Available online at www.sciencedirect.com

ScienceDirect

Applied Soft Computing 8 (2008) 706–721

Applied Soft Computing

www.elsevier.com/locate/asoc

Development of scheduling strategies with Genetic Fuzzy systems

Carsten Franke [a,1], Frank Hoffmann [b], Joachim Lepping [a,*], Uwe Schwiegelshohn [a]

[a] *Robotics Research Institute, Section Information Technology, University Dortmund, D-44221 Dortmund, Germany*
[b] *Control System Engineering, University Dortmund, D-44221 Dortmund, Germany*

Received 5 May 2006; received in revised form 4 April 2007; accepted 30 May 2007
Available online 2 June 2007

Abstract

This paper presents a methodology for automatically generating online scheduling strategies for a complex objective defined by a machine provider. To this end, we assume independent parallel jobs and multiple identical machines. The scheduling algorithm is based on a rule system. This rule system classifies all possible scheduling states and assigns a corresponding scheduling strategy. Each state is described by several parameters. The rule system is established in two different ways. In the first approach, an iterative method is applied, that assigns a standard scheduling strategy to all situation classes. Here, the situation classes are fixed and cannot be modified. Afterwards, for each situation class, the best strategy is extracted individually. In the second approach, a Symbiotic Evolution varies the parameter of Gaussian membership functions to establish the different situation classes and also assigns the appropriate scheduling strategies. Finally, both rule systems will be compared by using real workload traces and different possible complex objective functions.
© 2007 Elsevier B.V. All rights reserved.

Keywords: Scheduling algorithm development; Online scheduling; Genetic Fuzzy system; Symbiotic Evolution

1. Introduction

In this paper, we address an online scheduling problem with independent jobs submitted by different users. Unfortunately, for those problems, most common simple scheduling objectives, like the makespan [20], the total weighted completion [31] or response time [47] are not sufficient to represent the intentions of the machine provider. As example take the scheduling of computational jobs on a parallel computer system. There the users typically are partitioned into a small set of user groups which are assigned different priorities. Today, parallel computer systems are increasingly part of Computational Grids, that is, users from other sites with often low priority use those systems as well. As the common scheduling objectives are not suited to efficiently handle those priorities, the machine provider often sets quotas to prevent lower priority groups to occupy too many resources. However, those quotas tend to reduce machine utilization significantly.

Similar problems occur in many other application areas of practical importance, like telecommunications and logistics. Nevertheless, we focus in this paper on parallel computer systems as for those systems, real trace data are available that can be used to evaluate new approaches, see, for instance, the Standard Workload Archive [16], described by Chapin et al. [10]. It is vital to use real data to develop good scheduling algorithms as there is no method that guarantees an optimal or almost optimal solution for all input data.

The scheduling of parallel computer systems is an online problem as jobs are submitted over time and the precise processing times of those jobs are frequently not known in advance. Furthermore, information about future jobs are usually not available [25]. Formally, each job j is part of a job system τ and belongs to a user group. It is characterized by its degree of parallelism m_j, its processing time p_j, and additional criteria, see Feitelson and Nitzberg [17]. During the execution phase, job j requires the concurrent and exclusive access to $m_j < m$ processing nodes with m being the total number of nodes on the parallel computer system. The number of required processing nodes m_j is available at the release date r_j of job j and does not change during the execution. As the network does not favor any subset of the nodes and all nodes of a parallel computer system are either identical or very similar, we assume that a job j can be processed on any subset of m_j

* Corresponding author. Tel.: +49 231 755 4657; fax: +49 231 755 3251.
E-mail addresses: carsten.franke@udo.edu (C. Franke), frank.hoffmann@udo.edu (F. Hoffmann), joachim.lepping@udo.edu (J. Lepping), uwe.schwiegelshohn@udo.edu (U. Schwiegelshohn).
[1] Born Carsten Ernemann.

1568-4946/$ – see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.asoc.2007.05.009

nodes of the system. Note that in the rest of this paper, we replace the expression node by machine as this is the common terminology in scheduling problems.

As already mentioned, the processing time p_j is not known before the job has been completed. However, some systems require the users to provide an estimate $\bar{p}_j$ of this processing time. If the actual processing time of a job exceeds this estimate, the job is canceled to protect the system and the user from the costs of additional resource consumption by a possibly faulty job. In addition, this estimate is also used for some scheduling algorithms, like Backfilling [33]. Unfortunately, those estimates are of limited use for scheduling purposes as users tend to overestimate the processing time of their jobs in order to avoid the termination of correct jobs, see, for example, Lee et al. [32]. Most current real installations of parallel computers do not use preemption but let all jobs run to completion unless they are terminated as discussed above. The completion time of job j within the schedule S is denoted by $C_j(S)$. Furthermore, precedence constraints between jobs are rare and it is almost impossible for the machine owner to detect them if they exist. Hence, we assume that all jobs are independent.

Contrary to many theoretical online scheduling problems addressed previously, see, for instance, Albers [2], the scheduling of job j does not need to take place directly after its release date r_j. Instead, the allocation is established when sufficient machines become available, that is just before the start of job j at time $(C_j - p_j)$. Due to the frequency of job submissions and the size of m for many parallel computers, scheduling decisions must still be made within a short period of time after the release of a job. As information about future job submissions is not available compute intensive quasi offline scheduling algorithms cannot be used in this scenario.

In general, it is very difficult to optimally solve most scheduling problems. This is even true for offline problems with simple scalar objective functions as many of them are NP-hard [21]. Therefore, existing online scheduling systems at real installations mainly use heuristics, like Greedy Scheduling [24] in combination with a First Come First Serve (FCFS) approach and the above-mentioned Backfilling [33,22]. Although those algorithms produce a very low utilization in the worst case [41] they yield reasonably good results in practice [18]. But they do not allow different priorities for jobs or user groups apart from exclusively assigning computer partitions to user groups [28], limiting the total processing time of a user group, or statically weighing the waiting time of jobs. All those algorithms are rather cumbersome and often lead to a low utilization as already mentioned. Recently, there have been suggestions to consider market oriented scheduling approaches [8] which accept more flexible objectives and are able to adapt to variations in the job submission pattern. However, existing research projects in this area only use two simple linear objective functions: time and cost minimization [7]. So far no really flexible scheduling algorithm has been developed for parallel computer systems. Furthermore, our method is applicable to a broad range of resource allocation and scheduling problems in grid application. Aggarwal and Kent [1] present a scheme to adapt scheduling decisions to the grid infrastructure, dynamic workload and envrionmental conditions. The adaptation scheme employs a heuristic for optimal allocation of jobs to nodes within the grid, but does not consider costs or rewards explictly to adapt the scheduling algorithm in the context of the dynamic workload.

Within this work, we present a methodology to automatically generate a rule-based scheduling system that is able to produce good quality schedules with respect to a given complex objective. This objective defines the prioritization of different user groups. Even if different providers use the same basic objectives for the various groups the transformation of a generic multi-objective scenario into a specific scheduling problem with a single objective depends on the actual priorities assigned to the user groups and is likely to be individual. Hence, we focus on the development of a suitable methodology. To this end, we present a rule-based scheduling that is able to adapt to various scenarios. So far, the use of rule-based systems in scheduling environments is rare. Nevertheless, first attempts [13,9] have shown the feasibility of such an approach. However, those scheduling systems are all based on single objective evaluation functions that are not optimized.

The proposed scheduling process is divided into two steps. In the first step, the queue of waiting jobs is reordered according to a sorting criterion. Then an algorithm uses this order to schedule the jobs onto the available machines in the second step. Based on the present scheduling state, the rules determine the sorting criterion and the scheduling algorithm. In order to guarantee general applicability, the system classifies all possible scheduling states. This classification considers the actual schedule, the current waiting queue, and additional external factors, like the time of day.

The problem of sequential making decision making in complex domains has gained substantial attention in the past [38]. Classical approaches for dynamic programming and reinforcement learning soon become infeasible due to the complexity of the state-action space. Therefore, Schoknecht et al. [38] present an attempt to reduce the search space for the optimal policy by adapting and restricting the set of possible alternative actions in order to accelerate the learning. Even though the application is concerned with optimal control, the method is applicable to general Markov decision processes. This paper pursues a similar objective, namely reduction of the search space complexity, but differs in so far as it granularizes the state space rather than the action space. The fuzzy partition of the state space lumps together originally distinct states and associates them with the same action.

As already mentioned, the development of scheduling strategies for parallel computers is typically based on real workload traces. Those traces implicitly include all relevant characteristics and hidden properties, like temporal patterns, dependencies between jobs, or certain user behaviors. As local scheduling decisions influence the allocation of future jobs, the effect of a single decision cannot be determined individually. Therefore, the whole rule base is only evaluated after the complete scheduling of all jobs belonging to a workload trace. This has a significant influence on the learning method to