META-HEURISTICS SELF-CONFIGURATION FOR SCHEDULING

Ana Madureira, GECAD – Knowledge Engineering and Decision Support Group, Institute of Engineering Polytechnic of Porto, Portugal, anamadur@dei.isep.ipp.pt
Nuno Fonseca, Institute of Engineering – Polytechnic of Porto, Portugal, nunofmf@dei.isep.ipp.pt
Ivo Pereira, GECAD – Knowledge Engineering and Decision Support Group, Institute of Engineering – Polytechnic of Porto, Portugal, i020541@dei.isep.ipp.pt

Keywords: Intelligent_decision_support_system; distributed_agent system; meta-heuristics; computational_intelligence; dynamic_scheduling;

INTRODUCTION

Scheduling resolution requires the intervention of highly skilled human problem-solvers. This is a very hard and challenging domain because current systems are becoming more and more complex, distributed, interconnected and subject to rapidly changing. A natural Autonomic Computing evolution in relation to Current Computing is to provide systems with Self-Managing ability with a minimum human interference. This paper addresses the resolution of complex scheduling problems using cooperative negotiation. A Multi-Agent Autonomic and Meta-heuristics based framework with self-configuring capabilities is proposed.

Considering that Autonomic Computing is a grand-challenge vision of the future in which computing systems will manage themselves in accordance with high-level objectives specified by humans[1], we pretend with the proposed system to give a meaningful contribution in the field of Autonomic Computing application for dynamic scheduling in Manufacturing Systems.

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 behavior leading to the solution of alleged almost intractable problems.

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.

Hybridization and combination of different approaches seems to be a promising research field of computational intelligence focusing on the development of the next generation of intelligent systems.

The remaining sections are organized as follows: initially issues and links for Autonomic Computing are presented. Then some related work on dynamic scheduling and Meta-heuristics applications for scheduling are summarized. After that the proposed AutoDynAgents System and implemented mechanisms are described. Finally, the paper presents some conclusions and puts forward some ideas for future work.

AUTONOMIC COMPUTING

Autonomic computing was first introduced by IBM in 2001, the purpose of autonomic computing is reducing human intervention in configuration tasks, due to complexity and size of computers systems the time spent in configuration is bigger, occupying human resources in tasks not profiting to enterprises and organizations, with autonomic computing human resources have more time to focus in tasks more suitable to the organization goals. Some benefits of autonomic computing could be [1], [2]:

- The ability to release resources to more important tasks than maintenance of computer systems.
- Guarantee of managing ability through all process of business.
- Collaboration of different information sources to solve problems, information is always available.
- Mass simulation, calculations made 24 hours a day seven days a week.

Self-CHOP

CHOP stands for Configure, Healing, Optimizing and Protect. These characteristics are probably the most important for autonomic computing systems. These characteristics give the possibility of the autonomic computing system to adjust to the surrounding environment (Fig.1).

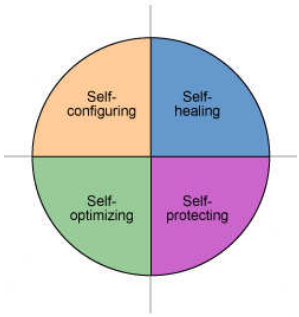

Fig. 1. Autonomic Computing attributes [2].

Self-configuration allows the system to adapt to the surrounding environment by adjusting components of the system. It may turn on components that were offline and vice-versa.

Self-optimization is concerned with tuning the several components in order to get the best performance of them. Or change priorities of tasks in scheduling.

Self-healing is responsible the resume the good functioning of the system after a crash, or part of it. It must be able to restore components in order to they resume their functioning.

Self-protect must be able to detect threats to the system, and take actions to prevent them, it must monitor the surrounding environment as well the system components searching for weaknesses in order prevent possible attacks or malfunctions [1], [2].

Autonomic computing elements

An autonomic computing system is composed by several elements. Each element in an autonomic computing architecture is responsible for certain tasks, the well functioning and synchronization of these elements are responsible for the success of the system [2]. It is possible to identify the following elements:

- Touch points are the components that allow the communication between the resources that are managed and the managers. The touch point is composed by sensors and actuators. The sensors allow receiving information from the managed resource, the actuators are responsible to send information to resource, by others words it controls the resource.
- Autonomic manager is the element responsible for the management for the resources. It must be able to gather information in an autonomic way and be able to analyze it in order to make any possible changes. The autonomic managers constantly perform a cycle usually called MAPE (Monitor, Analyze, Plan and Execute). This cycle is responsible for the constant monitoring and configurations of a resource.
- Knowledge sources are implementations of registers, data bases or other knowledge sources. In these knowledge sources exists information about the components of the autonomic system, policies to adopt, and even configurations that were made in order to look for similar solutions with similar symptoms.
- Manual manager is an implementation of an interface that allows human managing of the system. Usually is a console where user can make configurations and receive information from the autonomic managers.

All these components make the autonomic computing system. Autonomic managers can be managers of resources or orchestration managers, managers that manage others autonomic managers. All these components together form the autonomic computing architecture. IBM proposes an architecture that is only one possibility it is not a standard. There may be others architecture valid as this one.

DYNAMIC SCHEDULING

Dynamic scheduling can be defined as the constant maintenance of operations given to resources that are needed to maintain the scheduling up to date [3]. Problems with scheduling are clearly dynamic. There are too many variables to be able to elaborate an optimized plan, where we can predict all the possible failures. Some characteristics of dynamic scheduling are the ability to schedule in an environment where not all the information is available or it may change, be able to change plans already created according to information received in real time minimizing the disturbance of actual plans.

There are two types of events that are considered in dynamic scheduling. Events related with resources and events related with operations [4] [5]. Events related with resources can be considered machines breakdown, workers absenteeism, limit in production capacity and lack of correct tools to specific tasks. Events related with operations are operations that arrive to the system too late or too soon, new

operations not initial predicted, delivery dates changed, priorities changed.

In order to react to these disturbances that totally or partial invalid production plans, dynamic scheduling permits to resume the plans by making changes to existing plans or making news ones. Rescheduling takes less evaluation effort than changing the actual scheduling. When rescheduling is made a new plan is created. These guarantees that the plan is up to date, generating requires time, if there are many disturbances it is generated news plans spending much time in this process, the implementation of a new plan has costs. Changing the existing plans takes more effort because it analyzes existing plans not creating new ones from beginning. The cost of implementation these plans are less than implement a totally new one. If changes are minimal is faster to incorporate them. By changing parts of the plan, others parts may not be up to date [5].

Changing parts of plans may prove to be good to minimal changes, having a reduced cost. However it takes more processing effort and may leave parts of the plan not updated. Creating a new plan guarantee that all parts of the plan are updated and are working, it takes less processing effort, but the cost and time to implement a new plan is usually bigger than implement part of it, if constants changes are made the production system may become instable [4][5].

META-HEURISTICS APPLICATIONS FOR SCHEDULING

Meta-heuristics are the set of computing techniques inspired by biologically systems that are derived from nature. 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 [8].

The interest of this class of approaches is that they converge, in general, to satisfactory solutions in an effective and efficient way (computing time and implementation effort).

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.

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 [2][7][8].

AUTODYNAGENTS 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.

A natural Autonomic Computing evolution in relation to Current Computing is to provide systems with Self-Managing ability with a minimum human interference. Considering that AC is a grand-challenge vision of the future in which computing systems will manage themselves in accordance with high-level objectives specified by humans, we pretend with AUTODYNAGENTS (Autonomic Agents with Self-Managing Capabilities for Dynamic Scheduling Support in a Cooperative Manufacturing System) project to give a meaningful contribution in the field of Autonomic

Computing application for dynamic scheduling in Manufacturing Systems.

The concept of developing the next era of computing systems is driven by the convergence between Biological Systems and the Digital Computing Systems. AutoDynAgents is a project envisaging the use of Multi-Agent Systems paradigm for supporting dynamic and distributed scheduling in Manufacturing Systems with Autonomic properties, in order to reduce the complexity of managing systems and human interference.

As AutoDynAgents objectives we will try to make studies to prove the following assertions:

- Use Autonomic Computing to reproduce Life-like behavior in computation to explain, predict, reconstruct and deploy complex systems, with a minimum human interference.
- Multi-agent Systems are adequate to model and support dynamic and distributed scheduling with Cooperative Negotiation;
- Multi-agent paradigm is often inspired by biologically systems. Observing Manufacturing Systems like evolution-based social systems will be important in order to allow a better understanding and integration between the machinery and humans;
- Learning in Multi-agent Autonomic Systems is a challenging problem, so does Optimization. Optimization in such environments must deal with dynamism;
- Bio-Inspired Techniques can be adapted to deal with dynamic problems, reusing and changing solutions in accordance with system dynamism.

The final product of AutoDynAgents is an Autonomic Scheduling System in which communities of agents model a real manufacturing system subject to perturbations. Agents must be able to learn and manage their internal behaviour and their relationships with other autonomic agents, by cooperative negotiation in accordance with business policies defined by user manager. Cooperative Negotiation is quite important at this approach; we consider a Multi-dimensional Negotiation process depending upon the effort that the agents want to expend based on business.

AUTODYNAGENTS Architecture

The main purpose of AUTODYNAGENTS is to create a Multi-Agent system where each agent represents a resource (Machine Agents) in a Manufacturing System. Each Machine Agent must be able: to find an optimal or near optimal local solution trough an Meta-heuristics and to deal with system dynamism.

The Scheduling problem defined in [5], is decomposed into a series of Single Machine Scheduling Problems (SMSP). The Machine Agents (which has an Meta-heuristic associated) obtain local solutions and later cooperate in order to overcome inter-agent constraints and achieve a global plan solution.

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 (Machine Agent) is responsible for optimize the scheduling of operations for one machine through Tabu Search or Genetic Algorithms according to problem characteristics. It is based on three different types of agents. In order to allow a seamless communication with the user, a User Interface 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.

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.

Finally, Machine Agent is responsible for the scheduling of the operations that require processing in the machine supervised by the agent. This agent will implement meta-heuristic 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.

The architecture was implemented using the Java Agent Development framework (JADE).

Coordination Considerations

In a real manufacturing system a product is produced, step by step, passing on several machines. In each machine it will be performed at least one operation (job) of the process plan. In our approach we have one agent for each machine. However, if we join solutions obtained by our machine agents we will observe that, some times, they will not be feasible. In fact, if operation Op_1 (in machine m_1) precedes operation Op_2 (in machine m_2) and Op_2 precedes Op_3 (in machine m_3) in a manufacturing process,

it is not guaranteed that the initial time for Op_2 in m_2 will be after the end of Op_1 in m_1 nor that the end of Op_2 in m_2 will be before the start of Op_3 in m_3.

Two possible approaches, to deal with this problem, could be used. In the first, the AUTODYNAGENTS system waits for the solutions obtained by the machine agents and then apply a repair mechanism to shift some operations in the generated schedules till a feasible solution is obtained (Repair Approach) [5][9]. In the second, a coordination mechanism is established between related agents in the process, in order to interact with each other to pursuit common objective trough cooperation. These coordination mechanisms are prepared to accept agents subjected to dynamism (new jobs arriving, cancelled jobs, changing jobs attributes). The latter approach is the one implemented in the proposed system.

Self-Managing Mechanisms for Autonomic Agents

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 are internal to the system e.g. are autonomous. This field of research has received much attention trough Autonomic Computing paradigm [2] [10].

We envisage to define Self-Managing mechanisms for a Cooperative Scheduling System considering that AutoDynAgents must be able to perform scheduling in highly dynamic environments where there is incomplete information and changes often occur; modify previously formed schedules considering 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 flexibility to react robustly to any disruption in an efficient and timely manner. We intend to define the following Self-Managing mechanisms:

- **Self-Configuring** - enable agents to adapt to changing conditions by changing their own configurations, allowing the addition and removal of resources without service disruption. Machine Agents will be prepared to handle dynamism by adapting the solutions to external perturbations.
- **Self-Optimizing** - ability of the agent to monitor its state and performance and proactively tune itself to respond to environmental stimuli. Each machine agent adopt and provides self-parameterization of the solving method in accordance with the problem being solved (parameters can change in run-time).
- **Self-Healing** – giving agents the capacity to diagnose deviations from normal conditions and take proactively action to normalize them and avoid service disruptions.

Self-Configuring Mechanism

In this work each resource will have an autonomic manager. These autonomic managers will be responsible for monitoring, and send instructions to the resource, by doing this it can detect changes and disturbances to the original plan. By detecting the changes, the plan can be corrected. Information about the scheduling will be stores in a knowledge source, there will also exists information about older scheduling and changes effectuated in order to solve problems previously occurred. These older scheduling and special the changes previously made will be important in order to find a solution that incorporates the disturbances that happen. By searching in the knowledge source similar disturbances and their solutions, it can be adapted to current disturbance in order to adapt it and rearrange the plan.

Rescheduling is necessary due to two classes of events [5]: **Partial events** imply variability in jobs/operations attributes such as processing times, due dates or release times; and **Total events** imply variability in neighbourhood/population structure, resulting from new job arrivals, job cancellations, machines breakdown, etc.

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.

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 redefinition of 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.

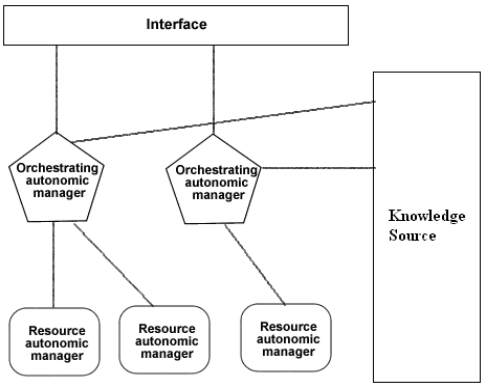

Fig. 2. Autodynagents project architecture.

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.

There will also exist orchestrating autonomic managers. These autonomic managers are at a superior level, they permit to resources autonomic managers to communicate with each others. They serve as an intermediary. They also have the access to knowledge source, in this way there will be a more controlled access to information. Orchestrating managers will also provide information to users interfaces. The number of orchestrating managers may vary depending on the number of resource autonomic managers, if there are too many resource autonomic managers they should be divided to several orchestrating autonomic managers to have more efficient response times(Fig.2).

CONCLUSIONS AND FUTURE WORK

We believe that a new contribution for the resolution of more realistic scheduling 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 through cooperation mechanism as a way to find an optimal or near-optimal global schedule.

As result of this project we expect to prove some ideas for which we are now claiming. From these we refer the use of Autonomic Computing to reproduce Life-like behaviour in computation to explain, predict, reconstruct and deploy complex systems, with a minimum human interference.

ACKNOWLEDGEMENTS

The authors would like to acknowledge FCT, FEDER, POCTI, POCI for their support to R&D Projects and GECAD - Knowledge Engineering and Decision Support Group Unit.

References

[1] IBM; Autonomic Computing White Paper- An architectural blueprint for autonomic computing; 2006.
[2] EMA. Practical Autonomic Computing: Roadmap to Self Managing Technology - A White Paper Prepared for IBM, Ent. Manag. Associates; 2006.
[3] M. Selim Akturk, Elif Gorgulu; Theory and Methodology Match-up Scheduling under a machine breakdown; European Journal of Operational Research 112, 81-97; 1999.
[4] Aytug, H., Lawley, M. A, McKay, K., Mohan, S.& Uzsoy, R., Executing production schedules in the face of uncertainties: A review and some future directions. European Journal of Operational Research, Volume 16 (1), 86-110, 2005.
[5] Madureira, A., Aplicação de Meta–Heurísticas ao Problema de Escalonamento em Ambiente Dinâmico de Produção Discreta. 2003 (in portuguese).
[6] Teofilo, F. Gonzalez, Handbook of Approximation Algorithms and Metaheuristics, Chapman&Hall/Crc Computer and Information Science Series, 2007.Parashar, Manish, Hriri, Salim; Autonomic Computing, Concepts, infrastructures and applications; CRC Press; 2006.
[7] Madureira, A., Santos, J., Gomes,N., Hybrid Multi-Agent System for Cooperative Dynamic Scheduling through Meta-Heuristics, 6th International Conference on Intelligent System Design and Applications, Rio de Janeiro (Brasil), 2007, pp.9-14, ISBN: 0-7695-2976-3.]
[8] Luck, M., McBurney, P., Shehory, O., Willmoth, S. , Agent Technology: Computing as Interaction. A Roadmap for Agent-Based Computing, AgentLink III, 2005.
[9] Madureira, A., Santos, J., Gomes, N., Ramos,C., Proposal of a Cooperation Mechanism for Team-Work Based Multi-Agent System in Dynamic Scheduling through Meta-Heuristics, 2007 IEEE Int. Symposium on Assembly and Manufacturing(ISAM07), Ann Arbor(USA), 2007, pp. 233-238, ISBN: 1-4244-0563-7.

[10] Monostoria L., Vánczaa, J., and S.R.T Kumara, Agent-Based Systems for Manufacturing , CIRP Annals - Manufacturing Technology, Volume 55, Issue 2, Pages 697-720, 2006.