

# Tuning Meta-heuristics Using Multi-agent Learning in a Scheduling System

Ivo Pereira<sup>1</sup>, Ana Madureira<sup>1</sup>, P. B. de Moura Oliveira<sup>2</sup>, and Ajith Abraham<sup>3,4</sup>

<sup>1</sup>GECAD - Knowledge Engineering and Decision Support Group, Institute of Engineering – Polytechnic of Porto, Porto, Portugal, {iaspe, amd}@isep.ipp.pt

<sup>2</sup>INESC TEC - INESC Technology and Science, Department of Engineering, University of Trás-os-Montes e Alto Douro, Vila Real, Portugal, oliveira@utad.pt

<sup>3</sup>Machine Intelligence Research Labs (MIR Labs). Scientific Network for Innovation and Research Excellence, Auburn, USA

<sup>4</sup>IT4Innovations - Center of Excellence, VSB-Technical University of Ostrava, Czech Republic  
ajith.abraham@ieee.org

**Abstract.** In complexity theory, scheduling problem is considered as a NP-complete combinatorial optimization problem. Since Multi-Agent Systems manage complex, dynamic and unpredictable environments, in this work they are used to model a scheduling system subject to perturbations. Meta-heuristics proved to be very useful in the resolution of NP-complete problems. However, these techniques require extensive parameter tuning, which is a very hard and time-consuming task to perform. Based on Multi-Agent Learning concepts, this article propose a Case-based Reasoning module in order to solve the parameter-tuning problem in a Multi-Agent Scheduling System. A computational study is performed in order to evaluate the proposed CBR module performance.

**Keywords:** Case-based Reasoning, Learning, Metaheuristics, Parameter tuning, Scheduling

## 1 Introduction

Recently, the interest in decentralized approaches for the resolution of complex real world problems, like Scheduling, is gaining much attention due to its wide applications. Several of these approaches belong to Distributed Systems research area, where a number of entities work together to solve problems in a cooperative way. In this area, it is possible to emphasize Multi-Agent Systems (MAS), concerning the coordination of agent's behaviors 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.

Multi-Agent Systems typically operate in open, complex, dynamic, and unpredictable environments. Therefore, learning becomes crucial. Learning is a relevant area from Artificial Intelligence (AI) as from human intelligence. Plaza et al. [1] defined learning as “the process of improving individual performance, precision (or quality)

of solutions, efficiency (or speed) of finding solutions and scope of solvable problems". Although this definition is very useful, it is a severe and constricted view of learning. In a more general way, it is possible to define learning as the acquisition of new knowledge or updating existing knowledge.

As per Alonso et al. [2], intelligence implies a certain degree of autonomy, which requires the capacity of taking decisions autonomously. Thus, agents should have the appropriate tools to take such decisions. In dynamic domains 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. This is especially true in MAS, where in many cases the global behavior emerges instead of being pre-defined. Consequently, learning is a crucial component of autonomy and pro-activeness, which must be a study target of agents and MAS [2].

The adaptation of ideas from different research areas, inspired from nature, led to the development of Meta-Heuristics (MH), which are techniques aiming to solve complex generic problems of combinatorial optimization, in which the scheduling problem is included.

Meta-heuristics are very useful to achieve good solutions in reasonable execution times. Sometimes they even obtain optimal solutions. However, to achieve near-optimal solutions, it is required the appropriate tuning of parameters.

Parameter tuning of MH has a great influence in the effectiveness and efficiency of the search process. The definition of the parameters is not obvious because they depend on the problem and the time that the user has to solve the problem [3]. Therefore, this paper proposes the use of a learning mechanism in order to perform the MH parameter tuning in the resolution of the scheduling problem. Case-based Reasoning (CBR) was chosen since it assumes that similar problems may require similar solutions.

As a MAS is used to model a dynamic scheduling system, with agents representing both tasks and resources, it is proposed that each resource agent have their own CBR module, allowing a multi-apprentice learning. With this type of learning, agents learn how to perform their own single-machine scheduling problem.

The proposed system adopts and provides parameter tuning of MH through CBR, with the possibility that parameters can change in run-time. According to the current situation being treated, the system must be able to define which MH should be used and define the respective parameters. It is even possible to change from one MH to another, according to current state and previous information, through learning and experience.

The paper is organized as follows: Section 2 describes the scheduling problem and Section 3 describes MH. Section 4 describes the Multi-Agent Learning and CBR is explained along section 5. In Section 6, the implemented MAS is explained with CBR integrated. A computational study is presented in Sections 7 and 8 and some conclusions and future work are presented.

## 2 Scheduling Problem

Scheduling problems are present in a large set of domains, from transports to manufacturing, computer environments, hospital settings, etc., most of them characterized by a vast amount of uncertainty leading to a considerable dynamism in the systems. Thereby, dynamic scheduling is getting an increased attention by researchers and practitioners [4][5].

The scheduling problem treated in this paper is named Extended Job-Shop Scheduling Problem (EJSSP), described by A. Madureira in 2003 [6], and has some major extensions and differences when compared to the classic JSSP, in order to better represent reality.

JSSP has a set of tasks processing in a set of machines, with each task following an ordered list of operations, each one characterized by the respective processing time and machine where is processed.

The main elements of JSSP problem are:

- a set of multi-operation jobs  $J_1, \dots, J_n$  to be scheduled on a set of machines  $M_1, \dots, M_n$
- $d_j$  represents the due date of job  $J_j$
- $t_j$  represents the initial processing time of job  $J_j$
- $r_j$  represents the release time of job  $J_j$

EJSSP problems consist in JSSP problems with additional restrictions, to better represent reality. Some of those restrictions are:

- Different release and due dates for each task
- Different priorities for each task
- Possibility that not every machine is used for all tasks
- A task can have more than one operation being processed in the same machine
- Two or more operations of the same task can be processed simultaneously
- Possibility of existence of alternatives machines, identical or not, for processing the operations

In this work, we define a job as a manufacturing order for a final product that can be Simple or Complex. It may be Simple Product, like a part, requiring a set of operations to be processed. Complex Products, require processing of several operations on a number of parts followed by assembly operations at several stages.

For a better understanding the EJSSP, the reader may consider the following example: a certain company produces some complex products. During the production process, new orders may arrive, some orders can be canceled, and some orders may be changed (due dates priorities, etc.). With the EJSSP modeling, it is possible to specify different priorities for each order, change the due dates, etc. But the most important contribution of this modeling strategy is that it is possible to have: i) many machines producing the same pieces; ii) more than one piece of each job processed in the same machine; iii) two or more pieces of the same job being processed at the same

time. The three last aforementioned aspects are an important enhancement, not considered in the classic JSSP definition.

Scheduling problems belongs to the NP-complete class [4]. Methods for their resolution can be categorized in exact and approximation algorithms [5][6]. In the former, an exhaustive solutions space search is made and it is ensured the optimal solution, but on the other hand they are very time consuming. The latter includes heuristics and MH and do not guarantee the optimal solution since they have the objective to find a good solution in an acceptable amount of time. For this reason, they are used in this work integrated with MAS.

### **3 Metaheuristics Parameter Tuning**

Meta-heuristics have gained popularity over the past two decades in the resolution of many types of real-life problems, including Scheduling, since they allow the resolution of large dimension problems by obtaining satisfactory solutions in satisfactory execution times. The term "meta-heuristic" was introduced by Fred W. Glover in 1986 [7].

These techniques have the objective of guiding and improving the search process in a way to overcome local optimal solutions, which represent a limitation of Local Search algorithm, and obtain solutions with satisfactory quality, very close to the optimal solution, in reasonable execution times [3][6].

Meta-heuristics consist on iterative or recursive methods with the objective of obtaining solutions the closest as possible to the global optimum for a given problem. Assuming that all solutions are interrelated, it is possible to obtain the set of solutions for a given problem.

In this work some of the most well known MH are used [3][6][8]: Tabu Search, Genetic Algorithms, Simulated Annealing, Ant Colony Optimization, and Particle Swarm Optimization.

Tabu Search (TS) was introduced by Fred Glover [7] and consists in a Local Search algorithm with the main objective to escape from local minimum. It uses a tabu list to memorize the last solutions trajectory, prohibiting the moves to solutions already visited in a short term memory.

Simulated Annealing (SA) was proposed by Kirkpatrick et al. [9] and Cerny [10]. It has connections to thermodynamics and metallurgy [11], and the original motivation is based on the process in which molten metal is slowly cooled, with a tendency to solidify in a structure of minimum energy. This MH has a statistical basis and is based on allowing the movement to a worst solution, with the objective to escape from local optimum.

In beginning of 1970, John Holland, together with his students and colleagues, developed research and studies based on natural selection of species, reaching a formal model designated by Genetic Algorithms (GA) [12]. In the 1980s, David Goldberg, Holland's student, implemented and published the first well successful applications of these algorithms [13].

Proposed by Dorigo et al. [14], Ant Colony Optimization (ACO) is based on a behavior that allows ants to find the shortest path between a food source and the respective colony [7]. Ants deposit in the ground a substance named pheromone, and when choosing a path, they opt, with greater probability, by the one that has more quantity of pheromone, which corresponds to the path followed by the higher number of ants.

Particle Swarm Optimization (PSO) was developed by James Kennedy and Russell Eberhart [15] with the objective to simulate a simplified social system. The basic idea was to demonstrate the behavior that flocks of birds or schools of fishes assume in their random local trajectories, but globally determined. Flocks of birds or schools of fishes make coordinated and synchronized movements as a way of finding food or as a mechanism of self-defense.

As mentioned, MH can be used for the resolution of many kinds of problems. However, to solve a specific problem it is necessary to choose a MH, which is considered a difficult task, requiring a study about the problem type and about the chosen technique. Furthermore it is also necessary to define the respective parameters.

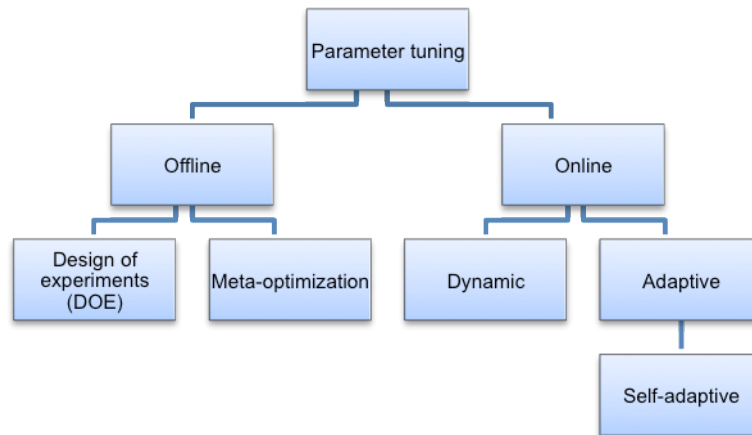
The parameter tuning of MH allows greater flexibility and robustness but requires a careful initialization, since parameters have a great influence on the efficiency and effectiveness of the search [3].

El-Ghazali Talbi [3] has identified two different approaches for MH parameter tuning: offline and online (Fig. 1). In offline tuning, the values for the parameters are defined before the execution of MH. In online tuning the parameters are controlled and updated in a dynamic or adaptive way, throughout the execution of MH.

Usually, when using MH, practitioners tune one parameter at a time and its optimal value is determined in an empiric way. However, this tuning strategy cannot guarantee the optimal parameter configuration.

To overcome this problem, design of experiments (DOE) [16] is used. Nevertheless, before using DOE it is necessary to take into account diverse factors which represent the parameters variation and the different values for each parameter (that can be quantitative and qualitative).

The greatest disadvantage about using DOE is the high computational cost when there is a large number of parameters, and when the domains of the respective values are also high since it is necessary to perform a large number of experiments [17]. To overcome this disadvantage, it is possible to use, e.g., racing algorithms [18][19].



**Fig. 1.** Parameter tuning [3]

On the other hand, in Meta-optimization, (meta) heuristics can be used to find the optimal parameters like in optimization problems. Meta-optimization consists in two levels: meta-level and base level. In the meta-level, solutions represent the parameters to optimize, such as the size of the tabu list in Tabu Search, the cooling rate in Simulated Annealing, the crossover and mutation rates of a Genetic Algorithm, etc. At this level, the objective function of a solution is the best solution found (or another performance indicator) by the MH with the specified parameters. Thus, for each solution in the meta-level there is an independent MH in the base level.

The drawback of offline approaches is the high computational cost, especially if used for each instance of the problem. In fact, the optimum values for the parameters depend on the problem to solve and on the different instances (e.g. larger instances may require different parameter settings). Thus, to increase the effectiveness and robustness of offline approaches, these should be applied to all instances (or class of instances) of a given problem [3].

Online approaches arise in order to try to achieve better results and they can be divided in dynamic and adaptive approaches [3]. In dynamic approaches, changes in parameter values are performed at random or deterministic ways, without taking into account the search process. In adaptive approaches, parameter values change according to the search process through the use of memory. A subclass, often used in the evolutionary computation community, is identified as self-adaptive, consisting in parameters evolution during the search. Therefore, the parameters are encoded and are subject to change, such as solutions to the problem.

This problem of finding the most suitable parameter configuration is related with the notion of hyper-heuristic [20][21][22]. Hyper-heuristic methods try to automate the process of selecting, combining or adapting several heuristics (or MH) in order to solve problems in an efficient manner.

The term “hyper-heuristic” was introduced in 1997 [23] to describe a procedure combining different AI methods. This idea became pioneer in the 1960s with the combination of scheduling rules [24][25] and has been used to solve many optimiza-

tion problems [21]. The term “hyper-heuristic” was independently used in 2000 [26] to describe “heuristics that choose heuristics” in the context of combinatorial optimization. In this context, a hyper-heuristic is a high-level approach which, given a particular instance of the problem and a number of low-level heuristics, can choose and apply an appropriate low level heuristic at each decision point [27][28].

In the literature it is possible to find a wide variety of hyper-heuristic approaches using high-level methodologies along with a set of low level heuristics applied to different optimization problems. However, there is no reason to limit the a high-level strategy to a heuristic. In fact, the sophisticated knowledge-based techniques such as CBR have been employed to this end with successful results for solving the university timetables problem [29]. This led to a more general definition for the term "hyper-heuristic", whose goal is to capture the idea of a method to automate the design of heuristics and the selection process: “A hyper-heuristic is an automated methodology for selecting or generating heuristics to solve hard computational search problems” [20].

The defining characteristic on hyper-heuristics research is that it investigates methodologies operating within a search space of heuristics rather than directly on a search space of problem solutions. This feature provides the potential to increase the level of general research methods. Several approaches for hyper-heuristics have been proposed that incorporate different research paradigms and machine learning [20].

The research on hyper-heuristics is based on the compromise between search methodologies and machine learning. Machine learning is a well established field of AI and its exploitation to automate the design of heuristics is still at the beginning, but it is expected big developments in the future [20].

## 4 Multi-agent Learning

In AI, machine learning is a research area concerning the development of algorithms and techniques in order to provide computers with learning faculties. Commonly accepted in the literature, machine learning algorithms and techniques can be classified in three categories:

- Supervised learning (where data have labels or classes);
- Unsupervised learning (data have no labels);
- Reinforcement learning (where the objective is to maximize a reward).

Some authors refer another category, placed between Supervised and Unsupervised learning, named Semi-Supervised learning, that uses both labeled and not labeled data. It is also very common the reference to another category, known by Instance-based Learning [30] or Non-Parametric Methods [31], where CBR can be included, described in the next section.

It is possible to apply machine learning concepts to many research areas, including natural language processing, pattern recognition, market analysis, DNA sequences classification, speech and handwriting recognition, object recognition in computer vision, game playing and robot locomotion.

Panait and Luke [32] have focused machine learning application to problems related with MAS. They use machine learning in order to explore ways to automate the inductive process, e.g., put a machine agent to find by itself how to solve a task or minimize error. They have referred that machine learning is a popular approach for the resolution of MAS problems because the complexity intrinsic to many of those problems can make solutions prohibitively hard to obtain.

In the next subsections, it will be described four learning techniques used in MAS, namely Reactive learning, Social learning, Team learning and Concurrent learning.

#### **4.1 Reactive learning**

In reactive systems, the cooperative behavior emerges from the interaction between agents. Instead of implementing coordination protocols or providing complex recognition models, it is assumed that agents work with value-based information (e.g. the distance they should keep from neighbors) which produces the social behavior. Once internal processing is avoided, these techniques allow MAS reacting to changes in an efficient way [2].

As a collateral effect, agents do not know the domain, which is crucial to take decisions in complex and dynamic scenarios. In this view, it is not possible to simulate complex social interactions and, in order to have high-level behaviors, agents need to summarize experiences in concepts. An entity that can conceptualize can also transform experience in knowledge and guide the vital resources until necessary [2].

#### **4.2 Social Learning**

Social learning is composed by learning mechanisms arising from AI and Biology.

In persistent MAS, where new agents enter a world already populated with experienced agents, a new agent starts with a blank state and has not had yet the opportunity to learn about the environment. However, a new agent does not need to discover everything about the environment since it can benefit from the accumulated learning from the experienced population of agents [2].

An important difference between artificial agents and animals is that, in the first, it is possible to simulate a completely cooperative scenario, where exists a common utility function. Even though cooperation occurs in many animal species, the possibility of conflicts emerging is always present, due to the competition in genes' self-replication of evolutionary process [2].

There are several different ways to an agent learn from other agents behaviors. Despite the existence of imitation (direct copy from other agents behaviors), this has proved to be complex since it involves not only the behaviors' understanding and reproducing but also the understanding of the changes in the environment caused by these behaviors [2].



### 4.3 Team Learning

In Team Learning it only exists an apprentice. However, it has the objective to discover a subset of behaviors for a team of agents, instead for a unique agent. It is a simple approach to Multi-Agent learning because the apprentice can use machine learning techniques, which avoid the difficulties emerging from the co-adaptation of multiple agents in Concurrent Learning approaches. Another advantage in the using of a unique apprentice agent is that it only cares about the team performance, and not with itself. For this reason, Team learning approaches can ignore the inter-agent credit assignment that is usually hard to determine [32].

However, Panait and Luke [32] also pointed some disadvantages in the use of Team learning. The main problem refers to the large state space for the learning process, which can be devastating for learning methods that explore the utility state space (such as Reinforcement learning) but cannot affect so drastically techniques that explore the behaviors space (such as Evolutionary computing). A second disadvantage refers to the learning algorithm centralization problem: every resource need to be available in the same place where the program will be executed. This can be uncomfortable for domains where data are inherently distributed.

Team learning can be divided in homogeneous and heterogeneous [32]. Homogeneous apprentices develop an unique identical behavior for each agent, even if agents are different. Heterogeneous apprentices must deal with a large search space, but with the guarantee to get better solutions through agents' specialization. The choice between approaches depends if experts are necessary in the team.

### 4.4 Concurrent Learning

The most common alternative to Team learning is Concurrent learning, where multiple apprentices try to improve parts from the team. Typically, each agent has its own learning process to modify the behaviors [32].

The main difficulty subjacent to Concurrent learning is to know in which domains it achieves better results when compared with Team learning. Jansen and Wiegand [33] argue that Concurrent learning can perform better in domains where decomposition is possible and helpful (such as Scheduling), and when it is useful to focus each sub-problem regardless others. This happens because Concurrent learning separates the search space into smaller ones. If the problem can be decomposed, such that agents' individual behaviors are relatively disjoint, it can result in a significant reduction of the search space and computational complexity. Another advantage is that decomposing the learning process into smaller pieces allows a greater flexibility using computational resources in each process learning, since they can, at least partially, be learned regardless others.

The main challenge of Concurrent learning consists in the adaption of each apprentice behaviors to the context of others, which its cannot control. In single agent scenarios, an apprentice explores his environment and improves his behavior. But things are quite different when using multiple apprentices: while agents learn, they change the behaviors, which can ruin the learned behaviors by other agents, making outdated

assumptions [34][35]. A simple approach to deal with this co-adaptation is to treat other apprentices as part of the dynamic environment for which each apprentice must adapt [36].

In this research, we propose a concurrent learning approach, in which several agents learn about their internal behaviors and environment.

## 5 Case-based Reasoning

Case-Based Reasoning (CBR) is an Artificial Intelligence technique that aims to solve new problems by using information about the resolution of previous similar problems [37]. As previously described, CBR represents a method of ML Instance-based Learning and uses the principle that similar problems may require similar solutions [38] on a direct analogy to learning based on past experience.

CBR roots are found in the work of Roger Schank about dynamic memory and how the memory of previous situations can affect problems' resolution and learning processes [39]. There are also references about the study of analogical reasoning [40].

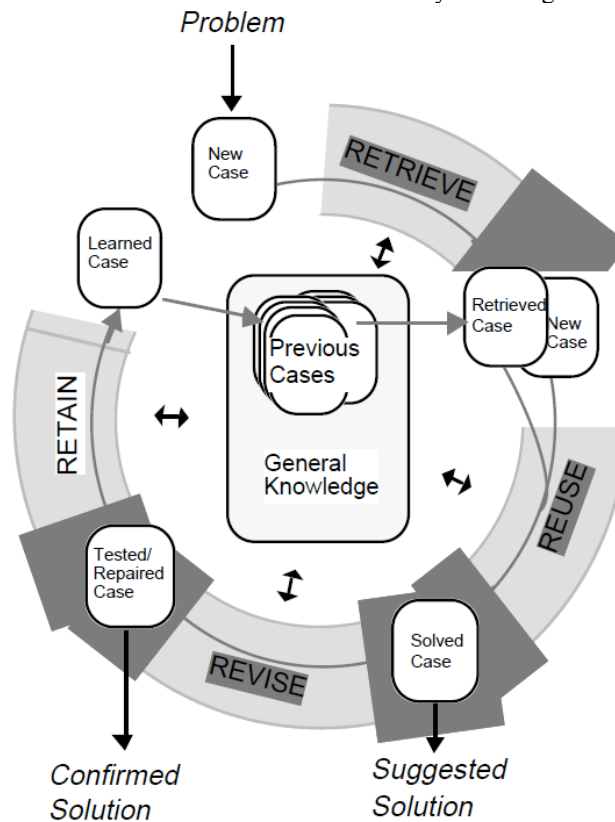


Fig. 2. The CBR cycle [42]

CYRUS system, developed by Janet Kolodner [37] was the first known CBR system. It was based on Schank's dynamic memory model [39] and, basically, consisted of a question-answer system with knowledge about the different travels and meetings of USA ex-Secretary of State Cyrus Vance. Another first system to use CBR was PROTOS, developed by Bruce Porter et al. [41], which dealt with ML classification problem.

The CBR cycle is illustrated in Fig. 2 and consists of four main phases [38][42]:

1. Retrieve the most similar case or cases
2. Reuse the retrieved information and knowledge
3. Revise the proposed solution
4. Retain the revised solution for future use

In CBR, previous solved cases and their solutions are memorized as cases in order to be reused in the future [38]. These cases are stored in a repository named casebase. Instead of defining a set of rules or general lines, a CBR system solves a new problem by reusing similar cases that were previously solved [43].

A new case of the problem to be solved is used to retrieve an old case from the casebase. In the Reusing phase, the retrieved case is analyzed in order to suggest a solution for the resolution of the new case. In the Revising phase, this suggested solution is tested, for example, by executing it in the system, and repaired if it fails. In the Retaining phase, the useful experience is retained for future use, and the casebase is updated with the new learned case (or by modifying some existing cases).

In the Reusing phase, it is possible to reuse a solution or a method. In solution reuse, the past solution is not directly copied to the new case, but there is some knowledge allowing the previous solution to be transformed into the new case solution. In case of method reuse, it is observed how the problem was solved in the retrieved case, which has information about the method used for the problem resolution, including an explanation about the used operators, sub-objectives considered, generated alternatives, failures, etc. The retrieved method is then reused to the new problem resolution, in the new context.

The objective of Revising phase is to evaluate the retrieved solution. If this solution is well succeeded it is possible to learn about the success, otherwise the solution is repaired using some problem domain's specific knowledge. The evaluating task applies the proposed solution in an execution environment and the result is evaluated. This is usually a step outside the CBR, once the problem may be executed in an application.

Finally, the Retaining phase consists in the integration of the useful information about the new case resolution into the casebase. It is necessary to know which information is important to retain, how to retain it, how to index the case for a future retrieve, and how to integrate the new case in the memory structure.

Burke et al. [44] referred that CBR is an appropriate approach for scheduling systems with expertise knowledge, and highlighted a research potential in dynamic scheduling.

Generally, CBR applications for scheduling domain can be classified in three categories [43]:

- Algorithms reuse - assume that it is probable that an effective approach for a specific problem's resolution will also be effective in the resolution of a similar problem. In these systems, a case consists in a representation of the problem and in a known effective algorithm for its resolution. Schmidt [45] designed a CBR structure to choose the most appropriate method for the resolution of scheduling problems in production scheduling. Schirmer [46] implemented a CBR system for selecting scheduling algorithms for the resolution of project scheduling problems. It was experimentally shown that some scheduling algorithms work better than others, in some instances of problems.
- Operators reuse - reuse the operators for the resolution of the new problem [44]. A case describes a context in which a useful scheduling problem is used for repairing/adapting a scheduling plan to improve its quality, in terms of constraints satisfaction [38]. Burke et al. [44] have proposed a case-based hyper-heuristic to solve timetabling problems. Beddoe et al. [38] have developed a CBR system to solve nurse scheduling problems.
- Solutions reuse - it is used the whole or part of previous problems' solutions to construct the solution of the new problem. A case contains the description of a problem and its solution, or part of solution. This method was used for the resolution of manufacturing scheduling problems [47][48] and university courses timetabling [44]. It was also used for constructing MH' initial solutions, as Genetic Algorithms [49] and Simulated Annealing [50].

## 6 Multi-agent Scheduling System

The developed MAS for the resolution of Scheduling problem consists in a hybrid autonomous architecture [51]. As illustrated in Fig. 3, there are three kinds of agents.

The proposed MAS have agents representing jobs/tasks and agents representing resources/machines. The system is able to find optimal or near optimal solutions through the use of MH, dealing with dynamism (arriving of new jobs, cancelled jobs, changing jobs attributes, etc.), change/adapt the parameters of the algorithm according to the current situation, switch from one MH to another, and perform a coordination between agents through cooperation or negotiation mechanisms.

Job agents process the necessary information about the respective job. They are responsible for the generation of the earliest and latest processing times on the respective job and automatically separate each job's operation for the respective Resource Agent.

Resource agents are responsible for scheduling the operations that require processing in the machine supervised by the agent. These agents implement MH in order to find the best possible single-machine schedules/plans of operations and communicate those solutions to the AgentUI for later feasibility check.

Since it is impossible to predict each problem to treat, the system should be capable of learning about its experience during lifetime, as humans do. To perform this learning mechanism, it is proposed the use of CBR within Resource agents.

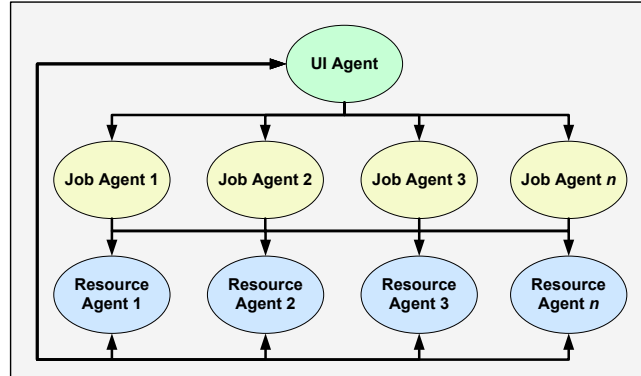


Fig. 3. Multi-agent Scheduling System

### 6.1 CBR module

The proposed CBR approach [51] consists in retrieving the most similar case or cases to the new problem, regardless the MH to be used, as well as its parameters. It is important for the system to decide which technique and respective parameters may be used, because not every MH is suitable to all types of problems.

The main objective of CBR module is to choose a MH to be used by the respective Resource Agent in which the CBR is included. The secondary objective is to perform the parameter tuning of MH, according to the problem to solve. Based on past experience, each case contains the MH and the respective parameters. If the parameters were effective and efficient in the resolution of a similar case, then they have a great probability to be effective and efficient in the resolution of the new problem. It is possible to describe our CBR module as a hyper-heuristic approach but since it performs a self-parameterization of MH it is more appropriate to see it as a parameter tuning approach.

It is important to notice that, like previously described in Fig. 2, every new problem or perturbations occurred leads to a new case in the system, with the previous most similar cases being retrieved from the casebase. After that, the better case is reused, becoming a suggested solution. After the solution revision, the case is executed in the MAS. This revision is performed to allow escaping from local optimal solutions and MH stagnation, since it is used some disturbance in the parameters of the proposed solution. After the conclusion of the MAS execution, the case is confirmed as a good solution, being retained on the database as a new learned case, for future use.

Figure 4 illustrates the inclusion of CBR in the system. Each Resource Agent has its own CBR module. With this approach, different MH may be chosen in the resolution of the same Job-Shop problem. This can be considered as an advantage because the Resource Agents can have different number of operations to schedule. Some MH are more suitable to schedule problems with large number of operations than others.

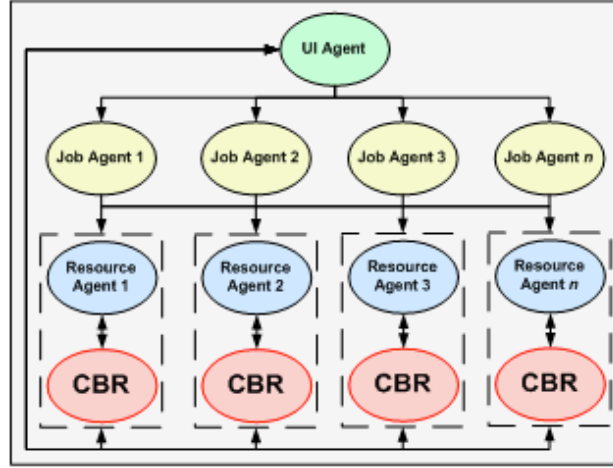


Fig. 4. CBR module within Resource agents

The most important part of a CBR module is its similarity measure because it decides how much two cases are similar between each other. The similarity measure of the proposed CBR module is very simple and is defined in equation (1).

$$Similarity = \frac{NumberOperations_{CaseA}}{NumberOperations_{CaseB}} \quad (1)$$

As previously mentioned, each Resource Agent has a number of operations to schedule. This number of operations can be different, depending on the problem to treat, and is enough to define a problem. The MH and the respective parameters may be chosen according to the dimension of the problem to treat. So, with this similarity measure it is possible to have a ratio between two cases. The similarity is a value in the interval  $[0,1]$ , whose limits correspond to non similar and completely similar cases, respectively. If there are more than one case very similar to the problem to be solved, the most effective and efficient case is reused.

If some perturbations occur in the problem, the MH and the parameters may change, because a different problem may be solved. For example, if new jobs arrive or if some jobs are canceled, the problem's dimension is different and so other MH and/or other parameters may be used. This decision is autonomously performed by the CBR module in run time.

## 7 Computational Results

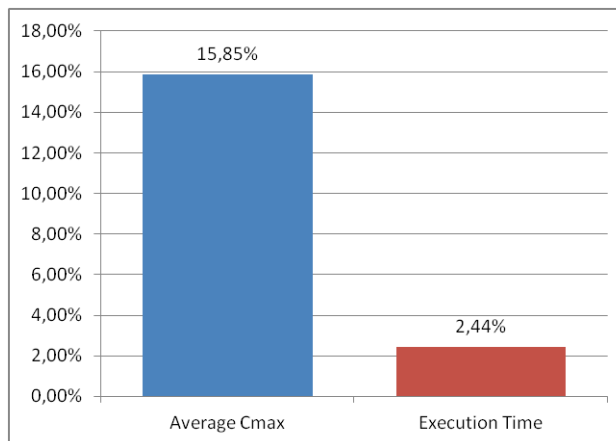
The main objective of this computational study is to analyze the integration of CBR in an effective and efficient way, comparing the system's performance with CBR included versus the system's performance before the integration of CBR. Another objective is to obtain some conclusions about the usage of MH in the resolution of Job-Shop instances, after the integration of CBR.

For the computational study, all instances from OR-Library Job-Shop Scheduling problems were used [52] (a total of 82 instances), proposed by Adams, Balas and Zawack [53], Fisher and Thompson [54], Lawrence [55], Applegate and Cook [56], Storer, Wu and Vaccari [57], and Yamada and Nakano [58]. These instances cover problems with 10, 20, 30, and 50 jobs and they were executed five times (before and after CBR integration).

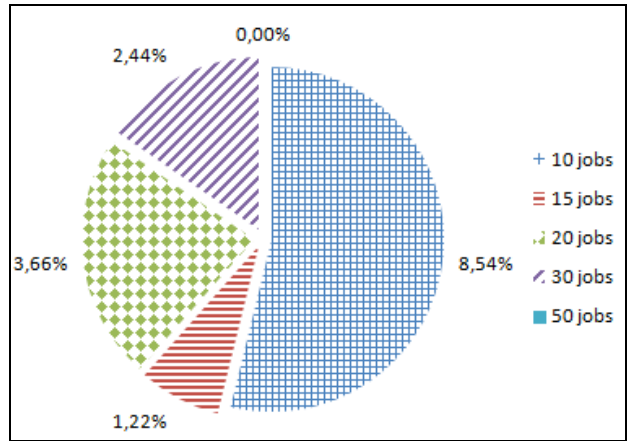
The machine used for the computational study is a HP Z400 Workstation, with the following main characteristics: Intel® Xeon® CPU W3565 @ 3.20 GHz, 6GB RAM, Samsung HD103SJ disk with 1TB, and Windows 7, 64-bit.

To conclude about the effectiveness and efficiency of the proposed CBR module, the average makespan (Cmax - conclusion time) and execution time were analyzed (Fig. 5). About the effectiveness, the average makespan was improved in 15,85% of the cases. This is considered a good improvement that can be better with the lifetime of CBR module.

Although a new module has been integrated into the MAS, the average execution times were improved in 2,44% of the cases when comparing to the previous obtained results (Fig. 5). It was not expected to improve this performance measure but it we can conclude that the parameters are becoming more efficient with the lifetime of CBR module.



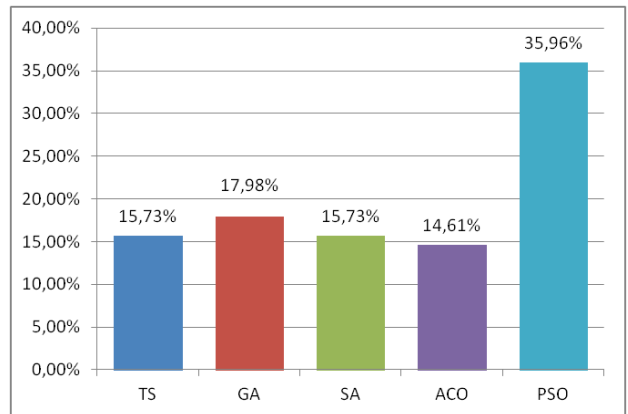
**Fig. 5.** Improvement of obtained average results (%)



**Fig. 6.** Improvement of obtained average  $C_{max}$  results separated by instances dimension (%)

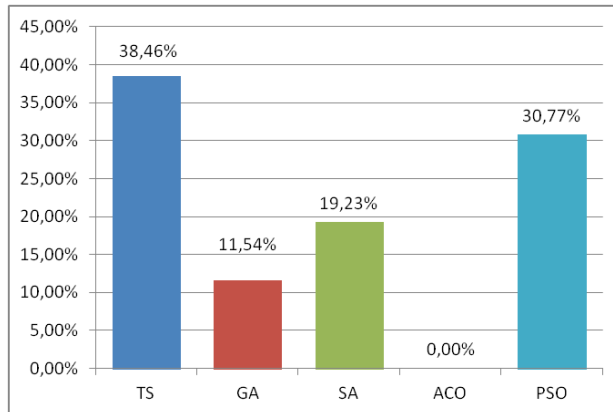
Figure 6 presents a detailed view about the improvement of average  $C_{max}$ . For 10 jobs instances, 8,54% of the results were improved. For 20 jobs instances, 3,66% results were improved. For 15 and 30 instances the results were improved only by 1,22% and 2,44% respectively. The obtained results for 50 jobs instances were not improved at all.

In addition to the obtained conclusions about the effectiveness and efficiency of CBR it is also possible to analyze the usage of MH. With this it is possible to know which MH were used most. In a global perspective (Fig. 7), PSO was the most used MH, in 36,96%, and then GA with 17,98%. TS and SA were used in 15,73% of the instances. Finally, ACO was the less used with 14,61%.

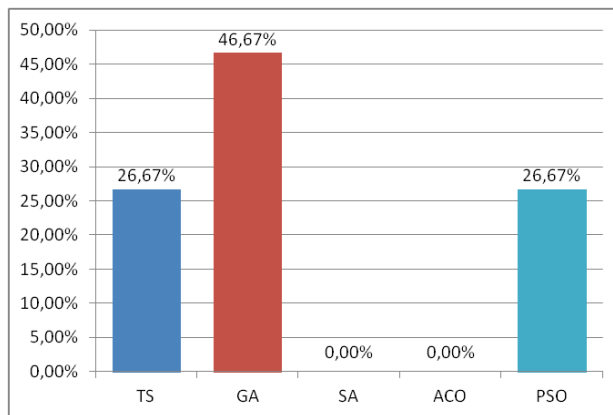


**Fig. 7.** Global use of MH





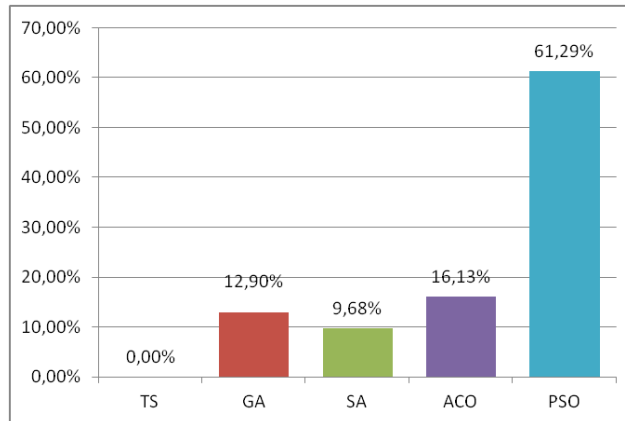
**Fig. 8.** MH use for 10 jobs instances



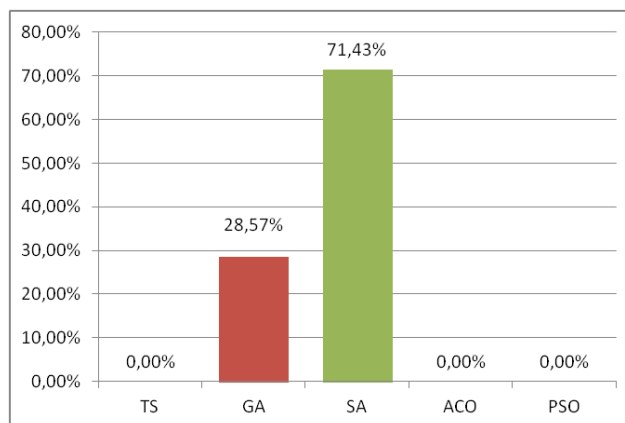
**Fig. 9.** MH use for 15 jobs instances

Figure 8 presents the MH use for 10 jobs instances. TS and PSO were the most used techniques in the smallest dimension problems' instances. ACO was not used at all.

Figure 9 presents the MH used for 15 jobs instances. GA was the most used MH with 46,67%. TS and PSO were the other techniques used in 26,67% of the cases. SA and ACO were not used in the resolution of this class of instances.



**Fig. 10.** MH use for 20 jobs instances



**Fig. 11.** MH use for 30 jobs instances

Only two MH were used in the resolution of 30 jobs instances, as shown in Fig. 11. SA was the most used technique in 71,43% of the cases. GA was the other used MH, in 28,57%.

Finally, for 50 jobs instances, ACO was the most used MH in 80% of the cases. The other used MH were SA and PSO in 10% of the cases each (Fig. 12).

Concluding, for small instances (10 and 15 jobs) TS, GA and PSO revealed to be the most used, but ACO was not used at all. For 20 and 30 jobs instances PSO and SA were the most used respectively. For large dimension instances ACO was the most used.



**Fig. 12.** MH use for 50 jobs instances

## 8 Conclusions

In this paper the use of CBR was proposed in order to perform MH parameter tuning in the resolution of Job-Shop scheduling problem.

The presented scheduling system consists in a MAS with different agents representing both jobs and resources. The proposed CBR module is included in resource agents with the objective to chose the best MH and perform the respective parameter tuning. The MH choice and parameters configuration is done based on past experience, since CBR assumes that similar cases may have similar solutions.

From the computational study presented it is possible to conclude that the system became more effective in 15,85% of the cases and more efficient in 2,44%.

It was also possible to conclude that, in the resolution of academic Job-Shop instances, PSO was, globally, the most used technique, in a global perspective. However, in the resolution of small dimension instances TS revealed to be the most used, and in the resolution of very large dimension instances ACO was the most used.

For future work we intend to do more experiments with the CBR module. It is expected that the effectiveness and efficiency of CBR improves during the lifetime.

### Acknowledgments

This work is supported by FEDER Funds through the “Programa Operacional Factores de Competitividade - COMPETE” program and by National Funds through FCT “Fundação para a Ciência e a Tecnologia” under the project: FCOMP-01-0124-FEDER-PEst-OE/EEI/UI0760/2011 and PTDC/EME-GIN/109956/2009. The first author would like also to thanks FCT for the Ph.D. scholarship SFRH/BD/63404/2009. This work is partially supported in the framework of the IT4 Innovations Centre of Excellence project, reg. no. CZ.1.05/1.1.00/02.0070 by operational programme ‘Research and Development for Innovations’ funded by the Struc-

tural Funds of the European Union and state budget of the Czech Republic, EU.

## References

1. E. Plaza, J. Arcos and F. Martin, Cooperative Case-Based Reasoning, in Weiss, G., ed. Distributed Artificial Intelligence Meets Machine Learning, Lecture Notes in Artificial Intelligence., Springer, 1996.
2. E. Alonso, M. D'inverno, D. Kudenko, M. Luch, and J. Noble, Learning in Multi-agent Systems, The Knowledge Engineering Review, volume 16 (3), pp.277-84, 2001.
3. El-Ghazali Talbi, Metaheuristics - From Design to Implementation, Wiley, 2009.
4. K. R. Baker and D. Trietsch, Principles of Sequencing and Scheduling, John Wiley & Sons, Inc., 2009.
5. M. Pinedo, Scheduling: Theory, Algorithms, and Systems, Fourth Edition, Springer, 2012.
6. A. Madureira, Meta-heuristics application to scheduling in dynamic environments of discrete manufacturing, Ph.D. thesis, University of Minho, Braga, Portugal, 2003, (in portuguese).
7. F. Glover, "Future paths for integer prog. and links to artificial intelligence", Comp. & Ops. Res., volume 5, pp.533-49, 1986.
8. C. Blum, and A. Roli, "Metaheuristics in combinatorial optimization: Overview and conceptual comparison", ACM Comput. Surv. 35, September, pp.268-308, 2003.
9. S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi, "Optimization by simulated annealing", Science, volume 220(4598), pp.671-680, 1983.
10. V. Cerny, "A thermodynamical approach to the travelling salesman problem: An efficient simulation algorithm", J. Optim. Theory Appl. 45, pp.41-51, 1985.
11. N. Metropolis, A.W. Rosenbluth, M.N Rosenbluth, A.H. Teller, and E. Teller, "Equations of State Calculations by Fast Computing Machines", Journal of Chemical Physics 21(6), pp.1087-1092, 1953.
12. J.H. Holland, Adaptation in Natural and Artificial Systems, University of Michigan, 1975.
13. D.E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, 1989.
14. M. Dorigo, V. Maniezzo, and A. Colorni, The ant system: an autocatalytic optimizing, Technical Report, TR91-016, Milano, 1991.
15. J. Kennedy, and R. Eberhart, "Particle swarm optimization", in IEEE International Conference on Neural Networks, 1995.
16. G. Box, J.S. Hunter, and W. G. Hunter, Statistics for Experimenters: Design, Innovation, and Discovery. Wiley, 2005.
17. J.D. Schaffer, R.A. Caruana, L. Eshelman, and R. Das, "A study of control parameters affecting online performance of genetic algorithms for function optimization", in J. D. Schaffer, editor, 3rd International Conference on Genetic Algorithms, Morgan Kaufman, San Mateo, CA, 1989, pp. 51-60.
18. O. Maron and A. W. Moore, "Hoeffding races: Accelerating model selection search for classification and function approximation", in Advances in Neural Information Processing Systems, Vol. 6. Morgan Kaufmann, San Francisco, CA, 1994, pp. 59-66.
19. M. Birattari, T. Stutzle, L. Paquete, and K. Varrentrapp, "A racing algorithm for configuring metaheuristics", in W. B. Langdon et al., editors, Proceedings of the Genetic and Evolution-

- ary Computation Conference (GECCO'2002), Morgan Kaufmann, San Francisco, CA, 2002, pp. 11–18.
20. E. Burke, M. Hyde, G. Kendall, G. Ochoa, E. Ozcan, E., and J. Woodward, “A classification of hyper-heuristics approaches”, Gendreau, M., Potvin, J. Y. (eds) Handbook of metaheuristics (2nd edition), Springer, 2009.
  21. Y. Hamadi, E. Monfroy and F. Saubion, “An Introduction to Autonomous Search”, Y. Hamadi et al. (eds.) Autonomous Search, ISBN 978-3-642-21433-2, Springer, 2012.
  22. P. Cowling, G. Kendall, and E. Soubeiga, “Hyperheuristics: a tool for rapid prototyping in scheduling and optimisation”, in Applications of Evolutionary Computing, EvoWorkshops 2002, Lecture Notes in Computer Science, vol., 2279, pp, 1–10, Springer, 2002.
  23. J. Denzinger, M. Fuchs, and M. Fuchs, “High performance ATP systems by combining several AI methods”, in Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI 97), pp. 102–107, USA, 1997.
  24. W. Crowston, F. Glover, G. Thompson, and J. Trawick, Probabilistic and parametric learning combinations of local job shop scheduling rules, Tech. rep., ONR Research Memorandum No. 117, GSIA, Carnegie-Mellon University, Pittsburg, 1963.
  25. H. Fisher and L. Thompson, “Probabilistic learning combinations of local job-shop scheduling rules”, Industrial Scheduling, Prentice Hall, 1963.
  26. P. Cowling, G. Kendall, and E. Soubeiga, “A hyperheuristic approach for scheduling a sales summit”, in Selected Papers of the Third International Conference on the Practice And Theory of Automated Timetabling, PATAT 2000, Lecture Notes in Computer Science, pp. 176–190, Springer, 2000.
  27. E. Burke, E. Hart, G. Kendall, J. Newall, P. Ross, and S. Schulenburg, “Hyper-heuristics: An emerging direction in modern search technology”, in F. Glover and G. Kochenberger (eds.), Handbook of Metaheuristics, pp. 457–474, 2003.
  28. P. Ross, “Hyper-heuristics”, in E. K. Burke and G. Kendall (eds.) Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques, chapter 17, pp. 529–556. Springer, Berlin, 2005.
  29. E. Burke, S. Petrovic, and R. Qu, “Case based heuristic selection for timetabling problems”, Journal of Scheduling, volume 9(2), pp. 115–132, 2006.
  30. T. Mitchell, Machine Learning. McGraw-Hill Education, ISE Editions, 1997.
  31. E. Alpaydin, Introduction to Machine Learning, Adaptive Computation and Machine Learning, The MIT Press, 2004.
  32. L. Panait and S. Luke, Cooperative Multi-Agent Learning: The State of the Art, Autonomous Agents and Multi-Agent Systems, pp.387-434, 2005.
  33. T. Jansen and R.P. Wiegand, Exploring the explorative advantage of the cooperative coevolutionary (1+1) EA, in Cantú-Paz, E. et al., eds. Genetic and Evolutionary Computation - GECCO 2003, Chicago, Springer-Verlag, 2003.
  34. T. Sandholm and R.H. Crites, On multiagent Q-learning in a semi-competitive domain, in Adaption and Learning in Multi-Agent Systems, 1995.
  35. M. Weinberg and J. Rosenschein, Best-response multiagent learning in non-stationary environments, in AAMAS-2004 Proceedings of the Third International Joint Conference on Autonomous Agents and Multi Agent Systems, 2004.
  36. J. Schmidhuber and J. Zhao, Multi-agent learning with the success-story algorithm, in ECAI Workshop LDAIS/ ICMAS Workshop LIOME, 1996.
  37. J. Kolodner, Case-Based Reasoning, Morgan Kaufmann Publishers Inc, 1993.
  38. G. Beddoe, S. Petrovic, and J. Li, “A hybrid metaheuristic case-based reasoning system for nurse rostering”, Journal of Scheduling, volume 12, number 2, April, pp. 99-119, 2009.

39. R. Schank, *Dynamic memory; a theory of reminding and learning in computers and people*. Cambridge University Press, 1982.
40. [40] D. Gentner, "Structure mapping - a theoretical framework for analogy. *Cognitive Science*", volume 7, pp. 155-170, 1983.
41. B. Porter and R. Bareiss, *PROTOS: An experiment in knowledge acquisition for heuristic*. In *Proceedings of the First International Meeting on Advances in Learning (IMAL)*, Les Arcs, France, 1986.
42. A. Aamodt, and E. Plaza, "Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches", *Artificial Intelligence Communications*, volume 7, pp. 39-52, 1994.
43. S. Petrovic, Y. Yang, and M. Dror, "Case-based selection of initialisation heuristics for metaheuristic examination timetabling", *Expert Syst. Appl.* 33, October, pp. 772-785, 2007.
44. E.K. Burke, B.L. MacCarthy, S. Petrovic, and R. Qu, *Knowledge Discovery in a Hyper-Heuristic for Course Timetabling Using Case-Based Reasoning*. PATAT 2002, 2002.
45. G. Schmidt, *Case-based reasoning for production scheduling*. *International Journal of Production Economics*, 56-57, 537-546, 1998.
46. A. Schirmer, *Case-based reasoning and improved adaptive search for project scheduling*. *Naval Research Logistics* 47, 201-222, 2000.
47. J. Coello. and R. Santos, *Integrating CBR and heuristic search for learning and reusing solutions in real-time task scheduling*. In *Proceedings of 3rd International Conference on Case-Based Reasoning*, Germany, Springer-Verlag, 1999.
48. B. MacCarthy and P. Jou, *Case-based reasoning in scheduling*. In MK, K. & CS, W., eds. *Proceedings of the Symposium on Advanced Manufacturing Processes, Systems and Techniques (AMPST96)*, MEP Publications Ltd, 1996.
49. S. Oman and P. Cunningham, *Using case retrieval to seed genetic algorithms*. *International Journal of Computational Intelligence and Applications*, 1, 1, 71-82, 2001.
50. P. Cunningham and B. Smyth, *Case-Based Reasoning in Scheduling: Reusing Solution Components*. *The International Journal of Production Research*, 35, 2947-2961, 1997.
51. I. Pereira, A. Madureira, and P. de Moura Oliveira, "Multi-apprentice learning for metaheuristics parameter tuning in a Multi Agent Scheduling System", *Nature and Biologically Inspired Computing (NaBIC)*, 2012 Fourth World Congress on, pp.31-36, 2012.
52. OR-Library, <http://people.brunel.ac.uk/~mastjjb/jeb/info.html>
53. J. Adams, E. Balas, and D. Zawack, D., "The shifting bottleneck procedure for job shop scheduling", *Management Science* 34, pp. 391-401, 1988.
54. H. Fisher, and G.L. Thompson, "Probabilistic learning combinations of local job-shop scheduling rules", Muth, J.F. and Thompson, G.L. (eds.), *Industrial Scheduling*, Prentice Hall, pp. 225-251, 1963.
55. S. Lawrence, "Resource constrained project scheduling: an experimental investigation of heuristic scheduling techniques" (Supplement), *Carnegie-Mellon University*, Pittsburgh, 1984.
56. D. Applegate, and W. Cook, "A computational study of the job-shop scheduling instance", *ORSA Journal on Computing* 3, 149-156, 1991.
57. R.H. Storer, S.D. Wu, and R. Vaccari, "New search spaces for sequencing instances with application to job shop scheduling", *Management Science* 38, 1495-1509, 1992.
58. T. Yamada, and R. Nakano, "A genetic algorithm applicable to large-scale job-shop instances", R. Manner, B. Manderick (eds.), *Parallel instance solving from nature 2*, North-Holland, 281-290, 1992.