Genetic Algorithm Calibration for Two Objective Scheduling Parallel Jobs on Hierarchical Grids

Victor Hugo Yaurima-Basaldua[1], Andrei Tchernykh[2], Yair Castro-Garcia[3], Victor Manuel Villagomez-Ramos[1], and Larisa Burtseva[3]

[1] CESUES Superior Studies Center, San Luis R.C., Mexico
vyaurima@yahoo.com, vicmvr@msn.com
[2] CICESE Research Center, Ensenada, Mexico
chernykh@cicese.mx
[3] Autonomous University of Baja California, Mexicali, Mexico
yair.castro.garcia@gmail.com, lpb@iing.mxl.uabc.mx

Abstract. This paper addresses non-preemptive offline scheduling parallel jobs on a Grid. We consider a Grid scheduling model with two stages. At the first stage, jobs are allocated to a suitable Grid site, while at the second stage, local scheduling is independently applied to each site. In this environment, one of the big challenges is to provide a job allocation that allows more efficient use of resources and user satisfaction. In general, the criteria that help achieve these goals are often in conflict. To solve this problem, two-objective genetic algorithm is proposed. We conduct comparative analysis of five crossover and three mutation operators, and determine most influential parameters and operators. To this end multi factorial analysis of variance is applied.

Keywords: Offline scheduling, Grid, Genetic Algorithm, Crossover Operator, Mutation Operator.

1 Introduction

In this paper, we present experimental work in the field of multi-objective scheduling in a two layer Grid computing. At the first layer, we select the best suitable machine for each job using a given criterion. At the second layer, local scheduling algorithm is applied to each machine independently. In such an environment, one of the big challenges is to provide scheduling that allows more efficient use of resources, and satisfies other demands. The optimization criteria are often in conflict. For instance, resource providers and users have different objectives: providers strive for high utilization of resources, while users are interested in a fast response. We provide solutions that consider both goals. The aggregation method of criteria and a scalar function to normalize them are used. We examine the overall Grid performance based on real data and present a comprehensive comparative analysis of five crossover operators, three operators of mutation, five values of crossover probability, and five values of mutation probability. To genetic algorithm tune up the multifactorial analysis of variance is applied.

After formally presenting our Grid scheduling model in Section 2, we discuss related work in Section 3. We introduce genetic scheduling algorithms and discuss their application for Grid scheduling in Section 4. Genetic algorithm calibration is presented in section 5. Finally, we conclude with the summary in Section 6.

2 Model

We address an offline scheduling problem: n parallel jobs $J_1, J_2, ..., J_n$ must be scheduled on m parallel machines (sites) $N_1, N_2, ..., N_m$. Let m_i be the number of identical processors or cores of machine N_i. Assume without loss of generality that machines are arranged in non-descending order of their numbers of processors, that is $m_1 \leq m_2 \leq ... \leq m_m$ holds.

Each job J_i is described by a tuple $(size_j, p_j, p_j^{'})$: its size $1 \leq size_j \leq m_m$ also called processor requirement or degree of parallelism, execution time p_j and user runtime estimate $p_j^{'}$. The release date of a job is zero; all the jobs are available before scheduling process start. Job processing time is unknown until the job has completed its execution (non-clairvoyant case). User runtime estimate $p_j^{'}$ is provided by a user. A machine must execute a job by exclusively allocating exactly $size_j$ processors for an uninterrupted period of time p_j to it. As we do not allow multisite execution and co-allocation of processors from different machines, a job J_j can only run on machine N_i if $size \leq m_j$ holds.

Two criteria are considered: *makespan*: C_{max}, $C_{max} = max(C_i)$, where C_i is the maximum completion time on N_i machine $(i = 1, 2, 3, ..., N_m)$ and *mean turnaround time*: $TA = \frac{1}{n} \sum_{j=1}^{n} c_j$, where c_j is the completion time of job J_j.

We denote our Grid model by GP_m. In the three field notation $(\alpha \left|\beta\right| \gamma)$ introduced in [6], our scheduling problem is characterized as $GP_m \left| size_j, p_j, p_j^{'} \right| OWA$, where OWA is the value of the multi-criteria aggregation operator ($OWA = w_1 C_{max} + w_2 TA$), and w_i is the linear combination weights. The problem of scheduling on the second stage is denoted as $P_m \left| size_j, p_j, p_j^{'} \right| C_{max}$.

3 Related Work

3.1 Hierarchical Scheduling

Scheduling algorithms for two layer Grid models can be split into a global allocation part and a local scheduling part. Hence, we regard MPS (Multiple machine Parallel Scheduling) as a two stage scheduling strategy: $MPS = MPS\_Alloc + PS$ [20]. At the first stage, we allocate a suitable machine for each job using a genetic algorithm. At the second stage, PS (single machine Parallel Scheduling) algorithm is applied to each machine independently for jobs allocated during the previous stage.

3.2 Multi-criteria Scheduling

Several studies deal with scheduling in Grids considering only one criterion (e.g. EGEE Workload Management System [1], NorduGrid broker [4], eNANOS [16], Gridway [10]. Various Grid resource managements involve multiple objectives and may use multicriteria decision support. Dutot et al. [3] considered task scheduling on the resources taking into account maximum task completion time and the average completion time. Siddiqui, et al. [18] presented task scheduling based on the resource negotiation, advance reservation, and user preferences. The user preferences are modeled by utility functions, in which users must enter the values and levels of negotiation.

General multi-criteria decision methodology based on the Pareto optimality can be applied. However, it is very difficult to achieve fast solutions needed for Grid resource management by using the Pareto dominance. The problem is very often simplified to a single objective problem or objectives' combining. There are various ways to model preferences, for instance, they can be given explicitly by stakeholders to specify an importance of every criterion or a relative importance between criteria. This can be done by a definition of criteria weights or criteria ranking by their importance.

In order to provide effective guidance in choosing the best strategy, Ramirez et al. [15] performed a joint analysis of several metrics according to methodology proposed in [19]. They introduce an approach to multi-criteria analysis assuming equal importance of each metric. The goal is to find a robust and well performing strategy under all test cases, with the expectation that it will also perform well under other conditions, e.g., with different Grid configurations and workloads. Kurowski et al. [11] used aggregation criteria method for modeling the preferences of participants (owners, system managers and users). Authors considered two stage hierarchical grids, taking into account the stakeholders' preferences, assuming unknown processing times of the tasks, and studied the impact of the size of the batch of tasks on the efficiency of schedules. Lorpunmanee et al. [12] presented task allocation strategies to the different sites of a Grid and propose a model for task scheduling considering multiple criteria. They concluded that such scheduling can be performed efficiently using *GAs*.

In this paper, we consider two-criteria scheduling problem. We propose a genetic algorithm as a strategy for allocating jobs to resources. It uses an aggregation criteria method and the weight generating function representing the relative importance of each criterion. We present an experimental analysis of such a problem and compare obtained results with strategies aimed at optimizing a single criterion. In this paper, the Ordered Weighted Averaging (OWA) [11] is applied: $OWA(x_1, x_2, ..., x_n) = \sum_{c=1}^{k} w_c s(x)_{\sigma(c)}$, where w_c is the weight, $c = 1, ..., k$, x_c is a value associated with the satisfaction of the c criterion. Permutation ordering values: $s(x)_{\sigma(1)} \leq s(x)_{\sigma(2)} \leq ...s(x)_{\sigma(k)}$ is performed. Weights (w_c) are nonnegative and $\sum_{c=1}^{k} w_c = 1$. If all the weights are set to the same value, OWA behaves as the arithmetic mean. In this case, high values of some criterion compensate low values of the other ones. In OWA approach, a compromise solution is provided. The highest weight is w_1 and the subsequent ones are decreasing, but never reaching 0. That 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. The goal is to find a weighting scheme that provides the best possible mean value for all stakeholders' evaluations and the highest possible value for the worst case evaluation. To achieve this, weight w_1 must be relatively large, while weight w_k should be small, where k denotes the number of criteria. The remaining weights decrease in value from w_1 to w_k according to:

$$w_c = \begin{cases} 3/2k, & c = 1 \\ (3k - 2c - 1)/\left[2n(k-1)\right], & 1 < c \le k \end{cases} \tag{1}$$

4 Genetic Algorithm

Scheduling algorithms for two layer Grid models can be split into a global allocation part and a local scheduling part. Hence, we regard *MPS* as a two stage scheduling strategy: *MPS*=*MPS_Alloc*+*PS*. At the first stage, we use a genetic algorithm (*GA*) to allocate a suitable machine for each job (*MPS_Alloc*=*GA*). At the second stage, the parallel job scheduling algorithm *PS* is applied to each machine independently for jobs allocated during the previous stage. As *PS* we use the well-known strategy Backfilling-EASY [21,22].

GA is a well-known search technique used to find solutions to optimization problems [9]. Candidate solutions are encoded by chromosomes (also called genomes or individuals). The set of initial individuals forms the population. Fitness values are defined over the individuals and measures the quality of the represented solution. The genomes are evolved through the genetic operators generation by generation to find optimal or near-optimal solutions. Three genetic operators are repeatedly applied: selection, crossover, and mutation. The selection picks chromosomes to mate and produce offspring. The crossover combines two selected chromosomes to generate next generation individuals. The mutation reorganizes the structure of genes in a chromosome randomly so that a new combination of genes may appear in the next generation. The individuals are evolved until some stopping criterion is met. OWA operator as a fitness function is applied (Section 2).

Each solution is encoded by $n \cdot m$ matrix. Where m is a number of machines in a Grid, and n is a number of jobs. The $i = 0, ..., m-1$ row represents local queue of machine N_i. Note, that machines are arranged in non-descending order of their number of processors $m_0 \le m_2 \le m_{m-1}$. A job J_j can only run on machine N_i if $size_j \le m_i$ holds. The available set of machines for a job J_j is defined to be the machines with indexes $f_j, ..., m$, where f_j is the smallest index i such that $m_i \ge size_j$.

The selection picks chromosomes to produce offspring. The binary tournament selection known as an effective variant of the parents' selection is considered. Two individuals are drawn randomly from the population, and one with highest fitness value wins the tournament. This process is repeated twice in order to select two parents.

4.1 Crossover Operators

Crossover operator produces new solutions by combining existing ones. It takes parts of solution encodings from two existing solutions (parents) and combines them into single solution (child). The crossover operator is applied under a certain probability (P_c). In this paper, five operators are considered.

One Segment Crossover for Matrix (*OSXM*). It is based on the crossover operator *OSX - One Segment Crossover* [7]. In this crossover, two random points $S1$ and $S2$ from 0 to *vmax* (maximum index used) are selected. The child inherits columns from the 0 to $S1$ position from parent 1. It also inherits columns from $S1$ to $S2$ from parent 2, but only elements that have not been copied from parent 1. Finally, child inherits elements from parent 1 that have not yet been copied.

Two Point Crossover for Matrix (*TPM*). It is based on the crossover operator *Two Point Crossover* [13]. In this crossover, two random points $S1$ and $S2$ from 0 to *vmax* are selected. Columns from position 0 to $S1$ and from $S2$ to *vmax* are copied from parent 1. The remaining elements are copied from the parent 2 only if they have not been copied.

Order Based Crossover for Matrix (*OBXM*). It is based on the crossover operator *OBX - Order Based Crossover* [5]. A binary mask is used. The mask values equal to one indicate that the corresponding columns are copied from parent 1 to the child. The rest of elements are copied from parent 2, only if they have not been copied. The mask values are generated randomly and uniformly.

Precedence Preservative Crossover for Matrix (*PPXM*). It is based on the crossover operator *PPX - Precedence Preservative Crossover* [2]. Random binary mask values equal to one indicates that corresponding columns are copied from parent 1 to the child, and the values equal to zero indicate that columns are copied from parent 2, this is done by iterating the columns of parents who have not been copied in order from left to right.

Order Segment Crossover for Matrix with Setup (*OSXMS*). It is based on the crossover operator *OSX - Order Segment Crossover* [7]. It chooses two points randomly. Columns from position 1 to the first point are copied from parent 1. The elements are ordered by the number of processors required for subsequent insertion into the child in the position according to the number of required processors. Columns from first point to second point are copied from parent 2, only if the elements have not been copied. Finally, the remaining elements are copied from the parent 1, considering not copied elements.

4.2 Mutation

The mutation operator produces small changes in an offspring with probability P_m. It prevents falling of all solutions into local optimum and extends search space of the algorithm. Three operators *Insert*, *Swap* and *Switch* adapted for two-dimensional encoding are considered. 1)Queue_Insert. Two points are randomly selected. The element of the second point is inserted to the first point, shifting the rest. Note that this mutation preserves most of the order and the adjacency information. 2)Queue_Swap. This mutation randomly selects two points and swaps their elements. 3)Queue_Switch. This mutation selects a random column and swaps their elements with the next column.

5 GA Calibration

5.1 Workload

The accuracy of the evaluation highly relies upon workloads applied. For testing the job execution performance under a dedicated Grid environment, we use Grid workload based on real production traces. Carefully reconstructed traces from real supercomputers provide a very realistic job stream for simulation-based performance evaluation of Grid job scheduling algorithms. Background workload (locally generated jobs) that is an important issue in non-dedicated Grid environment is not addressed.

Four logs from PWA (Parallel Workloads Archive) [14] (Cornell Theory Center, High Performance Computing Center North, Swedish Royal Institute of Technology and Los Alamos National Lab) and one from GWA (Grid Workloads Archive) [8] (Advanced School for Computing and Imaging) have been used: The archives contain real workload traces from different machine installations. They provide information of individual jobs including submission time, and resource requirements.

For creating suitable grid scenarios, we integrate several logs by merging users and their jobs. The premise for the integration of several logs of machines in production use into a Grid log is based on the following. Grid logs contain jobs submitted by users of different sites; a Grid execution context could be composed by these sites. Unification of these sites into a Grid will trigger to merger users and their jobs. It should be mentioned that merging several independent logs to simulate a computational Grid workload does not guarantee representation of the real Grid with the same machines and users. Nevertheless, it is a good starting point to evaluate Grid scheduling strategies based on real logs in the case of the lack of publicly available Grid workloads. Time zone normalization, profiled time intervals normalization, and invalid jobs filtering are considered.

5.2 Calibration Parameters

A method of experimental design is adapted from Ruiz and Maroto [17], where the following steps are defined: (a) test all instances produced with possible combinations of parameters; (b) obtain the best solution for each instance; (c) apply the Multifactor Variance Analysis (ANOVA) with 95% confidence level to find the most influential parameters; (d) set algorithm parameters based on selected parameters values; (e) calculate relative difference of the calibrated algorithm and other adapted algorithms over the best solutions. Table 1 shows parameters that were set for the calibration. Hence, 5 x 3 x 5 x 5 = 375 different algorithms alternatives were considered. 30 executions of the workload were realized, in total 375 x 30 = 11,250 experiments.

Table 1. Calibration parameters

Instance	Levels
Crossover operators	OSXM, TPM, OBXM, PPXM, OSXMS
Mutation operators	Queue_Insert, Queue_Swap, Queue_Switch
Crossover probability	0.001, 0.01, 0.05, 0.1, 0.2
Population	50 individuals
Number of jobs in individual	5275
Selection	Binary tournament
Stop criterion	If the fitness value of the best chromosome is not improved 4 times, the algorithm is stopped.

The performance of the proposed algorithm is calculated as the percentage of the relative distance of the obtained solution from the best one (Relative Increment over the Best Solution - *RIBS*). The *RIBS* is calculated using the following formula: $RIBS = (Heu_{sol} - Best_{sol})/Best_{sol} \cdot 100$, where Heu_{sol} is the value of the objective function obtained by considered algorithm, and $Best_{sol}$ is the best value obtained during the testing all possible parameter combinations.

5.3 Analysis of Variance

To assess the statistical difference among the experimental results, and observe effect of different parameters on the result quality, the ANOVA is applied. The analysis of variance is used to determine factors that have a significant effect, and which are the most important factors. Parameters of the Grid scheduling problem are considered as factors, and their values as levels. We assume that there is no interaction between the factors.

Table 2. Analysis of variance for *RIBS* (Type III Sums of Squares)

Source	*Sum of Squares*	*Df*	*Mean Square*	*F-Ratio*	*P-Value*
MAIN EFFECTS					
A:cross	2111.03	4	527.758	152.00	0.0000
B:mut	47.4724	2	23.7362	6.84	0.0012
C:pc	20.9586	4	5.23964	1.51	0.1990
D:pm	27.5114	4	6.87785	1.98	0.0969
RESIDUAL	1249.99	360	3.4722		
TOTAL (CORRECTED)	3456.97	374			

The *F-Ratio* is the ratio between mean square of the factor and the mean square of residues. A high *F-Ratio* means that this factor affects the response variable (see Table 2). The value of *P-Value* shows the statistical significance of the factors. The factors, whose *P-Value* is less than 0.05, have statistically significant effect on the response variable (*RIBS*) with 95% level of confidence.

Here we see that the factors that significantly affect the response variable are the operators of crossover and mutation. According to the *F-Ratio*, the most important factor is the crossover operator.

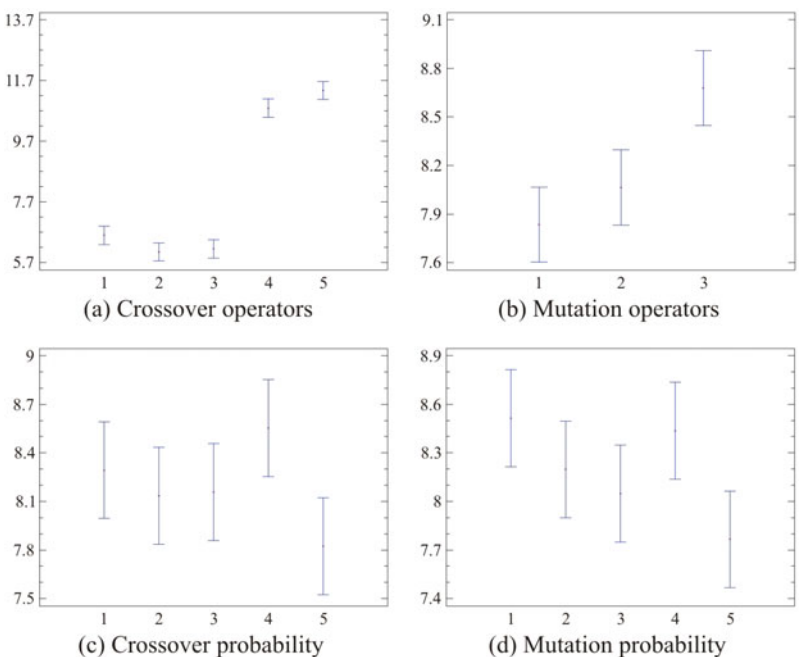

Fig. 1. Means and 95% LSD intervals

Figs. 1(a)-1(d) show means and 95% LSD intervals of the most influential factors. Fig. 1(a) shows the results obtained for crossover operators. Five operators are presented in the following order: 1-*OSXM*, 2-*TPM*, 3-*OBXM*, 4-*PPXM* and 5-*OSXMS*. The vertical axis is the values of *RIBS*. We can see that crossover *TPM* is the best crossover among the five ones tested, followed by *OBXM*. Figure 1(b) shows the results obtained for mutation operators. Three mutation operators are presented in the following order: 1- *Queue_Insert*, 2- *Queue_Swap* y 3- *Queue_Switch*. *Queue_Insert* is shown to be the best, followed by *Queue_Swap*. Figures 1(c) and 1(d) show the results for the crossover and mutation probability. The best crossover probability occurring is 0.9. The best mutation probability occurring is 0.2.

6 Conclusions and Future Work

We addressed a two-objective genetic algorithm calibration for scheduling jobs in a two stage computational Grid. We conduct comprehensive comparison of five known crossover operators, three mutation operators adapted for two dimension encoding, five values for the crossover probability, and five values for the mutation probability. 375 different genetic algorithms are analysed with 30 instances for each one. 11,250 experiments were evaluated. We use aggregation criteria method for modeling the preferences of two objectives: C_{max} and TA. To assess the statistical difference among the experimental results, and observe impact of different parameters on the result quality, the ANOVA technique was applied. The crossover operator is shown to have a statistically significant effect. It plays the most important role in genetic algorithms applied to a scheduling in computational Grid. Of five compared operators, the *TPM* obtained the best result, followed by the *OBXM*.

Obtained results may serve as a starting point for future heuristic Grid scheduling algorithms that can be implemented in real computational Grids. While the scope of this work is to determine most influential *GA* parameters and operators, in future work, we also intend to evaluate the practical performance of the proposed strategies, and the assessment of their cost. To this end, we plan simulations using real workload traces and corresponding Grid configurations. We will compare our *GA* with other existing Grid scheduling strategies which are typically based on heuristics. Future work needs for a better understanding of the scheduling with dynamic Grid configuration, resource failures and other real characteristics of the Grid.

References

1. Avellino, G., Beco, S., Cantalupo, B., Maraschini, A., Pacini, F., Terracina, A., Barale, S., Guarise, A., Werbrouck, A., Di Torino, S., Colling, D., Giacomini, F., Ronchieri, E., Gianelle, A., Peluso, R., Sgaravatto, M., Mezzadri, M., Prelz, F., Salconi, L.: The EU datagrid workload management system: towards the second major release. In: 2003 Conference for Computing in High Energy and Nuclear Physics. University of California, La Jolla (2003)
2. Bierwirth, C., Mattfeld, D., Kopfer, H.: On Permutation Representations for Scheduling Problems. In: Ebeling, W., Rechenberg, I., Voigt, H.-M., Schwefel, H.-P. (eds.) PPSN 1996, Part II. LNCS, vol. 1141, pp. 310–318. Springer, Heidelberg (1996)
3. Dutot, P., Eyraud, L., Mounie, G., Trystram, D.: Models for scheduling on large scale platforms: which policy for which application? In: Proceedings of 18th International Symposium on Parallel and Distributed Processing, p. 172 (2004)
4. Elmroth, E., Tordsson, J.: An interoperable, standards based Grid resource broker and job submission service. In: First International Conference on e-Science and Grid Computing, 2005, pp. 212–220. IEEE Computer Society, Melbourne (2005)
5. Gen, M., Cheng, R.: Genetic algorithms and engineering optimization, p. 512. John Wiley and Sons, New York (1997)
6. Graham, R.L., Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G.: Optimization and approximation in deterministic sequencing and scheduling: a survey. In: Hammer, P.L., Johnson, E.L., Korte, B.H. (eds.) Discrete Optimization II. Annals of Discrete Mathematics, vol. 5, pp. 287–326. North-Holland, Amsterdam (1979)

7. Guinet, A., Solomon, M.: Scheduling Hybrid Flowshops to Minimize Maximum Tardiness or Maximum Completion Time. Int. J. Production Research 34(6), 1643–1654 (1996)
8. GWA Grid Workloads Archive, `http://gwa.ewi.tudelft.nl`
9. Holland, J.: Adaptation in natural and artificial systems. University of Michigan Press (1975)
10. Huedo, E., Montero, R.S., Llorente, I.M.: Evaluating the reliability of computational grids from the end user's point of view. Journal of Systems Architecture 52(12), 727–736 (2006)
11. Kurowski, K., Nabrzyski, J., Oleksiak, A., Wglarz, J.: A multicriteria approach to two-level hierarchy scheduling in Grids. Journal of Scheduling 11(5), 371–379 (2008)
12. Lorpunmanee, S., Noor, M., Hanan, A., Srinoy, S.: Genetic algorithm in Grid scheduling with multiple objectives. In: Proceedings of the 5th WSEAS Int. Conf. on Artificial Intelligence, Knowledge Engineering and Data Bases, Madrid, Spain, pp. 429–435 (2006)
13. Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolutions Programs, 3rd edn. Springer, Heidelberg (1996)
14. PWA Parallel Workloads Archive, `http://www.cs.huji.ac.il/labs/parallel/workload/`
15. Ramirez-Alcaraz, J.M., Tchernykh, A., Yahyapour, R., Schwiegelshohn, U., Quezada-Pina, A., Gonzalez-Garcia, J.-L., Hirales-Carbajal, A.: Job Allocation Strategies with User Run Time Estimates for Online Scheduling in Hierarchical Grids. J. Grid Computing 9, 95–116 (2011)
16. Rodero, I., Corbalán, J., Badía, R.M., Labarta, J.: eNANOS Grid Resource Broker. In: Sloot, P.M.A., Hoekstra, A.G., Priol, T., Reinefeld, A., Bubak, M. (eds.) EGC 2005. LNCS, vol. 3470, pp. 111–121. Springer, Heidelberg (2005)
17. Ruiz, R., Maroto, C.: A genetic algorithm for hybrid flowshops with sequence dependent setup times and machine eligibility. European Journal of Operational Research 169, 781–800 (2006)
18. Siddiqui, M., Villazon, A., Fahringer, T.: Grid Capacity Planning with Negotiation-based Advance Reservation for Optimized QoS. In: Proceedings of the 2006 ACM/IEEE Conference on Supercomputing (SC 2006). ACM, New York, article 103 (2006), `http://doi.acm.org/10.1145/1188455.1188563`
19. Tsafrir, D., Etsion, Y., Feitelson, D.G.: Backfilling using system-generated predictions rather than user runtime estimates. IEEE Trans. Parallel Distrib. Syst. 18, 789–803 (2007)
20. Tchernykh, A., Schwiegelsohn, U., Yahyapour, R., Kuzjurin, N.: Online Hierarchical Job Scheduling on Grids with Admissible Allocation. Journal of Scheduling 13(5), 545–552 (2010)
21. Lifka, D.A.: The ANL/IBM SP Scheduling System. In: Feitelson, D.G., Rudolph, L. (eds.) IPPS-WS 1995 and JSSPP 1995. LNCS, vol. 949, pp. 295–303. Springer, Heidelberg (1995)
22. Skovira, J., Chan, W., Zhou, H., Lifka, D.: The EASY - LoadLeveler API Project. In: Feitelson, D.G., Rudolph, L. (eds.) IPPS-WS 1996 and JSSPP 1996. LNCS, vol. 1162, pp. 41–47. Springer, Heidelberg (1996)