Metaheuristics Parameter Tuning Using Racing and Case-Based Reasoning in Scheduling Systems

Ivo Pereira(✉), Ana Madureira, and Bruno Cunha

GECAD - Knowledge Engineering and Decision Support Research Center, Institute of Engineering – Polytechnic of Porto (ISEP/IPP), Porto, Portugal
{iaspe, amd, bmaca}@isep.ipp.pt

Abstract. Real world optimization problems like Scheduling are generally complex, large scaled, and constrained in nature. Thereby, classical operational research methods are often inadequate to efficiently solve them. Metaheuristics (MH) are used to obtain near-optimal solutions in an efficient way, but have different numerical and/or categorical parameters which make the tuning process a very time-consuming and tedious task. Learning methods can be used to aid with the parameter tuning process. Racing techniques have been used to evaluate, in a refined and efficient way, a set of candidates and discard those that appear to be less promising during the evaluation process. Case-based Reasoning (CBR) aims to solve new problems by using information about solutions to previous similar problems. A novel Racing+CBR approach is proposed and brings together the better of the two techniques. A computational study for the resolution of the scheduling problem is presented, concluding about the effectiveness of the proposed approach.

Keywords: Artificial intelligence · Case-based reasoning · Learning · Metaheuristics · Parameter tuning · Racing · Scheduling

1 Introduction

As defined by Pinedo [1], Scheduling is *"a decision-making process that is used on a regular basis in many manufacturing and services industries"*. Scheduling problems deal with the allocation of resources (e.g. machines) to tasks (or jobs) over given time periods. The goal is to optimize one or more objective functions. In other words, Scheduling problems could be stated as: given a set of jobs (composed by a set of operations), a set of resources, and an optimization criterion for performance measure, the optimal plan is the best way to allocate the resources to the operations considering that precedence relations and the resource availabilities are respected and performance is optimized [2].

The Scheduling problem treated in this work was discussed in [3] and referred as Extended Job-Shop Scheduling Problem (EJSSP). It has some major extensions and differences compared to the classic Job-Shop Scheduling Problem (JSSP), in order to better represent real world problems. JSSP has a set of tasks processing in a set of machines. Each task has an ordered list of operations, each one characterized by the

A.M. Madureira et al. (eds.), *Intelligent Systems Design and Applications*,
Advances in Intelligent Systems and Computing 557,
DOI 10.1007/978-3-319-53480-0_90

respective processing time and machine where is processed. EJSSP additional constraints are: different release and due dates for each job; different priorities for each job; the possibility that not every machines are used for all jobs; a job can have more than one operation being processed in the same machine; two or more operations of the same job can be processed simultaneously; the possibility of existence of alternative machines, identical or not.

Metaheuristics (MH) have gained recognition over the past two decades by successfully solving many real-world problems, including scheduling problems [4]. Indeed, they allow the resolution of large dimension problems by achieving satisfactory solutions in reasonable execution times. These techniques have the objective of guiding and improving the search process to overcome local optimal solutions. They can obtain good quality solutions, very close to the optimal, in reasonable execution times [4].

Metaheuristics parameter tuning has a great influence in the effectiveness and efficiency of the search process. Parameter tuning is known to be a difficult, time consuming and tedious task. There are no "universal" values for the MH parameters. The process of defining the parameters is not obvious because they depend on the problem and the time that the user has to solve the problem [4].

One of the oldest and more important goals in the field of MH is to transfer part of the parameters tuning effort to the algorithm itself. The objective is to provide MH with intelligent mechanisms to be capable of self-adaptation to the problem or situation. Learning techniques can help with this aspect.

Racing is known to be a machine learning approach that has been used to evaluate, in a refined and efficient way, a set of candidates and release those that appear to be less promising during the evaluation process. Racing can be used to effectively and efficiently perform parameter tuning of MH.

Case-based Reasoning (CBR) is a problem-solving paradigm in many aspects different from other Artificial Intelligence approaches. Instead of relying only on general knowledge, CBR is able to use specific knowledge of concrete situations (cases) of a problem. Together with Racing, CBR usage seems very promising.

A hybrid Racing+CBR framework to perform MH parameter tuning is proposed in this paper, which brings together the better of the two learning techniques. A computational study for the resolution of the scheduling problem is carried out in order to validate its influence in the global system's performance. With this work we intend to make a contribution for the resolution of the MH parameter tuning problem, for solving real manufacturing scheduling problems.

The remaining sections of the paper are organized as follows: Sect. 2 describes the proposed Racing+CBR approach, as a novel contribution to the literature; in Sect. 3, the computational results are presented, stating the quality of the proposed hybrid approach; finally, conclusions and future work are presented in Sect. 4.

2 Racing+CBR Learning Approach

In this paper, a hybrid approach for parameter tuning is proposed, joining Racing and CBR together. As illustrated in Fig. 1, Racing is executed before the CBR, where Racing output (tuned parameters) acts as an input for the casebase. After the Racing

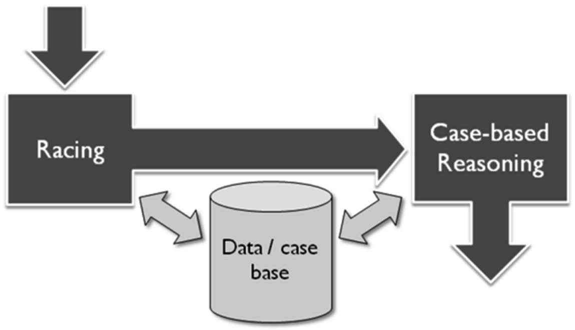

Fig. 1. Racing+CBR architecture [5]

module processing ends, CBR starts its own independent process, using the Racing output parameters to input the casebase. CBR evolves naturally through time and learns with experience. Both modules communicate through a data/casebase.

The idea of joining these two techniques relies on the fact that they can help each other while performing the learning process. Racing approaches perform an intelligent evaluation of learning candidates and gives suitable results, allowing e.g. categorization of candidates and generalization, it lacks on efficiency.

The most common Racing approaches need a lot number of instances to run, which make these learning techniques not very suitable for dynamic systems, as a Manufacturing Scheduling System is. Therefore, is not feasible to run over and over again a Racing algorithm to perform the recommendation of MH parameters each time one needs to optimize the system.

Case-based Reasoning comes into action as a very efficient technique that uses past knowledge to give recommendations. One of the most common problems using CBR is based on the fact that the casebase needs to be populated with good knowledge; otherwise CBR will need many cycles until achieve fairly good results. However, using an initial casebase populated by Racing, it is a very good starting point and the efficiency of the learning system increases.

The main objective of this paper is to show and understand how Racing and CBR can work together based on these premises. In the following subsections, each module is described in detail.

2.1 Racing Module

Racing techniques has been used to evaluate, in a refined and efficient way, a set of candidates and discard those that appear to be less promising during the evaluation process. Instead of training *N* times the structure of the model, a Racing algorithm can speed up the search for the best structure models, discarding the lower candidates once enough evidence against them are obtained.

The apparent advantage of Racing approaches on brute-force is based on the fact that it provides a better allocation of computational resources among the candidate configurations. Instead of wasting time to accurately estimate the performance of the lower candidates, Racing approaches focus on the most promising. This aspect allows a more informed selection of the best candidate. In Racing approaches, a candidate is dropped from future assessments so that there is sufficient evidence that it is not the best. Throughout the evaluation process, the approach of Racing focuses increasingly on the most promising candidates. On the other hand, the brute-force approach tests all candidates in the same number of instances.

The algorithm of the Racing module (Table 1) undergoes a study of various combinations of MH parameters – candidates. Thus, as input a list of MH to validate is necessary. The information that follows is based on [5, 6].

Table 1. Racing algorithm

Step	Description
1	Get all the problem instances
	Get all the candidate parameters
2	Create race to evaluate candidates
3	For each instance still in the race
	For each candidate still in the race
	Perform a run to evaluate candidate
	Remove worst candidates
4	Return best candidate

The first step involves obtaining a list of candidate parameters of each MH to make a race between them. These races are held in a number of instances, which are also obtained in the first step. For each instance, each combination of parameters is tested in the system and the data are stored in the database. At each instance, the candidates who obtained the worst results are removed. At the end of the race, the best candidate is the one who was able to survive through the various instances.

The most important of this whole process is the remove candidates' algorithm, since it decides which candidates will be eliminated from the race. This algorithm takes as input parameters the current race and the list of candidates' parameters, returning the list of survivors' parameters. The first test to perform is check if the list of candidates has more than one combination; otherwise we are looking at the best case. Then we need to check which statistical test to be used. If the candidate list has more than two elements, Friedman's test is applied. Otherwise, Wilcoxon test is used. Both statistical tests are described in [7].

To apply Friedman's test, at least two blocks of results are required, i.e. all candidates have been carried out in at least two instances. This implies that all candidates survive at the end of the first instance. To apply the Friedman's test, one should first obtain the ordered rankings of the candidate parameters and then calculate the sum of each ranking for each candidate.

The number of survivors at each phase will be the best n, calculated using the following equation:

$$sobrev = round\left(\frac{1}{\log_2(inst + 1)} \times cand\right) \quad (1)$$

where *cand* is the number of candidates and *inst* is the number of instances.

The number of survivors at each step relies on the number of existing candidates in that step and on the number of instances already executed. In this work, the number of survivors is calculated according to the inverse base-2 logarithm [5]. With this function, it is possible for even a small number of instances, to obtain a quick convergence to the best candidate. This point is an important contribution of this proposal, since the most common racing algorithms are not possible to apply in a small set of instances.

Birattari [6, 8] described that the Wilcoxon matched-pairs signed-ranks can be used when there are only two candidates running, since it has proved more robust and efficient. Following these guidelines, the Wilcoxon test is applied when there are only two candidates.

2.2 Case-Based Reasoning Module

Case-based Reasoning is an Artificial Intelligence methodology aiming to solve new problems by using information about solutions to previous similar problems [9]. On a direct analogy to human learning based on past experience, CBR uses the principle that similar problems may require similar solutions.

Reusing past cases is a powerful method for human and often applied to problem solving. Part of CBR grounds is its psychological plausibility. In CBR, a new problem is solved by searching for a similar past case, and reuse in the new situation. Another important aspect is that CBR used an incremental and sustained learning approach, since a new experience is retained each time a problem is solved, being readily available for future problems [9].

A CBR cycle consists in four main phases [9]:

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.

The database or case repository is designated as casebase. In the first phase, a new case to be solved by CBR is used to retrieve an old case from the casebase. At the second phase, the objective is to retrieve the most similar previous case that was found in the casebase. In the Reusing phase, this retrieved case is used to suggest a solution to solve the new case. This means that, e.g., the solution used to solve this retrieved case can be used to solve the new case, since they are similar between them. In the Revising phase, the suggested solution is tested, e.g., by executing the system, and adapted if necessary. Finally, in the Retaining phase, the information is retained for future use, and the casebase is updated with the new learned case [9].

The objective of this CBR submodule is to recover the most similar case with the new problem, regardless the used MH. Thus, the returned case contains the MH and the respective used parameters. This approach turns out to be suited to the problem at hand because it is important for the system to decide, at each time, what technique and parameters should be used.

As shown in Table 2, the CBR module consists in retrieving the most similar case or cases to the new problem, regardless the MH to use. Therefore, the case(s) containing the MH and its parameters are retrieved. It is important for the system to decide which MH to use and the respective parameters, because not all MH are suitable to every type of problems. As a consequence of the application of this approach, it will be possible to know e.g. which MH are more suitable to a particular class of problem.

Table 2. Case-based reasoning algorithm

Step	Description
1	Retrieving phase
2	Reusing phase Get retrieved case Get similarity between retrieved and new case
3	Revising phase Parameters tuning revision Running experience in the system
4	Retaining phase

At the start, every new problem leads to a new case for the system. In the first stage, the previous most similar cases are retrieved from the database. After recovering a list of most similar cases, the best case is reused, becoming in a suggested solution. In the Revise stage, the problem is executed using the given solution. To escape from local optimal solutions and stagnation, the use of some disturbance in the parameters of the proposed solution is proposed. In the end, the case are confirmed or not as a good solution, and if so, it is retained on the database as a new learned case, for future use.

In the Retrieving phase, previous cases are analyzed one by one, in order to select those that are similar enough with the new case, by calculating its similarity measure. The attributes to consider in the similarity measure are some characteristics from scheduling problem: number of jobs/tasks, number of machines/resources, problem type (some characteristics can be categorized to better identify a type of problem), number of precedence levels, etc. These attributes will depend exactly on the type of problem to solve and should be differently weighted, as shown in Eq. 2.

$$similarity = \sum_{i=0}^{n} w_i \times sim_i \tag{2}$$

where w_i is the weight of characteristic i and sim_i is the similarity of characteristic i

The different similarities can be calculated depending on the type of attribute. For example, the similarity of number of jobs and number of machines corresponds to the division of the lower value by the higher value, being in the interval [0;1]. The problem type similarity is a Boolean, representing if the attributes are the same or not. In the end of this phase, a list of most similar cases is retrieved.

In the Reusing phase, a case is selected from the list of most similar cases returned by the Retrieving phase, being the respective solution suggested as a candidate solution to solve the new case. So, the MH and respective parameters are returned to be used in the resolution of the new case.

In the Revising phase, the suggested solution is adapted, since the direct use of the solutions leads the system to stagnate and cannot evolve to the best results. So, to avoid local optimal solutions and system's stagnation, an algorithm that applies some diversity and perturbation to the suggested solution's parameters, using a global credit to assign different credit to the different parameters. The global credit is inversely proportional to the similarity of the reused case. This means that the less similar a case is the most perturbation is included in the suggested MH parameters.

Finally, in the Retaining phase, it becomes necessary to store the information in the casebase. First, values of conclusion and execution times are collected, resulting from the execution of the new case. The case is stored and, with this phase, the CBR cycle is concluded. In the next execution, this newly solved case will be available to be used in the resolution of a new case.

3 Computational Study

In this section, the computational study is presented that allows us to conclude about the effectiveness of the proposed Racing+CBR approach, in the resolution of the Scheduling problem. OR-Library[1] instances were used.

To solve the aforementioned Scheduling problem, a Multi-Agent System was proposed, consisting in Task and Resources agents, in which Resource Agents apply different MH, in order to produce effective scheduling plans. Used MH includes Tabu Search, Simulated Annealing, Genetic Algorithms, Ant Colony Optimization, Particle Swarm Optimization, and Artificial Bee Colony. The Multi-Agent System is detailed described in [5, 10] and previous work can be found in [3, 11, 12].

In order to take advantage of the potential of Racing and CBR modules, in this work a hybrid approach (Racing+CBR) is proposed, which employs Racing to initialize the casebase and uses CBR to solve new cases based on past experience. The proposed approach aims to select a technique and perform parameter tuning, depending on the instance to solve.

All obtained results correspond to makespan minimization (completion time - Cmax). Each instance was performed 5 times, and the average of the values obtained was calculated. In order to normalize the values, the ratio between the optimum value and the average value of Cmax is used, to estimate the deviation of the value obtained from the value of the optimal solution referenced in the literature.

In this hybrid approach, 150 initial cases were included, with values corresponding to the parameters in instances throughout the study Racing (6 results in 25 MH instances). Using the initial case base filled with the results obtained from Racing, it was carried out after the development of the study of the use CBR, running the system

[1] http://people.brunel.ac.uk/∼mastjjb/jeb/info.html.

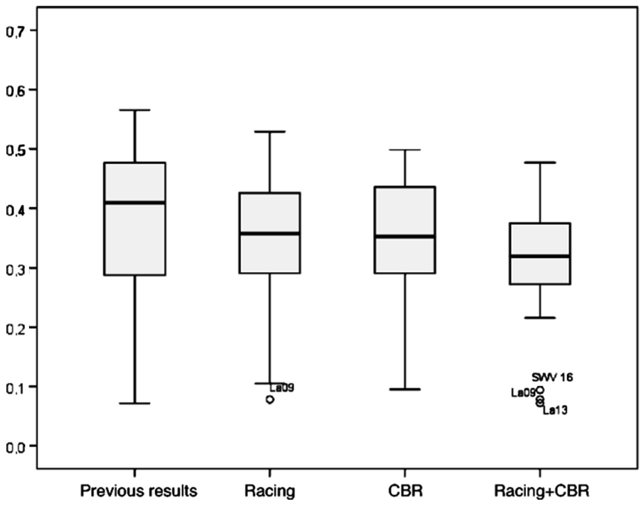

Fig. 2. Average values ratio comparison between initial (without learning), Racing, CBR and Racing+CBR results

in test instances. Table 3 shows the analysis of the ratio of the mean Cmax with all learning all approaches, including the Racing+CBR. From Fig. 2 chart analysis it is possible conclude on the advantage of the hybrid approach analyzing median and dispersion of data.

Table 3. Analysis of average values ration from initial (without learning), Racing, CBR and Racing+CBR results

	Min. value	Max. value	Avg.	Std. dev.
Initial	0.07	0.57	0.3792	0.13217
Racing	0.08	0.53	0.3474	0.11834
CBR	0.09	0.50	0.3388	0.10736
Racing+CBR	0.07	0.48	0.3142	0.10999

Comparing the results of the hybrid approach with the results of Racing module, we can see a slight improvement of the results, especially in the dispersion, which indicates that the inclusion of CBR module can maintain greater consistency of results. As shown in Table 3, there is an average and a standard deviation lower than in the Racing module, and moreover a minimum value and a maximum value below which is important since this is a minimization problem.

Comparing the hybrid approach with the results of CBR module, the improvement is not as significant, which expects to join the two modules can be advantageous. In Table 3, improvements are noted in minimum and maximum value, and the average, but the standard deviation is slightly higher, although the difference is minor.

The results of the Racing+CBR approach were compared with the initial results. And here, clear improvements are shown, which indicates a statistically significant advantage in the use of learning algorithms in MH parameter tuning for optimization problems. In Table 3 one can note significant improvements, especially in the middle of the results and the standard deviation. The minimum value was equal but the maximum value is significantly lower. Finally, comparing the first two approaches with the hybrid approach, you can check on the advantage of use of the latter.

At this point it becomes important to analyze the statistical significance of these results. Comparing the previous results and the results of the hybrid approach (Table 4) the values t (29) = 5.475, $p < 0.05$ allows to conclude, with a degree of confidence of 95%, that there are statistically significant differences between the initially obtained results (without training) and the results obtained by the Racing+CBR approach. Observing this, we can conclude about the advantage of using the Racing+CBR approach. In the other hand, comparing the results of the Racing+CBR approach with the results of Racing, and observing the values t (29) = 5.068, $p < 0.05$, one can also say with a confidence level of 95%, there are statistically significant differences between the two approaches, with advantages for the hybrid approach. Finally, comparing the results of CBR with the results of the hybrid approach, you can also complete, with a confidence level of 95%, there are statistically significant differences between the two approaches, as t (29) = 2.581; $p < 0.05$.

Table 4. Student's *t* test for paired samples

	Avg.	Std. dev.	*t*	DoF	p-value
Initial vs. Racing+CBR	0.06496	0.06499	5.475	29	0.000
Racing vs. Racing+CBR	0.03314	0.03582	5.068	29	0.000
CBR vs. Racing+CBR	0.02454	0.05208	2.581	29	0.015

4 Conclusions and Future Work

The objective of this paper was to present a novel approach for the selection and tuning of MH, in the resolution of Scheduling problems. The proposed hybrid approach is based on Racing and CBR. While Racing is used to evaluate a set of candidates and release those that appear to be less promising during the evaluation process, in a refined and efficient way, CBR which uses previous similar cases to solve new cases, providing a learning from experience. Considering the importance of tuning in the process of designing and implementing MH, a computational study was performed to evaluate the effectiveness of the proposed Racing+CBR approach.

The presented computational study aimed to evaluate the performance of a Multi-Agent Scheduling System, with the incorporation of different learning modules. The results of the hybrid Racing+CBR approach were compared with the results obtained previously, without the incorporation of learning mechanisms. All results were validated by analyzing the statistical significance and a significant statistically advantage in the use of Racing+CBR learning module was confirmed. All the presented approaches have improved the results obtained the system in relation to the previous results, however the hybrid Racing+CBR approach achieved better average results. From the results, it was possible to conclude about the existence of statistical evidence on the inclusion of learning on the process of MH tuning.

In the future, we intend to make a more extensive computational study and also compare our novel contribution with other learning techniques, as such Reinforcement Learning, for instance.

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/UI0760/2014.

References

1. Pinedo, M.: Scheduling: Theory, Algorithms, and Systems. Springer, Heidelberg (2016). ISBN 978-3-319-26580-3
2. Baker, K.R., Trietsch, D.: Principles of Sequencing and Scheduling. Wiley, New York (2009)
3. Madureira, A.: Meta-heuristics application to scheduling in dynamic environments of discrete manufacturing, Ph.D. Dissertation, University of Minho (2003). (in portuguese)
4. Talbi, E.-G.: Metaheuristics - From Design to Implementation. Wiley, New York (2009)
5. Pereira, I.: Intelligent system for scheduling assisted by learning, Ph.D. thesis in Electrical and Computer Engineering, Department of Electrical and Computer Engineering, University of Trás-os-Montes and Alto Douro (2014). (in portuguese)
6. Birattari, M., Balaprakash, P., Dorigo, M.: The ACO/F-RACE algorithm for combinatorial optimization under uncertainty. In: Doerner, K.F., Gendreau, M., Greistorfer, P., Gutjahr, W., Hartl, R.F., Reimann, M. (eds.) Metaheuristics. Operations Research/Computer Science Interfaces Series, vol. 39, pp. 189–203. Springer, Heidelberg (2007)
7. Box, G., Hunter, J., Hunter, W.: Statistics for Experimenters: Design, Innovation, and Discovery. Wiley, New York (2005)
8. Birattari, M., Stutzle, T., Paquete, L., Varrentrapp, K.: A racing algorithm for configuring metaheuristics. In: Langdon, W.B., et al. (eds.) Genetic and Evolutionary Computation Conference, pp. 11–18. Morgan Kaufmann Publishers Inc., San Francisco (2002)
9. Kolodner, J.: Case-Based Reasoning. Morgan Kaufmann, San Francisco (2014)
10. Madureira, A., Santos, J., Pereira, I.: A hybrid intelligent system for distributed dynamic scheduling. In: Chiong, R., Dhakal, S. (eds.) Natural Intelligence for Scheduling, Planning and Packing Problems, Studies in Computational Intelligence. SCI, vol. 250, pp. 295–324. Springer, Heidelberg (2009)
11. Madureira, A., Santos, J., Pereira, I.: MASDSheGATS – scheduling system for dynamic manufacturing environments. In: MultiAgent Systems. In-Tech (2009)
12. Pereira, I., Madureira, A.: Self-Optimization module for Scheduling using Case-based Reasoning. Appl. Soft Comput. **13**(3), 1419–1432 (2013). Elsevier