17

MASDScheGATS - Scheduling System for Dynamic Manufacturing Environmemts

Ana Madureira, Joaquim Santos and Ivo Pereira
Computer Science Department, Institute of Engineering - Polytechnic of Porto
GECAD – Knowledge Engineering and Decision Support Research Group
Portugal

1. Introduction

This chapter addresses the resolution of scheduling in manufacturing systems subject to perturbations. The planning of Manufacturing Systems involves frequently the resolution of a huge amount and variety of combinatorial optimisation problems with an important impact on the performance of manufacturing organisations. Examples of those problems are the sequencing and scheduling problems in manufacturing management, routing and transportation, layout design and timetabling problems.

The classical optimisation methods are not enough for the efficient resolution of those problems or are developed for specific situations (Brucker, 2004) (Blazewicz et al., 2005) (Pinedo, 2005) (Madureira, 2003).

New organizational and technological paradigms are needed to reply to the modern manufacturing systems challenges. The traditional structure of manufacturing industries is constructed upon the three pillars of land, labour and capital. The challenge is to move towards a new structure, which can be described as innovating manufacturing, founded on knowledge and capital. Future manufacturing solutions must identify multiple perspectives and linkages between novel approaches to customization, customer response, logistics and maintenance. The current typically linear approach to research, development, design, construction and assembly will be replaced by simultaneous activity in all areas to satisfying global demand and shorten time-to-market (MANUFUTURE, 2004).

Multi-agent paradigm is emerging for the development of solutions to very hard distributed computational problems. This paradigm is based either on the activity of "intelligent" agents which perform complex functionalities or on the exploitation of a large number of simple agents that can produce an overall intelligent behaviour leading to the solution of alleged almost intractable problems. The multi-agent paradigm is often inspired by biological systems.

Meta-Heuristics (MH) form a class of powerful and practical solution techniques for tackling complex, large-scale combinatorial problems producing efficiently high-quality solutions. From the literature we can conclude that they are adequate for static problems. However, real scheduling problems are quite dynamic, considering the arrival of new orders, orders being cancelled, machine delays or faults, etc. Scheduling problem in dynamic environments have been investigated by a number of authors, see for example (Aytug et al., 2005) (Branke, 2000) (Cowling & Johansson, 2002) (Madureira et al., 2004).

Source: Multiagent Systems, Book edited by: Salman Ahmed and Mohd Noh Karsiti,
ISBN 978-3-902613-51-6, pp. 426, February 2009, I-Tech, Vienna, Austria

In this chapter we will model a Manufacturing System by means of Multi-Agent Systems (MAS) and Meta-Heuristics technologies, where each agent may represent a processing entity (machine). The objective of the system is to deal with the complex problem of Dynamic Scheduling in Manufacturing Systems. Our approach shows that a good global solution for a scheduling problem may emerge from a community of machine agents solving locally their schedules while cooperating with other machine agents that share some relations between the operations/jobs.
The remaining sections are organized as follows: Section 2 summarizes some works related on Meta-Heuristics and Multi-Agent Systems applications. In section 3 the scheduling problem under consideration is described. Section 4 presents the MASDScheGATS Systems and describes implemented mechanisms. Section 5 present a computational study and puts forward results discussion. Finally, the chapter presents some conclusions that were obtained from our model and puts forward some ideas for future opportunities of research and development work.

2. Related work

The planning of Manufacturing Systems involves frequently the resolution of a huge amount and variety of combinatorial optimisation problems with an important impact on the performance of manufacturing organisations. Examples of those problems are the sequencing and scheduling problems in manufacturing management, routing and transportation, layout design and timetabling problems.
Scheduling 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 constraints imposed on jobs and resources. So, a scheduling problem 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 time, release and due dates, precedence constraints and resource availability, can affect scheduling decisions (Brucker, 2004) (Blazewicz et al., 2005) (Pinedo, 2005).
Frequently classical optimization methods are not efficient enough for the resolution of Job-Shop Scheduling problems (Blazewicz et al., 2005) (Pinedo, 2005). In most cases they are good for solving only some specific and small size ones. The interest of new approaches, namely Meta-Heuristics such as Tabu Search, Simulated Annealing, and Genetic Algorithms, based on local search, is that they lead, in general, to good solutions in an efficient way, i.e. short computing time and small implementation effort.
Meta-Heuristics is the set of computing techniques inspired by biologically systems that are derived from nature. The distinction between Nature-inspired techniques and Meta-Heuristics is largely counterproductive. Although surface-level dissimilarities, the central themes underlying these two classes of heuristic are nearly identical, e.g., intensification versus diversification, mechanisms for escaping local optimum, intelligent design of selection/mutation/crossover operators, and the structure of the fitness landscape. The family of Meta-Heuristics includes, but it is not limited, to Tabu Search, Simulated Annealing, Adaptive Memory procedures, Scatter Search, Soft Computing, Evolutionary Methods, Ant Systems, Particle Swarm Optimization and their hybrids. For literature on this subject, see for example (Gonzalez,2007) (Xhafa & Abraham, 2008)

In last decades, there has been a significant level of research interest in Meta-Heuristics approaches for solving large real world scheduling problems, which are often complex, constrained and dynamic. 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 is the case with most algorithms for solving the so-called static scheduling problem for different setting of both single and multi-machine systems arrangements. This reality, motivated us to concentrate on tools, which could deal with such dynamic, disturbed scheduling problems, even though, due to the complexity of these problems, optimal solutions may not be possible to find.

Considering the complexity inherent to the manufacturing systems, dynamic scheduling is considered an excellent candidate for the application of agent-based technology. In many implementations of MAS systems for manufacturing scheduling, the agents model the resources of the system and the tasks scheduling is done in a distributed way by means of cooperation and coordination amongst agents (Lu & Yih, 2001) (Nwana et al., 1996) (Madureira et al., 2007). When responding to disturbances, the distributed nature of multi-agent systems can also be a benefit to the rescheduling algorithm by involving only the agents directly affected, without disturbing the rest of the community that can continue with their work.

Hybridization of intelligent systems is a promising research field of computational intelligence focusing on combinations of multiple approaches to develop the next generation of intelligent systems. An important stimulus to the investigations on Hybrid Intelligent Systems area is the awareness that combined approaches will be necessary if the remaining tough problems in artificial intelligence are to be solved. Meta-Heuristics, Bio-Inspired Techniques, Neural computing, Machine Learning, Fuzzy Logic Systems, Evolutionary Algorithms, Agent-based Methods, among others, have been established and shown their strength and drawbacks. Recently, hybrid intelligent systems are getting popular due to their capabilities in handling several real world complexities involving imprecision, uncertainty and vagueness (Boeres et al., 2003), (Madureira et al., 2004) (Bartz- Beielstein et al., 2007) (Hasan Kamrul et al., 2007).

3. Extended job shop scheduling problem definition

Real world scheduling problems have received a lot of attention in recent years. In this work we consider the resolution of realistic 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.

In practice, many scheduling problems include further restrictions and relaxation of others (Portmann, 1997). Thus, for example, precedence constraints among operations of the different jobs are common because, often, mainly in discrete manufacturing, products are made of several components that can be seen as different jobs whose manufacture 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 (concurrent or simultaneous processing).

Moreover, in practice, scheduling environment tends to be dynamic, i.e. new jobs arrive at unpredictable intervals, machines breakdown, jobs can be cancelled and due dates and processing times can change frequently.

The main elements of the Extended Job-Shop Scheduling Problem (EJSSP) problem could be modeled as shown in the following subsections.

3.1 Jobs

- A set of multi-operation jobs $J_1,\ldots,J_n$ has to be scheduled. d_j is the due date of job J_j. t_j is the initial processing time of job J_j . r_j is the release time of job J_j.
- The existence of operations on the same job, on different parts and components, processed simultaneously on different machines, followed by components assembly operations (multi-level jobs).
- The existence of different job release dates r_j and due dates d_j.
- The possibility of job priorities definition, reflecting the importance of satisfying their due dates, being similar to the weight assigned to jobs in scheduling theory.
- Precedence constraints among operations of the different jobs.
- The existence of operations on the same job, with different parts and components, processed simultaneously on different machines.
- New jobs can arrive at unpredictable intervals.
- Jobs can be cancelled.
- Changes in task attributes can occur: Processing times, date of deliver and priorities.

3.2 Operations

- Each operation O_{ijkl} is characterized by the index (i, j, k, l), where i defines the machine where the operation k of job j is processed and l the graph precedence operation level (level *1* correspond to initial operations, without precedents).
- Precedence constraints among operations of the different jobs.
- Each job J_j consists of one or more operations O_{ijkl}, where:
  - IO_{ijkl} is the time interval for starting operation $_{Oijkl}$
  - r_{ijkl} is the release time of operation $O_{ijk}l$
  - t_{ijkl} is the earliest time at which O_{ijkl} can start
  - T_{ijkl} is the latest time at which O_{ijkl} can start
  - p_{ijkl} is the processing time of the operation $O_{ijk}l$
  - C_{ijkl} is the k operation completion time from job j, level l on the machine i
- Each operation O_{ijkl} must be processed on one machine of the set M_i, where p_{ijkl} is the processing time of operation O_{ijkl} on machine M_i.
- The existence of operations on the same job, on different parts and components, processed simultaneously on different machines, followed by components assembly operations (multi-level jobs).

3.3 Machines

- The shop consists of a set of machines $M_1,\ldots,M_n$.
- A machine can process more than one operation of the same job (recirculation).
- The existence of alternative machines, identical or not..

In this work, we define a job as a manufacturing order for a final item that could be Simple or Complex). It may be Simple, like a part, requiring a set of operations to be processed. We define it as 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 consider the existence of two different types of tasks:

- Jobs with linear structure – where operations are sequentially processed, considering that an operation can be processed when its precedent has already been finished. Job-Shop benchmark tests referred on literature are of this type (Madureira, 2003).
- Jobs with concurrent operations – where operations of same task can be processed simultaneously. An operation can have more than one precedent operation and more than one succeeding operation. This category is common in Complex Final items.

Moreover, in practice, scheduling environment tend to be dynamic, i.e. new jobs arrive a unpredictable intervals, machines breakdown, jobs are cancelled and due dates and processing times change frequently. This non-basic JSSP (Portmann,), 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 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.

4. MASDScheGATS system

Distributed environment approaches are important in order to improve scheduling systems flexibility and capacity to react to unpredictable events. It is accepted that new generations of manufacturing facilities, with increasing specialization and integration, add more problematic challenges to scheduling systems. For that reason, issues like robustness, regeneration capacities and efficiency are currently critical elements in the design of manufacturing scheduling system and encouraged the development of new architectures and solutions, leveraging the MAS research results.

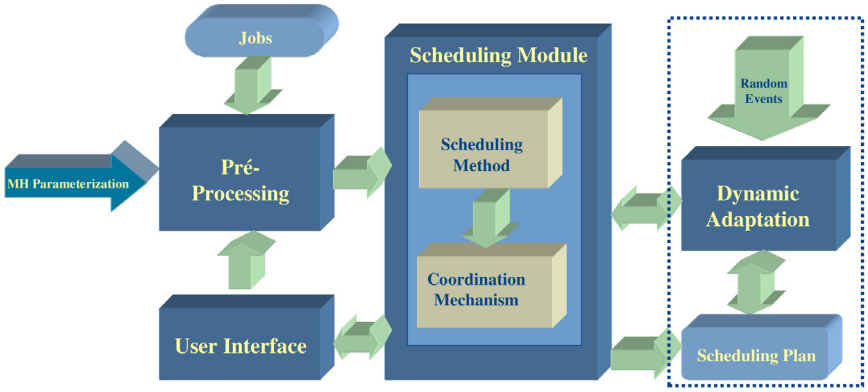

Fig. 1. MASDSCHEGATS System

4.1 MASDScheGATS scheduling system

It starts focusing on the solution of the dynamic deterministic EJSSP problems. For solving these we developed a framework, leading to a dynamic scheduling system (Fig. 1) having as

a fundamental scheduling tool, a hybrid scheduling system, with two main pieces of intelligence.
One such piece is a Hybrid Scheduling Module that could be a combination of Tabu Search and Genetic Algorithm based method and a mechanism for inter-machine activity coordination. The objective of this mechanism is to coordinate the operation of machines, taking into account the technological constraints of jobs, i.e. job operations precedence relationships, towards obtaining good schedules. The other piece is a dynamic adaptation module that includes mechanisms for neighbourhood/population regeneration under dynamic environments, increasing or decreasing it according new job arrivals or cancellations.

4.1.1 Hybrid scheduling module

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

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

Table 1. Notation

Initially, we start by decomposing the deterministic EJSSP problem into a series of deterministic Single Machine Scheduling Problems (SMSP). We assume the existence of different and known job release times rj, prior to which no processing of the job can be done and, also, job due dates dj. Based on these, release dates and due dates are determined for each SMSP and, subsequently, each such problem is solved independently by a TS or a GA (considering a self-parameterization issue). Afterwards, the solutions obtained for each SMSP are integrated to obtain a solution to the main EJSSP problem instance. The scheduling method is described in table 2.
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.

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 $1 \mid r_j \mid C_{max}$ based on information defined on Step1 and Step 2.
Step 4	Solve all SMSP $1 \mid r_j \mid C_{max}$ with those release times and due dates using TS or 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.

Table 2. Scheduling method algorithm

$$C_{ijk\text{-}1l} = C_{ijkl} - p_{ijkl} \tag{1}$$

This procedure begins with the last job operation and ends with the first. When an operation is precedent to more than one operation, i.e. there exists a multilevel structure, the completion due time is the lower value as defined on equation 2.

$$C_{ijkl\text{-}1} = min\{C_{ijkl} - p_{ijkl}\} \forall l > l\text{-}1 \tag{2}$$

The operations starting due time intervals $[t_{ijkl}, T_{ijkl}]$ are also defined considering the job release times and the operation processing times. The earliest starting time t_{ijkl} corresponds to the time instant from which the operation processing can be started. The latest starting time T_{ijkl} 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 $[t_{ijkl}, T_{ijkl}]$ 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 considered for defining completion and starting times.
The starting time interval (STI) for operations without precedents is defined as follows:

$$STI_{ijkl} = [r_{ijkl}, C_{ijkl} - p_{ijkl}] \tag{3}$$

The starting time interval of an operation with one precedent operation is defined by equation 4.

$$STI_{ijkl} = [t_{ijk\text{-}1l}, T_{ijk\text{-}1l}] + p_{ijk\text{-}1l} \tag{4}$$

The starting 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 (Equation 5).

$$STI_{ijkl} = [t_{ijK1L}+p_{ijK1L},\ T_{ijK1L}+p_{ijK1L}] \cap [t_{ijK2L}+p_{ijK2L},\ T_{ijK2L}+p_{ijK2L}] \cap ... \cap [t_{ijKnL}+p_{ijKnL},\ T_{ijKnL}+p_{ijKnL}] \text{ with } Kn<k \text{ and } L<l \quad (5)$$

Following this procedure it is easy to deal with situations were an operation has more than one precedent operation, i.e. there exists a multilevel task structure involving assembly operations. In this case the previous operations may be processed simultaneously, and therefore a relaxation of an underlining characteristic of the EJSSP problems is assumed. This situation is typical of real world manufacturing requirements. This means that a more generalized and realistic problem is dealt with the scheduling approach adopted in this work.
At this stage, only technological precedence constraints of operations and job due dates will be considered for defining completion and starting times (Fig.2).

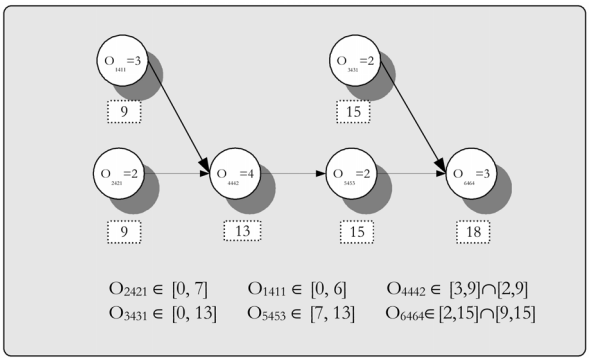

Fig. 2. Processing Precedence Graph with operation completion due times and starting times

The release date r_j correspond to the earliest starting times of each operation. The due date d_j correspond to the operation completion times. The notation r_j and d_j used at this point, considers that we are dealing with single machine problems.

Coordination Mechanism

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) 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, unchanged. The IMACM mechanism establishes the starting and the completion times for each operation. It ensures that the starting time for each operation is the higher of the two following values (Table 3):

Step 1	Start from initial operations, without precedents, which correspond the level 1 on the sequence for processing on the machine. At this level, the starting and the completion times are the same, i. e., they are equal to those defined by the scheduling algorithm on the previous phase.
Step 2	At level 2, we will have all the operations which immediately precedents (defined on the precedence graph) and on the machine sequence has been already scheduled.
Step 3	The process will be repeated until all the operations have been scheduled.

Table 3. Inter-Machine Activity Coordination Mechanism

- 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.

4.1.2 Dynamic adaptation module

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.

In the scheduling system for EJSSP, rescheduling is necessary due to two classes of events (Madureira, 2003):

- 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.

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 genes, population regeneration is done by updating the size of the population 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 process, the scheduling method could be applied and search for better solutions with the modified solution illustrated in Fig.1.

4.1.3 Meta-heuristics self-configuration properties
Generally, self-organization can be defined as the process by which systems tend to reach a particular objective with no external interference. All the mechanisms dictating its behaviour is internal to the system e.g. are autonomous. This field of research has received much attention trough Autonomic Computing paradigm (EMA, 2006).
In this paper we consider that Meta-Heuristics self-parameterization could permit a better adaptation to the dynamic situation being considered. The idea is that each agent adopts the MH (TS or GA) in accordance with the problem being solved: the method and/or parameters can change in run-time, the agents can use different MH according with problem characteristics (namely problem size, for smaller use GA for problems with more jobs using TS, considering efficiency constraints for example).
Meta-Heuristics can be adapted to deal with dynamic problems, reusing and changing solutions/populations in accordance with the dynamism. We will use the Dynamic Adaptation Mechanisms defined in (Madureira, 2000) for SMSP that includes a method for neighbourhood regeneration under dynamic environments, increasing or decreasing it according to new job arrivals or cancellations.

4.2 Hybrid multi-agent architecture
The work described in this chapter is a system where a community of distributed, autonomous, cooperating and asynchronously communicating machines tries to solve scheduling problems.
The proposed Team-Work based approach is rather different from the ones found in the literature; as we try to implement a system where each agent (Resource Agent) is responsible for optimize the scheduling of operations for one machine through TS or GA. This consider a specific kind of social interaction that is cooperative problem solving (CPS), where the group of agents work together to achieve a good solution for the problem.
Each Resource Agent must be able: to find an optimal or near optimal local solution trough Tabu Search meta-heuristics (or Genetic Algorithms); to deal with system dynamism (new jobs arriving, cancelled jobs, changing jobs attributes, etc); to change/adapt the parameters

of the basic algorithm according to the current situation; to switch from one Meta-Heuristic algorithm to another and to cooperate with other agents.

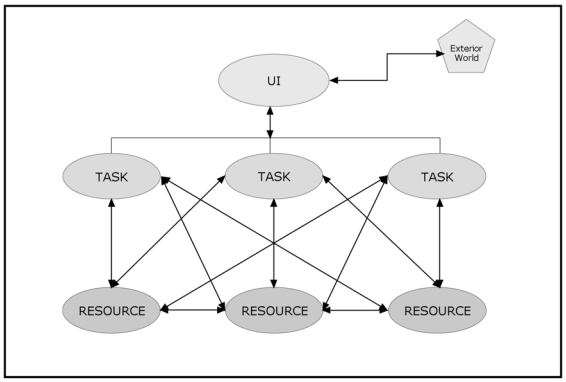

Fig. 3. MASDScheGATS System Architecture

The original Scheduling problem defined in section 3, is decomposed into a series of Single Machine Scheduling Problems (SMSP) (Madureira, 2001). The Resource Agents (which has a Meta-Heuristic associated) obtain local solutions and later cooperate in order to overcome inter-agent constraints and achieve a global schedule.

The proposed Team-Work architecture is based on three different types of agents (Fig. 5). In order to allow a seamless communication with the user, a User Interface (UI) Agent is implemented. This agent, apart from being responsible for the user interface, will generate the necessary Task Agents dynamically according to the number of tasks that comprise the scheduling problem and assign each task to the respective Task Agent(Fig. 3).

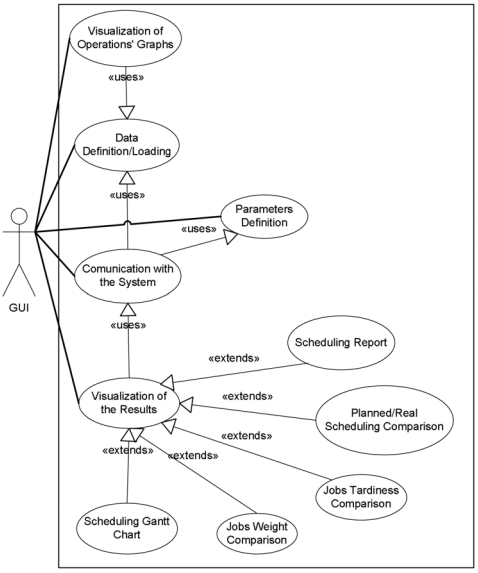

Fig. 4. User Interface Agent Functionalities

The Task Agent will process the necessary information about the job. That is to say that this agent will be responsible for the generation of the earliest and latest processing times, the verification of feasible schedules and identification of constraint conflicts on each job and the decision on which Machine Agent is responsible for solving a specific conflict.

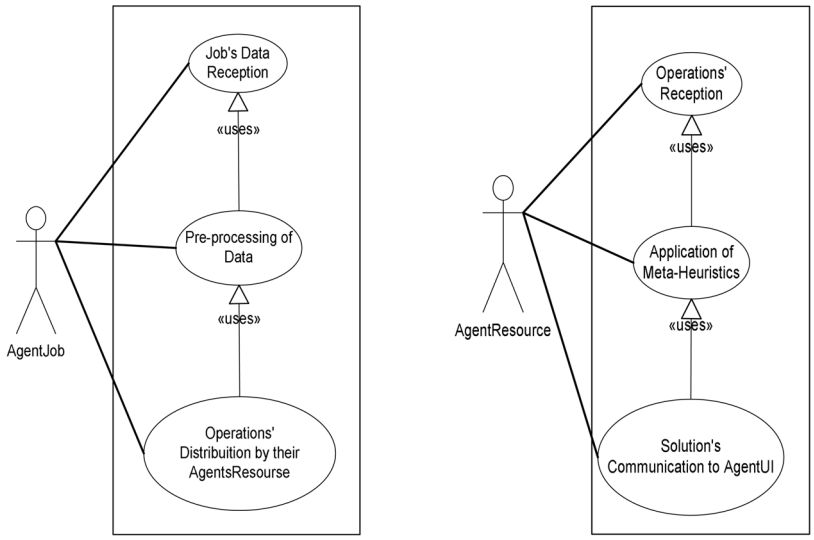

Fig. 5. Agent Job and Agent Resource functionalities

Finally, the Resource Agent is responsible for the scheduling of the operations that require processing in the machine supervised by the agent. This agent will implement metaheuristic and local search procedures in order to find best possible operation schedules and will communicate those solutions to the Task Agent for later feasibility check (Fig. 5).

5. Computational study

The proposed architecture was implemented using the Java Agent Development framework (JADE). Some computational tests were carried out to evaluate the performance of the referred scheduling systems under different manufacturing scenarios.
This section presents the results obtained by MASDScheGATS with TS and GA on the resolution of a set of academic instances of the Job-Shop problem (OR-library), considering the difficulties in finding test problems and computational results for EJSSP. The MASDScheGATS performance will be compared with MAPS - MultiAgent Production Planning System (Wellner and Dilger, 1999).

5.1. Tabu search parameterization

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, and evaluation of the solutions, such as neighbourhood size, tabu list length, tabu list attributes and stop criteria.

Details of the algorithm parameterization are briefly described as follows:

- Solution Representation - The solutions are encoded by the natural representation, where the schedule is described as a sequence of operations, i.e., each position represents an operation index.
- 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.
- Tabu list attributes and length - It is used a tabu list that stores the pairs of jobs involved on exchanging of positions (on the neighbourhood generation process), with length 4.
- As stopping criteria in the Tabu Search algorithm, we use a maximum of 100 iterations.

5.2 Genetic algorithms parameterization

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

Details of the algorithm parameterization are briefly described as follows:

- Solution Encoding - In this work, solutions are encoded by the natural representation (Davis, 1991). In this representation each gene represents a operation index. The geneposition in a chromosome represents the operation position in a sequence, defining, therefore, the operation processing order or priority. The number of genes in the chromosome represents the number of operation in a solution.
- Genetic Operators - Individuals, i.e. solutions, are randomly selected from the population and combined to produce descendants in the next generation. Depending on the problems to solve and their encoding, several crossover operators may be used namely one point, two points, uniform and order crossover(Davis, 1991). Here, we use the single point crossover operator with probability Pc=0.8. The single point crossover operator will be applied to M pairs of chromosomes randomly chosen, with M=N/2, where N is the size of the population. The mutation operator is applied with probability Pm=0.001, to prevent the lost of diversity. Thus, a single point in a chromosome is randomly selected, the current select resource, for the task, is replaced for another in the set of alternatives resources.
- Replacement Scheme - When creating a new population by crossover and mutation we must avoid loosing the best chromosomes or individuals. To achieve this, the replacement of the less fit individuals of the current population by offspring is based on elitism (Davis, 1991) . Thus, the best individuals, i.e. solutions, will survive into the next generation.
- As stopping criteria in the Genetic Algorithm we use a maximum of 100 generations.

5.3 Computational results

For our experiments, we consider some benchmark problems (OR-library). For release dates, we consider zero for all instances. Due dates are considered to be the optimal makespan

value. The obtained results with our method based on Tabu Search (TS) and Genetic Algorithms (GA) are compared with those obtained from using the MAPS system (Wellner & Dilger, 1999).
With a simple implementation of the TS and GA and a small parameterization effort it was possible to achieved good performance for most instances of the problem when compared with MAPS system (table 4).
It is important to refer that our scheduling framework, which here uses TS and GA, is flexible in several ways. It is prepared to use other Local Search Meta-Heuristics and to drive schedules based on practically any performance measure. Moreover, the framework is not restricted to a specific type of scheduling problem, as is the case with many methods.
One novel approach, rarely addressed in the literature, but very important in practice, is considered in our scheduling framework, namely that of being able to schedule jobs with complex processing structures, i.e. with both parallel processing of product component parts followed by their assembly at several stages.
Additionally the proposed coordination mechanism is of very simple implementation. We consider that with a more effective cooperation mechanism (that is on ongoing developing) it is possible to improve MASDScheGATS performance.

Instância	*n*	*m*	C_{max} Óptimo	MAPS - TCS			MAPS - LCS			MASDScheGATS T S					MASDScheGATS GA		
				Average	Best	Worst	Average	Best	Worst	F. Obj.	Average	Best	Worst		Average	Best	Worst
FT06	6	6	55	69.6	62	82	65.8	57	76	C_{max}	61.4	59	64		78.4	65	97
										WT	63.2	59	68		74	65	89
										L_{max}	99	99	99		74.6	66	96
FT10	10	10	930	1203.4	1164	1242	1277	1172	1373	C_{max}	1519	1408	1612		1476.4	1448	1540
										WT	1368.8	1319	1416		1627.2	1427	1816
										L_{max}	1634.8	1542	1739		1504.6	1399	1611
FT20	20	5	1165	1603.4	1513	1723	1470	1417	1503	C_{max}	1715	1707	1726		1761.4	1692	1952
										WT	1678.8	1664	1714		1696	1662	1755
										L_{max}	1784.6	1686	1889		1740.6	1714	1833
La01	10	5	666	782	725	832	861	822	891	C_{max}	812.2	764	858		919.8	810	1160
										WT	772.6	740	795		865	782	921
										L_{max}	1059	947	1122		1049.2	888	1164
La02	10	5	655	886.6	840	911	821.6	744	873	C_{max}	869.6	839	920		914.4	847	985
										WT	859.4	839	876		940.8	839	1117
										L_{max}	968.4	849	1099		979	871	1152
La03	10	5	597	778.8	726	823	793.6	716	858	C_{max}	834.6	776	916		888	804	997
										WT	794.2	749	830		849	809	894
										L_{max}	947.4	875	990		898.2	823	988
La04	10	5	590	835	801	894	786.2	724	819	C_{max}	865.8	798	924		904.8	746	979
										WT	875.2	767	989		846.6	773	950
										L_{max}	959.2	842	1085		915.4	822	1003

Table 4. Computational Results

6. Conclusions and future work

This chapter presented MASDScheGATS Scheduling System that assumes the combination of different Meta-Heuristics and Multi-Agent Systems potentialities. To solve the scheduling

problem, Machine Agents and Task Agents must interact and cooperate with other agents in order to obtain optimal or near-optimal global performances trough Meta-heuristics. The idea is that from local, autonomous and often conflicting agent's objectives, a global solution emerges from a community of machine agents solving locally their schedules while cooperating with other machine agents. Agents have to manage their internal behaviours and their relationships with other agents via cooperative negotiation in accordance with business policies defined by the user manager.

We believe that a new contribution for the resolution of more realistic scheduling problems (Extended Job Shop Problems) was described in this paper. The particularity of our approach is the procedure to schedule operations, as each machine will first find local optimal or near optimal solutions, succeeded by the interaction with other machines trough cooperation mechanism as a way to find an optimal or near-optimal global schedule.

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

Considering that natural evolution is a process of continuous adaptation, it seemed us appropriate to consider Genetic Algorithms and Tabu Search for tackling Dynamic Scheduling Problems. Thus, the MASDScheGATS based scheduling system developed adapts the resolution of the deterministic problem to the dynamic one in which changes may occur continually. A population/solution regenerating mechanism is put forward, for adapting the population/neighborhood of solutions, according to disturbances, to a new population, which increases or decreases according to new job arrivals or cancellations.

7. Acknowledgments

The authors would like to acknowledge FCT, FEDER, POCTI, POSI, POCI 2010 for their support to R&D Projects and the GECAD Unit.

8. References

Aytug, H., Lawley, M.A., McKay, K.,S. M. and Uzsoy, R. (2005). Executing production schedules in the face of uncertainties: A review and some future directions, European Journal of Operational Research,Volume 16 (1), pp. 86 - 10.

Bartz-Beielstein, Thomas, Blesa, M.J., Blum, C., Naujoks, B., Roli, A., Rudolph, G. &Sampels, M.(2007). Hybrid Metaheuristics. Proceedings of 4th International Workshop H. Dortmund, Germany, Lecture Notes in Computer Science. Vol. 4771. ISBN: 978-3-540-75513-5.

Blazewicz, Jacek., Ecker, Klaus H,.& Trystram, Denis(2005). Recent advances in scheduling in computer and manufacturing systems. European Journal of Operational Research, 164(3), 573 574.(1), pp.86-110.

Boeres, Cristina, Lima, Alexandre, Vinod, E.&Rebello, F.(2003). Hybrid Task Scheduling: Integrating Static and Dynamic Heuristics. 15th Symposium on Computer Architecture and High Performance Computing, 199.

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

Cowling, P. and Johansson, M. (2002). Real time information for effective dynamic scheduling. European J. of Operat. Research,139 (2), 230-244.

Davis, Lawrence(1991). Handbook of Genetic Algorithms, Van Nostrand Reinhold, New York. EMA (2006). Practical Autonomic Computing: Roadmap to Self Managing Technology – A White Paper Prepared for IBM , Enterprise Management Associates.

Gonzalez, Teofilo F. (2007). Handbook of Approximation Algorithms and Metaheuristics. Chapman & Hall/Crc Computer and Information Science Series

Hasan Kamrul, S.M.; Sarker, R.; Cornforth, D. (2007). Hybrid Genetic Algorithm for Solving Job-Shop Scheduling Problem, In: ICIS 2007 - 6th IEEE/ACIS International Conference pp:519-524.

Horling, Brian, Lesser: Victor. (2005). A Survey of Multi-Agent Organizational Paradigms, University of Massachusets.

Lu, T., Yih, Y. (2001). An Agent-Based Production Control Framework for Multiple-Line Collaborative Manufacturing, International Journal of Production Research, 39/10: pp.2155--2176.

Madureira, A., Ramos, C. and Silva, S. (2004). Toward Dynamic Scheduling Through Evolutionary Computing. In WSEAS Transactions on Systems. Issue 4. Volume 3, pp.1596--1604.

Madureira, A., Santos J., Gomes N. and Ramos C. (2007). Proposal of a Cooperation Mechanism for Team-Work Based Multi-Agent System in Dynamic Scheduling through Meta-Heuristics,In Proc. 2007 IEEE International Symposium on Assembly and Manufacturing (ISAM07), University of Michigan, Ann Arbor (USA), pp. 233--238, ISBN: 1-4244-0563-7.

Madureira, A. (2003). Meta-Heuristics Application to Scheduling in Dynamic Environments of Discrete Manufacturing. Ph.D. Thesis, University of Minho, Braga, Portugal, (in portuguese).

Madureira, Ana M., Ramos, Carlos and Silva, Sílvio. (2000). 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), pp 315—323.

Madureira, Ana M., Ramos, Carlos and Silva, Sílvio. (2001). 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.

MANUFUTURE High level group. (2004). MANUFUTURE A vision for 2020, Report of High level group, European Commission.

Nwana, H., Lee, L. and Jennings, N. (1996). Coordination in software agent systems, In: BT Technol J, Vol 14 No 4 October. OR-library, http://people.brunel.ac.uk/~mastjjb/jeb/info.html

Pinedo, M. (2005). Planning and Scheduling in Manufacturing and Services, Springer-Verlag, New York, ISBN:0-387-22198-0.

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

Wellner, Jörg and Dilger, Werner (1999). Job Shop Scheduling with Multiagents, In workshop planen und Konfigurieren.

Xhafa, Fatos, Abraham, Ajith.(2008). Metaheuristics for Scheduling in Industrial and Manufacturing Applications Series: Studies in Computational Intelligence , Vol. 128 (Eds.), ISBN: 978-3-540-78984-0.