 the Genetic algorithm. Afterwards, the solutions obtained for each SMSP are integrated to obtain a solution to the main problem instance, i.e. the instance of the EJSSP.

Table 1- GA based scheduling method

1st PHASE	**Finding a 1st 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 between 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 GA.
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 IMACM 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 due times for each operation of a job are derived from job due dates and processing times by subtracting the processing time from the completion due time of the immediately succeeding job operation
This procedure begins with the last job operation and ends with the first. The operations starting due time intervals $[t_{ijk},T_{ijk}]$ are also defined considering the job release times and the operation processing times. The earliest starting time t_{ijk} corresponds to the time instant 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 completion due time (due date). This means that no further delay is allowed. When an operation has more than one precedent operation, i.e. there exists a multilevel structure, the interval $[t_{ijk},T_{ijk}]$ is the interval intersection from precedent operations correlated by the respective processing times.
At this stage, only technological precedence constraints of operations and job due dates will be consid-ered for defining completion and starting times.
The integration of the SMSP solutions may give an unfeasible schedule to the EJSSP. This is why schedule repairing may be necessary to obtain a feasible solution. The repairing mechanism named Inter-Machine Activity Coordination Mechanism (IMACM) [3] carries this out. The repairing is carried out through coordination of machines activity, having into account job operation precedence and other problem constraints. This is done keeping job allocation order, in each machine, un-changed. The IMACM mechanism establishes the start-ing and the completion times for each operation. It en-sures that the starting time for each operation is the highest of the two following values:

- the completion time of the immediately precedent operation in the job, if there is only one, or the highest of all if there are more
- the completion time of the immediately precedent operation on the machine.

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 unable to find methods to solve the EJSSP as here described.

5.2 Dynamic Adaptation Mechanism

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, 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 [12]: 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 of events[1] [2]:

- **Partial events** which imply variability in jobs or operations attributes such as processing times, due dates and release times.
- **Total events** which imply variability in neighbourhood 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 solution 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 solution is imperative. In this work, this is carried out by mechanisms described in [1] 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.

The occurrence of a partial event requires redefining job attributes and a re-evaluation of the schedule objective 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 positions, neighbourhood regeneration is done by updating the size of the neighbourhood and ensuring a structure identical to the existing one. Then the scheduling module can apply the search process for better solutions with the new modified solution.

a) 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 the current solution/chromosome or use some intelligent mechanism to insert this operation in the schedules, based on job priority, for example.

b) Job elimination mechanism

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

c) Regeneration mechanisms

After integration/elimination of operations is carried out, by inserting/deleting positions/genes in the current solution/chromosome, population regeneration is done by updating its size. The population size for SMSP is proportional to the number of operations.

After dynamic adaptation processs, the scheduling method could be applied and search for better solutions with the modified solution.

6 Conclusions and Further Work

The handling of dynamism or uncertainty is a significant issue in many real-world problems. Uncertainty can arise from incomplete knowledge of the problem at hand, or from incomplete knowledge of how the problem changes over time. 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 [23].

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.

This paper has outlined an important gap between static and dynamic scheduling. Dynamic scheduling or rescheduling maintains schedules in a dynamic environment where a variety of unexpected events can occur at any time.

We have identified two types of perturbations commonly considered in research: perturbations related to resources (machine breakdown, operator illness, unavailability of specific tools, loading limits, etc.), and perturbations related to jobs (jobs arrive at random, new jobs, dynamic priorities, updating the deadlines, rush jobs, etc.). We have discussed two alternatives to deal with the problem of updating schedules in the most effective way in the presence of real-time information: rescheduling and schedule revise. Rescheduling is often not satisfactory because it is time consuming, and can cause instability. Schedule revise is more practical and maintains system stability.

This work is concerned with the resolution of realistic Job-Shop Scheduling Problems (EJSSP). Thus, not only it focuses on the solution of the dynamic EJSSP 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 EJSSP.

Considering that natural evolution is a process of continuous adaptation, it seemed us appropriate to consider Genetic Algorithms for tackling real Non-Deterministic Scheduling Problems. Thus, the GA based scheduling system developed adapts the resolution of the 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.

We recognize the need for further testing, particularly for better evaluation of the suitability the proposed framework and mechanisms under dynamic Extended Job-Shop environments. We also recognize that this is not an easy task because it is difficult to find in the literature test problems with the job structure that we selected and think important, in industrial practice, namely jobs made from several parts to be manufactured and assembled through several assembly operations and stages.

Work still to be done includes more testing of the proposed algorithms and mechanisms under dynamic Job-Shop environments [4]. We are also working on the improvement of the scheduling mechanisms and developing further testing to improve the quality of solutions. 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).

References:

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

[2]. Ana M. Madureira, Carlos Ramos and Sílvio do Carmo Silva, A Genetic Algorithm for The Dynamic Single Machine Scheduling Problem, *In 4th IEEE/IFIP Intl. Conf. on Information Technology for Balanced Automation Systems in Production and Transportation*, Berlim (Germany), 2000, pp 315-323.

[3]. Ana M. Madureira, Carlos Ramos and Sílvio do Carmo Silva, 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, 2001.

[4]. Ana M. Madureira, Carlos Ramos e Sílvio C. Silva, Resource-Oriented Scheduling for Real World Manufacturing Systems, *IEEE International Symposium Assembly and Task Planning (ISATP'2003)*, Besançon (França), 2003, pp.140-145.

[5]. Ana M. Madureira, Carlos Ramos e Sílvio C. Silva, Vertical Scheduling Approach to Dynamic Scheduling Problems Using Tabu Search, *5th MetaHeuristics International Conference (MIC'2003)*, Fukuoka (Japão), 2003.

[6]. Ana M. Madureira, Meta-heuristics for the Single-Machine Scheduling Total Weighted Tardiness Problem, *ISATP'99 (IEEE International Symposium Assembly and Task Planning)*, Portugal, 1999.

[7]. C. Bierwirth and D.C. Matfeld, Production scheduling and rescheduling with genetic algorithms, *Evolutionary Computation*, nº 7, 1999, pp 1-17.

[8]. E. Morton and D. W. Pentico, Heuristic Scheduling Systems, John Wiley & Sons, 1993.

[9]. H. V. D. Parunak, Characterizing the Manufacturing Scheduling Problem, *Journal of Manufacturing Systems*, nº 10, 1992, pp.241-259, 1992.

[10]. H. S. Lee, S. S. Murthy, S. W. Haider and D.V., Primary production scheduling at steel making industries, *IBM Journal Research Development*, Vol. 40, No. 2, 1996.

[11]. Hsiao-Lan Fang, Peter Ross and Dave Corne, A promising genetic algorithm approach to Job Shop scheduling, rescheduling and open shop scheduling problems, In Morgan Kaufmann (eds) *Proceedings of the Fifth International Conference on Genetic Algorithms*, 1993, pp. 375-382.

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

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

[14]. J. Branke, Evolutionary Approaches to Dynamic Optimization Problems – A Survey, *GECCO Workshop on Evolutionary Algorithms for Dynamic Optimization Problems*, 1999, pp. 134-137.

[15]. J. Branke, Efficient Evolutionary Algorithms for Searching Robust Solutions, Fourth Intl. Conf. on Adaptive Computing in Design and Manufacture (ACDM 2000), 2000, pp. 275-286.

[16]. Jurgen Dorn. Reactive scheduling improving the robustness of schedules and restricting the effects of shop floor disturbances by fuzzy reasoning. International *Journal Human Computer Studies*, 42, 1995, pp. 687-704.

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

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

[19]. Lee C.Y., S. Piramuthu and Y.K. Tsai, Job Shop Scheduling With a Genetic Algorithm and Machine Learning, *Intl. Journal Prod. Research*, vol.35, nº4, 1997, 1997, pp.1171-1191.

[20]. Lin Shyh-Chang, E. D. Goodman and William F. Punch, A genetic algorithm approach to dynamic scheduling Job Shop Problem, *Proceedings of the Seventh International Conference on Genetic Algorithms*, 1997, pp 481-489.

[21]. M. C. Portmann, Scheduling Methodology: optimization and compu-search approaches, in P*lanning and scheduling of production systems*", Chapman &Hall, 1997.

[22]. M. Pinedo, *Scheduling – Theory, Algorithms and Systems*, 2nd edição, Prentice-Hall, 2002.

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

[24]. P. I. Cowling and M. Johansson, Using real-time information for effective dynamic scheduling, *European Journal of Operational Research*, vol. 139, no. 2, 2001, pp. 230-244

[25]. Paul P.M. Stoop, The complexity of scheduling in practice, *International Journal of Operations and Production management*, Vol. 16, No. 10, 1996, pp. 37-53.

[26]. Peter Brucker, *Scheduling Algorithms*, Springer, 3rd edition, New York, 2001.

[27]. R. W. Conway, W. L. Maxwell and L. W. Miller, *Theory of scheduling*, Addison-Wesley Publishing Company, 1967.

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

[29]. S Jain. and S. Meeran, Deterministic Job Shop scheduling: past, present and future, *European Journal of Operational Research*, nº113, 1999, pp.390-434.

[30]. S. C. Graves, A review of production scheduling, *Operations Research*, 29 (4), 1981, pp. 646-675.

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

[32]. Thomas E. Vollmann, William L. Berry and D. Clay Whybark, *Manufacturing Planning and Control Systems*, McGraw-Hill, New York, 1997.

[33]. V. J. Leon, S. D.Wu, and e R. H. Storer, Robustness measures and robust scheduling for job shops, *IIE Transactions*, 26(5), 1994, pp. 32–43.