Computer Networks xxx (2010) xxx–xxx

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Robust scheduler for grid networks under uncertainties of both application demands and resource availability

Daniel M. Batista, Nelson L.S. da Fonseca [*]

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

ARTICLE INFO

Article history:
Received 25 February 2009
Received in revised form 8 July 2010
Accepted 16 July 2010
Available online xxxx
Responsible Editor: R. LoCigno

Keywords:
Grid networks
Task scheduling
Uncertainty
Integer programming
Fuzzy optimization

ABSTRACT

Imprecise input data imposes special 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 that of three non-fuzzy schedulers, as well as to that of a fuzzy scheduler which considers only uncertainties of application demands. Results of simulations considering diverse scenarios, based on characteristics of real networks and grid applications, evince the advantages of the proposed scheduler.

© 2010 Published by Elsevier B.V.

1. Introduction

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

Furthermore, the scheduling problem is an NP-hard problem [1], and feasible solutions in real time require the use of either heuristics or approximations [2]. 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 to be transferred by different 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 [3,4]. Schedules produced by deterministic schedulers and based on imprecise input data can be quite different from optimal ones.

Adaptive scheduling, dynamic scheduling and self-adjusting scheduling have all been proposed in the literature [5–8] to deal with fluctuations of resource availability. All of these schemes were designed to minimize the makespan. Resource monitoring and task migration are used in these approaches to react to fluctuations in the grid state. However, continuous monitoring can increase the actual uncertainty level due to the intrusion effect, while unnecessary task migration can increase overhead, thus enlarging the makespan.

Another approach to minimize the negative impact of uncertainties is to take them into account in both

[*] Corresponding author. Tel.: +55 (19) 3521 5878; fax: +55 (19) 3521 5847.

E-mail addresses: batista@ic.unicamp.br (D.M. Batista), nfonseca@ic.unicamp.br (N.L.S. da Fonseca).

1389-1286/$ - see front matter © 2010 Published by Elsevier B.V.
doi:10.1016/j.comnet.2010.07.009

application demands and resource availability in the input data. One of the advantages of this approach is a reduction in the number of reconfigurations necessary to reduce the makespan of the applications; another 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 for the consideration of the uncertainties of application demands and resource availability. To our knowledge, there is no other proposal in the literature that simultaneously takes into account both of these sources of uncertainties in grid scheduling. The IP-FULL-FUZZY is a revised version of the scheduler presented in [9] and differs from that in [10], which considers only the uncertainties of application demands.

The ranges of potential values for the application demands and availability of grid resources in the IP-FULL-FUZZY scheduler are modeled by fuzzy triangular numbers [11,12] and they are denominated projected uncertainty levels. Expressing resource availability as ranges of values in the problem formulation is a realistic approach, since several popular monitoring and measurement tools furnish their results as ranges of values. For instance, the `pathload` [3] and the `abget` [4] tools express the end-to-end available bandwidth between a pair of hosts as a range of bandwidth values; such tools are adequate for grid systems as shown in [13].

The performance of IP-FULL-FUZZY was compared to that of a scheduler which considers only application demand uncertainties (IP-APP-FUZZY) and that of a non-fuzzy scheduler (RANDOM). Grid topologies similar to that of the Internet and several DAGs were employed in the evaluation. We simulated scenarios using characteristics of real networks and grid applications. It was shown that the IP-FULL-FUZZY scheduler is robust for both types of uncertainty, producing speedup values 33% and 21% higher than those given by IP-APP-FUZZY and RANDOM, respectively. Moreover, the execution time can be 38% lower than that of the other two schedulers. The proposed scheduler was also compared to the deterministic schedulers HEFT and CPOP, which are quite popular in grid scheduling analysis. Results indicate that the speedup produced by the IP-FULL-FUZZY is on average 25% higher than that of the HEFT and 17% higher than that of the CPOP. Furthermore, the scheduling decisions made by IP-FULL-FUZZY tend to decrease the network utilization when the level of uncertainty in data transfer demand is high.

This paper is organized as follow. Section 2 presents definitions of terms used in the rest of the paper. Section 3 reviews previous work. Section 4 justifies the need for the proposed scheduler. Section 5 introduces the IP-FULL-FUZZY scheduler, a scheduler based on fuzzy optimization theory which considers uncertainties of both application demands and of resource availability estimations. Section 6 evaluates the proposed scheduler and Section 7 draws some conclusions.

2. Terminology and definitions

This section presents, in alphabetical order, the definition of some terms used in the paper.

- *Application demands* – requirements of utilization of computational and communication resources to process an application. In this paper these requirements are, respectively, given by the number of instructions and the number of bits. For example, if an application has two tasks which transfer 8 Gb of data between themselves with 1×10^6 and 2×10^6 instructions, the application demands are represented by the set $\{\{1 \times 10^6$ instructions, 2×10^6 instructions$\}, \{1\ \text{Gb}\}\}$ (please see the definition of the term "instruction" and the term "task" mentioned below);
- *Directed Acyclic Graph (DAG)* – "DAG G is a pair (V, E), where V is a finite set and E is a binary relation on V. The set V is called the vertex set of G, and its elements are called vertices (singular: vertex). The set E is called the edge set of G, and its elements are called edges or arcs, so that there is no path in V that forms a cycle (a path $\langle v_0, v_1, \ldots, v_k\rangle$ forms a cycle if $v_0 = v_k$ and the path contains at least one edge)" [14]. Fig. 1 illustrates a DAG with $V = \{t0, t1, t2, t3, t4, t5\}$ and $E = \{(t0, t1), (t0, t2), (t0, t3), (t0, t4), (t1, t5), (t2, t5), (t3, t5), (t4, t5)\}$. Vertices are represented by ellipses in the figure, and edges are represented by arrows. DAGs are used to represent the dependencies among the tasks of an application [15,16]. The DAG in Fig. 1 represents the tasks of an application. The weights on the arcs connecting two nodes express the number of bits to be exchanged by two dependent tasks. The weights on the nodes represent the amount of instructions to be processed. The strings between parenthesis are used to identify the tasks;
- *Execution time* – also known as response time, gives the total time required for the computer to complete a program, including disk accesses, memory access, I/O activities, operating system overhead, CPU execution time, and so on [17];
- *Fuzzy optimization* – an optimization problem in which the objective function or/and the constraints involve fuzzy numbers [18,19] (please see the definition of the term "membership function" mentioned below);
- *Grid applications* – applications that are executed on grids;
- *Grid schedule* – a mapping of the tasks of an application on the hosts of a grid jointly with the designated starting time of the execution of these tasks;
- *Grid scheduler* – a program that produces a grid schedule [20];

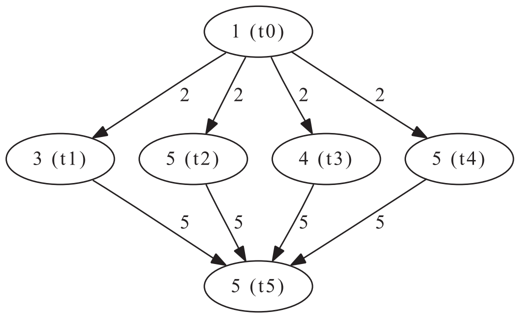


Fig. 1. Example of a DAG with 6 vertices and 8 arcs.

- *Grid scheduling* – process that aims to produce a grid schedule [21];
- *Heuristics* – or heuristic algorithms are algorithms that provide a sub-optimal solution, but without a guarantee on its quality. Although the running time is not guaranteed to be polynomial, empirical evidence suggests that some of these algorithms find a good solution fast. [22];
- *Instruction* – an assembly instruction executed in a processor. The computational demand of a task is represented by the number of machine instructions to be executed. In this paper we do not consider issues related to the compatibility of instruction sets of different processors. Virtualization techniques can be used to address this issue [23];
- *Integer programming or integer linear programming* – a deterministic optimization method. "Many problems can be formulated by maximizing or minimizing an objective, given limited resources and competing constraints. If it is possible to specify the objective as a linear function of certain variables, and if it is possible to specify the constraints on resources as equalities or inequalities on those variables, then we have a linear-programming problem. If we add to a linear program the additional requirement that all variables take on integer values, we have an integer linear program" [14];
- *Makespan* – elapsed time between the submission of a parallel application to a distributed/parallel system for processing and the ending time of the processing of this application [24]. Although the term "execution time" is also valid to define the time needed to process a parallel application, the term makespan is generally used;
- *Membership function* – or degree of truth $\mu(x)$, defines the degree of membership for the element x in a given fuzzy set [25]. When the fuzzy set represents an interval $\in \mathbb{R}$, it is also named a fuzzy number. Its elements vary in a range $[min, max]$ and they are associated with the membership function $\mu(x),\ x \in \mathbb{R},\ \mu(x) \in [0,1]$; For $x < min$ and $x > max$, the membership function is null. The greater the value of $\mu(x)$, the greater is the truth about the membership of x;
- *Projected uncertainty level* – the ranges of potential values for the application demands and for the availability of grid resources in the IP-FULL-FUZZY scheduler;
- *Quality of information (QoI)* – describes the degree of certainty in relation to a specific value [26] and it is given by:

$$\begin{aligned}&\text{if (true value} > \text{estimated)}\\ &\quad \rightarrow \text{QoI} = 100\% \times \left(1 - \frac{\text{true value} - \text{estimated}}{\text{true value}}\right),\\ &\text{else} \rightarrow \text{QoI} = 100\% \times \left(1 - \frac{\text{estimated} - \text{true value}}{\text{estimated}}\right),\end{aligned}$$

- *Resource availability* – existence of resource that can be used immediately;
- *Shortest path* – "given a weighted, directed graph $G = (V,E)$, with weight function $w : E \rightarrow \mathbb{R}$ mapping edges to real-valued weights. The weight of path $p = \langle v_0, v_1, \ldots, v_k \rangle$ is the sum of the weights of its constituent edges:

$$w(p) = \sum_{i=1}^{k} w(v_{i-1}, v_i).$$

The shortest-path weight from u to v is defined by:

$$\delta(u,v) = \begin{cases} min\{w(p) : u \overset{p}{\rightsquigarrow} v\} & \text{if there is a path from } u \text{ to } v,\\ \infty & \text{otherwise.}\end{cases}$$

A shortest path from vertex v to vertex u is then defined as any path p with weight $w(p) = \delta(u,v)$" [14];
- *Speedup* – ratio between the time taken to process sequentially the tasks of an application and the time to process the same set of tasks in a non-sequential fashion [20,27];

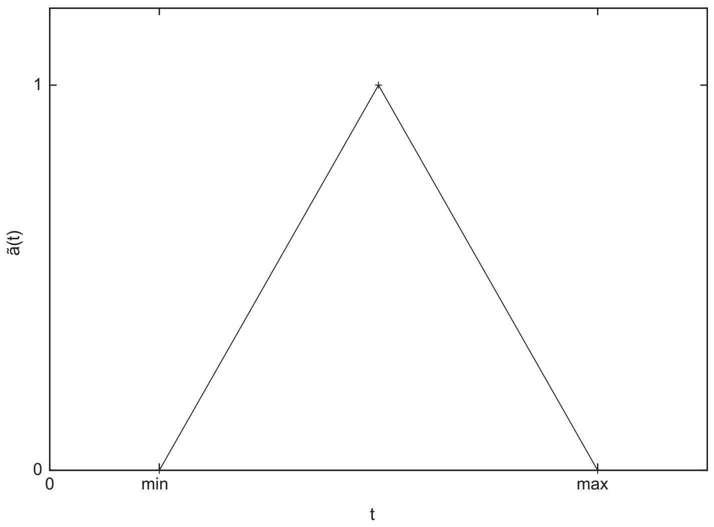


Fig. 2. Triangular fuzzy number [*min*, *max*].

- *Task* – sub-set of data and executable instructions of a parallel application. Some authors define a task as anything that needs resources [24]. Usually, a parallel application (program) is broken into several tasks for parallel processing. The division of applications into tasks is defined by grid users or by the application developers according to the application logical structure. For instance, a division that takes into consideration the nodes of a DAG [24];
- *Triangular fuzzy number* – a fuzzy number $\tilde{a}$ with membership function defined as [28]:

$$\tilde{a}(t) = \begin{cases} 1 - \frac{a-t}{\alpha} & \text{if } a - \alpha \leqslant t \leqslant a, \\ 1 & \text{if } a \leqslant t \leqslant b, \\ 1 - \frac{t-b}{\beta} & \text{if } a \leqslant t \leqslant b + \beta, \\ 0 & \text{otherwise.} \end{cases}$$

Fig. 2 illustrates a triangular fuzzy number $\tilde{a}$ with $a = b = (max + min)/2$ and $\alpha = \beta = (max - min)/2$.

3. Previous work

The ILPDT scheduler was introduced in [2]. 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 [10], which models the uncertainties of application demands as fuzzy numbers. Encouraging results derived when using the ILP-FUZZY scheduler motivated the development of the IP-FULL-FUZZY scheduler, which considers uncertainties in both application demands and resource availability.

The negative impact of the uncertainties in application demands in two approaches for scheduling DAGs of dependent tasks was presented in [29]. However, no scheduler accounting for the uncertainty in application demands was proposed. The present paper uses the same real network scenario and the same range of uncertainty level used in [29].

A dynamic approach for dealing with uncertainties was introduced in [7]. The IP-FULL-FUZZY differs from the scheduler in [7], however, since the latter did not take into account uncertainties related to the duration of the transfer of data. Moreover, evaluation of the scheduler in [7] did not include different uncertainty levels as has been done here.

As in the present paper, the scheduler proposed in [11] uses triangular fuzzy numbers, but it did not distinguish sources of misleading information. The work in [12] also assumes this type of shape but ignored weight values in the DAGs describing task dependencies.

In [30], two heuristic-based schedulers (called HEFT – Heterogeneous Earliest-Finish-Time and CPOP – Critical-Path-on-a-Processor) were introduced for the minimization of the makespan of applications submitted to heterogeneous parallel systems, while maintaining low computational complexity, and they are widely used in the literature for comparison of new proposed schedulers [7,31–34]. The HEFT schedules each task on the basis of criteria of the longest path up to the final task in the DAG. The CPOP scheduler also uses the longest path from the first task to the one being scheduled as a criterion and tasks on the critical path are scheduled to a single processor. Neither scheduler, however, considers uncertainties in the input data, which results in a negative impact on the performance. This also justifies the need for a scheduler such as that proposed in this paper.

There is a clear need for a scheduling framework based on fuzzy optimization, and in [9], we proposed the IP-FULL-FUZZY. We made a preliminary evaluation of its performance at that time, but have since then extended the scenarios and applications for which it has been evaluated. Such results present a stronger case for the effectiveness of the IP-FULL-FUZZY than does the work in [9]. Moreover, we present the transformation of the fuzzy optimization problem into an integer programming problem and the performance of IP-FULL-FUZZY is compared with that of four different schedulers, including the CPOP and the HEFT schedulers which is also a distinct aspect of this paper.

4. Motivation

The purpose of a grid scheduler is to determine the assignment of tasks to hosts, as well as the sequence of task executions, i.e., the schedule. Schedules are derived according to specific objectives related to the type of application. For instance, schedules for the execution of e-Science applications usually target the minimization of the makespan. The search for an optimum schedule is an NP-hard problem and heuristics and approximation techniques are generally employed to find a solution close to the optimal one. They receive as input the computational and the communication demands (Fig. 3(a)) and information about grid resource availability (Fig. 3(b)) to produce schedules which are used by grid management agents to allocate resources, dispatch tasks to hosts and initiate execution.

This section illustrates the need for fuzzy scheduling procedures. The example in Fig. 3 is adopted as a reference for arguments of motivation. Fig. 3(a) illustrates the DAG of an application. Fig. 3(b) represents the grid resources: nodes represent hosts with labels expressing the inverse of host capacity $(\text{instructions/unit of time})^{-1}$ and the edges represent the network links, with the labels giving the inverse of the available bandwidth $(\text{bits/unit of time})^{-1}$. The dotted lines denote the presence of other resources in the grid.

To produce input data with $\text{QoI} < 100\%$, the time to transfer one bit was increased, i.e., the available bandwidth was reduced, leading to overestimation of the bandwidth by the scheduler. Increases in bit transfer time of 25%, 50%, 100% and 200% imply QoI values of 80%, 66.67%, 50% and 33.33%, respectively.

The HEFT and CPOP schedulers were used to generate the scheduling of the application. As these schedulers do not deal with the uncertainty of application demands and resource availability, they were expected to produce poor schedules when input information did not correspond to the real demands. The CPOP and the HEFT schedulers were

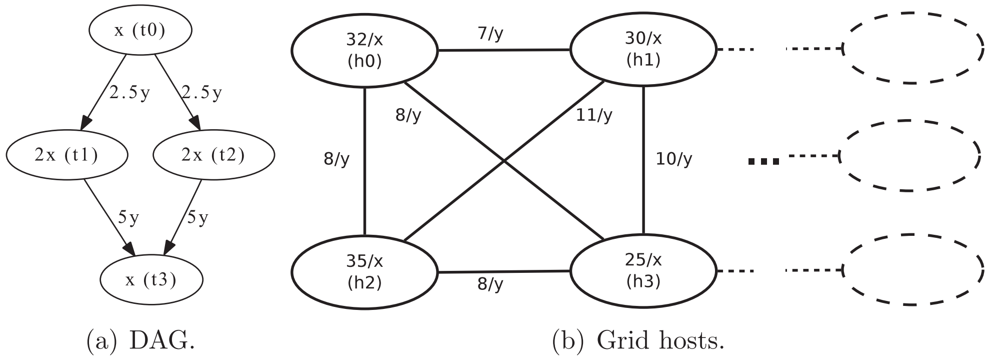


Fig. 3. An example to illustrate the need for fuzzy scheduling.

thus executed under two conditions: with 100% QoI input data and with input data with less than that.

Fig. 4 shows the makespans predicted by these schedulers when the true available bandwidth was known (curves labeled as "precise estimation") and when the QoI was less than 100% (curves labeled as "imprecise estimations"). The overlap of curves when QoI was less than 50% shows the importance of both magnitude of uncertainty and relevance on the performance. In this example, the uncertainty of the data transfer time had only a minor impact on the execution time for QoI $\leqslant$ 50% due to the parallelization adopted. However, when the transfer time was increased by more than 100%, the performance of deterministic schedulers fed by imprecise information underwent significant degradation.

The makespan produced by the deterministic schedulers with QoI < 100% increased, as expected, being 70% higher for increases of 200% for the time for the CPOP scheduler and 54% higher for HEFT. This suggests that deterministic schedulers produce poor schedules when resource availability is lower than expected. One way to counteract this deficiency is to consider the Quality of Information of estimated demand as input to the scheduler by estimating this demand as a fuzzy value. Furthermore, as will be shown in the next section, the IP-FULL-FUZZY scheduler is able to produce schedules that leads to performance when the QoI is less than 100% equivalent to the performance of schedules derived when the QoI is 100%.

5. The IP-FULL-FUZZY Scheduler

This section introduces the IP-FULL-FUZZY scheduler. Throughout the paper, we denote the uncertainty level existing in the input data describing the application or the grid resources as "actual uncertainty level" and the uncertainty level which the scheduler was designed to handle as "projected uncertainty level".

The IP-FULL-FUZZY scheduler is based on a fuzzy optimization formulation. This formulation is then transformed into a crisp equivalent model based on an integer programming formulation [25].

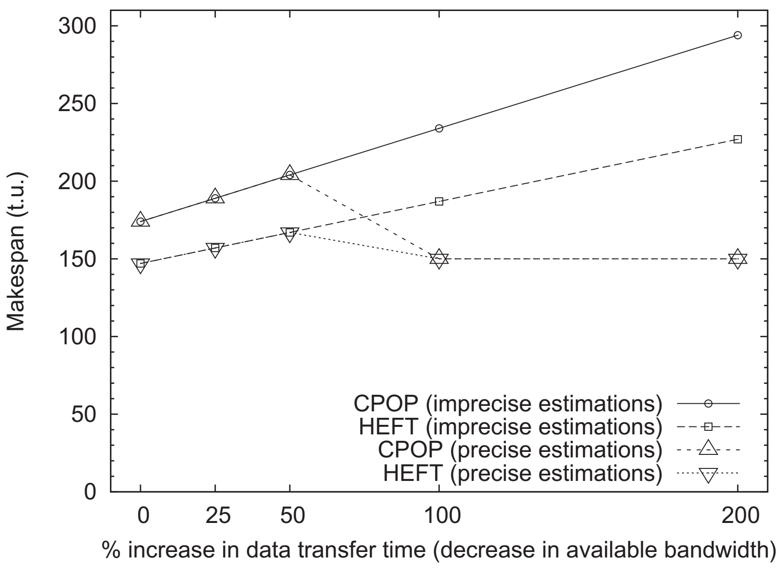


Fig. 4. Makespan in scenarios with precise and imprecise estimations.

We consider that the IP-FULL-FUZZY scheduler defines a schedule on a discrete timeline. Although the discretization of time introduces approximation and a consequent loss of precision, under certain circumstances, this loss may not be significant, and the savings in time can be quite attractive when compared to a corresponding scheduler which assumes time as a continuous variable. Uncertainties in both application demands and resource availability are represented by fuzzy numbers in the proposed formulation. The schedule given as a solution defines the mapping of tasks to hosts, as well as the timing for the initiation of the execution of tasks.

The IP-FULL-FUZZY scheduler accepts two graphs as input. The graph $H = (V_H, E_H)$ represents the grid topology, while the DAG $D = (V_D, A_D)$ indicates 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 labeled from 1 to m. In D, V_D is the set of n ($n = |V_D|$) tasks with integer numbers as labels, which allow a topological ordering of tasks, and A_D is the set of dependencies.

The IP-FULL-FUZZY scheduler considers that the input DAG have a single input task and a single output task. DAGs failing to satisfy this condition because they involve more than one input or output task can be easily modified by considering two null tasks with both processing time and communication weight equal to zero. IP-FULL-FUZZY provides as output, a list which provides information about the host on which each task should be executed, the starting time of that task, and the time when data transfer should take place.

Some characteristics of the DAGs are the following: I_i is the processing demand of the ith task, expressed as number of instructions to be processed by the ith task ($I_i \in \mathbb{R}_+$); $B_{i,j}$ is the number of bytes transmitted between the ith task and the jth task ($B_{i,j} \in \mathbb{R}_+$); $\mathcal{D}$ is the 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 is the time the kth host takes to execute 1 instruction ($TI_k \in \mathbb{R}_+$); $TB_{k,l}$ is the time for transmitting 1 bit on the link connecting the kth host to the lth host ($TB_{k,l} \in \mathbb{R}_+$); $\delta(k)$ is the set of hosts linked to the kth 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.

The values of I and B are represented by triangular fuzzy numbers. The ith task requires I_i instructions with a projected uncertainty level of σ% of this value; the quantity 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, communication demands are given by $\widetilde{B_{i,j}} = [\underline{B_{i,j}}, B_{i,j}, \overline{B_{i,j}}]$, with ρ% of projected uncertainty level, 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 lower the value of σ, the closer to the true computational demand are the weight values furnished by the DAG. Similarly, the lower the value of ρ, the closer to the true communication demand are the weight values furnished in the DAG.

The processing capacity of the kth 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 χ% representing the projected uncertainty level. 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 ω% projected uncertainty level, i.e., $\underline{TB_{k,l}} = TB_{k,l}(1 - \frac{\omega}{100})$ and $\overline{TB_{k,l}} = TB_{k,l}(1 + \frac{\omega}{100})$. The lower the value of χ, the closer is the estimation of the processing availability to the true value in the hosts. Note that the values of TI_k are estimated considering the existing processing load on the host. The greater the number of jobs competing for CPU, the higher are the TI_k values. Similarly, the lower the values of ω, the closer to the true available bandwidth are the estimated ones. Note that TB is a fuzzy triangular number which describes the estimations provided by popular available bandwidth estimators such as `abget` and `pathload` [35]. Note, also, that such estimations are furnished as input to the schedulers, since the role of the fuzzy scheduler is to make decisions which should be robust in the face of the uncertainties as a result of the use of monitoring/measuring tools.

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 on the fastest host, i.e., $T_{max} = \min(\{TI_{k|k \in V_H}\}) \times \sum_{i=1}^{n} I_i$. The minimum makespan achievable is obtained when all tasks on the shortest path (SP) of D (considering the number of instructions as weights), are executed on the fastest host. This minimum time is represented by 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 values of T_{min} and T_{max} are computed as if there were no actual uncertainty level in either the DAG or the grid. The variables σ and χ introduce the projected uncertainty level in T''_{max} and T''_{min}.

The IP-FULL-FUZZY scheduler solves a linear integer program which seeks the value of the 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 ith task finishes at time t on 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 ith task is finished ($f_i \in \mathbb{N}^*$). These variables are related as follows:

$$\forall i \in V_D,\ f_i = \sum_{t \in \mathcal{T}} \sum_{k \in \mathcal{H}} t x_{i,t,k}. \tag{1}$$

The IP-FULL-FUZZY is given by the following fuzzy optimization problem:

$$\text{Minimize } \widetilde{f_n}$$

$$\text{subject to } \sum_{t \in \mathcal{T}'} \sum_{k \in V_H} x_{j,t,k} = 1 \quad \text{for } j \in V_D; \tag{F'1}$$

$$x_{j,t,k} = 0 \quad \begin{array}{l} \text{for } j \in V_D, \quad k \in V_H, \\ t \in \{1, \ldots, \lceil \widetilde{I_j} \widetilde{TI_k} \rceil\}; \end{array} \tag{F'2}$$

$$\sum_{k \in \delta(l)} \sum_{s=1}^{\lceil t - \widetilde{I_j}\widetilde{TI_l} - \widetilde{B_{i,j}}\widetilde{TB_{k,l}} \rceil} x_{i,s,k} \geqslant \sum_{s=1}^{t} x_{j,s,l} \quad \begin{array}{l} \text{for } j \in V_D, \quad ij \in A_D, \\ \text{for } l \in V_H, \quad t \in \mathcal{T}'; \end{array} \tag{F'3}$$

$$\sum_{j \in V_D} \sum_{s=t}^{\lceil t + \widetilde{I_j}\widetilde{TI_k} - 1 \rceil} x_{j,s,k} \leqslant 1 \quad \begin{array}{l} \text{for } k \in V_H, \quad t \in \mathcal{T}', \\ t \leqslant \lceil T'_{max} - \widetilde{I_j}\widetilde{TI_k} \rceil; \end{array} \tag{F'4}$$

$$x_{j,t,k} \in \{0,1\} \quad \begin{array}{l} \text{for } j \in V_D, \quad l \in V_H, \\ t \in \mathcal{T}'. \end{array} \tag{F'5}$$

The objective function involves the minimization of a fuzzy variable as well as constraints (F′2)–(F′4) involve fuzzy variables. To solve this problem, the fuzzy constraints and the objective function must be transformed into corresponding crisp expressions, leading to an integer programming formulation of the original problem [25]. In this case, the equivalent integer programming problem is given by:

Maximize λ

subject to $f_n \leqslant (1-\lambda)T''_{max} + \lambda T''_{min};$ (F1)

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

$$x_{j,t,k} = 0 \quad \begin{array}{l}\text{for } j\in V_D,\ k\in V_H,\\ t\in\{1,\ldots,\lceil \underline{I_j}\underline{TI}_k\rceil\};\end{array} \tag{F3}$$

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

$$\sum_{j\in V_D}\sum_{s=t}^{\lceil t+\underline{I_j}\underline{TI}_k-1\rceil} x_{j,s,k} \leqslant 1 \quad \begin{array}{l}\text{for } k\in V_H,\ t\in\mathcal{T},\\ t\leqslant\lceil T''_{max}-\underline{I_j}\underline{TI_k}\rceil;\end{array} \tag{F5}$$

$$x_{j,t,k}\in\{0,1\} \quad \begin{array}{l}\text{for } j\in V_D,\ l\in V_H,\\ t\in\mathcal{T}.\end{array} \tag{F6}$$

The objective function of the integer programming version of the IP-FULL-FUZZY scheduler maximizes the degree of satisfaction λ ($\in[0,1]$), which is inversely proportional to the makespan of the application (f_n) given by a schedule. Constraint (F1) establishes the relationship between λ and f_n. This objective function jointly with constraint (F1) are equivalent to the objective function of the fuzzy optimization formulation.

Constraint (F2) determines that a task must be executed on a single host while (F6) defines the domain for variables $x_{j,t,k}$ in the formulation. These two constraints are equivalent to the constraints (F′1) and (F′5) of the fuzzy optimization formulation, respectively. As (F′1) and (F′5) did not involve fuzzy variables, the constraints remained the same.

Constraints (F3)–(F5) establish relationships using the fuzzy numbers I_j, $B_{i,j}$, TI_k, TI_l and $TB_{k,l}$, which should vary in the range allowed. The constraints (F3)–(F5) in the integer programming formulation correspond, respectively, to the constraints (F′2)–(F′4) in the fuzzy optimization formulation. Explanation on the transformation of these constraints is given next.

Constraint (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 processing capacity of the host, 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 underutilization.

The constraints in (F4) establish that the jth task cannot start execution until all predecessors have finished their execution and transferred the data required by the jth task. In this way, in order to prevent the potential execution of the jth task prior to the execution of its predecessors due to the projected uncertainty level, the $\widetilde{I_j}\times\widetilde{TI_k}$ and $\widetilde{B_{i,j}}\times\widetilde{TB_{k,l}}$ values are replaced by their maximum values, 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 a single task in execution on any one host at a specific time. To maximize the number of tasks on a host, the lowest time for tasks is used. Computational uncertainty thus yields the replacement of $\widetilde{I_j}\times\widetilde{TI_k}$ with $\underline{I_j}\times\underline{TI_k}$.

Although the solution of the optimization problem gives end times for tasks, the starting times are derived by using the relation $f_i-(\overline{TI_l}\times\overline{I_i})$ where $\overline{TI_l}$ and $\overline{I_i}$ account for the uncertainty of the true I_i and TI_i values.

Since applications are described as a set of dependent tasks with a single ending task, the tasks are ordered so that the nth task is the last one; the makespan is thus given by the ending time of the last task, f_n.

6. Performance evaluation

To evaluate the effectiveness of schedulers designed to deal with uncertainties (defined by σ, ρ, χ, and ω), DAGs of applications and graphs describing the grid resource availability are fed to the schedulers to produce a schedule.

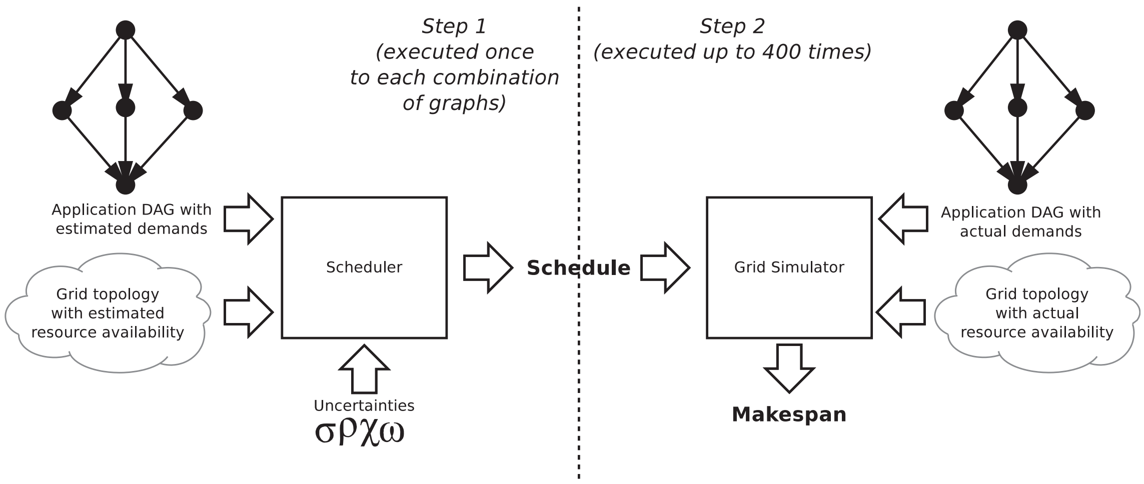


Fig. 5. Simulation process flowchart.

This schedule is then used as input for an event-driven grid simulator, developed by the authors, which also receives a DAG describing the actual application and a graph describing actual resource availability.

The weights in the graphs provided to the scheduler represent the estimations based on measurement/monitoring, while those in the graphs given to the simulator represent the actual values of the real system.

Each schedule produced was confronted with numerous combinations of actual application demand and resource availability. Such a confrontation serves to evaluate the robustness of the scheduler. Up to 20 DAGs representing actual application demands and 20 graphs representing resource availability were generated by randomly changing the weights of their edges.

The makespan reported is the result of the evaluation of all combinations of the DAGs for actual weight values.

Fig. 5 illustrates the process. In the first step a schedule is produced by a scheduler with a projected uncertainty level and in the second step a simulator, considering the schedule derived and numerous combinations of graphs with actual weight values, is executed.

The effectiveness of the IP-FULL-FUZZY scheduler is compared to that of four other schedulers: IP-APP-FUZZY, RANDOM, HEFT and CPOP. The IP-APP-FUZZY considers only uncertainties of application demand, i.e., it accepts input DAGs describing application demands when weights are expressed as fuzzy numbers. Comparison with the IP-APP-FUZZY scheduler allows the assessment of the impact of uncertainties of resource availability 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 [2]. One thousand drawings of random values are used in the search for a solution by RANDOM. Its execution time tends to be less than that of IP-FULL-FUZZY due to relaxation of these integrality constraints, and it is used here as a baseline for comparison. The HEFT and CPOP deterministic schedulers are also used in the comparison given their wide use in the literature.

For both IP-APP-FUZZY and RANDOM, the parameters of χ and ω are null, moreover, for RANDOM, both ρ and σ are also null. For this latter scheduler, the variables $x_{i,t,k}$ are real variables treated as probability values in order to find the final schedule.

Three DAGs of real applications were used to evaluate the schedulers. Since there are no standard benchmarks in this area, evaluation is based on DAGs of real applications, with the weight values of the DAGs varied according to a uniform distribution, as explained above. The first DAG was taken from an astronomy application called Montage (Fig. 6), which is widely used in grid evaluation [29,15], with weights randomly chosen from a uniform distribution with an edge weight mean of 72×10^6 bits, and a vertex weight mean of 31.5×10^{12} instructions. This DAG represents 26 tasks, with 39 dependencies among them. The recency of the emergence of the relevant technology, and the lack of established benchmarks has led to the heterogeneity in the applications presented here.

The second DAG corresponds to an application in quantum chemistry called WIEN2k [16] (Fig. 7) developed at the Vienna University of Technology [32]. This DAG represents 26 tasks with 43 dependencies; weights are assigned in the same way as those for the Montage DAG. The third DAG is a modified version of the WIEN2k with 48 dependencies involving parallel tasks in addition to the input and output tasks (Fig. 8).

Fifteen grids were generated for the evaluation of the IP-FULL-FUZZY scheduler by using the Doar–Leslie method [36], which is widely used to generate Internet topologies. Fifty hosts were considered, with a degree of node connectivity (β) of 0.98 and the ratio between the quantity of longest and shortest links being 0.98. The mean weight of the

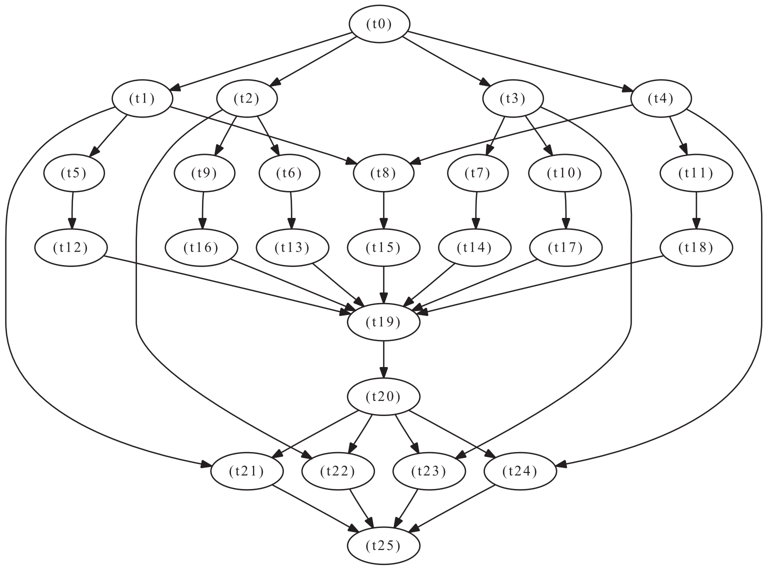


Fig. 6. Montage DAG.

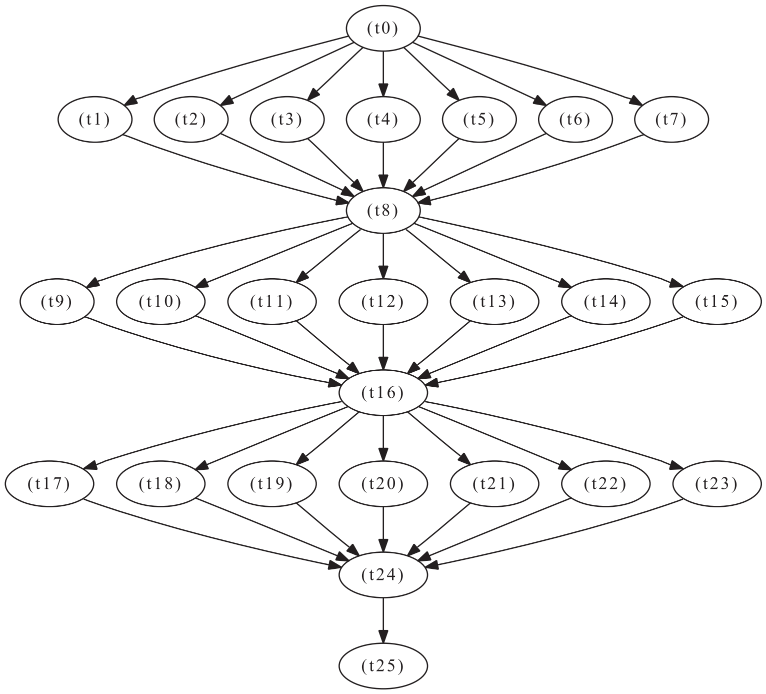


Fig. 7. WIEN2k DAG.

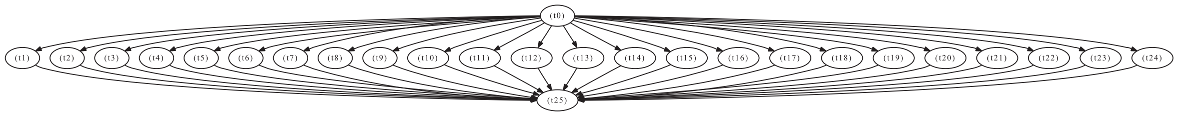


Fig. 8. Modified-WIEN2k DAG.

hosts is $\frac{9}{5.25\times10^{12}}$ min/instructions (9726 MIPS, which is equivalent to the capacity of an Intel Pentium IV processor) and the mean weight of the links is $\frac{2}{12\times10^{9}}$ min/bit (100 Mbps, the transmission rate of Fast Ethernet networks).

In other words, 45 different input combinations of 3 application DAGs and 15 grids were used as input for the IP-FULL-FUZZY scheduler.

The projected uncertainty levels adopted for application demands were 25%, 50%, 100%, 200% and for resource availability they were also 25%, 50%, 100%, 200%. These values were taken from previous studies in the literature [7,29]. Results considering the two types of uncertainties are shown. This is equivalent to saying $\sigma = 0$, $\rho \in \{25\%, 50\%, 100\%, 200\%\}$, $\chi = 0$ and $\omega \in \{25\%, 50\%, 100\%, 200\%\}$ in the notation adopted.

For the verification of the robustness of the schedules produced, twenty DAGs and twenty grids with randomly generated weights were fed into the schedulers. Each of these twenty DAGs was generated by increasing the weight of the edges (for $\rho \neq 0$), and each of the twenty grids was generated by increasing the weight of the edges (for $\omega \neq 0$) in a uniformly distributed range from 0 to x%, where $x \in \{25\%, 50\%, 100\%, 200\%\}$, x being the actual level of uncertainty. In this way, it was possible to evaluate how well a scheduler designed to operate under a specific projected level of uncertainty handles different levels of actual uncertainty.

The following subsections show the speedup, the network utilization, and the time required to produce the schedule (execution time) with a confidence level of 95%. Each confidence interval considers the combination of graphs used as input to the simulator.

Both schedulers and simulator were written in the C programming language, and the optimization library `Xpress` version `2006A.1` was used to develop them. Programs were executed on a Pentium IV, 3.2 GHz computer with 2.5 GB of RAM memory and a Debian GNU/Linux version Lenny operating system.

6.1. Speedup with uncertainties in application demands

Figs. 9–11 display, respectively, the mean speedup for the Montage, WIEN2k and modified-WIEN2k DAGs as a function of actual uncertainty levels in application demands for different projected levels of uncertainty in communication demands (ρ) that the scheduler was designed to meet. In this example, the available bandwidth is known ($\omega = 0$). As expected, the two fuzzy schedulers produce the

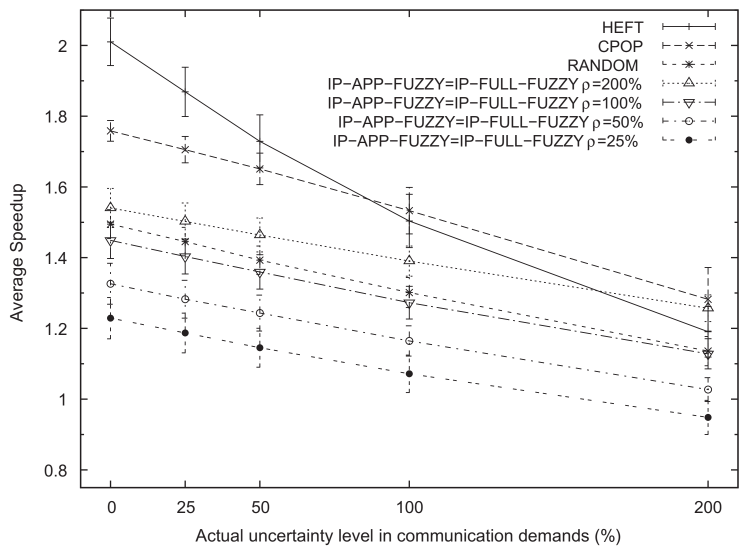


Fig. 9. Speedup produced by schedulers as a function of different actual uncertainty levels in communication demands (Montage DAG).

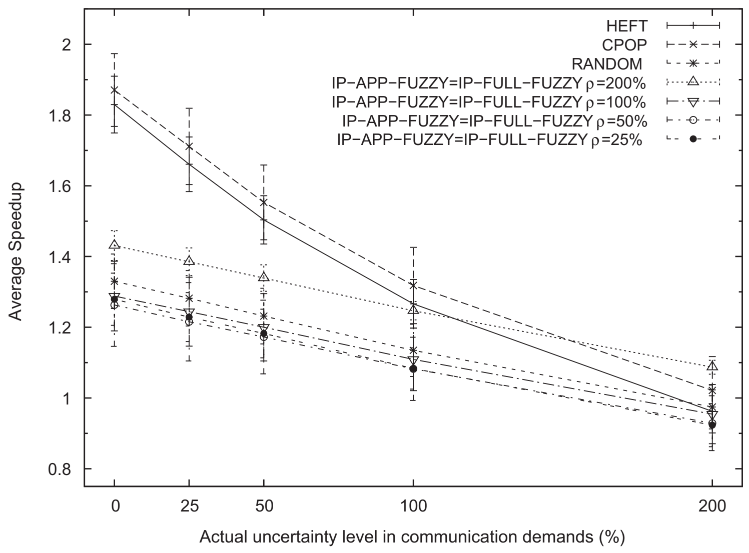


Fig. 10. Speedup produced by schedulers as a function of different actual uncertainty levels in communication demands (WIEN2k DAG).

same speedups, which shows the accuracy of the IP-FULL-FUZZY scheduler when the availability of resources is known.

Fig. 9 shows that the fuzzy schedulers perform worse than the deterministic RANDOM scheduler when the projected uncertainty level (ρ) is less than 100% for the Montage DAG. When the projected uncertainty level is 100%, all schedulers perform roughly the same. However, the fuzzy schedulers produce higher speedup values than those produced by the RANDOM one when the projected uncertainty level is high ($\rho = 200\%$). For instance, when $\rho = 200\%$ and $x = 200\%$, the speedup produced when IP-FULL-FUZZY is used can be up to 11% greater than that produced by RANDOM. High QoI values are usual in e-Science applications since data is produced (collected) in run time. When flexibility is quite limited (low QoI values), fuzzy solutions tend to be less precise. If flexibility increases, the enhanced ability to handle uncertainty of demands may overcompensate potential mistakes. Furthermore, the speedup produced by schedulers with a high projected uncertainty level is quite robust in relation to variations in the actual uncertainty levels of demands. Moreover, the decay of the speedup produced by fuzzy schedulers is less than that resulting from the use of RANDOM, which evidences its adequacy for scenarios with a high actual uncertainty level. Fig. 9 thus reinforces the fact that deterministic schedulers perform well when the degree of uncertainty is low (in this case, for uncertainties lower than 50%, regardless of the value of ρ), as was shown in Section 4. However, the performance of

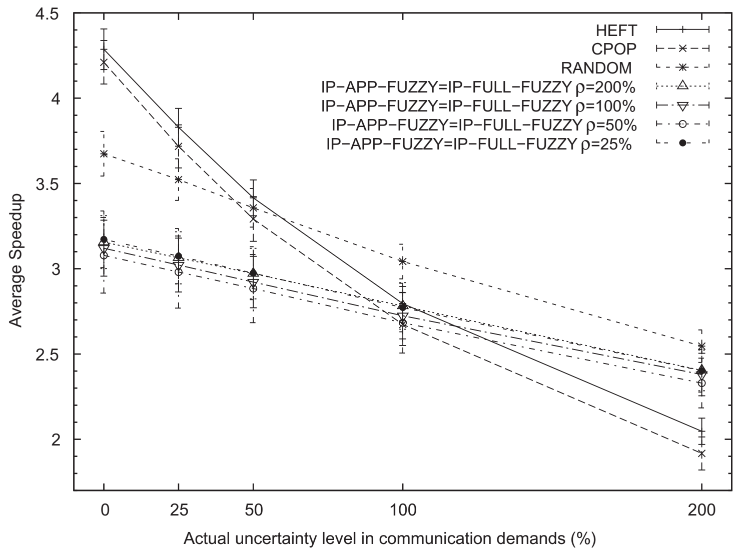


Fig. 11. Speedup produced by schedulers as a function of different actual uncertainty levels in communication demands (modified-WIEN2k DAG).

these schedulers gradually degrades as the degree of uncertainty increases, as can be seen by the decrease in speedup when values of uncertainty are equal to or higher than 100%.

The results shown in Fig. 10 reinforce those in Fig. 9. The IP-FULL-FUZZY scheduler produces higher speedup values than those given by RANDOM when $\rho = 200\%$, and their results are similar when when $\rho = 100\%$. Again, the speedup difference is about 10% when $\rho = 200\%$ and $x = 200\%$. The trend observed in Fig. 9 can also be observed in Fig. 10, showing the degradation in performance for deterministic schedulers when uncertainty values are equal to or higher than 100%.

Moreover, the rate of decay of speedup when RANDOM is considered is much sharper than that arising when fuzzy schedulers are employed for the modified-WIEN2K DAG (Fig. 11). This result is somehow dependent of the DAG topology. Uncertainties in the value of the weights of DAG edges have a great impact on speedup, since tasks along the longest path can be executed only after the execution of all previous tasks on which they depend. The use of HEFT and of CPOP schedulers result in speedup decreases for the modified-WIEN2k DAG due to its high degree of parallelism. Although the deterministic schedulers tend to produce schedules that yield higher network utilization, this does not happen when the IP-FULL-FUZZY is employed, since it is able to account for the existing uncertainties. The speedup of the IP-FULL-FUZZY can be up to 25% and 17% higher than those of the CPOP and HEFT schedulers, respectively.

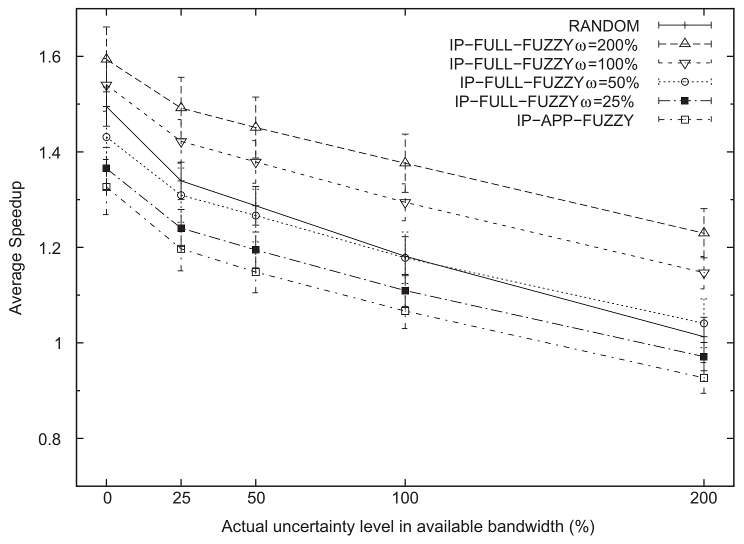


Fig. 12. Speedup produced by schedulers as a function of actual uncertainty level in the available bandwidth ($\rho = 50\%$ for IP-APP-FUZZY and IP-FULL-FUZZY) (Montage DAG).

6.2. Speedup with uncertainties in grid resources

Figs. 12–14 display, respectively, the speedup for the Montage, WIEN2k and modified-WIEN2k DAGs as a function of the actual uncertainty level in available bandwidth for different projected uncertainty levels the IP-FULL-FUZZY was designed for (ω). The projected uncertainty level in application demands (ρ) was fixed at 50%, since at this level IP-FULL-FUZZY produces bad results, as shown in Figs. 9–11.

From now on, for the sake of visual interpretation, results given by the deterministic schedulers CPOP and HEFT are omitted since the results given by the IP-FULL-FUZZY are at least 50% higher.

The poor performance of the IP-APP-FUZZY scheduler for the Montage DAG in Fig. 12 confirms the need for modeling the uncertainty of bandwidth availability. For the projected uncertainty level (ω) of 25%, the RANDOM scheduler over-performs the IP-FULL-FUZZY one due to the limited flexibility of the fuzzy scheduler. With a projected uncertainty level of 50%, the two provide roughly equal performance. When the projected uncertainty level increases to 100% or more, however, the IP-FULL-FUZZY scheduler produces higher speedup values than those produced by RANDOM. The

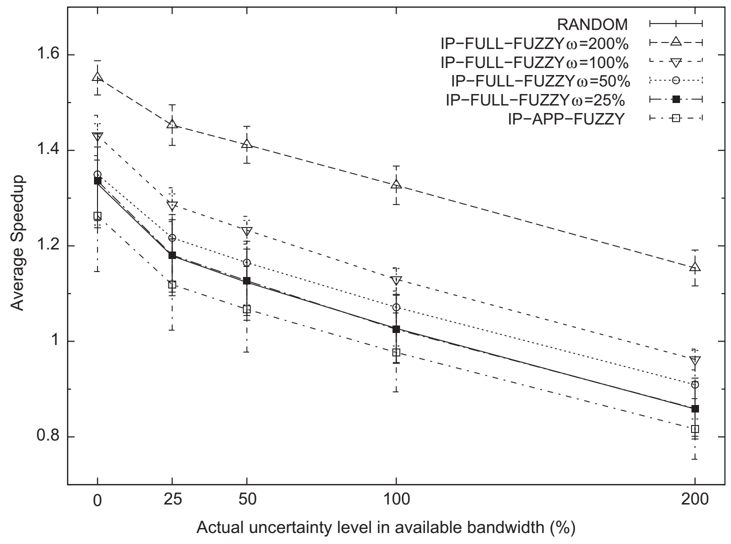


Fig. 13. Speedup produced by schedulers as a function of actual uncertainty level in the available bandwidth (ρ = 50% for IP-APP-FUZZY and IP-FULL-FUZZY) (WIEN2k DAG).

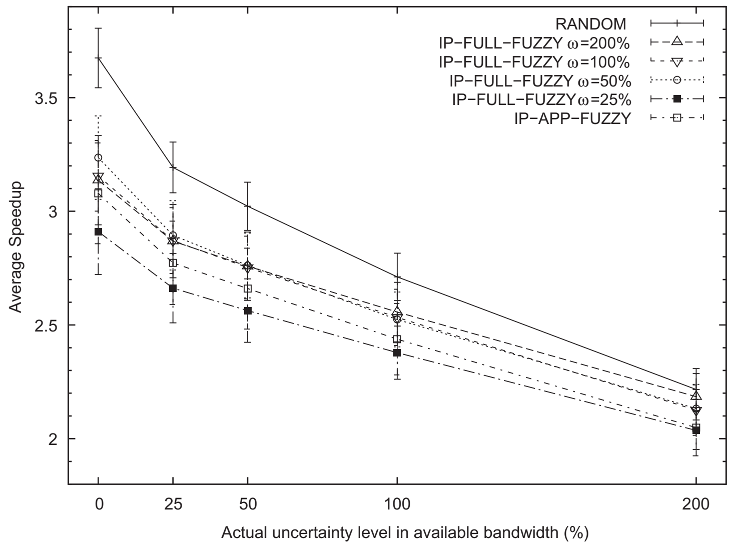


Fig. 14. Speedup produced by schedulers as a function of actual uncertainty level in the available bandwidth (ρ = 50% for IP-APP-FUZZY and IP-FULL-FUZZY) (modified-WIEN2k DAG).

speedup with the IP-FULL-FUZZY scheduler was 21% higher when $\omega = 200\%$ and $x = 200\%$. Besides that, the rate of decay of the speedup for RANDOM is sharper than that of the IP-FULL-FUZZY scheduler.

The results for the WIEN2k DAG (Fig. 13) were quite similar to those for the Montage DAG. With $\omega > 50\%$, the speedup of the IP-FULL-FUZZY scheduler can be up to 34% greater than that of RANDOM. Moreover, it can be up to 41% greater than that of IP-APP-FUZZY when $\omega = 200\%$ and $x = 200\%$.

The RANDOM scheduler which was not designed to deal with uncertainties produced for the modified-WIEN2k DAG speedup values which decay much faster than that given by the IP-FULL-FUZZY scheduler when the actual uncertainty level increases (Fig. 14). Differently than the results previously shown, the speedup of both schedulers are similar when $x = 100\%$.

6.3. Network utilization

The IP-FULL-FUZZY scheduler tends to assign dependent tasks to the same host, thus decreasing the degree of parallelism when the projected uncertainty level in the available bandwidth increases. This approach avoids unnecessary utilization of network links, as this could enlarge the makespan of the applications. Conversely, the IP-APP-FUZZY and the RANDOM schedulers consider the weights of the edges as deterministic values, which can lead to unnecessary allocation of network links, and an increase in network congestion, and, consequently, application makespan. The

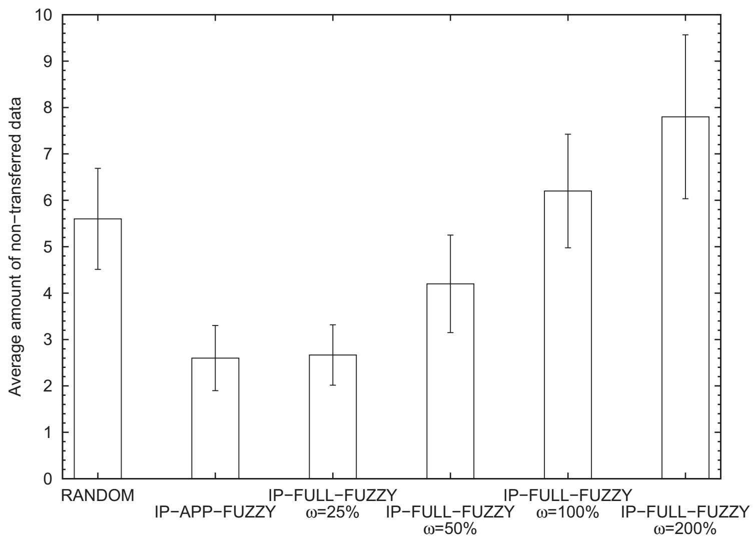


Fig. 15. Average amount of non-transferred data for the Montage DAG.

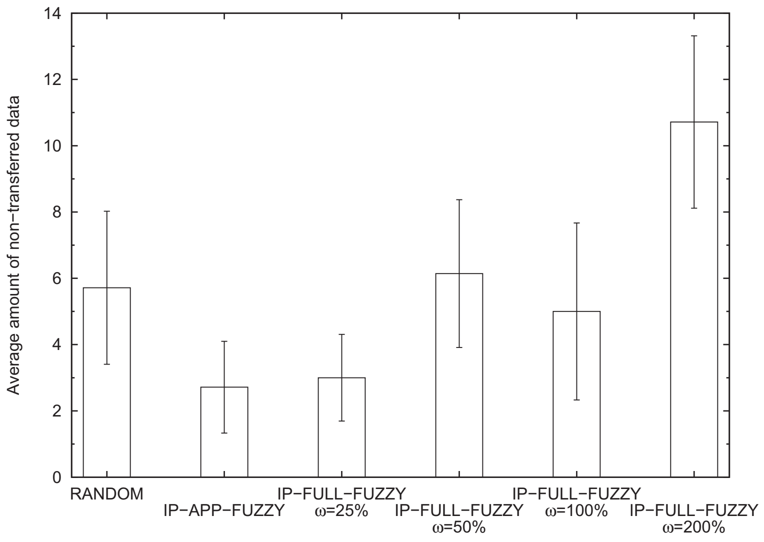


Fig. 16. Average amount of non-transferred data for the WIEN2k DAG.

impacts of this congestion affect not only a specific application, but others as well.

Fig. 15 shows the amount of non-transferred data due to the assignment of dependent tasks on the same host for the Montage DAG. A greater amount of non-transferred data implies lower network utilization. It can be seen in Fig. 15 that the amount of non-transferred data increases with the projected uncertainty level, which makes IP-FULL-FUZZY more robust in relation to misallocation of resources due to imprecise estimations of available bandwidth. In this specific example of the Montage DAG, data produced by ≅8 of the 39 task dependencies were not transferred via the network when the IP-FULL-FUZZY was designed with $\omega = 200\%$. In this case, the IP-FULL-FUZZY scheduler transferred half the data of the IP-APP-FUZZY and 40% of that of the RANDOM scheduler. The avoidance of this transfer led to an increase in speedup of 33% and 21% in comparison with IP-APP-FUZZY and RANDOM, respectively.

The decrease of network utilization with an increase in the value of ω value can also be seen in Figs. 16 and 17 for the WIEN2k and modified-WIEN2k DAGs, respectively.

6.4. Execution time

The execution time a scheduler takes to produce a schedule is quite relevant in dynamic environments; because a schedule can become sub-optimal since the grid is

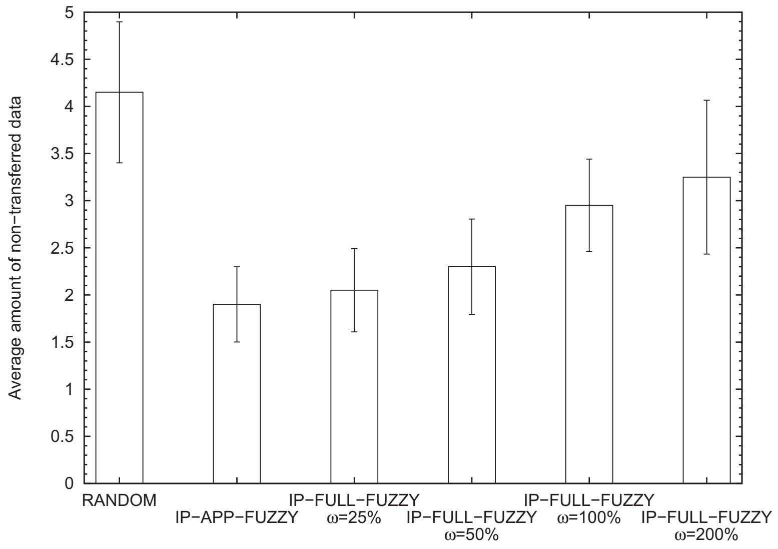


Fig. 17. Average amount of non-transferred data for the modified-WIEN2k DAG.

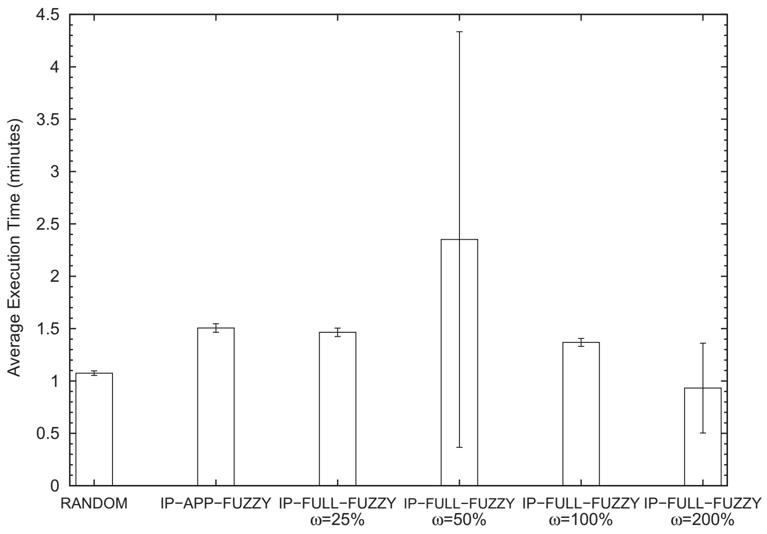


Fig. 18. Execution time of the schedulers considered ($\rho = 50\%$ for IP-APP-FUZZY and IP-FULL-FUZZY) for different projected uncertainty levels of the available bandwidth (Montage).

constantly changing during the period taken to produce the schedule. Figs. 18–20 compare the execution time of the schedulers involved in this study for the three DAGs considered.

As expected, RANDOM requires the least execution time, since the integrality constraints in the integer programming formulation have been relaxed. However, the IP-FULL-FUZZY scheduler with $\omega = 200\%$ is an average of 13.25% faster than RANDOM for the Montage DAG (Fig. 18) and 11% faster than the RANDOM scheduler for the WIEN2k DAG (Fig. 19). The longer average execution time when the projected uncertainty level is 50% for the Montage DAG is due to the fact that a single run required an execution time one order of magnitude longer than all other replications. The confidence interval for a 200% projected uncertainty level is wider than for the other projected uncertainty levels for the same reason.

Although the RANDOM scheduler requires a lower execution time for the modified-WIEN2k DAG than did the IP-FULL-FUZZY scheduler, the decay in execution time for the latter increases with the increase of the projected uncertainty level (Fig. 20).

The execution time of the IP-FULL-FUZZY scheduler was always less than that of the IP-APP-FUZZY scheduler when high projected uncertainty levels were involved ($\omega > 50\%$).

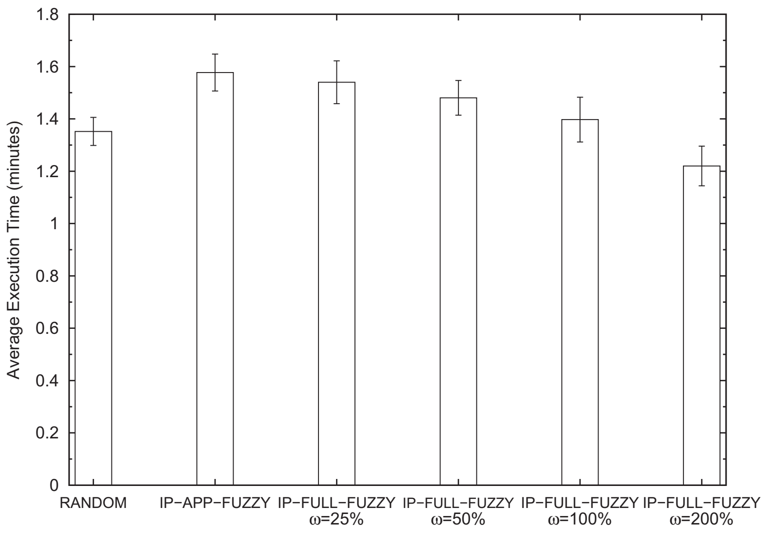


Fig. 19. Execution time of the schedulers considered ($\rho = 50\%$ for IP-APP-FUZZY and IP-FULL-FUZZY) for different projected uncertainty levels of the available bandwidth (WIEN2k).

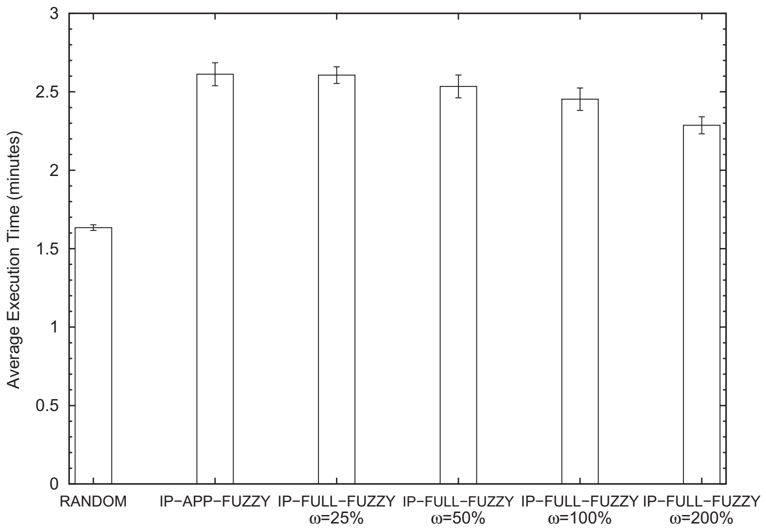


Fig. 20. Execution time of the schedulers considered ($\rho = 50\%$ for IP-APP-FUZZY and IP-FULL-FUZZY) for different projected uncertainty levels of the available bandwidth (modified-WIEN2k).

7. Conclusion and future work

In the present paper, the negative impact of uncertainties on deterministic schedulers was used to justify the use of a scheduler based on fuzzy optimization to schedule grid tasks involving uncertainty in both demands and resource availability. The results show that the speedup produced by the proposed IP-FULL-FUZZY scheduler is greater than that produced by a fuzzy scheduler which does not consider uncertainties of resource availability and by a deterministic (non-fuzzy) scheduler with improvements of 33% and 21%, respectively; moreover, the execution time required of the IP-FULL-FUZZY can be up to 38% and 13.25% less than those taken by the other schedulers. Furthermore, both network utilization and execution time of the IP-FULL-FUZZY scheduler decrease with an increase in the projected uncertainty level of input data.

The results have indicated that the effectiveness of the proposed approach relies on its ability to cope with a high level of uncertainty. As several grid applications, especially those of e-Science, generate huge amounts of data during their execution, the approach proposed in this paper seems to be quite attractive for future implementation in grid middlewares.

Although the scheduler introduced here represents a preventive approach for the handling of imprecise information, it was not designed to replace reactive approaches, such as self-adjusting scheduling. The integration of the two approaches seems to be a promising option. Currently, we are investigating the trade-offs between the solutions provided by fuzzy schedulers and those resulting from self-adapting schemes. Evaluation of the proposed scheduler when fed by data generated by different measurement tools is also under investigation.

Acknowledgements

The authors thank the anonymous referees and the Associate Editor for their valuable comments which greatly helped us to improve the paper. This work was financially supported by the Brazilian National Research Council – CNPq.

References

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

[2] D.M. Batista, N.L.S. da Fonseca, F.K. Miyazawa, A set of schedulers for grid networks, in: Proc. of ACM SAC'07, 2007, pp. 209–213, doi: <http://doi.acm.org/10.1145/1244002.1244057>.

[3] M. Jain, C. Dovrolis, End-to-end available bandwidth: measurement methodology, dynamics, and relation with TCP throughput, ACM SIGCOMM Comput. Commun. Rev. 32 (4) (2002) 295–308. doi: <http://doi.acm.org/10.1145/964725.633054>.

[4] D. Antoniades, M. Athanatos, A. Papadogiannakis, E.P. Markatos, C. Dovrolis, Available bandwidth measurements as simple as running wget, in: Proc. of Passive and Active Measurement Conference (PAM 2006), 2006, pp. 61–70.

[5] G. Allen, D. Angulo, I. Foster, G. Lanfermann, C. Liu, T. Radke, E. Seidel, J. Shalf, The cactus worm: experiments with dynamic resource discovery and allocation in a grid environment, Int. J. High Perform. Comput. Appl. 15 (4) (2001) 345–358.

[6] S.S. Vadhiyar, J.J. Dongarra, A performance oriented migration framework for the grid, in: Proc. of CCGRID'03, 2003, pp. 130–137.

[7] R. Sakellariou, H. Zhao, A low-cost rescheduling policy for efficient mapping of workflows on grid systems, Sci. Program. 12 (2004) 253–262.

[8] D.M. Batista, N.L.S. da Fonseca, F.K. Miyazawa, F. Granelli, Self-adjustment of resource allocation for grid applications, Comput. Networks 52 (9) (2008) 1762–1781. doi: <http://dx.doi.org/10.1016/j.comnet.2008.03.002>.

[9] D.M. Batista, A.C. Drummond, N.L.S. da Fonseca, Robust scheduler for grid networks, in: SAC '09: Proceedings of the 2009 ACM Symposium on Applied Computing, ACM, New York, NY, USA, 2009, pp. 35–39, doi: <http://doi.acm.org/10.1145/1529282.1529289>.

[10] D.M. Batista, A.C. Drummond, N.L.S. da Fonseca, Scheduling grid tasks under uncertain demands, in: Proc. of ACM SAC'08, 2008, pp. 2041–2045, doi: <http://doi.acm.org/10.1145/1363686.1364181>.

[11] C. Fayad, J.M. Garibaldi, D. Ouelhadj, Fuzzy grid scheduling using tabu search, in: Proc. of IEEE Int. Fuzzy Systems Conference, 2007, pp. 1–6.

[12] I. González-Rodrígues, C.R. Vela, J. Puente, A memetic approach to fuzzy job shop based on expectation model, in: Proc. of IEEE Int. Fuzzy Systems Conference, 2007, pp. 7–12.

[13] D.M. Batista, L.J. Chaves, N.L.S. da Fonseca, A. Ziviani, Performance analysis of available bandwidth estimation tools for grid networks, in: CAMAD '09: Proceedings of the IEEE 14th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks, 2009, pp. 1–5, doi:10.1109/CAMAD.2009.5161472.

[14] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, Introduction to Algorithms, second ed., The MIT Press, 2001.

[15] NASA, Applications of Montage, <http://montage.ipac.caltech.edu/applications.html> (retrieved 24.02.09).

[16] P. Blaha, K. Schwarz, G. Madsen, D. Kvasnicka, J. Luitz, WIEN2k, 2009, <http://www.wien2k.at/> (retrieved 24.02.09).

[17] D.A. Patterson, J.L. Hennessy, Computer Organization and Design, third ed., Elsevier, 2005. p. 242.

[18] B. Möller, W. Graf, W. Stransky, Fuzzy-optimization of structures, in: S.N. Atluri, S.J.N. Tadeu (Eds.), Proceedings of ICCES04, Tech Science Press, 2004, pp. 1765–1770.

[19] B. Möller, M. Beer, W. Graf, W. Stransky, Dynamic structural analysis considering fuzziness, in: M. Potier-Ferry, L.S. Toth (Eds.), Fourth Euromech Solid Mechanics Conference, Euromech, 2000, p. 616.

[20] T. Fahringer, R. Prodan, R. Duan, J. Hofer, F. Nadeem, F. Nerieri, S. Podlipnig, J. Qin, M. Siddiqui, H.-L. Truong, A. Villazon, M. Wieczorek, ASKALON: a development and grid computing environment for scientific workflows, Workflows for e-Science – Scientific Workflows for Grids, Springer-Verlag, 2007, pp. 450–471.

[21] A.S. McGough, W. Lee, J. Cohen, E. Katsiri, J. Darlington, ICENI, Workflows for e-Science – Scientific Workflows for Grids, Springer-Verlag, 2007, pp. 395–415.

[22] D. Bertsimas, J.N. Tsitsiklis, Introduction to Linear Optimization, Athena Scientific, 1997. p. 480.

[23] G. Juve, E. Deelman, Scientific workflows and clouds, ACM Crossroads 16 (3) (2010) 14–18.

[24] M. Mika, G. Waligóra, J. Weglarz, A Metaheuristic Approach to Scheduling Workflow Jobs on a Grid, in: Grid Resource Management – State of the Art and Future Trends, Kluwer Academic Publishers, 2003, pp. 295–318.

[25] H.-J. Zimmermann, Fuzzy Set Theory—And Its Applications, fourth ed., Kluwer Academic Publishers, 2001. pp. 11–13; 336–347.

[26] H. Casanova, D. Zagorodnov, F. Berman, A. Legrand, Heuristics for scheduling parameter sweep applications in grid environments, HCW '00: Proceedings of the Ninth Heterogeneous Computing Workshop, IEEE Computer Society, Washington, DC, USA, 2000, p. 349.

[27] K. Hwang, F.A. Briggs, Computer Architecture and Parallel Processing, McGraw-Hill, 1984.

[28] R. Fullér, On product-sum of triangular fuzzy numbers, Fuzzy Sets Syst. 41 (1) (1991) 83–87. doi: <http://dx.doi.org/10.1016/0165-0114(91)90158-M>.

[29] J. Blythe, S. Jain, E. Deelman, Y. Gil, K. Vahi, A. Mandal, K. Kennedy, Task scheduling strategies for workflow-based applications in grids, in: Proc. of CCGRID'05, vol. 2, 2005, pp. 759–767.

[30] H. Topcuouglu, S. Hariri, M. you Wu, Performance-effective and low-complexity task scheduling for heterogeneous computing, IEEE Trans. Parall. Distrib. Syst. 13 (3) (2002) 260–274. doi: <http://dx.doi.org/10.1109/71.993206>.

[31] H. Zhao, R. Sakellariou, Scheduling multiple DAGs onto heterogeneous systems, in: Proc. of the 20th International Parallel and Distributed Processing Symposium (IPDPS 2006), 2006, pp. 130–143.

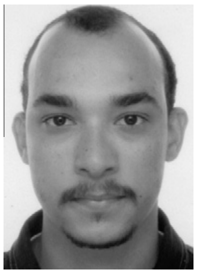
[32] M. Wieczorek, R. Prodan, T. Fahringer, Scheduling of scientific workflows in the ASKALON grid environment, SIGMOD Record 34 (3) (2005) 56–62. doi: <http://doi.acm.org/10.1145/1084805.1084816>.

[33] Z. Yu, W. Shi, An adaptive rescheduling strategy for grid workflow applications, in: IPDPS 2007: Proceedings of the IEEE International Parallel and Distributed Processing Symposium, 2007, pp. 1–8.

[34] J. Kim, J. Rho, J.-O. Lee, M.-C. Ko, CPOC: effective static task scheduling for grid computing, in: Proceedings of the First International Conference on High Performance Computing and Communications (HPCC 2005), 2005, pp. 477–486.

[35] Cooperative Association for Internet Data Analysis (CAIDA), CAIDA: Tools: Taxonomy, January 2009, <http://www.caida.org/tools/taxonomy/performance.xml#bw> (retrieved 06.09.09).

[36] M. Doar, I.M. Leslie, How bad is naive multicast routing? in: Proc. of IEEE INFOCOM'93, 1993, pp. 82–89.



Daniel M. Batista received a B.Sc. degree in Computer Science from Federal University of Bahia in 2004 and his M.Sc. degree in Computer Science from State University of Campinas in June 2006. He is now a Ph.D. candidate at the Institute of Computing, State University of Campinas, Campinas, Brazil and he is affiliated with the Computer Networks Laboratory at the same University. His research interests include traffic engineering and grid networks. His current research addresses robust mechanisms for grid networks.

Nelson L.S. da Fonseca received his Electrical Engineer (1984) and M.Sc. in Computer Science (1987) degrees from The Pontificial Catholic University at Rio de Janeiro, Brazil, and the M.Sc. (1993) and Ph.D. (1994) degrees in Computer Engineering from The University of Southern California, USA. Since 1995, he has been affiliated with the Institute of Computing of The State University of Campinas, Campinas – Brazil where is currently a Full Professor. He is also a Consulting Professor to the Department of Informatics and Telecommunications of the University of Trento, Italy. He is the Editor-in-Chief of the IEEE Communications Surveys and Tutorials. He served as Editor-in-Chief of the IEEE Communications Society Electronic Newsletter and Editor of the Global Communications Newsletter. He is a member of the editorial board of: Computer Networks, IEEE Communications Magazine, Peer-to-Peer Networking and Applications and International Journal of Communication Systems. He served on the editorial board of the IEEE Transactions on Multimedia, Brazilian Journal of Computer Science, and on the board of the Brazilian Journal on Telecommunications. He is the recipient of Elsevier Computer Networks Editor of the Year 2001, USC International Book award and of th eBrazilian Computing Society First Thesis and Dissertations award. He is an active member of the IEEE-Communications Society. He is currently ComSoc Director for Latin America. He served as ComSoc Director of On-line Services (2002–2003) and served as technical chair for several ComSoc symposia and workshops.