Scheduling Grid Tasks in Face of Uncertain Communication Demands

Daniel M. Batista *Nelson L. S. da Fonseca*

Technical Report - IC-12-06 - Relatório Técnico

January - 2012 - Janeiro

Scheduling Grid Tasks in Face of Uncertain Communication Demands

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

Abstract

Grid scheduling is essential to Quality of Service provisioning as well as to efficient management of grid resources. Grid scheduling usually considers the state of the grid resources as well application demands. However, such demands are generally unknown for highly demanding applications, since these often generate data which will be transferred during their execution. Without appropriate assessment of these demands, scheduling decisions can lead to poor performance. Thus, it is of paramount importance to consider uncertainties in the formulation of a grid scheduling problem. This paper introduces the IPDT-FUZZY scheduler, a scheduler which considers the demands of grid applications with such uncertainties. The scheduler uses fuzzy optimization, and both computational and communication demands are expressed as fuzzy numbers. Its performance was evaluated, and it was shown to be attractive when communication requirements are uncertain. Its efficacy is compared, via simulation, to that of a deterministic counterpart scheduler and the results reinforce its adequacy for dealing with the lack of accuracy in the estimation of communication demands.

1 Introduction

Resource management [1] is fundamental to efficient operation of grid networks as well as to Quality of Service provisioning of grid applications. To facilitate simultaneous (parallel) use of computational resources, grid applications are divided into smaller pieces of code, called tasks and, the scheduling of these tasks to resources is central to efficient service provisioning. Indeed, effective usage of grid resources is possible only when tasks are properly scheduled [2]. Essentially, scheduling is the decision-making process which matches

This paper was published in D. M. Batista, N. L. S. da Fonseca, *Scheduling Grid Tasks in Face of Uncertain Communication Demands*. IEEE Transactions on Network and Service Management, vol. 8, no. 2, pp. 92–103, 2011. DOI: `http://dx.doi.org/10.1109/TNSM.2011.050311.100060`

[*]Department of Computer Science, University of São Paulo, Rua do Matão, 1010 - Cidade Universitária – 05508-090, São Paulo – SP, Brazil (e-mail:batista@ime.usp.br)

[†]Institute of Computing, State University of Campinas, Avenida Albert Einstein, 1251 – 13084-971, Campinas – SP, Brazil (e-mail:nfonseca@ic.unicamp.br).

application demands to grid resources, and includes the specification of the specific time at which these resources should be used to satisfy these demands. Grid resources comprise the computational capacity of the hosts as well as network bandwidth.

The grid scheduling problem is an NP-hard problem [3], and either heuristics or approximations to obtain schedules in acceptable time frames are employed [4] [5] [6]. 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 as a function of other loads on the grid [7] [8], making the original schedule sub-optimal. In this paper, what is meant by uncertainty is the lack of assurance of a definite value. Uncertainties of schedule input values can result from many reasons such as fluctuation of resource availability, imprecise measurement tools and incapacity of estimating the true demand of applications which, in certain cases, are known only at execution time.

One approach for solutions to deal with such a dynamic environment is the implementation of a self-adaptive procedure for resource allocation [9] [10] [11] [6] [12]. Such a solution, however, implies a certain overhead, due to the need to monitor resources to detect changes in resource availability, as well as due to task migration. Moreover, it has not yet been proved that this type of solution leads to stable grids under situations of intense competition for resources. Furthermore, this approach considers only the uncertainties of resource availability, but not the uncertainties of application demands. As pointed out in [12] and [13], robustness to uncertainty in system and application information has been neglected and this can have a negative impact on the capability to provide services. Addressing such common issue in (grid) networks is certainly a significant advance in the management and operation of grids [14].

Uncertainties in relation to both computational and communication demands can jeopardize the optimality of schedules produced by deterministic grid schedulers. Furthermore, such uncertainties can also jeopardize the whole network operation [5] [6] as well as the profit made by service provider due to misclassification of the expected load [15]. Such uncertainties can also mislead the decision making process on the proper actions and mechanism to adopt in a dynamic environment [16].

Uncertainties in crucial information for grid operation arises from the difficulty in estimating application demands, especially those of communication (For example, in several e-Science applications, data are generated on-the-fly). Although uncertainties in application demands have been addressed in parallel systems [17] [18] [19], in grid networks, the existing techniques can only be partially utilized, since those systems are usually tightly coupled, and communication demands have little impact on the performance of applications. In grids, however, computers are connected by shared communication links, and the time required to transfer data across them impacts significantly the application makespan, i.e. the time taken to execute the application.

Both reactive [20] [21] [22] [5] [6] [23] and proactive [24] [25] [26] solutions have been proposed to deal with uncertainties in application demands. However, both types of solutions fail to incorporate uncertainties related to data transfer between dependent tasks, although data transfers can have a significant impact on the performance when executing demanding applications such as those of e-Science, which often transfer data in the order of

Petabytes [27] [6]. Therefore, it is of paramount importance to derive scheduling solutions to deal with these uncertainties. Moreover, some existing solutions [21] [28] for parallel processing assume that grids are homogeneous computing systems, which is a non-realistic assumption [7] and therefore can not be applied to grids context.

This paper introduces the IPDT-FUZZY scheduler, a scheduler based on fuzzy optimization for dealing with uncertainties in applications demands. The scheduler accepts as input a set of dependent tasks, described by Directed Acyclic Graphs (DAGs). The IPDT-FUZZY scheduler allows as input a single DAG, which is not a restrictive assumption since this DAG can represent the aggregation of several others [29]. In the proposed solution, fuzzy numbers represent application demands and fuzzy optimization techniques are employed to determine the schedule of tasks. Although previous work has adopted fuzzy theory to represent the uncertainty of grid application demands [21] [26], none of these papers has taken into account the uncertainty of communication demands, which is a unique contribution of the present work. The consideration of this uncertainty is of paramount importance for e-Science applications, where processing has only been possible since the emergence of grid network technology. To our knowledge, no scheduler based on fuzzy optimization with constraints given by fuzzy numbers has ever been proposed to cope with the uncertainties of estimating the amount of data transferred by application tasks. Actually, the lack of such capacity can lead to the degradation of performance when scheduling decisions are based on misleading estimations of communication demands. Moreover, the scheduler also considers uncertainties of processing demands.

The IPDT-FUZZY scheduler implements an integer linear program, which is the result of the mapping of the fuzzy formulation onto a crisp formulation. The integer program minimizes the makespan of the application represented by the DAG furnished as input. The scheduler also receives as input a graph representing the availability of computation and communication resources of the grid.

The makespans values furnished by our scheduler are compared, via simulation, with those produced by the scheduler based on classical optimization procedures proposed in [11]. Three different applications were simulated over several grid network configurations and significant speedup values were produced by the IPDT-FUZZY scheduler, when designed to operate under high levels of uncertainty. It was observed that the IPDT-FUZZY scheduler can produce speedups 30% greater than those of the classical scheduler. Moreover, the performance of the IPDT-FUZZY scheduler is similar to that of the classical one when the true degree of uncertainty is less than expected, in spite of the length of the dependence chain in the DAG. Furthermore, the time required by the IPDT-FUZZY scheduler to produce feasible schedules can be 85% less than that required by its counterpart classical scheduler, which is desirable in a dynamic environment where time for making decisions is crucial for the optimality of a schedule.

This paper extends and revises our preliminary work [30], in which we proposed a scheduler to deal with the uncertainties in the demands of specific applications. The formulation of the IPDT-FUZZY scheduler presented here has a reduced number of equations when compared to that proposed in [30]. Besides that, the preliminary evaluation of the performance of the scheduler in [30] has been extended by including new scenarios and applications. Results obtained in this extended evaluation present a stronger case for the effectiveness of the IPDT-FUZZY than does the work in [30]. Moreover, we present the transformation of the fuzzy optimization problem into an integer programming problem which is then implemented in the C language.

The rest of the paper is organized as follows. Section 2 reports previous work. Section 3 provides the motivation for the design of the IPDT-FUZZY scheduler. Section 4 introduces the IPDT-FUZZY scheduler and Section 5 evaluates its efficacy and compares it to that of a deterministic scheduler (IPDT). In this paper, we refer to schedulers based on optimization theory as deterministic schedulers, whereas those based on fuzzy optimization as fuzzy schedulers. Section 6 concludes the paper.

2 Related Work

The impact of uncertainty in application demands over two different scheduling approaches was evaluated in [31]. One considered only computational demands, whereas the other also included communication demands. Both of these approaches schedule DAGs of dependent tasks without full knowledge of them and are used in a reactive mechanism to react to changes. It was shown that the makespan of applications increases with degree of uncertainty. Errors up to 400% in bandwidth demand estimation occurred, which implied a seventeen-fold increase in execution time. Although the results are unquestionably evidence of the need for schedulers that use uncertain estimations, no scheduler to account for the uncertainty in application demands was proposed. Since there is no widely accepted benchmark available for grid applications, the numerical examples in the present paper will consider the same real scenario of the work in [31].

Previous work has investigated the issue of demand uncertainty in parallel systems. In [21], a proposal oriented to cluster and I/O bound applications was developed. That proposal classifies applications according to their demands as I/O bound, communication intensive and computation intensive. By using fuzzy logic theory and Bayesian estimators, this classification is done during the execution of the application, with a specific scheduler available for each type of application. A change in classification occurs once the application demands change. Although this approach considers communication costs, it is a reactive scheme, whereas the one proposed here is a proactive one. Our proposal does not require the overhead of monitoring and task re-scheduling necessary for the proposal in [21], which does not take into consideration the uncertainties in the demands of each type of application. For instance, if estimation of the number of bytes transferred in a communication intensive application is wrong, no action is taken to counteract the negative impact of this estimation. Moreover, the evaluation of the proposal in [21] was conducted using a homogeneous computing system, which is not realistic for grid environments.

The research reported in [24] surveys metrics of the performance of parallel applications in supercomputers. Probability distributions of performance metrics, such as makespan and host utilization, are used to generate synthetic loads for the prediction of performance. These predictions are quite useful for increasing performance, although the procedure cannot be used when executing the application for the first time, nor when the application is executed only once. Our proposal, however, can cope with uncertainties regardless the existence of the application execution history. Moreover, that proposal differs from the present proposal in the fact that it terminates the execution of applications if they exceed a specific predicted value. Furthermore, only independent tasks are considered.

Another approach, based on the impact of individual tasks on the makespan of an application, was adopted in [32]. This impact is estimated by considering the degree of uncertainty in application demands, as well by taking into consideration the relation between the tasks in the application DAG. Tasks with a greater impact are assigned to a host dedicated to their execution. However, no communication demands are considered, which limits the use of this approach in grid scheduling. The results reported in [32] are not very good for low degrees of uncertainty (only a maximum of 1.5% decrease in execution time was observed when the degree of uncertainty of processing demand was of 40%) and, in contrast to our proposal, the proposal in [32] was not evaluated when high degrees of uncertainty ($\geq$ 100%) were prevalent.

In [28], a module considering the available capacity of a grid was introduced. This module is based on "Support Infrastructure to Mobile Applications" middleware (ISAM). Measurements are made and used by Bayesian models to predict performance for real applications. Uncertainties are introduced as part of the performance predictions, although uncertainties in application demands are not considered, especially those related to the transfer of data between tasks. This is clearly different from our proposal, which does consider communication demands. As in the proposal in [21], it does not consider heterogeneous scenarios which limits its use for grid scheduling.

The research described in [25] does not deal with uncertainties in applications demands. It compares the predicted makespan with that which is measured to infer the discrepancies of makespan predictions. Experiments using the "Enabling Grids for E-sciencE" (EGEE) grid were conducted, although only CPU intensive applications in a specific cluster were considered, and not much about other scenarios can be inferred.

The scheduler proposed in [26] does not distinguish sources of misleading information, although as in this paper, triangular fuzzy numbers are considered. It uses tabu search and fuzzy sets to represent the uncertainties of the execution time of tasks. The lack of user knowledge about application demands is what motivates the use of fuzzy sets as in our proposal. Triangular fuzzy numbers are used to describe host utilization. Such shape is justified by the subjectiveness of any other potential shape, since resource utilization is very specific and depends on the specific application. Utilization is computed multiplying the application demand by the resource capacity. Since the latter is a crisp value, this utilization is expressed in fuzzy numbers. Different from our proposal, however, the approach in [26] is used for the admission control of applications to the grid system, i.e., it uses fuzzy numbers to classify requests for application processing, whereas our approach uses fuzzy numbers for scheduling application tasks.

A dynamic approach for dealing with uncertainties was introduced in [22]. Similar to the proposal in [21], tasks are monitored during execution so that rescheduling can be contemplated in an attempt to achieve the shortest execution time. The scheduler introduced in this paper, the Integer Programming with Discrete Time under Fuzzy Optimization (IPDT-FUZZY) scheduler, differs from that in [22], however, since the latter does not take into consideration uncertainties relating to the duration of data transfer. Moreover, evaluation of the scheduler in [22] included only a single degree of uncertainty, in contrast with that available in the present paper.

The self-adjusting procedure [11], previously introduced by the authors of the present paper, relies on an initial schedule produced by the deterministic scheduler Integer Programming with Discrete Time (IPDT). Motivated by results found in that previous paper, the scheduler introduced here (IPDT-FUZZY) extends the principles of resource allocation adopted for the IPDT and describes application demands using triangular fuzzy numbers. The performance of the schedules produced by the two schedulers is compared in this paper.

Table 1 summarizes the characteristics of related solutions. It specifies the contribution of previous work. More precisely, if the approach employed considered either reactive or proactive mechanisms, as well as whether it was originally designed to grids. Moreover, it details the type of mechanism used to cope with imprecise input values and if unknown application demands are considered by the mechanism. As can be observed, most of the systems are based on reactive mechanisms and no previous work is based on fuzzy optimization.

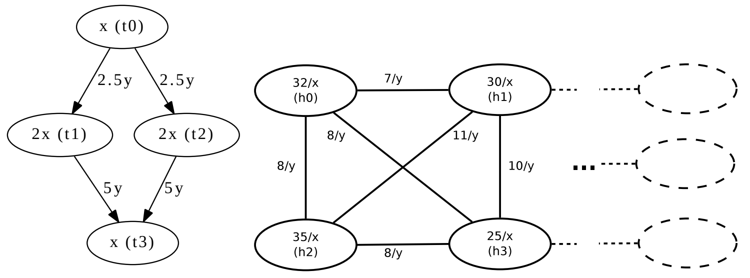
Table 1: Summary of related solutions.

Reference	Approach	Contribution	It deals with uncertainties by using	Is it grid-oriented?	Does it consider uncertainties of network demands?
[31]	Reactive	Scheduler	Rescheduling	Yes	No
[21]	Reactive	Scheduler / Classifier	Fuzzy logic / Bayesian estimator	No	No
[24]	N/A	Load Predictor	Statistical Models	No	No
[32]	Proactive	Scheduler	Heuristic (Prioritizes critical tasks)	Yes	No
[28]	N/A	Performance Predictor	Bayesian network	No	Yes
[25]	N/A	Performance Evaluation	N/A	Yes	No
[26]	Proactive	Scheduler / Admission Controller	Tabu search and fuzzy numbers	Yes	No
[22]	Reactive	Scheduler	Rescheduling	Yes	No
[11]	Reactive	Scheduler	Rescheduling	Yes	Yes

In summary, previous work has adopted a reactive approach which implies high overhead since it requires need of monitoring and task migration. Moreover, previous research ignores the heterogeneous nature of grids, as well as the fact that the network bandwidth is a resource to be allocated. The failure to consider these two issues makes the solutions inappropriate for adoption for real grids. The proposal introduced in this paper, however, is a proactive approach which avoids the huge overhead of monitoring and migration. Not only does it deal with the heterogeneity of grids, but it also considers the uncertainties of network demands. In our proposal, schedules are derived considering estimation uncertainties which are found in grid processing. The IPDT-FUZZY scheduler was designed to handle applications with dependent tasks which must transfer huge amounts of data, such as is typical in e-Science applications.

3 Illustrative Example of the Negative Impact of Uncertainties in Grid Scheduling

This section illustrates the negative impact of the absence of mechanisms to deal with uncertainty in descriptive values on the performance of grid applications. Figure 1(a) illustrates the DAG of an application. The weight (label) of the edge connecting two nodes indicates the number of bits to be exchanged by two dependent tasks, and the labels of the nodes represent the quantity of instructions to be processed. The order of precedence of tasks is given by the direct edges. The task id is represented by the label between parentheses. Figure 1(b) represents the grid resources on which the application described in Figure 1(a) is executed: nodes represent hosts, with labels expressing the inverse of host capacity (instructions/unit of time)$^{-1}$, and edges represent network links, with labels indicating the inverse of the available bandwidth (bits/unit of time)$^{-1}$. The host id is also represented by the labels in parentheses. The dotted lines denote the existence of other resources in the grid. To compute the time taken for data transfer, it is necessary to multiply the label on the respective edge of Figure 1(a) (number of bits to be transferred) by the label on the edge of Figure 1(b) in which the transfer will occur. For example, to compute the time taken to transfer data from task $t0$ to task $t1$, located respectively in the hosts $h0$ and $h2$, it is necessary to compute $2.5y \times \frac{8}{y} = 20$ units of time (u.t.). Similarly, to compute the processing time of a task, one needs to multiply the label on the respective node of Figure 1(a) (number of instructions to be processed) by the label on the node in Figure 1(b) corresponding to the host in which the task will be executed. For example, to compute the processing time of task $t0$ on host $h1$, it is necessary to compute $x \times \frac{30}{x} = 30$ u.t..



(a) DAG (vertices weights in instructions and arcs weights in bits).

(b) Grid hosts (vertices weights in (instructions/u.t.)$^{-1}$ and edges weights in (bits/u.t.)$^{-1}$).

Figure 1: An example to illustrate the negative impact of uncertainties on grid scheduling.

One approach to quantify such uncertainties is to employ the Quality of Information index (QoI) [20]. The QoI index expresses the level of confidence of a demand estimation. A value of 100%, the maximum QoI value, means that there is no doubt about the estimation, whereas values lower than 100% indicate that estimations are not precise. The lower the QoI value, greater the uncertainty of an estimation. The QoI is given by the following expression:

$$\begin{array}{rcl} \text{if } (tv > ev) & \rightarrow & \text{QoI} = 100\% \times [1 - \frac{(tv-ev)}{tv}] \\ \text{else} & \rightarrow & \text{QoI} = 100\% \times [1 - \frac{(ev-tv)}{ev}] \end{array}$$

where tv is the true value and ev is the estimated value.

For example, if the expected quantity of byte transfer is 1GB, although up to 2GB can be transferred when the application is actually executed, the QoI value is $100\% \times [1-(2-1)/2] = 50\%$.

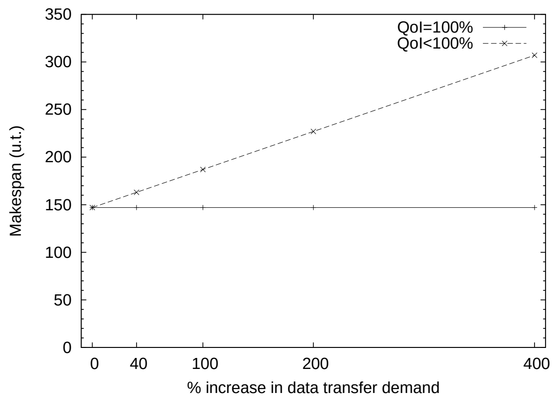


Figure 2: Makespan when QoI=100% and QoI<100%.

To evaluate the impact of uncertainties in the scheduling of the application shown in Figure 1(a), the deterministic scheduler IPDT was used to generate a schedule and the communication demands between tasks $t1$ and $t3$ were increased by up to 400%.

Figure 2 illustrates the makespan of the IPDT deterministic scheduler when considering exact estimates of the amount of data to be transferred from $t1$ to $t3$ (QoI equals 100%), and when that estimate was imprecise (QoI less than 100%).

Note that as the actual demand surpasses the estimated one, the makespan of the scheduler with QoI $<$ 100% also increases, being 100% higher for increases of 400% in demand. This shows that deterministic schedulers tend to produce poor schedules when demands are greater than those they anticipate. Moreover, it is not possible to feed schedulers with input with QoI=100%, since several applications such as those in e-Science generate data in real time. Thus, one way of counteracting this problem is to consider the Quality of Information of the estimated demand as input to the scheduler by using fuzzy values as estimates for these demands.

4 The IPDT-FUZZY Scheduler

The IPDT-FUZZY scheduler introduced in this paper was designed to produce optimal schedules when grid applications are faced with uncertainties in communication and computation demands. The numerical examples given in this paper, however, focus on communication demands only, since this has been so largely neglected in previous work. Moreover, we consider highly demanding applications such as those in e-Science.

The IPDT-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 [33]. This transformation is needed because only the crisp model can be implemented and solved by a computer program.

The use of linear programming in grid scheduling problem has given effective solutions [4] [5] [6]. However, heuristics has been employed to reduce the long execution times demanded by integer programming; an effective heuristic is to discretize the time line. Indeed, we consider that the IPDT-FUZZY scheduler defines a schedule on a discrete time line. 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.

The IPDT-FUZZY scheduler requires two graphs as input. The graph $H = (V_H, E_H)$ represents the grid topology, while the DAG $D = (V_D, A_D)$ gives the dependencies among the tasks. In H, V_H is the set of m ($m = |V_H|$) hosts connected by the set of links E_H, with the hosts labelled from 1 to m. In D, V_D is the set of n ($n = |V_D|$) tasks with integer numbers as labels which allows a topological ordering of tasks and A_D is the set of dependencies.

The IPDT-FUZZY scheduler considers that input DAGs have a single input task, as well as 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 zero processing time and communication weights. The output of the IPDT-FUZZY scheduler is 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 of the characteristics of the DAGs are as follows: I_i is the processing demand of the i^{th} task, expressed as number of instructions to be processed by the i^{th} task ($I_i \in \mathbb{R}_+$); $B_{i,j}$ is the number of bytes transmitted between the i^{th} task and the j^{th} task ($B_{i,j} \in \mathbb{R}_+$); $\mathcal{D}$ is the set of arcs $\{ij$ with $i < j$, 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 k^{th} host takes to execute a single instruction ($TI_k \in \mathbb{R}_+$); $TB_{k,l}$ is the time for transmitting a single bit along the link connecting the k^{th} host to the l^{th} host ($TB_{k,l} \in \mathbb{R}_+$); $\delta(k)$ is the set of hosts linked to the k^{th} host in the network, including the host k itself.

The IPDT-FUZZY scheduler deals with uncertainties by considering edge and vertex weights (I_i and $B_{i,j}$) as fuzzy numbers. As in [26], these fuzzy numbers are triangular. Since there is no benchmark available that would indicate a more specific type of fuzzy numbers, triangular fuzzy numbers are adopted, as is usual when no information is available to suggest the adoption of a more specific shape. In this way, the i^{th} task requires I_i instructions and is subject to an uncertainty of $\sigma\%$ of this value; the number of instructions is represented by $[\underline{I_i}, I_i, \overline{I_i}]$ where $\underline{I_i} = I_i(1 - \frac{\sigma}{100})$ and $\overline{I_i} = I_i(1 + \frac{\sigma}{100})$. Communication demands are treated in a similar manner, being represented by $[\underline{B_{i,j}}, B_{i,j}, \overline{B_{i,j}}]$, with a level of uncertainty of $\rho\%$. It is important to note that the values of σ and ρ are not random, but are rather set by the grid users to express their level of confidence in application demands.

For convenience, time is represented by $\mathcal{T}' = \{1, \ldots, T'_{max}\}$ where $T'_{max} = T_{max}(1 + \frac{\sigma}{100})$ and T_{max} is the maximum makespan of the application, i.e., the time it would take to execute all the tasks serially on the fastest host of the grid ($T_{max} = \min(\{TI_{k|k \in V_H}\}) \times \sum_{i=1}^{n} I_i$). Moreover, the minimum makespan is given by $T'_{min} = T_{min}(1 - \frac{\sigma}{100})$ where T_{min} is the time taken to execute all the tasks in the longest dependence chain on the fastest host

($T_{min} = \min(\{TI_{k|k\in V_H}\}) \times \sum_{i\in \text{ longest chain}} I_i$). T_{max} and T_{min} can be directly derived by the DAG furnished as input; there is no need to run the application serially on the fastest host. This is also true for T'_{max} and T'_{min}.

The IPDT-FUZZY scheduler solves an integer linear program to determine 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 i^{th} task is finished at time t on host k; otherwise this variable assumes a value of 0; f_i is a variable that records the time at which the execution of the i^{th} task is finished ($f_i = \sum_{t\in\mathcal{T}'}\sum_{k\in V_H} t \times x_{i,t,k}$).

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

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

subject to

$$\sum_{t\in\mathcal{T}'}\sum_{k\in V_H} x_{j,t,k} = 1 \qquad \text{for } j \in V_D; \qquad (\text{F}'1)$$

$$x_{j,t,k} = 0 \qquad \text{for } j \in V_D,\quad k \in V_H,\quad t \in \{1,\dots,\lceil \widetilde{I_j}\, TI_k \rceil\}; \qquad (\text{F}'2)$$

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

$$\sum_{j\in V_D} \sum_{s=t}^{\lceil t+\widetilde{I_j}\,TI_k-1\rceil} x_{j,s,k} \leq 1 \qquad \text{for } k \in V_H,\quad t \in \mathcal{T}',\quad t \leq \lceil T'_{max} - \widetilde{I_j}\,TI_k\rceil; \qquad (\text{F}'4)$$

$$x_{j,t,k} \in \{0,1\} \qquad \text{for } j \in V_D,\quad l \in V_H,\quad t \in \mathcal{T}'. \qquad (\text{F}'5)$$

The objective function involves the minimization of a fuzzy variable, as well as constraints (F'2), (F'3) and (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 [33]. In this case, the equivalent integer programming problem is given by:

$$\text{Maximize } \lambda$$

subject to

$$1 - \frac{f_n - T'_{min}}{T'_{max} - T'_{min}} \geq \lambda \tag{F1}$$

$$\sum_{t \in \mathcal{T}'} \sum_{k \in V_H} x_{j,t,k} = 1 \qquad \text{for } j \in V_D; \tag{F2}$$

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

$$\sum_{k \in \delta(l)} \sum_{s=1}^{\lceil t - \overline{I_j}\, TI_l - \overline{B_{i,j}}\, TB_{k,l} \rceil} x_{i,s,k} \geq \sum_{s=1}^{t} x_{j,s,l} \qquad \text{for } j \in V_D, \quad ij \in A_D, \quad \text{for } l \in V_H, \quad t \in \mathcal{T}'; \tag{F4}$$

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

$$x_{j,t,k} \in \{0,1\} \qquad \text{for } j \in V_D, \quad l \in V_H, \quad t \in \mathcal{T}'. \tag{F6}$$

The objective function of the integer programming version of the IPDT-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. The constraint (F1) establishes the relationship between λ and f_n, which is illustrated in Figure 3. The range of values for the makespan $f_n \in [T'_{min}, T'_{max}]$ is accounted for by the F1 restriction. The objective function jointly with constraint (F1) are equivalent to the objective function of the fuzzy optimization formulation.

Constraints (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), (F4) and (F5) establish inequalities using the fuzzy numbers I_i and $B_{i,j}$, which can vary in their allowed range. The constraints (F3), (F4) and (F5) in the integer programming formulation correspond, respectively, to the constraints (F'2), (F'3) and (F'4) in the fuzzy optimization formulation. Explanation on the transformation of these constraints is given next.

The constraints (F3) determine that a task (j) cannot terminate until all of its instructions have been completely executed. Since it is not possible to know the exact number of instructions, the minimum value of $\widetilde{I_j} \times TI_k$, given by $\underline{I_j} \times TI_k$, is used in (F3) to avoid

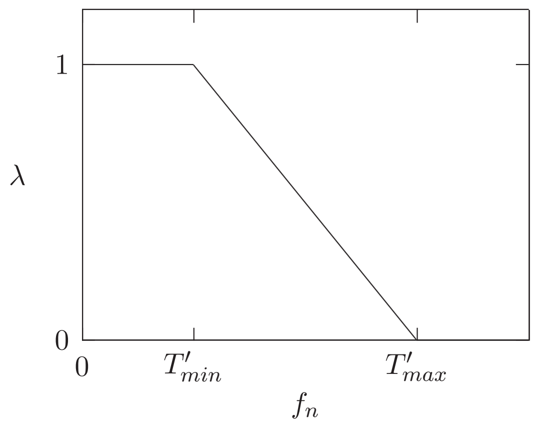


Figure 3: Satisfaction degree.

resource under-utilization.

The constraints in (F4) establish that execution of the j^{th} task cannot start until all its predecessors have been completed and the data required by the j^{th} task transferred. In order to prevent the potential execution of the j^{th} task prior to the execution of its predecessors due to existing uncertainties, the values of $\widetilde{I_j} \times TI_k$ and $\widetilde{B_{i,j}} \times TB_{k,l}$ are replaced by their maximum values, given by $\overline{I_j} \times TI_k$ and $\overline{B_{i,j}} \times TB_{k,l}$, respectively.

The constraints in (F5) establish that there is at most one task in execution on any one host at any given time. To maximize the number of tasks executed by a host, the lowest execution time for these tasks is established, with computational uncertainty $\widetilde{I_j} \times TI_k$ being replaced by $\underline{I_j} \times TI_k$.

The scheduler receives the input DAG and translates the demands into the constraint values of the optimization problem for which the solution is the set of $x_{i,t,k}$ values denoting on which host k the task i should start execution at time t.

Although the IPDT-FUZZY scheduler can handle uncertainties of both computational and communication demands, only communication demands will be the focus here since such analysis has been neglected in the literature. There is, however, a real demand driven by emerging e-Science applications. This demand can be in the order of Petabytes per year for each application [27] [6]; when dealing with this order of magnitude, deterministic decisions based on misleading information can lead to much longer execution times.

5 Performance Evaluation

In this section, the efficacy of the IPDT-FUZZY scheduler is assessed and compared via simulation to that of its deterministic counterpart (IPDT), since the aim is to evaluate the advantages of using fuzzy optimization in dealing with imprecise input values to the scheduler. Both the IPDT and the IPDT-FUZZY solve a linear programming problem and aim to minimize the application makespan. The variables used in the formulation of the two are similar, except that the latter introduces the uncertainty parameters ρ and σ.

Moreover, there is a one to one correspondence between the restrictions of the two schedulers ($\mathrm{D}_i \Leftrightarrow \mathrm{F}_{i+1}$). The formulation of the IPDT problem is the following:

$$\text{Minimize} \sum_{t \in \mathcal{T}} \sum_{k \in \mathcal{H}} t\, x_{n,t,k}$$

subject to

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

$$x_{j,t,k} = 0 \qquad \text{for } j \in V_D, \quad k \in V_H, \quad t \in \{1, \ldots, \lceil I_j\, TI_k \rceil\}; \qquad \text{(D2)}$$

$$\sum_{k \in \delta(l)} \sum_{s=1}^{\lceil t - I_j\, TI_l - B_{i,j}\, TB_{k,l} \rceil} x_{i,s,k} \geq \sum_{s=1}^{t} x_{j,s,l} \qquad \text{for } j \in V_D, \quad ij \in A_D, \quad \text{for } l \in V_H, \quad t \in \mathcal{T}; \qquad \text{(D3)}$$

$$\sum_{j \in V_D} \sum_{s=t}^{\lceil t + I_j\, TI_k - 1 \rceil} x_{j,s,k} \leq 1 \qquad \text{for } k \in V_H, \quad t \in \mathcal{T}, \quad t \leq \lceil T_{max} - I_j\, TI_k \rceil; \qquad \text{(D4)}$$

$$x_{j,t,k} \in \{0, 1\} \qquad \text{for } j \in V_D, \quad l \in V_H, \quad t \in \mathcal{T}. \qquad \text{(D5)}$$

where $\mathcal{T} = \{[1, \ldots, T_{max}\}$.

The IPDT scheduler is equivalent to the IPDT-FUZZY scheduler when the parameters ρ and σ are null. In this case, the fuzzy numbers of the IPDT-FUZZY formulation are crisp, as in the IPDT formulation. Consequently, all the curves representing the results of the IPDT scheduler in this paper can be interpreted as representing the results of the IPDT-FUZZY scheduler when $\rho = 0$ and $\sigma = 0$.

The input to this scheduler is composed of two graphs, one representing the tasks involved in the application, and the other the grid. These graphs represent a diverse set of task dependencies and topologies. Three DAGs for real applications were used to evaluate the schedulers. The first DAG was taken from an astronomy application called Montage (Figure 4), which is widely used in grid evaluation [31] [34] [6]. The weight of the DAGs are randomly chosen from a uniform distribution. The distribution for the weights of the edges has a mean of 72×10^6 bits, whereas the mean for the vertex weight is 31.5×10^{12} instructions. This DAG represents 26 tasks with 39 dependencies.

The second DAG corresponds to an application in quantum chemistry called WIEN2k [35] (Figure 5), developed at the Vienna University of Technology [36]. This DAG represents 26 tasks with 43 dependencies; weights are assigned randomly from a uniform distribution, as they were for the Montage DAG. The third DAG is a modified version of WIEN2k, but with 48 dependencies involving parallel, input and output tasks (Figure 6).

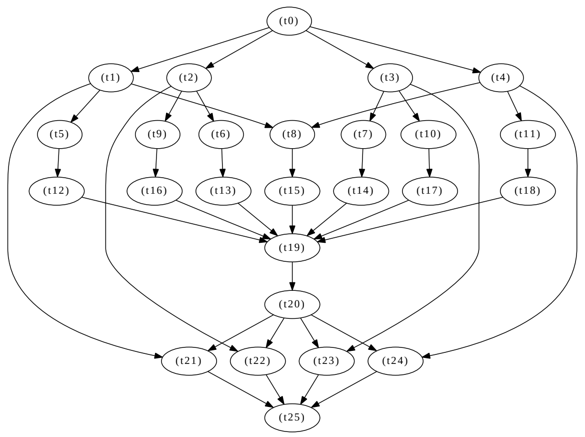


Figure 4: Montage DAG.

For the evaluation of the schedulers, twenty grids were generated employing the Doar-Leslie method [37], which is widely used for the generation of Internet topologies. The number of hosts was set at 50; the degree of node connectivity (β) was 0.98, and the ratio between the longest and the shortest links was 0.98. The mean weight of the hosts was $\frac{9}{5.25\times10^{12}}$ minutes/instruction (9726MIPS, which is equivalent to the performance of an Intel Pentium IV processor), and the mean weight of the links was $\frac{2}{12\times10^{9}}$ minutes/bit (100Mbps, the transmission rate of Fast Ethernet networks).

The degree of uncertainty adopted for application demands was $\rho \in [40\%, 100\%, 200\%, 400\%]$, which corresponds, respectively, to a QoI of $\{71.43\%, 50\%, 33.33\%, 20\%\}$ (derived by using the definition of QoI and the definition of ρ) These values were taken from previous studies in the literature [22] [31]. Although the characterization of typical levels of uncertainty that users have about the demand of their applications is beyond the scope of the present paper, this is certainly a promising area for future investigation. Moreover, results are presented only for uncertainties in relation to the amount of data to be transferred ($\sigma = 0$), as previously explained.

There is no standard benchmark available for grid applications. The heterogeneity of this emerging application and the recent emergence of the relevant technology have influenced this lack of information [38]. The DAGs used in this paper are DAGs representing the computation of real grid applications. To generate a variety of scenarios based on these DAGs, a range of weight values for the DAG edges was used. Such a range is not, however, related to the expected uncertainty level, ρ, to which the scheduler was designed. For a fixed value of ρ, a whole range of DAG edge weight values is considered for the analysis of the robustness of the scheduler when dealing with uncertainty levels different from the uncertainty level of the application for which they were designed for.

For each scheduler designed to operate with a specific uncertainty level, DAGs with different uncertainty levels were used as input. Twenty DAGs were used for each level of uncertainty. In this way, it was possible to evaluate how well a scheduler designed to operate for a specific uncertainty level handled different degrees of uncertainty. Each of

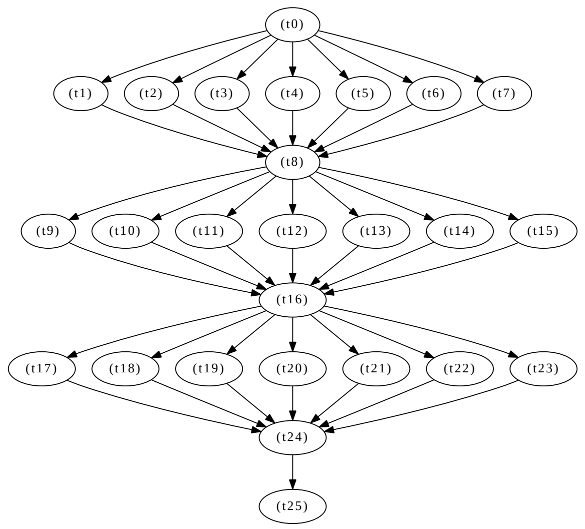


Figure 5: WIEN2k DAG.

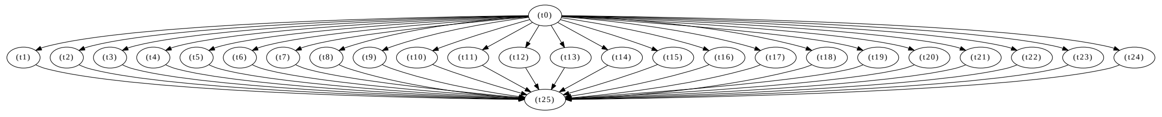


Figure 6: Modified WIEN2k DAG.

these twenty DAGs was generated by increasing the weight of the edges, with the increase based on uniform distributions from 0 to $x\%$, where $x \in \{40, 100, 200, 400\}$.

Various metrics were used for evaluation. The speedup, given by the ratio T_{max}/makespan, indicates how much the computation was speeded up in relation to sequential computation [39]. The higher the speedup, the more efficient is the schedule. The execution time provides the time required to produce a schedule, a metric which is quite important for the timely utilization of scheduling decisions in a changeable environment. The lower the execution time, the more efficiently a schedule can be used. Network utilization was also assessed, since unnecessary transfer enlarges the makespan and overloads network resources.

A confidence interval of 95% was adopted. Schedulers were written in the C programming language, and the `Xpress` version `2006A.1` optimization library was used. All simulations were executed in a Pentium IV, 3.2GHz personal computer with 2.5GB RAM memory and a Debian GNU/Linux (Lenny version) operating system.

Various characteristics can impact the performance of a scheduler. Our previous work [11] based on linear programming analyzes this impact, although it is beyond the scope of the present paper to do so. The present paper focus rather on the robustness of a scheduler for different levels of uncertainties in application demands.

5.1 Speedup

Figures 7, 8 and 9 show the speedup for, respectively, the Montage, WIEN2k and modified-WIEN2k DAGs as a function of the degree of uncertainty experienced. Each graph contains a curve corresponding to the results given by the IPDT scheduler, as well as four others corresponding to the results given by the IPDT-FUZZY scheduler for different degrees of uncertainty (ρ).

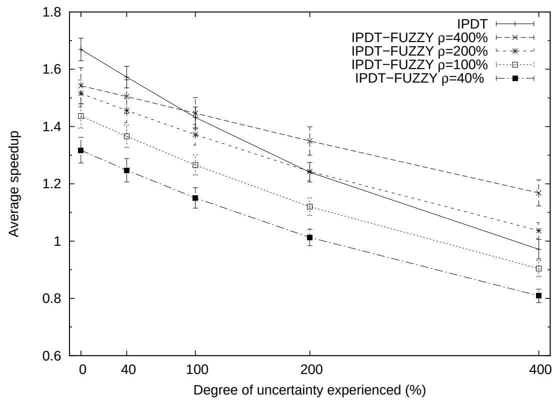


Figure 7: Average speedup for Montage DAG.

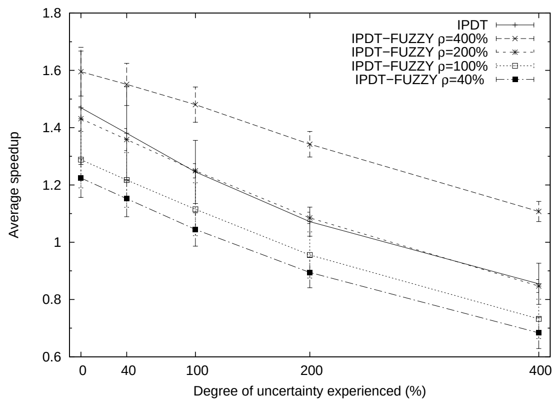


Figure 8: Average speedup for WIEN2k DAG.

It can be seen in Figure 7 that the IPDT-FUZZY scheduler produces higher speedup values than does the IPDT when the degree of uncertainty it was designed for was 400%, although the actual uncertainty was greater than 40%. When the uncertainty experienced was 400%, the speedup of the IPDT-FUZZY scheduler was $\cong$ 20.27% greater than that of the IPDT. The IPDT-FUZZY scheduler reveals speedup values greater than those furnished by the IPDT scheduler when the planned degree of uncertainty was 200% and actual uncertainty was higher than 100%. Moreover, the decay rate of speedup produced by the IPDT scheduler was twice as high as that of the IPDT-FUZZY when $\rho = 400\%$. This provides

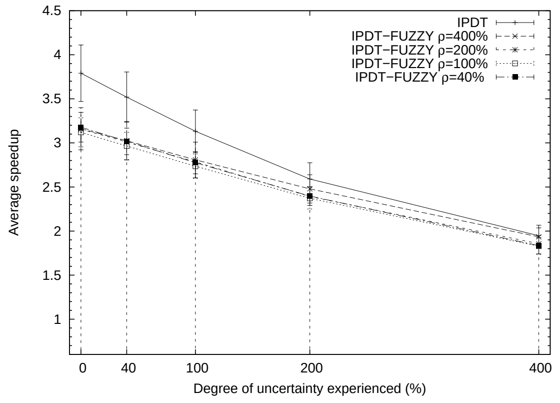


Figure 9: Average speedup for modified WIEN2k DAG.

clear evidence of the robustness of the IPDT-FUZZY scheduler, since its performance undergoes less degradation than does that of the IPDT scheduler as the degree of uncertainty increases. The performance of the IPDT scheduler is worse than that of the IPDT-FUZZY, even when the tasks were serially executed on the fastest host.

The same trends were observed when the WIEN2k DAG was employed. However, the speedup value of the IPDT-FUZZY scheduler was 29.55% greater than that of the IPDT scheduler for $\rho = 400\%$. For $\rho = 400\%$, the IPDT-FUZZY scheduler gives greater speedups than those of the IPDT. For $\rho = 200\%$, the speedups were equivalent.

The topology of the modified WIEN2k DAG consists basically of a set of tasks executed in parallel. This topology can negatively impact the speedup when there are discrepancies between the actual and the designed levels of uncertainty. Figure 9 illustrates this impact, showing that the IPDT-FUZZY scheduler produces a speedup comparable to that of the IPDT scheduler only when the uncertainty experienced is greater or equal to 200% and $\rho = 400\%$.

Figures 7, 8, and 9 evince that the IPDT scheduler achieves its best performance when there was no uncertainty. In some cases (Figure 7 and Figure 9), the average speedup of the IPDT scheduler was greater than that of the IPDT-FUZZY scheduler. Such results were expected, since these cases correspond to deterministic scenarios and the IPDT-FUZZY scheduler produced schedules considering uncertainties that indeed did not exist. However, were both values of ρ and σ null, the IPDT-FUZZY scheduler would have the same performance of that of the IPDT scheduler.

It has been concluded that the IPDT-FUZZY scheduler produces the fastest speedups when it is designed for a greater degree of uncertainty. It produces high speedup values even when the actual degree of uncertainty is less than that for which it was designed. When designed for a low degree of uncertainty, the limited flexibility provided for adjustment does not cope well with the actual uncertainty experienced.

The intersection points of the curves give the level of uncertainty above which the IPDT-FUZZY scheduler outperforms the IPDT. As expected, the advantages of the IPDT-FUZZY scheduler decrease as the degree of uncertainty experienced tends to zero (0%). This happens because the IPDT-FUZZY scheduler tries to decrease the degree of parallelism to avoid unnecessary transfer of data between two grid nodes for an expected level of uncertainty. When the degree of uncertainty decreases, the degree of parallelism also decreases, and the speedup values are lower than those given by the IPDT scheduler. However, when the degree of uncertainty increases, the low level of parallelism tends to decrease the time spent on data transfer, and as a consequence, the IPDT-FUZZY scheduler outperforms the IPDT.

5.2 Network Utilization

The IPDT-FUZZY scheduler tends to assign dependent tasks to the same host when the degree of uncertainty in communication demands increases, thus decreasing the degree of parallelism. This procedure avoids unnecessary utilization of network links, which would enlarge the makespan of the applications. Conversely, the IPDT scheduler considers the weights of the edges as deterministic values, leading to unnecessary allocation of network links and increasing congestion, with a consequent increase in the application makespan. This congestion impacts not only on the specific application, but also on other types of applications in the network.

Figure 10 shows the amount of data which is not transferred due to the assignment of dependent tasks to the same host for the Montage DAG. More non-transferred data implies lower network utilization. Figure 10 shows that the amount of non-transferred data increases with the degree of uncertainty, which makes the IPDT-FUZZY scheduler more robust in relation to misallocation of resources due to imprecise estimations of communication demands. In this specific example of the Montage DAG, the data produced by 11 of the 39 dependent tasks were not transferred via the network when the IPDT-FUZZY scheduler was designed to deal with $\rho = 400\%$. In this case the IPDT scheduler transferred 266% more data than did the IPDT-FUZZY, i.e., there was a 72.68% decrease in the amount of data transferred when the IPDT-FUZZY scheduler was used. This avoidance of transfer led to an increase in the speedup of the IPDT-FUZZY scheduler (20.27% greater; Figure 7).

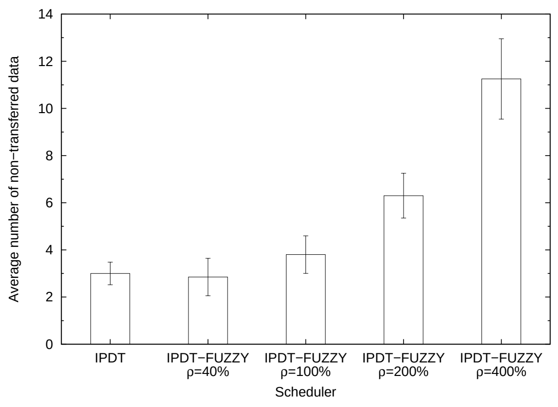


Figure 10: Average amount of non-transferred data for Montage DAG.

5.3 Execution time

The time a scheduler takes to produce a schedule, i.e., the execution time of the scheduler, is quite important in dynamic environments, since a prepared schedule may already be sub-optimal due to changes which have taken place in the grid during the time required to produce it.

Figures 11 to 13 show the execution time for the three DAGs considered. The execution time of the IPDT-FUZZY scheduler is considerably less than that required by the IPDT (approximately 85.08%, 95.67% and 94% less than that required for the Montage, WIEN2K and modified-WIEN2k DAGs, respectively).

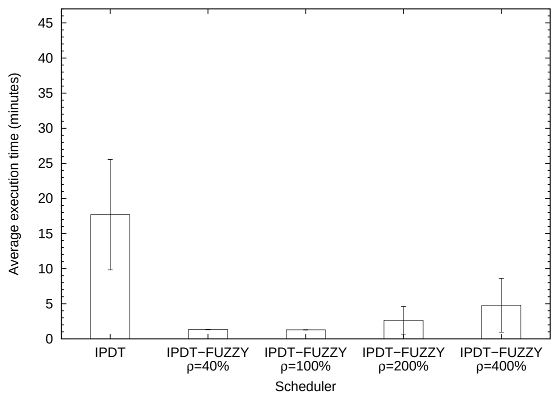


Figure 11: Average execution time for Montage DAG.

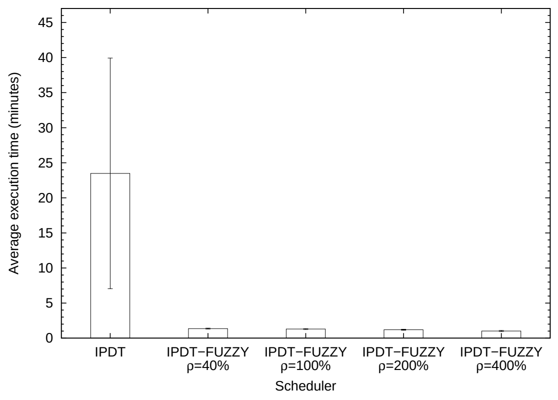


Figure 12: Average execution time for WIEN2k DAG.

One of the most widely used for solving linear programming (LP) problems is the simplex method. Although this method does not involve polynomial time complexity, in practice it is in fact very fast. The complexity of an algorithm is derived based on the worst possible computation time, rather than the average, although the present paper does not discuss such issues. However, even large LP problems with thousands of variables and constraints

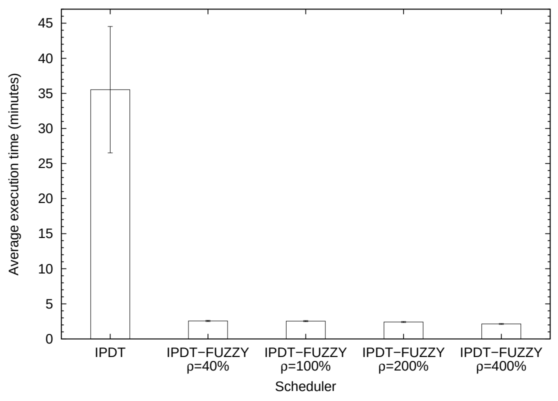


Figure 13: Average execution time for modified WIEN2k DAG.

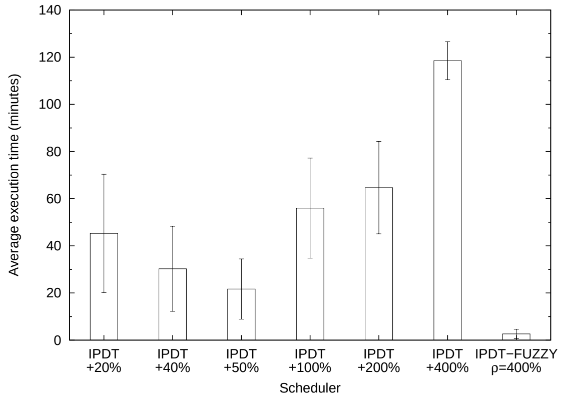


Figure 14: Average execution time of the IPDT (Inflated by the degree of uncertainty experienced) and of the IPDT-FUZZY ($\rho = 400\%$) for Montage DAG.

can be solved in seconds/minutes as can be verified in [40].

To further evaluate the efficiency of the IPDT-FUZZY scheduler, the execution time with $\rho = 400\%$ was compared to those produced by the IPDT scheduler when it was fed with a DAG with edge weights increased by specific percentages. Figure 14 shows that the slowest execution time of the IPDT scheduler, for an increase of 50%, was 21.68 minutes, while the execution time of the IPDT-FUZZY scheduler was 2.64 minutes (The execution time of the IPDT scheduler decreases when the increase in edge weight varies from 20 to 50%, since the computational demands are higher than are those of the communications. For increments greater than 50%, communication demands surpass computational ones.) When compared to the execution time of the IPDT scheduler with edge weights inflated by 400%, the execution time of the IPDT-FUZZY scheduler is 97.77% lower. This low execution time of the IPDT-FUZZY scheduler, however, does not jeopardize its efficacy, as can be seen in Figure 15, which compares the speedup of both schedulers.

The confidence intervals shown in Figures 11 to 14 indicate that the variability in execu-

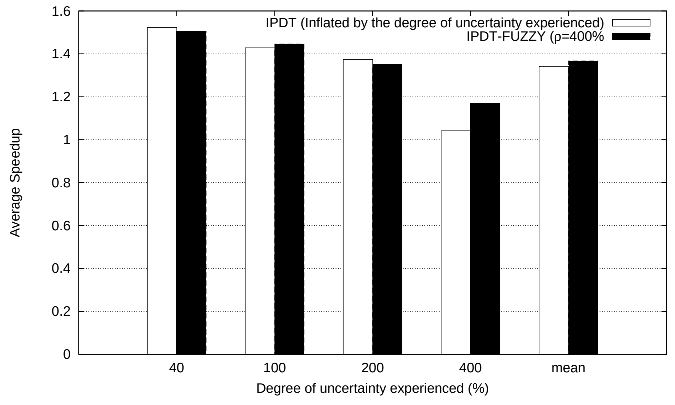


Figure 15: Average speedup given by IPDT (Inflated by the degree of uncertainty experienced) and IPDT-FUZZY ($\rho = 400\%$) for Montage DAG.

tion time of IPDT scheduler is higher than that for those of the IPDT-FUZZY scheduler for various values of ρ. For certain combinations of DAG and grid parameters, the execution times of the IPDT scheduler differed by one order of magnitude, which did not happen when using the IPDT-FUZZY scheduler.

In summary, in addition to greater speedup and a more efficient use of network links, the fuzzy scheduler runs faster than does the deterministic one.

6 Conclusions and Future Work

Inaccuracies in the estimation of demands can considerably enlarge the makespan of an application when tasks are scheduled by a deterministic scheduler. Moreover, accurate estimations are not only difficult to derive but may be impossible for some applications, especially those with real-time generated data.

Therefore, the consideration of information inaccuracy is essential for grid scheduling, which is, actually, of paramount importance for grid management. One of the approaches used to deal with this type of uncertainty is to use fuzzy schedulers for the scheduling of the application tasks. Although scheduling based on fuzzy schedulers has been used by previous investigations, none of these has considered the uncertainties in communication demands for applications.

This paper introduces the IPDT-FUZZY scheduler, a fuzzy scheduler which does consider communication uncertainties when deriving the application scheduling. Simulation experiments compare the effectiveness of the IPDT-FUZZY scheduler to that of a deterministic scheduler when communication demands are uncertain. The results, based on DAGs from real grid applications, have shown that the fuzzy scheduler can produce speedups 29.55% greater than those produced by the deterministic scheduler, which is quite significant when one considers the typical makespan length of e-Science applications. Moreover, the schedules derived by the deterministic scheduler can require the transfer up to 266% more data than do those derived by the fuzzy scheduler. Furthermore, the time required for the derivation of schedules by the fuzzy scheduler is 85% less than that required by the deterministic one.

Although the IPDT-FUZZY scheduler was compared to the IPDT scheduler when the latter had full knowledge of the application demands, the speedup produced by the former was similar to that given by the latter, yet it can have execution times 97.77% lower. This and other results show the advantages of using fuzzy numbers to represent uncertainties in application demands as well as in using fuzzy optimization for producing schedules. The efficiency of the IPDT-FUZZY scheduler was demonstrated using the two most relevant metrics in heterogeneous systems: the speedup and the execution time of the scheduler.

The evidence of the advantages of the unique approach introduced in this paper leads us to conclude that the insertion of the IPDT-FUZZY scheduler in grid middlewares can yield enhanced performance, since in the operation of real grids, users usually have limited knowledge about their application requirements, although they have stringent QoS requirements.

One interesting investigation that could be pursued is the integration of fuzzy schedulers with procedures for self-adjustment of resource allocation. Such an integration might lead to even more robust schedules, since the schedules provided by the IPDT-FUZZY scheduler require decreased monitoring and migration overhead, while self-adjusting procedures could compensate for eventual imprecise scheduling decisions.

The objective of the experiments in this paper was to evaluate the gains of the fuzzy optimization to deal with uncertainties in grid scheduling, so it was sufficient to compare the performance of the proposed fuzzy scheduler with its deterministic counterpart, the IPDT scheduler. An interesting future work is to compare the performance of the IPDT-FUZZY scheduler with that of other schedulers proposed in the literature, like the HEFT and CPOP schedulers [41].

References

[1] Martin Ansbjerg Kjær, Maria Kihl, and Anders Robertsson. Resource Allocation and Disturbance Rejection in Web Servers using SLAs and Virtualized Servers. *IEEE Transactions on Network and Service Management*, 6(4):226–239, Dec 2009.

[2] Jennifer M. Schopf. Ten Actions when Grid Scheduling – The User as a Grid Scheduler. In Jarek Nabrzyski and Jennifer M. Schopf and Jan Weglarz, editor, *Grid Resource Management: State of the Art and Future Trends*, chapter 2, pages 15–23. KAP, 2003.

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

[4] D. M. Batista, N. L. S. da Fonseca, and F. K. Miyazawa. A Set of Schedulers for Grid Networks. In *SAC '07: Proceedings of the 2007 ACM Symposium on Applied Computing*, pages 209–213, New York, NY, USA, 2007. ACM. DOI: `http://dx.doi.org/10.1145/1244002.1244057` .

[5] Pawel Garbacki and Vijay K. Naik. Efficient Resource Virtualization and Sharing Strategies for Heterogeneous Grid Environments. In *Proceedings of the 10th IFIP/IEEE IM*, pages 40–49. IEEE, May 2007.

[6] Gargi Dasgupta, Koustuv Dasgupta, and Balaji Viswanathan. Data-WISE: Efficient Management of Data-Intensive Workflows in Scheduled Grid Environments. In *Proceedings of the IEEE/IFIP NOMS 2008*, pages 488–495. IEEE, Apr 2008.

[7] I. Foster. What is the Grid? A Three Point Checklist. *GRIDToday*, 1(6), Jul 2002. `http://www-fp.mcs.anl.gov/~foster/Articles/WhatIsTheGrid.pdf` . Retrieved Sep 5, 2009.

[8] D. B. Skillicorn. Motivating Computational Grids. In *2nd IEEE/ACM International Symposium on Cluster Computing and the Grid(CCGRID'02)*, pages 401–406, May 2002.

[9] Sathish S. Vadhiyar and Jack J. Dongarra. A Performance Oriented Migration Framework for the Grid. In *3rd IEEE/ACM International Symposium on Cluster Computing and the Grid(CCGRID'03)*, pages 130–137, 2003.

[10] Xian-He Sun and Ming Wu. GHS: A Performance System of Grid Computing. In *19th IEEE International Parallel and Distributed Processing Symposium*, 2005. `http://doi.ieeecomputersociety.org/10.1109/IPDPS.2005.234` . Retrieved Sep 5, 2009.

[11] 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. DOI: `http://dx.doi.org/10.1016/j.comnet.2008.03.002` .

[12] D. M. Batista and N. L. S. da Fonseca. Self-Adjustment for Service Provisioning in Grids. In Nick Antonopoulos and Georgios Exarchakos and Maozhen Li and Antonio Liotta, editor, *Handbook of Research on P2P and Grid Systems for Service-Oriented Computing: Models, Methodologies and Applications*, volume 1, chapter 21, pages 495–518. IGI Global, 2010. `http://www.igi-global.com/chapter/handbook-research-p2p-grid-systems/40815` . Retrieved Dec 29, 2011.

[13] D. M. Batista and N. L. S. da Fonseca. A Survey of Self-Adaptive Grids. *IEEE Communications Magazine*, 48(7):94–100, 2010. DOI: `http://dx.doi.org/10.1109/MCOM.2010.5496884` .

[14] D. M. Batista and N. L. S. da Fonseca. Robust Scheduler for Grid Networks under Uncertainties of Both Application Demands and Resource Availability. *Computer Networks*, 55(1):3–19, 2011. DOI: `http://dx.doi.org/10.1016/j.comnet.2010.07.009` .

[15] Gabi Dreo Rodosek, Matthias Göhner, Mario Golling, and Michael Kretzschmar. Towards an Accounting System for Multi-Provider Grid Environments. In *Proceedings of the 10th IFIP/IEEE IM*, pages 60–69. IEEE, May 2007.

[16] C. Cárdenas, M. Gagnaire, V. López, and J. Aracil. Performance Evaluation of the Flow-Aware Networking (FAN) Architecture under Grid Environment. In *Proceedings of the IEEE/IFIP NOMS 2008*, pages 481–487. IEEE, Apr 2008.

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

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

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

[20] Henri Casanova, Dmitrii Zagorodnov, Francine Berman, and Arnaud Legrand. Heuristics for Scheduling Parameter Sweep Applications in Grid Environments. In *HCW '00: Proceedings of the 9th Heterogeneous Computing Workshop*, page 349, Washington, DC, USA, 2000. IEEE Computer Society.

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

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

[23] D. M. Batista, N. L. S. da Fonseca, F. Granelli, and D. Kliazovich. Self-Adjusting Grid Networks. In *ICC '07: Proceedings of the 2007 IEEE International Conference on Communications*, pages 344–349. IEEE, 2007. DOI: `http://dx.doi.org/10.1109/ICC.2007.64` .

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

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

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

[27] Worldwide LHC Computing Grid, 2009. `http://lcg.web.cern.ch/LCG/` . Retrieved Sep 5, 2009.

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

[29] Henan Zhao and Rizos Sakellariou. Scheduling Multiple DAGs onto Heterogeneous Systems. In *Proceedings of the 20h International Parallel and Distributed Processing Symposium (IPDPS 2006)*, pages 130–143, Apr 2006.

[30] Daniel M. Batista, André C. Drummond, and Nelson L. S. da Fonseca. Scheduling Grid Tasks under Uncertain Demands. In *Proceedings of the ACM SAC '08*, pages 2041–2045. ACM, 2008. DOI: `http://doi.acm.org/10.1145/1363686.1364181` .

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

[32] Ladislau Bölöni and Dan C. Marinescu. Robust Scheduling of Metaprograms. *Journal of Scheduling*, 5:395–412, 2002.

[33] H.-J. Zimmermann. *Fuzzy Set Theory—and its Applications*, pages 11–13;336–347. Kluwer Academic Publishers, 4th edition, 2001.

[34] NASA. Applications of Montage. `http://montage.ipac.caltech.edu/applications.html` . Retrieved Sep 5, 2009.

[35] P. Blaha, K. Schwarz, G. Madsen, D. Kvasnicka, and J. Luitz. WIEN 2k, 2009. `http://www.wien2k.at/` . Retrieved Sep 5, 2009.

[36] Marek Wieczorek, Radu Prodan, and Thomas Fahringer. Scheduling of scientific workflows in the ASKALON grid environment. *SIGMOD Rec.*, 34(3):56–62, 2005.

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

[38] Marios D. Dikaiakos. Grid Benchmarking: Vision, Challenges, and Current Status. *Concurrency and Computations: Practice and Experience*, 19(1):89–105, 2007.

[39] Kai Hwang and Fayé A. Briggs. *Computer Architecture and Parallel Processing*. McGraw-Hill, 1984.

[40] Hans Mittelmann. Decision Tree for Optimization Software, Aug 2009. `http://plato.asu.edu/bench.html` . Retrieved Sep 5, 2009.

[41] Haluk Topcuouglu, Salim Hariri, and Min you Wu. Performance-Effective and Low-Complexity Task Scheduling for Heterogeneous Computing. *IEEE Trans. Parallel Distrib. Syst.*, 13(3):260–274, 2002.