Scheduling Grid Tasks under Uncertain Demands

Daniel M. Batista batista@ic.unicamp.br
André C. Drummond andred@ic.unicamp.br
Nelson L. S. da Fonseca nfonseca@ic.unicamp.br

Institute of Computing, State University of Campinas
Avenida Albert Einstein, 1251 – 13084-971
Campinas – SP, Brazil

ABSTRACT

The uncertainty of the demands of grid applications can cause unpredicted performance and, consequently, can make ineffective schedules derived for target demand values. To produce effective results, schedulers need to take into account the difficulty in estimating the demands of applications. In this paper, a scheduler based on fuzzy optimization is proposed to deal with such uncertainties. It is shown, via numerical results, that the proposed scheduler presents advantages when compared to classical schedulers.

Categories and Subject Descriptors

I.2.3 [**Artificial Intelligence**]: Deduction and Theorem Proving—*Uncertainty, "fuzzy," and probabilistic reasoning*; I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*Scheduling*; C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*Distributed Applications*

Keywords

fuzzy optimization, uncertainty, grid networks, task scheduling, linear programming

1. INTRODUCTION

Grid Networks (Grids) have been designed to provide a distributed computational infrastructure for advanced science and engineering [1]. They involve coordinated resource sharing and problem solving in heterogeneous dynamic environments to meet the needs of a generation of researchers requiring large amounts of bandwidth and more powerful computational resources. Although in its infancy, cooperative problem solving via grids has become a reality, and various areas from aircraft engineering to bioinformatics have benefited from this novel technology. Grids are expected to evolve from pure research information processing to e-commerce, as has happened with the World Wide Web.

SAC'08 March 16-20, Fortaleza, Ceará, Brazil

Central to grid processing is the scheduling of application tasks to resources. Essentially, scheduling is the decision making process of matching applications demands to grid resources and the specification of the time at which resources should be used to satisfy these demands. Grid resources comprise the hosts computational and storage capacity as well as network bandwidth.

The lack of resource ownership by grid schedulers and fluctuations in resource availability requires mechanisms which will enable grids to adjust themselves to cope with fluctuations. A sudden increase in link load can, for example, increase the time for the transfer of data between the computers where two tasks reside, thus leading to the necessity of relocating the tasks to a third computer. Furthermore, the lack of a central controller implies a need for self-adaptation. Such autonomous operation of grids has been proposed and implemented in different systems [2].

Although uncertainty of the demands of applications have been addressed in parallel systems [3] [4] [5], the few existing techniques can only be partially utilized since these systems are usually tightly coupled and the communication demands have little impact on the performance of the applications. Conversely, the tasks of grid applications are connected by shared communication links.

Computational demands can be inferred by executing the application with a controlled number of parameters [6]. However, these techniques work only for applications which computational demands increase linearly with the number of parameters used.

Furthermore, the scheduling problem is an NP-hard problem [7] and feasible solutions in real time require either heuristics or approximations [8]. In addition, the computational complexity is increased by the need to account for heterogeneous resources and irregular topologies, which contrasts to what happens in multiprocessor systems.

This paper introduces a scheduler based on fuzzy optimization for dealing with uncertainties of the demands of applications. The scheduler accepts as input a set of dependent tasks described by Directed Acyclic Graphs (DAGs). Scheduling decisions account for unpredicted amounts of data to be exchanged among tasks.

The makespan given by the fuzzy solution is compared to those produced by schedulers based on classical optimization. Significant speedup values produced by schedulers designed to operate under high levels of uncertainty were observed in the experiments conducted. Moreover, the time taken by the fuzzy scheduler to produce feasible schedules is about the same the one taken by its corresponding classical

scheduler.

This paper is organized as follows. Section 2 introduces a novel scheduler based on fuzzy optimization theory and Section 3 evaluates the effectiveness of the proposed scheduler. Section 4 describes related works and Section 5 draws some conclusions.

2. A SCHEDULER FOR DEALING WITH UNCERTAINTY OF THE DEMANDS OF APPLICATIONS

In [8], it was introduced a set of eight schedulers offering solutions which differ in terms of their schedule length as well as computational complexity. These schedulers can be executed in parallel to obtain a schedule with the highest possible quality within a time period. One of these schedulers, called ILPDT, considers discrete intervals of time ($\in \mathbb{Z}_+$) and treats the scheduling problem as an integer linear programming problem. Although the discretization of time introduces approximation and a consequent loss of precision, under certain circumstances, this loss may not be significant, and the saving of time can be quite attractive when compared to a corresponding scheduler which assumes time as a continuous variable. A fuzzy version of this scheduler is presented in Subsection 2.2.

2.1 The ILPDT scheduler

The ILPDT scheduler accepts two graphs as input. The graph $G = (V_G, E_G)$ represents the grid topology while the DAG $D = (V_D, A_D)$ the dependencies among tasks. In G, V_G is the set of m ($m = |V_G|$) hosts connected by the set of links E_G. Hosts are labelled from 1 to m. In D, V_D is the set of ($n = |V_D|$) tasks with integer numbers as labels which allows a topological ordering of tasks and A_D is the set of dependencies.

The ILPDT scheduler considers that the input DAGs have a single input task and a single output task. DAGs failing to satisfy this condition because they have more than one input or output task can be easily modified by considering two null tasks with zero processing time and communication weights. Some characteristics of the DAGs are: I_i: processing demand of the i^{th} task, expressed as number of instructions to be processed by the i^{th} task ($I_i \in \mathbb{R}_+$); $B_{i,j}$: number of bytes transmitted between the i^{th} task and the j^{th} task ($B_{i,j} \in \mathbb{R}_+$); $\mathcal{D}$: set of arcs $\{ij : i < j$ and there exists an arc from vertex i to vertex j in the DAG$\}$. Moreover, grid resources composed of hosts and links have the following characteristics: TI_k: time the k^{th} host takes to execute 1 instruction ($TI_k \in \mathbb{R}_+$); $TB_{k,l}$: time for transmitting 1 bit on the link connecting the k^{th} host and the l^{th} host ($TB_{k,l} \in \mathbb{R}_+$); $\delta(k)$: set of hosts linked to the k^{th} host in the network, including the host k itself.

The weights of arcs (B) and nodes (I), representing respectively the amount of data to be transferred and the amount of processing, are furnished by the user. ILPDT outputs a Gantt diagram which provides information in which host each task should be executed, the starting time of task and the time which data transfer should happen.

The ILPDT scheduler solves a linear integer program which seeks the value of variables $x_{i,t,k}$ ($\in \{0,1\}$) and f_i ($\in \mathbb{N}^*$). $x_{i,t,k}$ is a binary variable that assumes a value of 1 if the i^{th} task finished at time t in the host k; otherwise this variable assumes a value of 0; f_i is a variable that stores the time at which the execution of the i^{th} task is finished ($f_i \in \mathbb{N}^*$).

The objective is to minimize the execution time of the application ($Minimize(f_n)$). The linear integer program is shown below. For convenience, the following notations is used: $\mathcal{T} = \{1, \ldots, T_{max}\}$ (T_{max} is the time that the application would take to execute serially all the tasks in the fastest host, i.e., $T_{max} = \min(TI) \sum_{i=1}^{n} I_i$), $\mathcal{J} = \{1, \ldots, n\}$ is the set of existing tasks of an application and $\mathcal{H} = \{1, \ldots, m\}$ is the set of hosts. The ILPDT formulation is given below:

Minimize f_n

such that

$$\sum_{t \in \mathcal{T}} \sum_{k \in \mathcal{H}} x_{j,t,k} = 1 \qquad \text{for } j \in \mathcal{J}; \qquad \text{(D1)}$$

$$f_j = \sum_{t \in \mathcal{T}} \sum_{k \in \mathcal{H}} t\, x_{j,t,k} \qquad \text{for } j \in \mathcal{J}; \qquad \text{(D2)}$$

$$x_{j,t,k} = 0 \qquad \text{for } j \in \mathcal{J}, \quad k \in \mathcal{H}, \quad t \in \{1, \ldots, \lceil I_j\, TI_k \rceil\}; \qquad \text{(D3)}$$

$$\sum_{k \in \delta(l)} \sum_{s=1}^{\lceil t - I_j\, TI_l - B_{i,j}\, TB_{k,l} \rceil} x_{i,s,k} \geq \sum_{s=1}^{t} x_{j,s,l} \qquad \text{for } j \in \mathcal{J}, \quad ij \in \mathcal{D}, \quad \text{for } l \in \mathcal{H}, \quad t \in \mathcal{T}; \qquad \text{(D4)}$$

$$\sum_{j \in \mathcal{J}} \sum_{s=t}^{\lceil t + I_j\, TI_k - 1 \rceil} x_{j,s,k} \leq 1 \qquad \text{for } k \in \mathcal{H}, \quad t \in \mathcal{T}, \quad t \leq \lceil T_{max} - I_j\, TI_k \rceil; \qquad \text{(D5)}$$

$$\sum_{l \in \mathcal{H}} VA_{j,l} = 1 \qquad \text{for } j \in \{2, ..., n\}; \qquad \text{(D6)}$$

$$(|\{i : ij \in \mathcal{D}\}| + 1)\, VA_{j,l} \leq \sum_{t \in \mathcal{T}} x_{j,t,l} + \sum_{i : ij \in \mathcal{D}} \sum_{k \in \delta(l)} \sum_{t \in \mathcal{T}} x_{i,t,k} \qquad \text{for } j \in \{2, ..., n\}, \quad l \in \mathcal{H}; \qquad \text{(D7)}$$

$$|\{i : ij \in \mathcal{D}\}| + VA_{j,l} \geq \sum_{t \in \mathcal{T}} x_{j,t,l} + \sum_{i : ij \in \mathcal{D}} \sum_{k \in \delta(l)} \sum_{t \in \mathcal{T}} x_{i,t,k} \qquad \text{for } j \in \{2, ..., n\}, \quad l \in \mathcal{H}; \qquad \text{(D8)}$$

$$VA_{j,l},\ x_{j,t,k} \in \{0,1\} \qquad \text{for } j \in \mathcal{J}, \quad l \in \mathcal{H}, \quad t \in \mathcal{T}. \qquad \text{(D9)}$$

Moreover, $T_{min} = \min TI \sum_{i \in SP} I_i$ is the time that the application would take to execute all the tasks of the shortest path in the fastest host (SP is the shortest path from task 1 to task n in terms of number of instructions). T_{max} and T_{min} are quite useful to reduce the number of variables needed in the fuzzy formulation.

2.2 The ILP-FUZZY scheduler

The uncertainties of the demands of the applications are represented by fuzzy numbers in the proposed formulation. The values of I and B are represented by triangular fuzzy numbers. In this way, the i^{th} task requires I_i instructions

with an uncertainty of $\sigma\%$ of this value; the amount of instructions is represented by $[\underline{I_i}, I_i, \overline{I_i}]$ where $\underline{I_i} = I_i(1 - \frac{\sigma}{100})$ and $\overline{I_i} = I_i(1 + \frac{\sigma}{100})$. In a similar way, the communications demands are given by $[\underline{B_{i,j}}, B_{i,j}, \overline{B_{i,j}}]$ with a $\rho\%$ level of uncertainty.

The objective function of ILPDT is modified so that the satisfaction degree λ ($\in [0,1]$) is maximized. The satisfaction degree is inversely proportional to the execution time given by a schedule. The values of T_{max} and T_{min} cannot be computed as in the ILPDT algorithm, given the uncertainty in the DAGs weights, therefore, $T'_{max} = T_{max}(1+\frac{\sigma}{100})$ and $T'_{min} = T_{min}(1 - \frac{\sigma}{100})$ are used instead. Figure 1 illustrates the relationship between λ and the application execution time (f_n). Moreover, the following constraint should be added to the formulation in order to include the range of values of f_n ($\in [T'_{min}, T'_{max}]$) and the relationship with the λ value:

$$1 - \frac{f_n - T'_{min}}{T'_{max} - T'_{min}} \geq \lambda \quad (1)$$

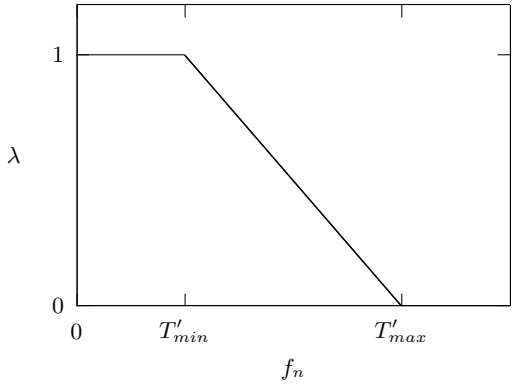

Figure 1: Satisfaction Degree.

Since I_i and B_i are now fuzzy numbers, the constraints of ILPDT need to be changed accordingly. The constraint D3 is:

$$x_{j,t,k} = 0 \quad \begin{array}{l}\text{for } j\in\mathcal{J}, \quad k\in\mathcal{H}, \\ t\in\{1,\ldots,\lceil \underline{I_j} TI_k \rceil\}\end{array} \quad (2)$$

These constraints determine that a task (j) cannot terminate until all its instructions have been completely executed. Since it is not possible to know the exact number of instructions, I_j is replaced by $\underline{I_j}$ given that the minimum number of instructions is the only value that is certainly known.

The constraints D4 are modified to the following formulation:

$$\sum_{k\in\delta(l)} \sum_{s=1}^{\lceil t - \overline{I_j} TI_l - \overline{B_{i,j}} TB_{k,l} \rceil} x_{i,s,k} \geq \sum_{s=1}^{t} x_{j,s,l} \quad \begin{array}{l}\text{for } j\in\mathcal{J}, \quad ij\in\mathcal{D}, \\ \text{for } l\in\mathcal{H}, \quad t\in\mathcal{T}\end{array} \quad (3)$$

The constraints in (3) establish that the j^{th} task cannot start execution before all its predecessors have finished their execution and transferred the required data by the j^{th} task. In this way, in order to prevent the potential execution of the j^{th} task previous to the execution of its predecessors due to the existing uncertainty on I and B values, I_j and B_{ij} are respectively replaced by their maximum values given by $\overline{I_j}$ and $\overline{B_{i,j}}$.

The last set of modified constraints is D5, given by:

$$\sum_{j\in\mathcal{J}} \sum_{s=t}^{\lceil t + \underline{I_j} TI_k - 1 \rceil} x_{j,s,k} \leq 1 \quad \begin{array}{l}\text{for } k\in\mathcal{H}, \quad t\in\mathcal{T}, \\ t\leq \lceil T'_{max} - \underline{I_j} TI_k \rceil\end{array} \quad (4)$$

The constraints in (4) establish that there is at most one task in execution at any one host at a specific time. Since the only know value of the number of instructions is the lowest value, the computational uncertainty yields to the replacement of I_j by $\underline{I_j}$.

These set of modifications leads to a new fuzzy scheduler called ILP-FUZZY.

3. EFFECTIVENESS OF THE ILP-FUZZY SCHEDULER

To assess the effectiveness of the fuzzy approach, the schedule produced by the ILP-FUZZY scheduler was compared to the one produced by the ILPDT scheduler. In the evaluation both schedulers received the same DAGs as input. The degree of uncertainty of the DAGs weights is provided as input to the ILP-FUZZY scheduler. The Montage astronomy [9] application was used to generate the input DAG shown in Figure 2. Weights were randomly chosen from a uniform distribution [9].

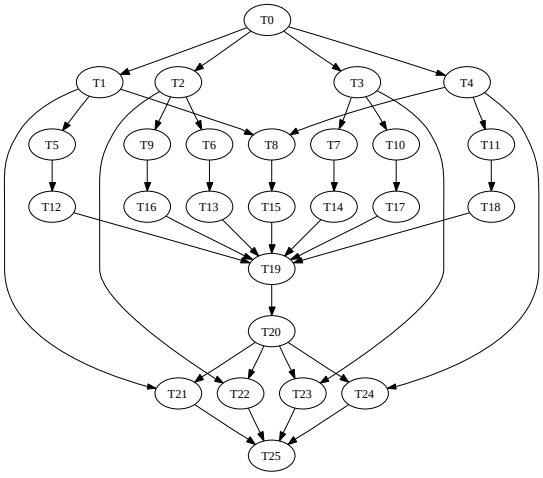

Figure 2: Tasks DAG used in the experiments.

Thirty grids were generated for the evaluation of the ILP-FUZZY scheduler by using the Doar-Leslie method which is largely used to generate Internet topologies. The number of hosts used is 50, the degree of node connectivity (β) is 0.98 and the ratio between the longest and the shortest links is 0.98.

Uncertainty values of 40%, 100%, 200% and 400% were considered. The maximum value of 400% was chosen following the recommendation in [9]. Although, uncertainty was place on both computational and communication demands, due to space limitation, results considering uncertainties only on B values ($\sigma = 0\%$ and $\rho \in \{40\%, 100\%, 200\%, 400\%\}$) are shown in this paper since communication demand is the most critical one given that data can be generated in real time.

For each scheduler designed to operate under a specific uncertainty level, DAGs with diverse uncertainty levels were used as input. Twenty DAGs with randomly generated weights were used for each level of uncertainty. In this way, it is possible to evaluate how well a scheduler designed to operate under a specific uncertainty level handles a whole range of uncertainty of the demands of the applications.

To compare the performance of different schedulers, the speedup was employed. The makespan produced by a scheduler and the T_{max} of the original DAG (with no uncertainty, $\sigma = 0$ and $\rho = 0$) were used in the computation of the speedup.

The schedulers were written in the C programming language and the optimization library `Xpress` version `2006A.1` was used. Programs were executed in a machine with Debian GNU/Linux version Lenny operating system.

Figure 3 displays the speedup of the ILP-FUZZY schedulers designed with different uncertainty levels as a function of the uncertainty of the input DAGs. The ILPDT produces higher speedup values than those produced by the ILP-FUZZY schedulers designed to operate under uncertainty levels equal or lower than 100%. ILPDT also performs better for low levels of uncertainty in the input DAG than the ILP-FUZZY scheduler designed for a 200% level of uncertainty. However, when the ILP-FUZZY scheduler is designed for a high degree of uncertainty (400%), it produces higher speedup values than those produced by the ILPDT scheduler. For instance, the speedup produced by ILP-FUZZY for $\rho = 400\%$ can be 26% higher than the ILPDT speedup. Moreover, the ILPDT speedup drops much faster than those given by the ILP-FUZZY scheduler as a function of the uncertainty of the DAGs weights.

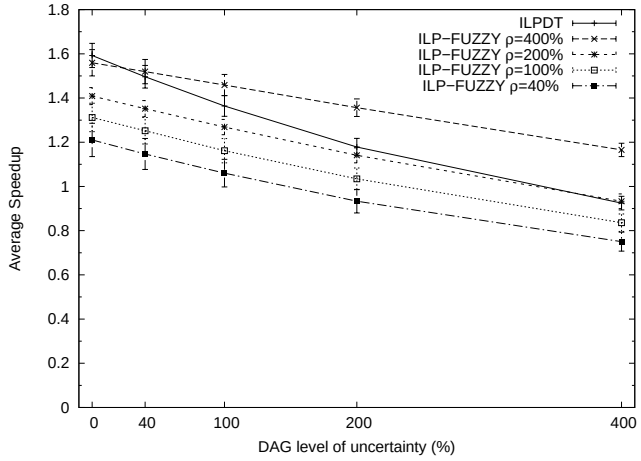

Figure 3: Speedup produced by schedulers for uncertain communication demands.

Results point out that the fuzzy approach is attractive if schedulers are designed to handle a high level of uncertainty. This can be understood by the lack of precision introduced by fuzzy solutions when their flexibility is quite limited. However, when the flexibility increases, the enhanced ability to handle uncertainty of demands overcompensates potential mistakes made when the range of variation of solutions is limited. Furthermore, the speedup produced by schedulers designed for a high level of uncertainty is quite robust to variations of uncertainties of demands.

Comparing the execution time of both schedulers, it follows that the 90th percentile of the execution time of the ILP-FUZZY scheduler is 32% lower than that of the ILPDT scheduler. However, the execution time of the ILP-FUZZY scheduler for two (out of 30 grids used in the evaluation) were ten times longer than that of the ILPDT scheduler. Although the use of the fuzzy scheduler is attractive not only for the speedup gain but also for time taken to derive a schedule, one needs to be aware that long execution times can occur.

4. RELATED WORK

Previous works investigated the issue of uncertainty of demands, specially in parallel systems. In [10], a proposal oriented to cluster and I/O bound applications was developed. The proposal gathers applications into classes according to their demands. Applications are monitored during their execution and change of class is pursued in case of changes of the demands of an application. Although this works considers communication costs, it is a reactive scheme while our proposal is a preventive one. Moreover, experiments were conducted in a homogeneous environment with a low number of hosts.

The research reported in [11] collects metrics of performance of parallel applications in supercomputers. Probability distributions for performance metrics such as execution time and host utilization are used to generate synthetic load to predict performance. This research differs from our work by the fact that the execution of applications are terminated in case they exceed a predicted value. Moreover, only independent tasks are considered.

In [12], a module that takes into account the available capacity of a grid is introduced. The module was based on the middleware "Support Infrastructure to Mobile Applications" (ISAM). Measurements are collected and used by bayesian models to predict the performance of real applications. Uncertainties are introduced in performance predictions. However, uncertainties of the demands of applications are not considered, specially those related to the transfer of data among tasks. The NWS (Network Weather Service) was employed in experiments with homogeneous hosts to predict the computational demands of the applications.

Two different approaches were compared in [9]. One that consider only computational demands and the other which also includes communication demands. These approaches schedule DAGs of dependent tasks without full knowledge of them. Uncertainty of DAGs weights is taken into account. It was shown that the makespan of applications increases with the degree of uncertainty. However, no scheduler that accounts the uncertainty of the demands of applications was proposed. Our work uses the same real scenario of the work in [9].

The research described in [13] compares the predicted execution time with the measured one. Experiments using a cluster of the "Enabling Grids for E-sciencE" (EGEE) grid were conducted. However, only CPU intensive applications in a specific cluster were considered and not much can be inferred about other scenarios.

Our work differs from others in the literature by considering both computation and communications uncertainties of grid applications with dependent tasks in heterogeneous grids.

5. CONCLUSION AND FUTURE WORK

The computation and communication demands of grid applications are usually informed by grid users who can only make a rough estimation of these demands. The uncertainty of these demands can cause unpredicted performance and can make ineffective schedules derived with different target demand values. Therefore, it is of paramount importance to take into account uncertainties when scheduling tasks. Schedules produced should be flexible enough to cope with uncertainties and yet produce suboptimal, if not optimal, solutions.

In this paper a scheduler based on fuzzy optimization was proposed to schedule grid tasks under uncertain knowledge about their demands. Results indicate that the effectiveness of this approach relies on the ability to cope with a high level of uncertainty.

As several grid applications, specially those of e-Science, generate huge amount of data during its execution, the approach proposed in this paper seems to be quite attractive for future implementation in grid systems. Currently we are investigating the trade-off between the solutions given by fuzzy schedulers and those given by self-adapting schemes.

6. REFERENCES

[1] I. Foster. What is the Grid? A Three Point Checklist. *GRIDToday*, 1(6), Jul 2002.

[2] Daniel M. Batista, Nelson L. S. da Fonseca, Fabrizio Granelli, and Dzmitry Kliazovich. Self-Adjusting Grid Networks. In *Proceedings of the IEEE International Conference on Communications – ICC 2007*, Jun 2007.

[3] Su-Hui Chiang, Rajesh K. Mansharamani, and Mary K. Vernon. Use of Application Characteristics and Limited Preemption for Run-to-Completion Parallel Processor Scheduling Policies. In *SIGMETRICS '94: Proceedings of the 1994 ACM SIGMETRICS Conference on measurement and Modeling of Computer Systems*, pages 33–44, New York, NY, USA, 1994. ACM Press.

[4] Su-Hui Chiang, Andrea Arpaci-Dusseau, and Mary K. Vernon. The Impact of More Accurate Requested Runtimes on Production Job Scheduling Performance. In *8th International Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP 2002)*, pages 103–127. Springer-Verlag Berlin Heidelberg, 2002.

[5] Cynthia Bailey Lee, Yael Schwartzman, Jennifer Hardy, and Allan Snavely. Are User Runtime Estimates Inherently Inaccurate? In *10th International Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP 2004)* , pages 253–263. Springer-Verlag Berlin Heidelberg, 2004.

[6] Radu Prodan and Thomas Fahringer. Dynamic Scheduling of Scientific Workflow Application on the Grid: a Case Study. In *SAC'05: Proceedings of the 2005 ACM Symposium on Applied computing*, pages 687–694, New York, NY, USA, 2005. ACM Press.

[7] Christos H. Papadimitriou and Kennet Steiglitz. *Combinatorial Optimization – Algorithms and Complexity*, pages 363–366. Dover Publications, 1998.

[8] Daniel M. Batista, Nelson L. S. da Fonseca, and Flavio K. Miyazawa. A Set of Schedulers for Grid Networks. In *SAC '07: Proceedings of the 2007 ACM Symposium on Applied Computing*, pages 209–213, New York, NY, USA, 2007. ACM Press.

[9] Jim Blythe, Sonal Jain, Ewa Deelman, Yolanda Gil, Karan Vahi, Anirban Mandal, and Ken Kennedy. Task Scheduling Strategies for Workflow-based Applications in Grids. In *IEEE International Symposium on Cluster Computing and Grids (CCGRID'05)*, volume 2, pages 759–767, May 2005.

[10] Fabricio Alves Barbosa da Silva and Isaac D. Scherson. Improving Parallel Job Scheduling Using Runtime Measurements. In *IPDPS '00/JSSPP '00: Proceedings of the Workshop on Job Scheduling Strategies for Parallel Processing*, pages 18–38, London, UK, 2000. Springer-Verlag.

[11] F. Cirne, W.; Berman. A Comprehensive Model of the Supercomputer Workload. In *IEEE International Workshop on Workload Characterization (WWC-4)*, pages 140–148, Dec 2001.

[12] R. Real, A. Yamin, L. da Silva, G. Frainer, I. Augustin, J. Barbosa, and C. Geyer. Resource Scheduling on Grid: Handling Uncertainty. In *Proceedings of the Fourth International Workshop on Grid Computing*, pages 205–207, Nov 2003.

[13] Michael Oikonomakos, Kostas Christodoulopoulos, and Emmanouel (Manos) Varvarigos. Profiling Computation Jobs in Grid Systems. In *Proceedings of the Seventh IEEE International Symposium on Cluster Computing and the Grid (CCGRID 2007)*, pages 197–204, May 2007.