Applied Soft Computing 13 (2013) 1419–1432

Contents lists available at SciVerse ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

# Self-Optimization module for Scheduling using Case-based Reasoning

I. Pereira*, A. Madureira

*GECAD, Knowledge Engineering and Decision Support Research Center, Institute of Engineering – Polytechnic of Porto (ISEP/IPP), Porto, Portugal*

ARTICLE INFO

*Article history:*
Received 20 December 2011
Received in revised form 27 January 2012
Accepted 3 February 2012
Available online 3 March 2012

*Keywords:*
Autonomic Computing
Case-based Reasoning
Learning
Meta-heuristics
Multi-Agent Systems
Scheduling

ABSTRACT

Metaheuristics performance is highly dependent of the respective parameters which need to be tuned. Parameter tuning may allow a larger flexibility and robustness but requires a careful initialization. The process of defining which parameters setting should be used is not obvious. The values for parameters depend mainly on the problem, the instance to be solved, the search time available to spend in solving the problem, and the required quality of solution.

This paper presents a learning module proposal for an autonomous parameterization of Metaheuristics, integrated on a Multi-Agent System for the resolution of Dynamic Scheduling problems.

The proposed learning module is inspired on Autonomic Computing Self-Optimization concept, defining that systems must continuously and proactively improve their performance. For the learning implementation it is used Case-based Reasoning, which uses previous similar data to solve new cases. In the use of Case-based Reasoning it is assumed that similar cases have similar solutions.

After a literature review on topics used, both AutoDynAgents system and Self-Optimization module are described. Finally, a computational study is presented where the proposed module is evaluated, obtained results are compared with previous ones, some conclusions are reached, and some future work is referred.

It is expected that this proposal can be a great contribution for the self-parameterization of Metaheuristics and for the resolution of scheduling problems on dynamic environments.

## 1. Introduction

The exponential growth of computational capacity together with the appearance of network communication, mainly Internet, led companies to invest significantly in infrastructures and computational applications. These systems are subject to failures, dynamism, overloads, and others, due to exponential growth of its complexity. Generally, organizations invest more in maintenance than in development. With the prediction of a “breaking point”, in October 2001, Paul Horn, vice-president of IBM Research, gave some visibility to this problem launching an IBM challenge named Autonomic Computing (AC) [1].

An autonomous system is viewed as a system with the ability of self-management. The objective of these systems is to free users from repetitive tasks, enabling a full-time execution, and providing enough intelligence to be possible to take decisions in order to reach an objective [2].

In recent years, there has been a growing interest in decentralized approaches for the resolution of complex real world problems, like Scheduling, as the number of proposed solutions and successful implementations is increasing. It is possible to highlight Multi-Agent Systems (MAS), which concern with behaviors' coordination of a set of agents, in order to share knowledge, abilities, and objectives, in the resolution of complex problems. Due to the exponential growing of system's complexity, it is important that MAS become more autonomous to deal with dynamism, overloads and failures recovery.

MAS typically operate in open, complex, dynamic, and unpredictable environments. It is not possible to predict every situation that an agent can find so it is necessary that agents have the ability to adapt to new situations. Since intelligence implies a certain degree of autonomy, requiring the capacity of taking decisions autonomously, agents must have the appropriate tools to take such decisions. Therefore learning becomes, many times, indispensable [3].

Biological and natural processes have been a source of inspiration for computer science and information technology, which led to the development of techniques that converge, in general, to satisfactory solutions in an effective and efficient way, generally named Meta-heuristics (MH) [4]. MH have often been shown to be effective for difficult combinatorial optimization problems appearing in several industrial, economic, and scientific domains [5–10].

Since MH parameterization revealed to be a hard task, requiring expertise knowledge about the application domain and which

* Corresponding author at: GECAD, Dr. António Bernardino de Almeida, 431, 4200-072 Porto, Portugal. Tel.: +351 22 8340500; fax: +351 22 8321159.
*E-mail address:* iasp@isep.ipp.pt (I. Pereira).
*URL:* http://www.gecad.isep.ipp.pt/ (I. Pereira).

1568-4946/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.asoc.2012.02.009

techniques and parameters should be used, it is important to turn systems capable of autonomously parameterize themselves. To validate this self-parameterization, we use learning about past experience, with Case-based Reasoning (CBR) revealing to be a promising approach. Using CBR, systems can remember past effectively solved cases and autonomously decide which MH and parameters to use for the resolution of a new similar problem [11].

With this, we propose using MAS, MH, AC, together with learning capabilities, to improve the resolution of the Dynamic Scheduling problem, with the objective of maximizing the quality of solutions, minimizing the computational time spent, and also reducing the human intervention.

The remaining sections are organized as follows. Section 2 describes the Dynamic Scheduling problem and Section 3 presents a literature review of AC, MH, MAS, and CBR applications for Scheduling. In Section 4 AutoDynAgents system is presented and the Self-Optimizing Module is described in Section 5, which was integrated in AutoDynAgents. The computational study is presented in Section 6 and, finally, the paper presents some conclusions and puts forward some ideas for future work.

### 1.1. Dynamic Scheduling problem

Scheduling problems arise in a diverse set of domains, ranging from manufacturing to transports, hospitals settings, computer and space environments, amongst others, most of them characterized by a great amount of uncertainty that leads to significant dynamism in the system. Such dynamic scheduling is getting increased attention between researchers and practitioners [12,13].

The dynamism can arise from requirements of a new user or from the evolution of the external environment. In a more general way, dynamic changes can be seen as a set of inserted and cancelled constraints.

Dynamic optimization problems environments are often impossible to avoid in practice. For these, the optimization algorithm must continuously find the optimum in dynamic environments, or find a robust solution that is capable of operate optimally in the occurrence of perturbations, instead of simply locate the global optimum solution [14].

In spite of all research made so far, the scheduling problem is still known to be NP-complete, even for static environments [12]. This fact presents serious challenges to conventional algorithms and incites researchers to explore new directions. Multi-Agent technology has been considered an important approach for developing industrial distributed systems [15].

The scheduling problem treated in this work has some major extensions and differences compared to the classic Job-Shop Scheduling Problem (JSSP), named Extended Job-Shop Scheduling Problem (EJSSP), proposed by Madureira [16]. The main elements of EJSSP problem can be modeled as:

(1) a *set* of multi-operation jobs $J_1, \ldots, J_n$ has to be scheduled on a set of machines $M_1, \ldots, M_n$.
(2) $d_j$ represents the due date of job $J_j$.
(3) $t_j$ represents the initial processing time of job $J_j$.
(4) $r_j$ represents 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).

Furthermore EJSSP should meet the following restricted conditions:

(a) The existence of different job release dates $r_j$ and due dates $d_j$.
(b) 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.
(c) Precedence constraints among operations of the different jobs.
(d) New jobs can arrive at unpredictable intervals. Jobs can be cancelled. Changes in task attributes can occur: processing times, due dates release dates and priorities.
(e) 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 $M_i$.
(f) A machine can process more than one operation of the same job (recirculation).
(g) The existence of alternative machines, identical or not.

The methods for the resolution of NP-hard combinatorial optimization problems, where Scheduling is included, can be divided in exact and approximation algorithms [4,13]. In the first, an exhaustive solutions space search is made and the optimal solution obtaining is ensured. The second type of algorithms has the objective to find a good solution in an acceptable period of time, but do not guarantee the optimal solution. MH are a more representative class of approximation algorithms, used in this work.

## 2. Literature review

This section will discuss some efforts related to literature on applications of different paradigms to the Dynamic Scheduling problem resolution, such as AC, MH, MAS, and CBR.

### 2.1. Autonomic Computing applications for Scheduling

Autonomic Computing (AC) is an IBM Grand Challenge proposed in 2001 by Paul Horn, Senior Vice-President of IBM Research [1]. Horn argues that the Information Technology (IT) industry is on constant expansion and will soon reach its breaking point. This can happen because massive data centers are built in organic, ad hoc ways, resulting in a heterogeneous composition where maintenance costs in terms of qualified staff, time and capital can soon exceed corporate capabilities [2].

AC represents a new computation paradigm in which IT systems have managing mechanisms embedded in the application, with the objective to automate the management. So, applications must be able to adapt, accommodate and protect themselves to the changes in the environment and in the objectives.

AC proposes a broad new field of research related to the automation of IT management processes, drawing inspiration from the human autonomous nervous system, since many essential functions to the welfare and regulation of living beings are not consciously triggered, e.g., heart beating, digestive system, etc. However, without an autonomous system to manage these mechanisms, the body would stop working or the concentration in other life aspects would not be possible.

From its inception, the AC concept involves four properties, generally referred as self-* properties, in which research efforts may be categorized [2,17]:

- **Self-configuration** deals with installation, configuration and integration of IT systems. When a new component is intended to insert the system, it is autonomously incorporated, like a new cell incorporates the human body, or even when a person incorporates a population. The installation procedures work by gathering information about the environment, figuring out the dependencies among needed tasks and also optimizing performance measures, and finally executing the tasks to perform changes.