Robust Scheduler for Grid Networks

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

Imprecise input data imposes additional challenges to grid scheduling. This paper introduces a novel scheduler based on fuzzy optimization called IP-FULL-FUZZY which considers uncertainties of both application demands and of resource availability. The effectiveness of the proposed scheduler is compared to those of a non-fuzzy scheduler as well as to those of a fuzzy scheduler which considers only uncertainties of application demands. Results evince the advantages of adopting the proposed scheduler.

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, bandwidth estimation

1. INTRODUCTION

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.

Furthermore, the scheduling problem is an NP-hard problem [12] and feasible solutions in real time require either heuristics or approximations [3]. Once tasks are allocated to hosts (grid nodes) according to a schedule, they are executed until all have been completed. However, due to the lack of ownership of resources, availability can change dynamically due to other loads on the grid. Thus, the original schedule may become sub-optimal.

Imprecise estimations of both applications demands and resource availability impose additional challenges to grid scheduling. Uncertainties of application demands arise from the lack of precision in estimating the amount of data transferred by applications. Uncertainty of available bandwidth is related to the nature of measurement and monitoring tools. Actually, estimations are quite often given in ranges rather than as deterministic values [10] [2]. Schedules produced by deterministic schedulers and based on imprecise input data can be quite different than an optimal one.

Adaptive scheduling, dynamic scheduling and self-adjusting scheduling have been proposed in the literature [1] [14] [13] [4]. All these schemes were designed to minimize the execution time of applications. Resource monitoring and task migration are used in these approaches to react to fluctuations of grid state. However, continuous monitoring can increase the degree of uncertainty due to intrusion effect while unnecessary task migration can increase overhead, enlonging the execution time.

Alternatively to these approaches, uncertainties of both application demands and resource availability can be accounted for in the input data to the scheduler. One of the advantages of this approach is to reduce the number of reconfigurations needed to reduce the execution time of the applications. Another advantage is to avoid poor operation as a result of misleading information.

This paper introduces a novel scheduler based on fuzzy optimization called IP-FULL-FUZZY which considers uncertainties of application demands as well as of resource availability. This paper differs from previous work [5] since the latter consider only uncertainties of application demands. It is our knowledge that there is no other proposal in the literature that takes into account the two mentioned sources of uncertainties in grid scheduling.

It is important to mention that although the scheduler introduced here represents a preventive approach towards the handling of imprecise information, it does not aim to replace reactive approaches such as self-adjusting scheduling. The integration of both approaches seems to be a promising scheme.

This paper is organized as follow. Section 2 reviews previous work. Section 3 introduces a novel scheduler based on fuzzy optimization theory. Section 4 evaluates the proposed scheduler and Section 5 draws some conclusions.

SAC'09 March 8-12, 2009, Honolulu, Hawaii, U.S.A.

2. PREVIOUS WORK

In [3], the ILPDT scheduler was introduced. It considers discrete intervals of time ($\in \mathbb{Z}_+$) and treats the scheduling problem as an integer linear programming problem. The ILPDT was employed for the design of the ILP-FUZZY scheduler [5], which models the uncertainties of application demands as fuzzy numbers. Results indicate the benefit of this approach, specially to scenarios with high degree of uncertainty. This was a first step towards the development of schedulers robust to imprecise input information. Findings pointed out the need for the development of schedulers which include in their definition uncertainties of both application demands and of resource availability.

Two approaches for scheduling DAGs of dependent tasks without full knowledge of them were compared in [6]. However, no scheduler that accounts for uncertainty of application demands was proposed. The present works uses the same real network scenario and the same range of degree of uncertainty used in [6].

A dynamic approach to deal with uncertainties was introduced in [13]. The IP-FULL-FUZZY differs from the scheduler in [13] since the latter does not take into account uncertainties of the duration of the transfer of data. Moreover, evaluation of the scheduler in [13] did not include different degrees of uncertainties as is carried out in this paper.

The scheduler proposed in [8] does not distinguish sources of misleading information. As in this paper, triangular fuzzy numbers are considered in [8]. The work in [9] also assumes this type of shape but ignores weight values in DAGs describing tasks dependencies.

3. THE IP-FULL-FUZZY SCHEDULER

This section introduces the IP-FULL-FUZZY scheduler which takes into account uncertainties of applications demands as well as of resource availability. Its design capitalizes on previous investigations [3] [5].

The IP-FULL-FUZZY scheduler is based on an integer programming formulation. 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. Uncertainties of both applications demands and resource availability are represented by fuzzy numbers in the proposed formulation. The schedule given as solution defines the mapping of tasks to hosts as well as the timing for tasks to start execution.

The IP-FULL-FUZZY scheduler accepts two graphs as input. The graph $H = (V_H, E_H)$ represents the grid topology while the DAG $D = (V_D, A_D)$ the dependencies among tasks. In H, V_H is the set of m ($m = |V_H|$) hosts connected by the set of links E_H. 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 IP-FULL-FUZZY 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. IP-FULL-FUZZY 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.

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.

Uncertainties of applications demands, as well as those of resource availability are represented by fuzzy numbers in the proposed formulation. The values of I and B are represented by triangular fuzzy numbers. The i^{th} task requires I_i instructions with an uncertainty of $\sigma\%$ of this value; the amount of instructions is represented by $\widetilde{I_i} = [\underline{I_i}, I_i, \overline{I_i}]$ where $\underline{I_i} = I_i(1 - \frac{\sigma}{100})$ and $\overline{I_i} = I_i(1 + \frac{\sigma}{100})$. Similarly, communications demands are given by $\widetilde{B_{i,j}} = [\underline{B_{i,j}}, B_{i,j}, \overline{B_{i,j}}]$ with $\rho\%$ level of uncertainty, i.e., $\underline{B_{i,j}} = B_{i,j}(1 - \frac{\rho}{100})$ and $\overline{B_{i,j}} = B_{i,j}(1 + \frac{\rho}{100})$.

The processing capacity of the k^{th} host is given by $\widetilde{TI_k} = [\underline{TI_k}, TI_k, \overline{TI_k}]$ where $\underline{TI_k} = TI_k(1 - \frac{\chi}{100})$ and $\overline{TI_k} = TI_k(1 + \frac{\chi}{100})$ with $\chi\%$ representing the uncertainty degree. Moreover, the available bandwidth between hosts k and l is given by $\widetilde{TB_{k,l}} = [\underline{TB_{k,l}}, TB_{k,l}, \overline{TB_{k,l}}]$ with $\omega\%$ degree of uncertainty, i.e., $\underline{TB_{k,l}} = TB_{k,l}(1 - \frac{\omega}{100})$ and $\overline{TB_{k,l}} = TB_{k,l}(1 + \frac{\omega}{100})$.

For convenience, the following notation is used: $\mathcal{T} = \{1, \ldots, T''_{max}\}$, where $T''_{max} = T_{max}(1 + \frac{\sigma}{100})(1 + \frac{\chi}{100})$ and T_{max} is the time that the application would take to execute serially all the tasks in the fastest host, i.e., $T_{max} = \min(\{TI_{k|k \in V_H}\}) \times \sum_{i=1}^{n} I_i$. The minimum execution time achievable is obtained when all tasks in the shortest path of D (nodes in the SP set), considering the number of instructions as weights, are executed in the fastest host; such minimum time is represented as T''_{min}, where $T''_{min} = T_{min}(1 - \frac{\sigma}{100})(1 - \frac{\chi}{100})$ and $T_{min} = \min(\{TI_{k|k \in V_H}\}) \times \sum_{i \in SP} I_i$.

The IP-FULL-FUZZY 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 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 ($f_i \in \mathbb{N}^*$). These variables are related by:

$$\forall i \in V_D, f_i = \sum_{t \in \mathcal{T}} \sum_{k \in \mathcal{H}} t x_{i,t,k} \quad (1)$$

The IP-FULL-FUZZY is given by the following integer programming problem:

Maximize λ
subject to

$$1 - \frac{f_n - T''_{min}}{T''_{max} - T''_{min}} \geq \lambda \qquad \text{(F1)}$$

$$\sum_{t \in \mathcal{T}} \sum_{k \in V_H} x_{j,t,k} = 1 \qquad \text{for } j \in V_D; \qquad \text{(F2)}$$

$$x_{j,t,k} = 0 \qquad \text{for } j \in V_D, k \in V_H, \quad \text{(F3)}$$
$$t \in \{1, \ldots, \lceil \underline{I_j}\,\underline{TI_k} \rceil\};$$

$$\sum_{k \in \delta(l)} \sum_{s=1}^{\lceil t - \overline{I_j}\ \overline{TI_l} - \overline{B_{i,j}}\ \overline{TB_{k,l}} \rceil} x_{i,s,k}$$
$$\geq \sum_{s=1}^{t} x_{j,s,l} \qquad \text{for } j \in V_D, ij \in A_D, \quad \text{(F4)}$$
$$\text{for } l \in V_H, t \in \mathcal{T};$$

$$\sum_{j \in V_D} \sum_{s=t}^{\lceil t + \underline{I_j}\,\underline{TI_k} - 1 \rceil} x_{j,s,k} \leq 1 \qquad \text{for } k \in V_H, t \in \mathcal{T}, \quad \text{(F5)}$$
$$t \leq \lceil T''_{max} - \underline{I_j}\,\underline{TI_k} \rceil;$$

$$x_{j,t,k} \in \{0, 1\} \qquad \text{for } j \in V_D, l \in V_H, \quad \text{(F6)}$$
$$t \in \mathcal{T}.$$

The objective function of IP-FULL-FUZZY maximizes the satisfaction degree λ ($\in [0, 1]$) which is inversely proportional to the execution time of the application (f_n) given by a schedule. Restriction (F1) establishes the relationship between λ and f_n.

Restriction (F2) determines that a task must be executed in a single host while (F6) defines the domain for variables $x_{j,t,k}$ in the formulation.

Restrictions (F3), (F4) and (F5) establish relationships using the fuzzy numbers I_i, $B_{i,j}$, TI_k e $TB_{k,l}$ which should vary in their allowed range.

Restriction (F3) determines that a task (j) cannot terminate until all its instructions have been completely executed. Since it is possible to know neither the exact number of instructions, nor the host processing capacity, the minimum value of $\widetilde{I_j} \times \widetilde{TI_k}$, given by $\underline{I_j} \times \underline{TI_k}$, is used in (F3) to avoid resource under-utilization.

The constraints in (F4) 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, the $\widetilde{I_j} \times \widetilde{TI_k}$ and $\widetilde{B_{i,j}} \times \widetilde{TB_{k,l}}$ values are replaced by their maximum value given by $\overline{I_j} \times \overline{TI_k}$ and $\overline{B_{i,j}} \times \overline{TB_{k,l}}$, respectively.

The constraints in (F5) establish that there is at most one task in execution at any one host at a specific time. To maximize the number of tasks in a host, it is used the lowest execution time of tasks. The computational uncertainty yields to the replacement of $\widetilde{I_j} \times \widetilde{TI_k}$ by $\underline{I_j} \times \underline{TI_k}$.

4. PERFORMANCE EVALUATION

The effectiveness of the IP-FULL-FUZZY scheduler is compared to two other schedulers: the IP-APP-FUZZY and the RANDOM scheduler. The IP-APP-FUZZY considers only uncertainties of applications demand, i.e., it accepts input DAGs with edge weights given by fuzzy numbers. Comparison with the IP-APP-FUZZY scheduler allows the assessment of the impact of resource availability uncertainties on task scheduling. The RANDOM scheduler is a deterministic (non-fuzzy) scheduler and consequently does not consider any type of uncertainty. RANDOM was constructed by the relaxation of the integrality constraints of the linear programming formulation in [3]. One thousand drawings of random values are used in the search of the solution. Its execution time tends to be lower than the execution time of IP-FULL-FUZZY due to relaxation of integrality constraints and it is used as baseline for comparison.

The input to all schedulers is composed by two graphs; one representing the tasks composing the application and the other the grid. The input DAGs are taken from the Montage application (Figure 1), an astronomy application executed in real grids [6] which is used in several experiments in grids [11]. The input DAG for each scheduler is modified accordingly to the degree of uncertainty involved. The weights of the DAG edges were chosen from an uniform distribution [6]. Several others DAGs were used in the assessment of the performance of the IP-FULL-FUZZY scheduler. However, only results using the Montage DAG are reported in this paper due to space limitation. Note that although the DAG is composed by dependent tasks, the schedulers accept Bag of Tasks type of application. For that, zero-weight virtual tasks and links need to be added to the DAG.

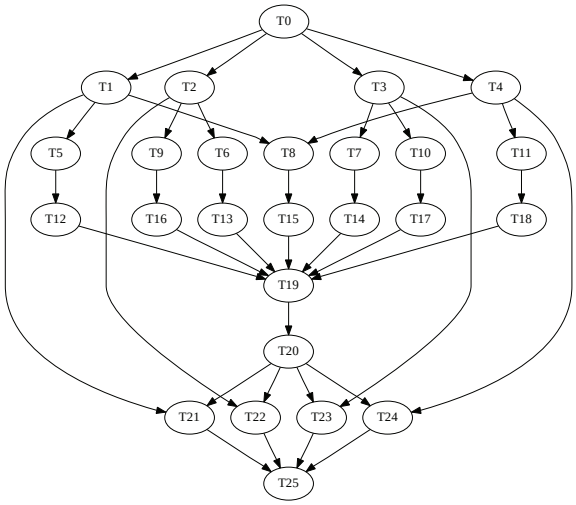

Figure 1: DAG of tasks used in the experiments.

Fifteen grids were generated for the evaluation of the IP-FULL-FUZZY scheduler by using the Doar-Leslie method [7] 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. The mean weight of the hosts is $\frac{9}{5{,}25 \times 10^{12}}$ minutes/instructions (9726MIPS, which is equivalent to the capacity of an Intel Pentium IV processor) and the mean weight of the links is $\frac{2}{12 \times 10^{9}}$ minutes/bit (100Mbps, the transmission rate of Fast Ethernet networks). Note that the scheduler works on the input topology according to the

degree of uncertainty assumed.

The degree of uncertainties adopted for application demands were $\{25\%, 50\%, 100\%, 200\%\}$ and for resource availability were $\{25\%, 50\%, 100\%, 200\%\}$. These values were taken from previous studies in the literature [13] [6]. Results considering the two types of uncertainties are shown. This is equivalent to say $\sigma = 0$, $\rho \in \{25\%, 50\%, 100\%, 200\%\}$, $\chi = 0$ and $\omega \in \{25\%, 50\%, 100\%, 200\%\}$ in the notation adopted.

For each scheduler designed to operate with a specific uncertainty level, DAGs and grids with different uncertainty levels were used as input. Twenty DAGs and twenty grids 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 different degrees of uncertainty of application demands. The following subsections show the speedup ($speedup = \frac{T_{max}}{makespan}$) and the time taken to produce the schedule (execution time) with confidence intervals with 95% confidence level. The capacity of the scheduler to deal with unforeseen situations is evaluated by varying the degree of uncertainty of the input data.

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 Pentium IV, 3.2GHz and 2.5GB RAM memory and with Debian GNU/Linux version Lenny operating system.

4.1 Speedup under Uncertainties of Application Demands

Figure 2 displays the mean speedup as a function of uncertainty of application demands for different levels of uncertainty of communication demands (ρ) the scheduler was designed for. In this example, the available bandwidth is known ($\omega = 0$). As expected, the two fuzzy schedulers produce the same speedups which show the correctness of IP-FULL-FUZZY when the availability of resources is known. The fuzzy schedulers perform worst than the deterministic RANDOM scheduler when the degree of uncertainty is less than 100%. When the degree is 100%, all schedulers perform roughly the same. However, the fuzzy schedulers produce speedup values higher than those produced by RANDOM when the degree of uncertainty is high (ρ=200%), which is common in e-Science applications given the amount of data produced in real time. 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. Moreover, the decay of the speedup produced by RANDOM increases with the degree of uncertainty which is an evidence of its inadequacy for scenarios with a high degree of uncertainty.

4.2 Speedup under Uncertainties of Grid Resources

Figure 3 displays the speedup as a function of the degree of uncertainty of available bandwidth for different degrees of uncertainty the IP-FULL-FUZZY was designed for (ω). The degree of uncertainty of the application demand (ρ) was fixed at 50% since for this value, the IP-FULL-FUZZY produces the worst results, as shown in Figure 2. In this way, the impact of the uncertainty of application demands on the speedup (Figure 3) does not mask the benefits of considering uncertainties of resource availability.

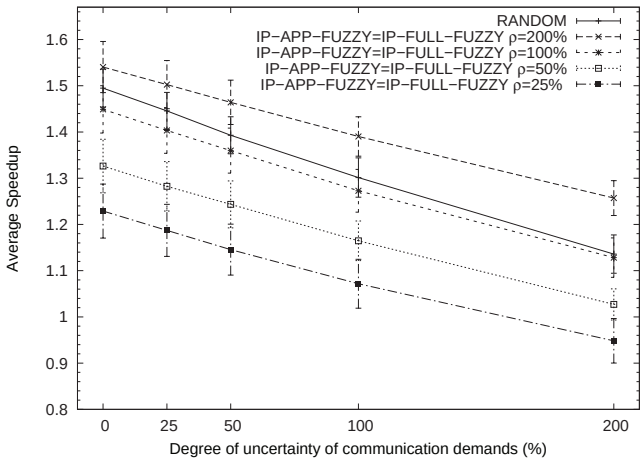

Figure 2: Speedup produced by schedulers as a function of different degrees of uncertainty of communication demands.

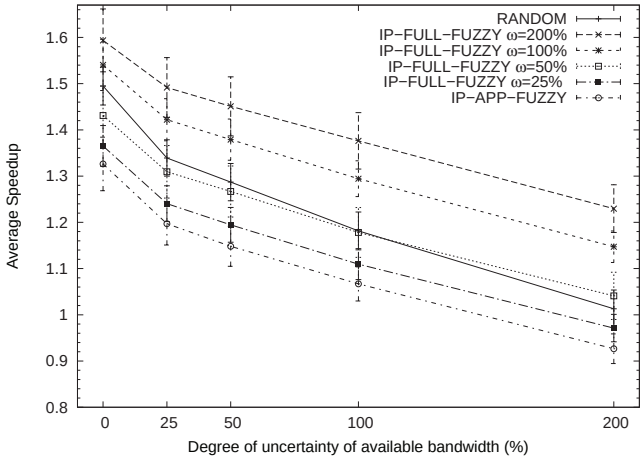

Figure 3: Speedup produced by schedulers as a function of uncertainty of the available bandwidth (ρ = 50% for IP-APP-FUZZY and IP-FULL-FUZZY).

The poor performance of the IP-APP-FUZZY scheduler in Figure 3 confirms the need for modeling the uncertainty of bandwidth availability. RANDOM overperforms IP-FULL-FUZZY for uncertainties of 25% given the limited flexibility of the fuzzy scheduler. With uncertainties of 50%, RANDOM and IP-FULL-FUZZY perform roughly the same. When the degree of uncertainties increases to 100% and over, the IP-FULL-FUZZY produces higher speedup values than those produced by RANDOM. The speedup is 0.22 higher than that of RANDOM for a degree of uncertainty of 200%.

4.3 Execution Time

Figure 4 compares the execution time of the schedulers involved in this study. As expected, RANDOM produces

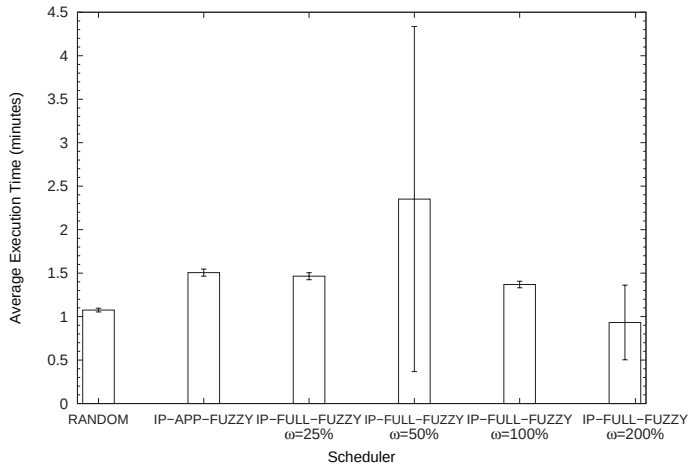

Figure 4: Execution time of the schedulers considered ($\rho = 50\%$ for IP-APP-FUZZY and IP-FULL-FUZZY) for different degrees of uncertainty of the available bandwidth.

the lowest execution time given the relaxation of the integrality constraints in the integer programming formulation. However, the execution time of the IP-FULL-FUZZY scheduler with $\omega = 200\%$ is on average 13,25% lower than that of RANDOM. The long execution time taken when the degree of uncertainty is 50% is due to a single run with execution time one order of magnitude longer than all other replications. The confidence interval for degree of uncertainty of 200% is wider than for others degrees of uncertainties due to the same reason.

5. CONCLUSION AND FUTURE WORK

Computation and communication demands of grid applications are usually informed by users who can only make a rough estimation of these demands. Uncertainty of these demands can cause unpredicted performance and can make scheduling ineffective. Moreover, the output of tools for estimating resource availability are typically given in ranges rather than as deterministic values. Besides that, results produced by these tools are imprecise by their own nature. An additional source of uncertainty is the fluctuation of resources availability during the elapsed time between the estimation of input data values and the production of a final schedule which can make ineffective a schedule based on deterministic data.

In this paper, a scheduler based on fuzzy optimization was proposed to schedule grid tasks under uncertainty of their demands as well as of resource availability. Results show that the speedup produced by IP-FULL-FUZZY is on average, 32% and 21% higher than those produced by a fuzzy scheduler which does not consider uncertainties of resource availability and by a deterministic (non-fuzzy) scheduler, respectively, and the execution time can be up to 38% and 13% lower than those taken by these schedulers. Results indicate that the effectiveness of the proposed 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 implementations. Currently, we are investigating the trade-off between the solutions given by fuzzy schedulers and those given by self-adapting schemes. Evaluation of the proposed scheduler fed by data generated by different measurement tools is also under investigation.

6. REFERENCES

[1] G. Allen, D. Angulo, I. Foster, G. Lanfermann, C. Liu, T. Radke, E. Seidel, and J. Shalf. The Cactus Worm: Experiments with Dynamic Resource Discovery and Allocation in a Grid Environment. *Int. Journal of High Performance Computing Applications*, 15(4):345–358, Nov. 2001.

[2] D. Antoniades, M. Athanatos, A. Papadogiannakis, E. P. Markatos, and C. Dovrolis. Available Bandwidth Measurements as Simple as Running wget. In *Proc. of Passive and Active Measurement Conference (PAM 2006)*, pages 61–70, 2006.

[3] D. M. Batista, N. L. S. da Fonseca, and F. K. Miyazawa. A Set of Schedulers for Grid Networks. In *Proc. of ACM SAC'07*, pages 209–213, 2007.

[4] D. M. Batista, N. L. S. da Fonseca, F. K. Miyazawa, and F. Granelli. Self-Adjustment of Resource Allocation for Grid Applications. *Computer Networks*, 52(9):1762–1781, 2008.

[5] D. M. Batista, A. C. Drummond, and N. L. S. da Fonseca. Scheduling Grid Tasks under Uncertain Demands. In *Proc. of ACM SAC'08*, pages 2041–2045, 2008.

[6] J. Blythe, S. Jain, E. Deelman, Y. Gil, K. Vahi, A. Mandal, and K. Kennedy. Task Scheduling Strategies for Workflow-based Applications in Grids. In *Proc. of CCGRID'05*, volume 2, pages 759–767, May 2005.

[7] M. Doar and I. M. Leslie. How Bad is Naive Multicast Routing? In *Proc. of IEEE INFOCOM'93*, pages 82–89, 1993.

[8] C. Fayad, J. M. Garibaldi, and D. Ouelhadj. Fuzzy Grid Scheduling Using Tabu Search. In *Proc. of IEEE Int. Fuzzy Systems Conference*, pages 1–6, Jul 2007.

[9] I. González-Rodrígues, C. R. Vela, and J. Puente. A Memetic Approach to Fuzzy Job Shop Based on Expectation Model. In *Proc. of IEEE Int. Fuzzy Systems Conference*, pages 7–12, Jul 2007.

[10] M. Jain and C. Dovrolis. End-to-end Available Bandwidth: Measurement Methodology, Dynamics, and Relation with TCP Throughput. *ACM SIGCOMM Comput. Comm. Rev.*, 32(4):295–308, 2002.

[11] NASA. Applications of Montage. `http://montage.ipac.caltech.edu/applications.html`. Accessed at July 6, 2008.

[12] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization – Algorithms and Complexity*, pages 363–366. Dover Publications, 1998.

[13] R. Sakellariou and H. Zhao. A Low-cost Rescheduling Policy for Efficient Mapping of Workflows on Grid Systems. *Scientific Programming*, 12:253 – 262, Dec 2004.

[14] S. S. Vadhiyar and J. J. Dongarra. A Performance Oriented Migration Framework for the Grid. In *Proc. of CCGRID'03*, pages 130–137, 2003.