J Sched (2008) 11: 371–379
DOI 10.1007/s10951-008-0058-8

A multicriteria approach to two-level hierarchy scheduling in grids

Krzysztof Kurowski · Jarek Nabrzyski · Ariel Oleksiak · Jan Węglarz

Published online: 8 February 2008

Abstract In this paper we address a multicriteria scheduling problem for computational Grid systems. We focus on the two-level hierarchical Grid scheduling problem, in which at the first level (the Grid level) a Grid broker makes scheduling decisions and allocates jobs to Grid nodes. Jobs are then sent to the Grid nodes, where local schedulers generate local schedules for each node accordingly. A general approach is presented taking into account preferences of all the stakeholders of Grid scheduling (end-users, Grid administrators, and local resource providers) and assuming a lack of knowledge about job time characteristics. A single-stakeholder, single-criterion version of the approach has been compared experimentally with the existing approaches.

Keywords Grid computing · Grid resource management and scheduling · Multicriteria decision support

1 Introduction

The Grid computing paradigm was developed in order to support computation intensive applications and collaborative environments. In such Grid environments usually there is one common, central Grid broker, that serves all the end-users and their jobs. The majority of centralized Grid environments are based on the *two-level hierarchy scheduling* approach consisting of many remote sites, each of which is usually managed by a local scheduling or cluster management system, such as LSF, PBS, SGE, Condor, etc. Thus, in the first step, the Grid broker assigns Grid jobs to resource providers and then local schedulers generate their schedules for resources they manage. This concept is very natural, since a Grid broker neither possesses a full knowledge of the local resource load nor has overall control of local resources. On the other hand, local schedulers do not know about any other Grid jobs and other resources available for these jobs (Open Grid Forum 2008).

In this paper we propose a multicriteria Grid scheduling approach for Grid environments with two-level hierarchy scheduling. The methodology takes into account preferences of all participants of a scheduling process, i.e. end-users, Grid administrators and resource providers altogether called *stakeholders*. We show how to aggregate various criteria to find a schedule that satisfies stakeholders' preferences and requirements in the best possible and flexible way. Moreover, we demonstrate how to apply our scheduling approach for both online (where a set of jobs is unknown beforehand by a Grid broker) and offline scheduling problems. Additionally, we perform various experiments to compare a single-stakeholder, single-criterion version of our approach with some existing scheduling strategies. In our qualitative performance analysis we use naive synthetic workload models to introduce a manageable experimental medium for Grid scheduling that is free of site-specific behavior of production systems and can be easily simulated by other researchers.

The rest of the paper is organized as follows. Section 2 contains state-of-the-art and related work. In Sect. 3 we present a multicriteria approach to two-level hierarchical

K. Kurowski · J. Nabrzyski (✉) · A. Oleksiak · J. Węglarz
Poznan Supercomputing and Networking Center, Poznan, Poland
e-mail: naber@man.poznan.pl

K. Kurowski
e-mail: krzystof.kurowski@man.poznan.pl

A. Oleksiak
e-mail: ariel@man.poznan.pl

J. Węglarz
Institute of Computing Science, Poznan University of Technology, Poznan, Poland

Grid scheduling problem. Section 4 describes a single-stakeholder, single-criterion version of the approach. In Sect. 5 the comparative simulation experiment is given. Section 6 contains final remarks and suggestions of future work.

2 Related work

In most approaches to Grid scheduling and resource management single criterion problems are considered, where a single stakeholder (usually a Grid administrator) imposes the scheduling strategy. Many systems have been developed so far, based on such an approach, e.g. EGEE Workload Management System (Avellino et al. 2003), NorduGrid broker (Elmroth and Tordsson 2005), eNANOS (Rodero et al. 2005), GridWay (Huedo et al. 2005), and others. Few groups worldwide are working on multicriteria approaches, although it seems straightforward to consider Grid resource management and scheduling as a multicriteria problem. In (Dutot et al. 2005, 2004; Dutot and Trystram 2005) the authors consider the bi-criteria algorithm for scheduling jobs on clusters of resources. Two pre-selected criteria are used, namely makespan and average completion time, for moldable jobs scheduling. Preferences of particular end-users have been taken into account in (Siddiqui et al. 2006) for negotiation-based scheduling using advance reservation. End-user preferences have been modeled as utility functions for which end-users have to specify required values and negotiation levels. In (Kurowski et al. 2001) we have presented user preference driven multi-objective scheduling strategies for Grids. The idea presented in (Kurowski et al. 2001) has then been extended to multi-stakeholder case and further developed in (Kurowski et al. 2003, 2006a, 2006b). The idea has been successfully implemented in *Grid Resource Management System (GRMS)* that operates today in several production Grid environments (GridSim 2008).

Unfortunately, it is still difficult to perform experimental performance evaluation studies of Grid scheduling algorithms. The use of real workloads (e.g. Grid5000, EGEE, DAS2, Grid Workloads Archive 2008) has serious drawbacks. This follows from the fact that real workloads usually come from single and often independent local clusters, collected under specific conditions, and do not contain information about the many parameters that could affect performance evaluation (Li et al. 2007). Many authors (Downey 1997a, 1997b), and (Kurowski and Nabrzyski 2000) stress that one should be careful when using real workloads, recommending that a workload derived from one system should not be used to evaluate another one. The reason for such recommendations is that real workload models could be affected by local site policies and constraints.

Another approach is to use synthetic workloads. Various synthetic workloads models for parallel machines have been recommended by numerous researchers to offer the convenience of a much more manageable experimental medium that is free of the idiosyncratic behavior of production systems. Interestingly, also the naive synthetic workload models have been proved to be useful in qualitative performance analysis. Comprehensive studies presented in (Lo et al. 1998; Lublin and Feitelson 2003) show that the choice of workload trace alone does not affect the relative performance of the resource management algorithms. Most workloads, regardless whether they were real or synthetic, across sites, or for different time periods at the same time, ranked scheduling algorithms in the same order from best to worst with respect to the slowdown ratio and system utilization. However, models for generating synthetic Grid workloads are still emerging and subject to further research. Taking all this into account we decided to use a naive synthetic workload to compare Grid scheduling strategies in this paper.

3 Multicriteria approach to two-level hierarchy scheduling

In our multicriteria approach, a Grid broker is the main decision point of the scheduling process. It takes into account the requirements and preferences of all the stakeholders and assigns end-user jobs to resources offered by resource providers in the Grid environment managed by a Grid administrator (see Fig. 1). Stakeholders, i.e. end-users, re-

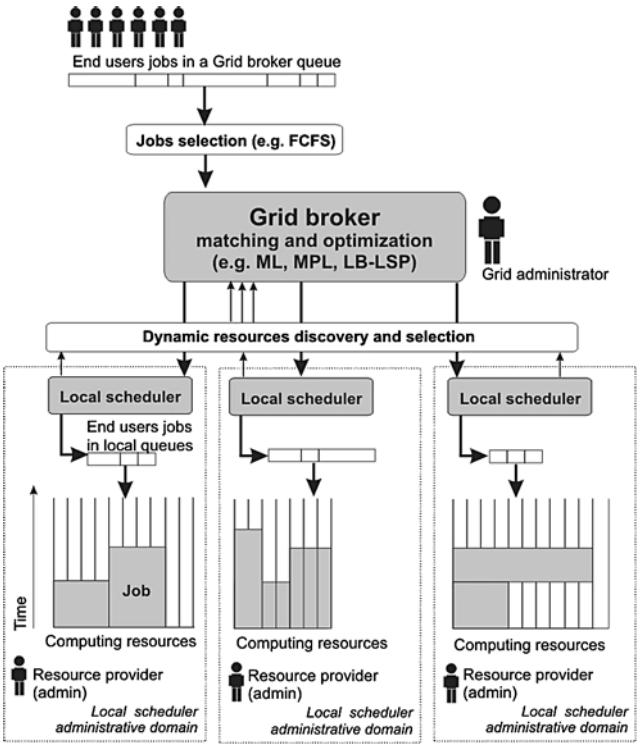
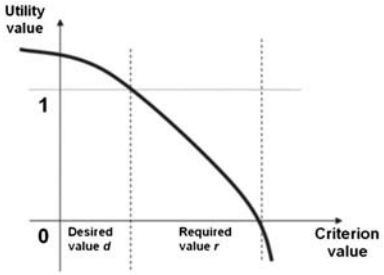


Fig. 1 The typical two-level configuration of Grid broker and local schedulers in computational grids

Fig. 2 The scaling function



$$s_i(g_i) = \begin{cases} \gamma \frac{d_i - g_i}{r_i - d_i} + 1, & \text{for } g_i \leq d_i; \\ \frac{r_i - g_i}{r_i - d_i}, & \text{for } d_i < g_i \leq r_i; \\ \delta \frac{r_i - g_i}{r_i - d_i}, & \text{for } g_i > r_i. \end{cases}$$

source providers, and Grid administrators, can express their requirements and preferences regarding various parameters and criteria of the scheduling process. In order to express their requirements they specify *hard constraints*, i.e. requirements that must be satisfied before optimization and assignment procedures are invoked in the Grid broker. In practice hard constraints determine the set of feasible solutions. Stakeholders' preferences are expressed by *soft constraints*. Soft constraints describe preferences regarding multiple criteria, e.g. various performance factors, QoS-based parameters, reliability as well as specific dynamic characteristics of local schedulers, local queues and resources. In contrast to hard constraints, which must be obeyed, soft constraint are of secondary significance. However, as we have demonstrated in (Kurowski et al. 2006b), soft constraints should be taken into consideration to improve scheduling procedures of Grid broker. They drive the process of selecting best solutions.

After assignment has been made at the Grid broker level, local schedulers can then allocate jobs from local queues to processors according to various scheduling strategies such as First Come First Served *(FCFS)*, Largest-Size-First *(LSF)* or Backfilling. Note that in the approach we do not assume any knowledge about job time characteristics (e.g. job arrival time, job execution time, job runtime, etc.).

3.1 Preference modeling

In the previous work (see Kurowski et al. 2003, 2005, 2006a, 2006b) we have proposed an approach for modeling and exploiting stakeholders' preferences in Grid scheduling. In this work we improve this model by adding a more convenient scaling function and a more universal aggregating operator, and by defining a local search procedure for finding good quality schedules.

For the sake of simplicity and efficiency, our approach is based on the functional model in which an explicit expression of preferences is requested from the stakeholders. Since the particular criteria are expressed using different units and scales, they are incomparable in their original form. Therefore, appropriate scaling functions are used. Scaling methods often use some reference values given by stakeholders to make sure these functions really reflect preferences concerning the particular criteria. For instance, this idea is used in reference point methods (Lewandowski and Wierzbicki 1989). In our approach the following scaling function has been used, assuming, without loss of generality, that the ith criterion is to be minimized, where g_i is the value of the ith criterion, d_i is the desired value of this criterion (also called an *aspiration level* Lewandowski and Wierzbicki 1989) and r_i is its required value (also called a *reservation level* Lewandowski and Wierzbicki 1989) (see Fig. 2). The desired value represents a value which satisfies a stakeholder, while the required one is the maximal value that is still profitable. This means that for values below d_i the utility function increases more slowly than between d_i and r_i (stakeholder's benefits from decreasing this value are small). Within this range a gradient of the utility function is defined by the γ coefficient such that $0 < \gamma < 1$. On the other hand, if a value of the ith criterion is above r_i, the function decreases faster than within the range (d_i, r_i), which ensures a certain penalty for bad values of the criterion. In this case, coefficient $\delta > 1$ is used. If the desired and required values are not provided by a stakeholder, then 0 and maximal available values are taken as the desired and required values, respectively (for minimization). However, it may lead to inaccurate preference modeling, since adding a solution with a very bad score on a given criterion distorts the evaluation of the remaining solutions. It is worthy of note that we chose a piecewise linear function for the sake of computational robustness and simplicity, although in general scaling could be more adequately expressed using non-linear functions.

The use of a scaling function allows modeling preferences for every single criterion. However, since we consider a multi-criteria problem, a method for aggregation of multiple criteria is needed. For the sake of several profitable properties we decided to apply the Ordered Weighted Averaging (OWA) operator to aggregate the stakeholder's criteria. The OWA operator has originally been introduced by Yager (1988) to provide a means for aggregating scores associated with the satisfaction of multiple criteria, which unifies the conjunctive and disjunctive behavior in one operator:

$$OWA(x_1, x_2, \ldots, x_n) = \sum_{j=1}^{n} w_j x_{\sigma(j)}, \tag{1}$$

where $x_i, i = 1, \ldots, n$ is a score associated with the satisfaction of the ith criterion and σ is a permutation that orders the

scores: $x_{\sigma(1)} \leq x_{\sigma(2)} \leq \cdots \leq x_{\sigma(n)}$. The weights are all non negative ($w_i \geq 0$) and their sum equals one ($\sum_{i=1}^{n} w_i$=1).

The operator turned out to be very useful because of its versatility. Compared to the common basic operators such as the weighted mean or the Chebyshev norm (as used in Kurowski et al. 2006b), OWA is more general. It is possible, by setting the weights in formula (1) conveniently, to transform an OWA operator to a chosen aggregation operator, e.g. minimum, maximum or arithmetic mean. To do so, a stakeholder may use a single "orness" parameter or other methods (Yager 1988). Setting the weight w_i to 1 and all the remaining weights to 0, the OWA operator becomes the minimum. In multicriteria analysis it means that we look for solutions that have good values of the worst criterion (i.e. good values of all criteria). If all the weights are set to the same value, OWA behaves as the arithmetic mean. In this case high values of some criteria compensate low values of the other ones.

In our approach we provide a compromise solution. The highest weight is w_1, and the subsequent ones are decreasing, but never reaching 0. This means that both the worst criterion and the mean of criteria are taken into account.

In this way stakeholders are able to evaluate schedules using multiple criteria. Additionally, a method taking into account partial evaluations made by all stakeholders is needed to evaluate all the schedules globally. Our approach to this problem is described in the next section.

3.2 Multi-stakeholder scheduling

In Grid environments it is generally difficult to allocate resources beforehand. Therefore, in most approaches to Grid scheduling, jobs first arrive at the Grid level queue and then, for each job separately, appropriate resources are selected. In this process, preferences of only one stakeholder, in this case end-user, are taken into account. There is a lack of studies on simultaneous scheduling of sets of Grid jobs taking into account preferences of many stakeholders. However, it is clear that an approach that takes into account multiple jobs, and therefore also preferences of multiple stakeholders, leads to better global performance.

The OWA operator can be applied for this purpose, where for multiple stakeholders we use a different weighting scheme than illustrated for the single stakeholder case in Sect. 3.1. Here the goal is to find a weighting scheme that provides the best possible mean value for all stakeholder evaluations and the highest possible value for the worst case evaluation. To achieve this, weight w_1 must be relatively large, while weight w_n should be small, where n now denotes the number of stakeholders. The remaining weights should decrease in value from w_1 to w_n. This formula is illustrated below:

$$w_i = \begin{cases} \frac{3}{2n}, & \text{for } i = 1; \\ \frac{1}{2n}, & \text{for } i = n; \\ \frac{3n-2i-1}{2n(n-1)}, & \text{for } 1 < i < n. \end{cases} \quad (2)$$

3.3 Scheduling procedure

Since jobs are continually submitted to a Grid broker, we take jobs from a Grid broker queue in *job packages (JP)*, where the *size of the package (JPS)* is a configurable parameter of a broker that should depend on properties of the Grid environment, workload, scheduling strategy, etc. We study the impact of job package size on the efficiency of scheduling in Sect. 5.2. For large job packages, it becomes too difficult to evaluate all the possible schedules. Therefore, one must apply some heuristic methods in order to find good solutions in a reasonable time. One possible method is a local search procedure.

In our approach, we use a greedy algorithm to generate an initial solution. We start by first sorting all the available resources for a particular job from best to worst by applying the OWA weighting scheme described in Sect. 3.2 to predefined criteria. Then for each job in the job package, we assign the best available resource from the ordered set of resources. We take the jobs from the job package in the order in which they were submitted to a Grid broker queue. Of course, the ranking of each resource will be affected after assigning jobs to it. Therefore, one must reorder the resource set after each job is assigned. Given this initial schedule, we want to see if there are better alternatives that can be generated through simple reassignment of jobs to resources. In general, there are many ways to generate a neighborhood of possible alternatives. The three approaches we considered included:

(1) Defining a set of schedules by swapping any two job allocations,
(2) Defining a set of schedules which differ by the allocation of a single job, where one moves a job from its current location in a queue to the end of the same queue or the end of another queue,
(3) Defining a set of schedules which differ by the allocation of a single job, where one moves a job from its current queue to an arbitrary position in the same queue or to an arbitrary position in another queue.

On the basis of experiments, the third approach proved to be the best one. The first approach generated neighborhoods that were too large to be efficiently evaluated, whereas the second generated neighborhoods that were too constrained, causing slow convergence. On the other hand, the third approach generated neighborhoods that were quite reasonable

in size and converged relatively quickly. The formal definition of the chosen neighborhood is given in Appendix. Note that we use a steepest local search algorithm to search generated neighborhoods.

4 Reduction to a single-stakeholder, single-criterion problem

In previous sections a general approach to multicriteria and multi-stakeholder scheduling in Grids was presented. According to our knowledge, other such approaches to two-level hierarchy scheduling do not exist. Therefore, in order to compare our approach with some existing ones, we had to reduce our model to a single-stakeholder, single-criterion case. In this case we assume that a Grid administrator is the only stakeholder imposing the scheduling objective.

An interesting analysis of typical two-level hierarchy scheduling strategies has been presented by Tchernykh et al. (2006) under the assumption that all the jobs are available at time 0. So we decided to compare the results produced by our reduced model with the Tchernykh's approach. In the aforementioned paper the authors have proposed to use, on a Grid broker level, the commonly known load balancing strategies to assign Grid jobs to local queues. Then, local scheduling has been performed using the LSF algorithm.

Based on the assumptions and formulations proposed in (Tchernykh et al. 2006) and (Kurowski et al. 2006b) we attempted to define the single-stakeholder, single criterion problem for two-level hierarchy Grid scheduling. Consider a set of jobs $J = (j_1, j_2, \ldots, j_n)$ in a Grid broker queue. In practice, depending on the utilization of processors, the submitted jobs have to wait in local queues before being executed. Thus, we assume that each queue has a certain number of waiting jobs $J_r = (r_1, r_2, \ldots, r_m)$.

Each job j requires a number of processors denoted as s_j. This parameter is also called *job size*. Each job is assigned to a local queue k for which $s_j \leq p_k$ in such a way that a certain criterion is optimized. In this model the co-allocation problem is not considered. Note that we consider a model with no time characteristics of jobs, and therefore the Grid broker is not able to build local schedules. In order to compare the two approaches we focused on the criteria that in (Tchernykh et al. 2006) are used as Grid level scheduling strategies:

- *Min-Load* (ML) takes the queue with the lowest load per processor (the number of jobs available in a local queue over the number of processors), i.e. $ML = \min \frac{r_k}{p_k}$, where k denotes a local queue.
- *Min-Parallel-Load* (MPL) takes the queue with the lowest parallel load per processor (the sum of job sizes over the number of available processors), i.e.

$$MPL = \min \frac{\sum_{j=0}^{j=r_k} s_j}{p_k}.$$

The other strategies that are used in (Tchernykh et al. 2006) assume that job time characteristics are known in advance, and thus they are not interesting from our model's point of view.

Both, ML and MPL are designed for simple list scheduling algorithms. They are used to choose the best queue for each job separately. As stated above, in our approach the Grid broker schedules many jobs at once. This is why the ML and MPL are not suitable for our reduced model. Therefore, we introduced another criterion to evaluate schedules. The criterion, called *Load Balance (LB)*, measures the balance of jobs in local queues and is calculated as a standard deviation of MPL values for all local queues:

- $LB = \min \sqrt{\frac{\sum (MPL - \overline{MPL})^2}{m}}$, where m denotes the number of available local queues for the Grid broker.

So, in our problem the stakeholder's objective is to schedule the jobs available in the Grid queue to available local queues, so that the LB criterion is minimized. To this end, we will apply the LSP procedure described in Sect. 3.3 denoting it by *LB-LSP*.

5 Simulation experiments

5.1 Description

We conducted our experiments with the Grid Scheduling Simulator (GSSIM) (GSSIM 2008). GSSIM is based on an enhanced version of GridSim (GridSim 2008; Milkiewicz 2005), where GridSim was extended to support additional simulation parameters, workloads and evaluation of multiple Grid scheduling algorithms. GSSIM allows generating workloads for a range of probability distributions of input parameters. It also offers a Grid scheduling benchmark suite that enables to compare experimental results. It is available, together with GSSIM, from the GSSIM web site.

In the experiments we used GSSIM to generate a synthetic naive workload to evaluate various scheduling strategies, i.e. ML, MPL, and LB-LSP, illustrated in the figure below.

The total number of jobs equals 500 in each experiment. Since in Grid environments the whole set of jobs is usually unknown in advance, we assumed that this set is divided into job packages as explained in Sect. 3.3. For each job package JP a scheduling procedure is applied. The set

Table 1 Parameters of workloads used in the evaluation experiments

Parameter	Mean	Standard deviation	Minimum	Maximum	Distribution
Job length	20TI	20TI	1		normal
Job size			1	16	uniform
Jobs' arrival ratio	3				Poisson

Table 2 The basic characteristics of queues and of processors used in the simulation experiments

	p_k	CPUSpeed [MIPS]
Local queue 1	8	2000
Local queue 2	8	2000
Local queue 3	16	2000
Local queue 4	16	2000
Local queue 5	32	2000
Local queue 6	32	2000
Local queue 7	64	2000
Local queue 8	64	2000
Local queue 9	128	2000

of jobs from a JP is scheduled up to the last one and then the subsequent JP is scheduled if available. We assumed that the first JP is available at time 0. Setting job package size JPS to a total number of jobs (500 in the case of our experiment) reduces the problem to a specific case that is offline scheduling problem as it has been originally considered for ML and MPL algorithms in (Tchernykh et al. 2006). The input workload and the resource demand for simulations were generated according to a normal distribution with the average job length = 20 trillions instructions *(TI)* and standard deviation = 20 TI. CPU requirements of these jobs were generated from the interval [1,16] according to a uniform distribution. The jobs' arrival ratio was based on a Poisson distribution with the average equal to 3. We repeated the experiments 50 times using different workloads generated with the same parameters presented in Table 1. The basic characteristics of queues and of processors used in the simulation experiments are summarized in Table 2.

5.2 Results

We illustrate the quality of our approach measured by the following evaluation criteria: *makespan*, *mean completion time*, *mean job waiting time*, *mean job execution time*.

In general, we observed that LB-LSP outperforms ML and MPL only for relatively big JPS, i.e. $JPS \geq 75$. A comparison of our LB-LSP scheduling procedure with ML and MPL for $JPS = 500$ is presented in Fig. 3. However, the LB-LSP generated solutions with more dispersed values of the evaluation criteria. Using standard deviation we measured how dispersed the values in a data set were (see Fig. 4).

Figure 5 presents values of the evaluation criteria with regard to job package size (*JPS*) used for the minimization of the LB criterion. As we see, the larger the sets of jobs are processed the better results are yielded by the LB-LSP.

Another parameter that significantly influences the efficiency of generated schedules is the job length and its diversity. Figures 6 and 7 present evaluation criteria for both, MPL and LB-LSP, as functions of the mean job length. Job length is given in Trillions Instructions *(TI)*.

5.3 Conclusions

In the experiments we observed that the performance of Grid level scheduling strategies strongly depends on the specific parameters of the workload and environment. Figure 3 shows that for relatively large job packages, scheduling simultaneously many jobs is more efficient than the list scheduling methods. However, increasing the size of job packages beyond a certain threshold does not affect the overall performance of the scheduling strategy. Figure 5 shows this "saturation" level. In the above experiments, this value was around 75 jobs per job package. Naturally, this threshold will depend on such parameters as arrival ratio, job length and size, etc., thus one will see different thresholds for different configurations.

Another parameter that significantly influences the efficiency of the considered method is the job length. In Fig. 6 one can see that the LB-LSP procedure outperforms the MPL for jobs longer than a specific length. In the above experiments, the use of LB-LSP procedure became profitable for jobs longer than 30 TI. All the considered evaluation criteria behave in a similar manner. Namely, we observed that if jobs are long enough then the order in which they arrive has a greater influence on the quality of the final schedule. Additionally, differences in job lengths also have a significant impact.

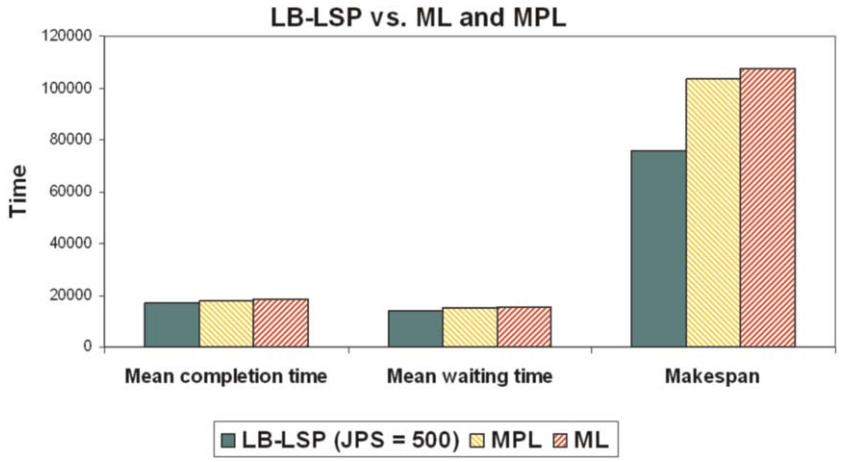


Fig. 3 Comparison of the LB-LSP with ML and MPL

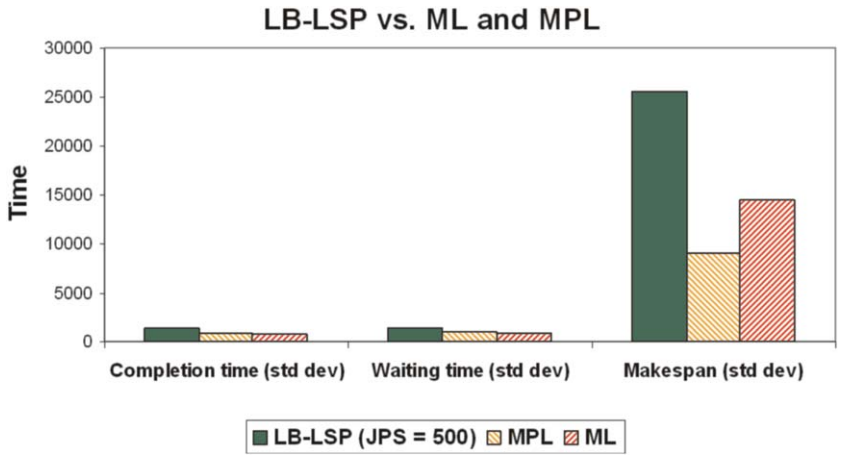


Fig. 4 Comparison of standard deviation of schedules generated by LB-LSP, ML, and MPL

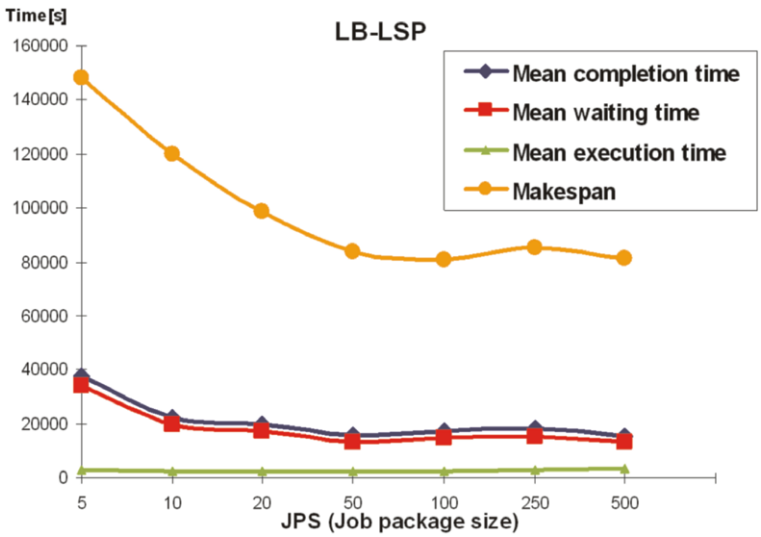


Fig. 5 Comparison of LB-LSP with regard to job package size (*JPS*)

6 Summary and future work

In this paper we presented a multicriteria approach for two-level hierarchy scheduling in Grids with unknown job time characteristics, where the basic idea is to schedule sets of jobs simultaneously, taking into account preferences of all stakeholders. To this end, we described an adopted scaling function and aggregating operator. We presented a simple and efficient local search procedure for scheduling jobs including a method for generating an initial solu-

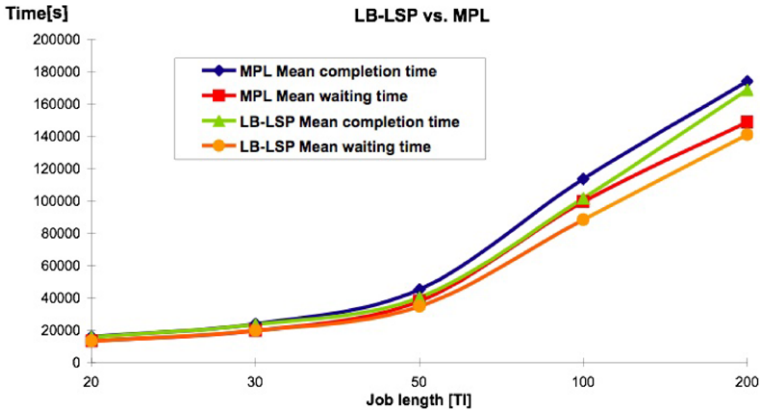


Fig. 6 Comparison of LB-LSP and MPL with regard to job length

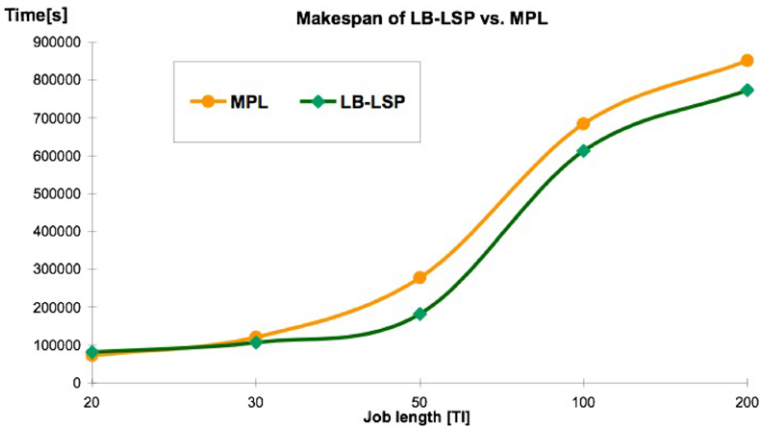


Fig. 7 Comparison of makespan of LB-LSP and MPL with regard to job length

tion and a definition of neighborhood. Finally, we applied this procedure to the single-criterion, single-stakeholder case and compared it experimentally to the existing approaches.

This research can provide the basis for further study on more advanced scheduling procedures, such as branch and bound algorithms, genetic algorithms, or modified versions of our local search procedure. Another direction to take this research would be to experiment with multicriteria optimization based on other available levels of knowledge about Grid environments. Workloads based on various real cases should be used to verify these results. Moreover, it would be useful to study the influence of job size (number of required processors), diversity of jobs, and resource configurations on the efficiency of different scheduling procedures. Such knowledge could then be used for dynamic and automated selection of scheduling algorithms by real Grid scheduling systems, depending on the workload and the state of the Grid environment.

Acknowledgements The authors are grateful to anonymous referees for their valuable remarks. Special thanks go to Michael Russell for his comments and fruitful discussions.

Appendix: Formal definition of neighborhood for the scheduling procedure described in Sect. 3.3

Let us consider the schedule $S = (J_1, \ldots, J_m)$, where m denotes a number of available local queues, J_i denotes the set of jobs in the ith queue, JP is the set of all jobs that are to be scheduled, i.e. $J_i \cap J_j = \emptyset$, $i \neq j$, $\bigcup J_i = JP$. Let us assume that there exist sets of jobs in queues such that $J_i = (j_{i1}, \ldots, j_{ik}, \ldots, j_{i|J_i|})$ and $J_j = (j_{j1}, \ldots, j_{jl}, \ldots, j_{j|J_j|})$, where $i, j = 1 \ldots m, k = 1 \ldots |J_i|, l = 1 \ldots |J_j|$, and $i \neq j$ or $(k \neq l$ and $k \neq l + 1)$. Then, a neighbor of this solution is an arbitrary solution $S' = (J_1, \ldots, J'_i, \ldots, J'_j, \ldots, J_m)$, where $J'_i = (j_{i1}, \ldots, j_{ik-1}, j_{ik+1}, \ldots, j_{i|J_i|})$ and $J'_j = (j_{j1}, \ldots, j_{jl}, j_{ik}, j_{jl+1}, \ldots, j_{j|J_j|})$.

References

Avellino, G., Barale, S., Beco, S., Cantalupo, B., Colling, D., Giacomini, F., Gianelle, A., Guarise, A., Krenek, A., Kouril, D., Maraschini, A., Matyska, L., Mezzadri, M., Monforte, S., Mulac, M., Pacini, F., Pappalardo, M., Peluso, R., Pospisil, J., Prelz, F., Ronchieri, E., Ruda, M., Salconi, L., Salvet, Z., Sgaravatto, M., Sitera, J., Terracina, A., Vocu, M., & Werbrouck, A. (2003). The EU DataGrid workload management system: towards the second major release. In *CHEP 2003*, La Jolla, CA, March 2003.

Downey, A. B. (1997a). A parallel workload model and its implications for processor allocation. In *Proceedings HPDC 97*.

Downey, A. B. (1997b). Predicting queue times on space-sharing parallel computers. In *Proceedings of the 2nd workshop on job scheduling strategies for parallel processing, IPPS*.

Dutot, P.-F., & Trystram, D. (2005). A best-compromise bicriteria scheduling algorithm for parallel tasks. In *Proceedings of WEA'05 4th international workshop on efficient and experimental algorithms*, Santorini Island, Greece, Poster.

Dutot, P.-F., Eyraud, L., Mounie, G., & Trystram, D. (2004). Bicriteria algorithm for scheduling jobs on cluster platforms. In *Symposium on parallel algorithm and architectures* (pp. 125–132), Barcelona.

Dutot, P.-F., Eyraud, L., Mounié, G., & Trystram, D. (2005). Scheduling on large scale distributed platforms: from models to implementations. *International Journal of Foundations of Computer Science*, *16*(2), 217–237.

Elmroth, E., & Tordsson, J. (2005). An interoperable standards-based grid resource broker and job submission service, e-Science 2005. In *First IEEE conference on e-science and grid computing* (pp. 212–220). Los Alamitos: IEEE Computer Society Press.

Huedo, E., Montero, R. S., & Llorente, I. M. (2005). Coordinated use of Globus pre-WS and WS resource management services with GridWay. In *LNCS: Vol. 3762. 2nd workshop on grid computing and its application to data analysis on the move federated conferences* (pp. 234–243). Berlin: Springer.

Kurowski, K., & Nabrzyski, J. (2000). Predicting job execution times in the grid. In *Proceedings of the 1st SGI 2000 international user conference*, Krakow, Poland.

Kurowski, K., Nabrzyski, J., & Pukacki, J. (2001). User preference driven multi-objective resource management in grid environments. In: *Proceedings of the first IEEE/ACM international symposium on cluster computing and grid, CCGRID 2001*, Brisbane, Australia.

Kurowski, K., Nabrzyski, J., Oleksiak, A., & Węglarz, J. (2003). Multicriteria aspects of grid resource management. In J. Nabrzyski, J. Schopf, & J. Węglarz (Eds.), *Grid resource management*. Boston: Kluwer Academic.

Kurowski, K., Oleksiak, A., Nabrzyski, J., Kwiecień, A., Wojtkiewicz, M., Dyczkowski, M., Guim, F., Corbalan, J., & Labarta, J. (2005). *Multi-criteria grid resource management using performance prediction techniques. Integrated research in grid computing*. Berlin: Springer.

Kurowski, K., Nabrzyski, J., Oleksiak, A., & Węglarz, J. (2006a). Grid multicriteria job scheduling with resource reservation and prediction mechanisms. In J. Józefowska & J. Węglarz (Eds.), *Perspectives in modern project scheduling* (pp. 345–373). New York: Springer.

Kurowski, K., Nabrzyski, J., Oleksiak, A., & Węglarz, J. (2006b). Scheduling jobs on the grid—multicriteria approach. In *Computational methods in science and technology*. Poznan: OWN.

Lewandowski, A., & Wierzbicki, A. P. (Eds.). (1989). *Aspiration based decision support systems—theory, software and applications*. Berlin: Springer.

Li, H., Muskulus, M., & Wolters, L. (2007). Modeling correlated workloads by combining model based clustering and a localized sampling algorithm. In *Proceedings of 21st ACM International Conference on Supercomputing (ICS07)*, Seattle, USA, June 16–20, 2007. New York: ACM Press.

Lo, M., Mache, J., & Windisch, K. J. (1998). A comparative study of real workload traces and synthetic workload models for parallel job scheduling. In *Lecture notes in computer science: Vol. 1459. Proceedings of the workshop on job scheduling strategies for parallel processing* (pp. 25–46). Berlin: Springer.

Lublin, U., & Feitelson, D. G. (2003). The workload on parallel supercomputers: modeling the characteristics of rigid jobs. *Journal of Parallel and Distributed Computing*, *63*(11), 1105–1122.

Milkiewicz, J. (2005). *Development of grid simulator framework*. Master thesis, Poznan University of Technology, Faculty of Computing Science and Management, Poznan.

Siddiqui, M., Villazon, A., & Fahringer, T. (2006). Grid capacity planning with negotiation-based advance reservation for optimized QoS. In *ACM/IEEE super computing (SC|06)*, Tampa, FL, November 11–17, 2006.

Rodero, I., Corbalt'an, J., Badia, R. M., & Labarta, J. (2005). eNANOS grid resource broker. In P. M. A. Sloot, et al. (Eds.), *Advances in grid computing, EGC 2005*.

Tchernykh, A., Ramírez, J., Avetisyan, A., Kuzjurin, N., Grushin, D., & Zhuk, S. (2006). Two-level job-scheduling strategies for a computational grid. In Wyrzykowski, et al. (Eds.), *Lecture notes in computer science: Vol. 3911. Parallel processing and applied mathematics*. The second grid resource management workshop (GRMW'2005) in conjunction with the sixth international conference on parallel processing and applied mathematics—PPAM 2005, Poznan, Poland, 11–14 September 2005 (pp. 774–781). Berlin: Springer.

Yager, R. R. (1988). On ordered weighted averaging aggregation operators in multicriteria decision making. *IEEE Transactions on Systems, Man and Cybernetics*, *18*, 183–90.

Open Grid Forum. (2008). Grid scheduling architecture research group. https://forge.gridforum.org/projects/gsa-rg/.

Grid Workloads Archive. (2008). http://gwa.ewi.tudelft.nl/.

GridSim project home page. (2008). http://www.gridbus.org/gridsim/.

GRMS home page. (2008). http://www.gridge.org/grms.

GSSIM project home page. (2008). http://www.gssim.org.