Computers & Operations Research 76 (2016) 95–117

Contents lists available at ScienceDirect

Computers & Operations Research

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

Adaptive online scheduling of tasks with anytime property on heterogeneous resources

István Módos [a,c,*], Přemysl Šůcha [a], Roman Václavík [a], Jan Smejkal [b], Zdeněk Hanzálek [a,c]

[a] Department of Control Engineering, Faculty of Electrical Engineering, Czech Technical University, Karlovo náměstí 13, 121 35 Prague 2, Czech Republic
[b] Merica, U Ládek 353/37, 251 01 Říčany – Strašín, Czech Republic
[c] Czech Institute of Informatics, Robotics, and Cybernetics, Czech Technical University in Prague, Zikova street 1903/4, 166 36 Prague 6, Czech Republic

ARTICLE INFO

Article history:
Received 4 August 2015
Received in revised form
18 March 2016
Accepted 10 June 2016
Available online 16 June 2016

Keywords:
Online scheduling
Anytime algorithms
Machine learning
Adaptive systems

ABSTRACT

An acceptable response time of a server is an important aspect in many client–server applications; this is evident in situations in which the server is overloaded by many computationally intensive requests. In this work, we consider that the requests, or in this case *tasks*, generated by the clients are instances of optimization problems solved by anytime algorithms, i.e. the quality of the solution increases with the processing time of a task. These tasks are submitted to the server which schedules them to the available computational resources where the tasks are processed. To tackle the overload problem, we propose a scheduling algorithm which combines traditional scheduling approaches with a quality control heuristic which adjusts the requested quality of the solutions and thus changes the processing time of the tasks. Two efficient quality control heuristics are introduced: the first heuristic sets a global quality for all tasks, whereas the second heuristic sets the quality for each task independently. Moreover, in practice, the relationship between the processing time and the quality is not known *a priori*. Because it is crucial for scheduling algorithms to know at least the estimation of these relationships, we propose a general procedure for estimating these relationships using information obtained from the already executed tasks. Finally, the performance of the proposed scheduling algorithm is demonstrated on a real-world problem from the domain of personnel rostering with very good results.

1. Introduction

An important aspect of client–server applications is the response time of the server. In a case of computationally intensive requests, e.g. optimization problems, the issue of the response time is even more pressing because the server can be easily overwhelmed even by a small number of requests.

Due to financial reasons, the computational capacity of a server is commonly scaled to handle a typical workload, i.e. the arrival rate and the computational complexity of the requests, so that the response time during this typical workload is kept at an acceptable level. In a case of sudden increase in the requests, the server may become easily overloaded and the response time increases significantly resulting in user dissatisfaction. One possibility of how to mitigate the increased response time during the overload is to buy more computational resources, but such solution is not financially suitable if the overload occurs a few times a day. However, if the requests or some of the requests are instances of optimization problems, it is possible to maintain an acceptable response time by moderate degradation of the solution quality, i.e. to trade-off a small decrease in a solution quality for a significantly shorter response time.

In this paper, we consider a scheduling problem illustrated in Fig. 1. *Users* work with *client applications* which generate *tasks*. The tasks are sent to a *scheduling system* which schedules the received tasks to *computational resources*. The resources are *heterogeneous*, i.e. each resource may have a different processing power and, therefore, the processing time of the tasks may vary on each resource. The task is processed on the assigned resource, and once it is finished, its result is sent back to the scheduling system which distributes the result to the respective client application. The scheduling system receives the tasks progressively through time, i.e. we are dealing with an *online scheduling* problem [14,26].

* Corresponding author.
E-mail addresses: modosist@fel.cvut.cz (I. Módos), suchap@fel.cvut.cz (P. Šůcha), vaclarom@fel.cvut.cz (R. Václavík), smejkal@merica.cz (J. Smejkal), hanzalek@fel.cvut.cz (Z. Hanzálek).

http://dx.doi.org/10.1016/j.cor.2016.06.008
0305-0548/

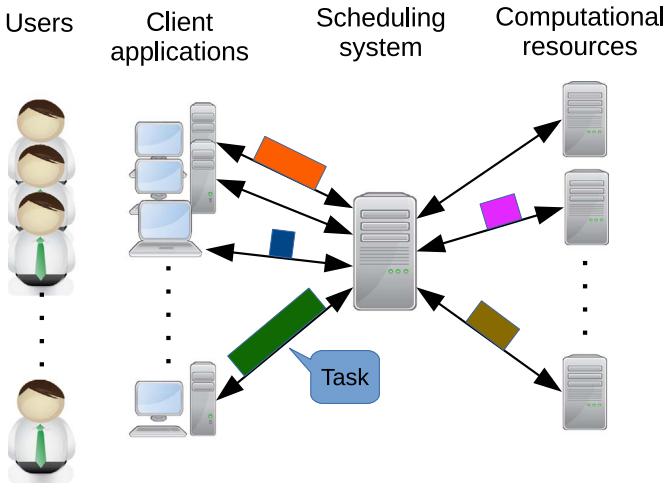

Fig. 1. Overview of the environment.

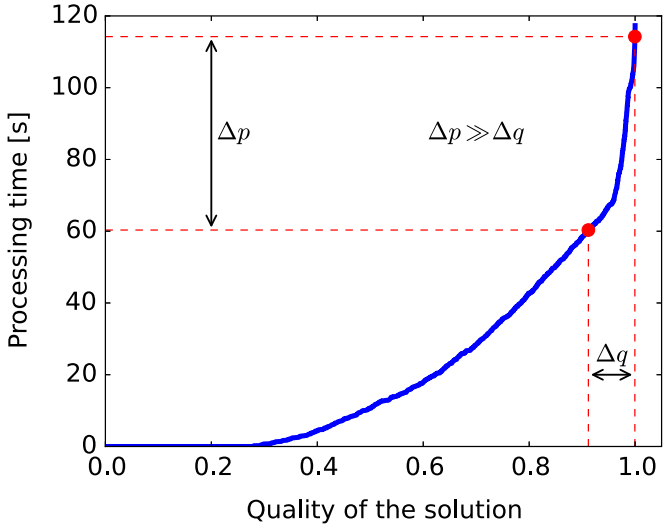

Fig. 2. A typical example of the processing time function of one task.

The tasks are instances of some optimization problems and are solved by *anytime algorithms*. The property of an anytime algorithm is that the processing of a task can be interrupted at any time and the algorithm returns a feasible solution if such solution exists. The quality of the solution depends on the processing time of a task, i.e. a longer processing time may result in a better solution (the quality of a solution is defined using the objective function of the tasks' optimization problem, i.e. how close the solution is to the optimal/near optimal solution). This behavior is typical for the majority of metaheuristics and hyperheuristics solving optimization problems. The relationships between the processing time and the solution quality are defined by *processing time functions*. A typical example of a processing time function of one task is illustrated in Fig. 2. In general, these functions have an increasing character: to get a better solution, an anytime algorithm must perform more operations or explore a larger part of the solution space. From Fig. 2, it can also be seen that a slight deterioration of the solution quality can significantly shorten the processing time of the task and thus reduce the response time of the system. From the user point of view, a good solution is better then excessive waiting time for the near-optimal/optimal solution.

In reality, the processing time functions are not known *a priori* as is usually considered in the related literature. The reason for this is that the anytime algorithms search the solution space and the algorithms are not generally aware where the good solutions are. Without any knowledge of the processing time functions, the scheduling system cannot guarantee the response time because the scheduling system does not know how long the processing of the tasks will take. However, using either statistical or machine learning methods, the processing time for the given quality can be estimated from the previous executions of similar tasks.

In this study, we focus on the situations in which the scheduling system is *overloaded*, i.e. the response time of the system increases significantly due to increased workload. The idea of how to tackle the system overload is to control the requested quality of solutions, i.e. when the overload of the system is detected, the system trades off the quality of solutions so that the response time is kept close to the acceptable level. On the other hand, the system requests the highest quality of the solutions if the overload is not detected.

We want to emphasize that our solution does not substitute *clouds* [9]. In fact, a cloud can be integrated into the scheduling system as a