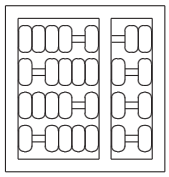

INSTITUTO DE COMPUTAÇÃO
UNIVERSIDADE ESTADUAL DE CAMPINAS

**A Set of Schedulers for Grid Networks**

*Daniel M. Batista* *Nelson L. S. da Fonseca*
*Flavio K. Miyazawa*

Technical Report - IC-12-03 - Relatório Técnico

January - 2012 - Janeiro

# A Set of Schedulers for Grid Networks

Daniel M. Batista* Nelson L. S. da Fonseca† Flavio K. Miyazawa

**Abstract**

Central to grid processing is the scheduling of application tasks to resources. Schedulers need to consider heterogeneous computational and communication resources, producing the shortest possible schedule under time constraints dictated by both the application needs and the frequency of fluctuation of resource availability. This paper introduces a set of schedulers with such characteristics.

## 1 Introduction

Grid Networks (Grids) involve coordinated resource sharing and problem solving in heterogeneous dynamic environments [1] [2]. Central to grid processing is the allocation of tasks to resources (task scheduling). In heterogeneous systems, the scheduling problem is a NP-complete problem [3]. Feasible solutions in real time to this problem require either heuristics or approximations [3] [4] [5]. Usually, such solutions need to be found previous to a given deadline which depends on the characteristics of the application and of the grid considered [6] [7]. One approach to obtain the best possible schedule within a time frame is to consider the schedules produced by different schedulers. These schedulers would have different computational complexity and would produce schedules with diverse quality. In general, the quality of a schedule depends on the time demanded to produce it. The schedulers could run in parallel and the desired schedule chosen among those fully produced within a given time period.

This paper introduces a set of eight schedulers offering solutions which differ in terms of their schedule length as well as computational complexity. These schedulers can be executed in parallel to obtain a schedule with the highest possible quality within a time period. One of the main characteristics of these schedulers is the consideration of network resource availability which is not widely considered in the literature [8] [9]. Moreover, the



*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).

performance of the proposed schedulers are evaluated considering various network topologies and applications composed by dependent tasks.

Although various scheduling schemes have been proposed for grids [10] [11] [12] [13] [14], most of them do not take into account heterogeneous resources, with the exception of [13] which focus on tasks with cyclical data dependencies. Some authors [12] propose schedulers to work under time constraints but no comparison with schedulers that take data dependencies into account has been carried out. The schedulers presented in this paper differ from the existing ones by both considering heterogeneous environments and working under time constraints.

This paper is organized as following. Section 2 introduces a set of novel schedulers. Experiments are shown in Section 3 and Section 4 provides some conclusions.

## 2 Proposed Set of Schedulers

Schedulers attempt to achieve scheduling objectives taking into consideration a description of the application and of the available grid resources. Applications are described by tasks and their dependencies are represented by directed acyclic graphs (DAGs). Tasks in these DAGs are represented by vertices and arcs represent dependencies between tasks. Vertex weights indicate the number of instructions that need to be executed by a task whereas arc weights the amount of bits to be transferred between two tasks. Grid resources are also represented by graphs. In these graphs, vertices represent hosts and their weights the time taken to execute 1 instruction, whereas edges represent full-duplex links and their weights indicate the time needed to transfer 1 bit.

The aim of all the schedulers proposed is the minimization of schedule length for grid applications under the following restrictions: i) the execution of a task should begin only after the completion of all the other tasks which the task depends on, as well as only after the reception of all data sent by these tasks. ii) each task can be mapped to only one host. iii) two dependent tasks can only be mapped to hosts which have a connecting link (each host is assumed to have a virtual link to itself with zero cost associated with that link). iv) each host can execute only a single task at anyone time.

The schedules produced by six of the eight schedulers proposed are derived from the solution of mixed/integer linear programming (LP) problems. Three of these schedulers consider time to be a continuous variable ($\in \mathbb{R}_+$) whereas the other three treat it as a discrete variable ($\in \mathbb{Z}_+$). The choice involves a certain trade-off between execution time and the schedule length. 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. The exact solution for a mixed/integer programming problem for both continuous and discrete time are derived and the other four schedulers are derived by employing two different relaxation techniques to the exact LP problems.

The schedulers which consider time as a continuous variable are formulated as a mixed linear programming problem (Subsection 2.1), whereas those that consider time as a discrete variable are formulated as integer linear programming problem (Subsection 2.2). In these

problems, variables $X_{i,k}$ define the mapping of tasks to hosts; $X_{i,k}$ is 1 if the $i^{th}$ task is mapped to $k^{th}$ host; otherwise, it is 0.

Although solving exact linear programming problems with integrality constraints leads to optimal or quasi-optimal solutions, it may take a very long time. An alternative is the obtainment of partial fractional solutions by considering relaxation of integrality constraints, with the option of conversion of these solutions to integer ones. In this case, the variables $(X_{i,k})$ are defined in the interval $[0, 1]$. Techniques for the relaxation of integrality constraints adopt randomized rounding techniques, in which the value of the variable $X_{i,k}$ is the probability of the $i^{th}$ task being mapped to the $k^{th}$ host. Two different randomized rounding techniques were adopted to define two different algorithms. The first technique (Algorithm 1) solves a linear programming problem once, with the value of the variables used as probabilities for a series of drawings, each defining a different schedule; the one yielding the shortest schedule is selected as the solution. In the second technique (Algorithm 2), an iterative randomized rounding procedure is adopted. In each step of this procedure, an LP is solved, and the task with the highest probability values is definitely mapped to a host. The procedure terminates when no more tasks are left to be mapped to a host.

The other two schedulers are based on random drawing. The schedule is the one of those produced during a series of drawings that minimizes the schedule length. The first step of each iteration of these algorithms is the assignment of an initial value to the variables $X_{i,k}$. The actual starting values constitute the only difference between the two algorithms. In one, it is based on a probability that is uniformly distributed among the hosts (Subsection 2.3), whereas in the other, it somehow translates the characteristics of tasks and hosts, and will be denominated "grid aware" (Subsection 2.4). In both algorithms, the dependency constraints shown in the DAG, the network topology and the resource capacity are observed. Moreover, these algorithms produce different schedule lengths itself as well as for their own execution time. The one using "grid aware" initial values tends to run for longer periods, but produces shorter schedule length.

Hosts are labelled from 1 to $m$, while tasks are identified by labels from 1 to $n$. Tasks are processed according to the topological order of the input DAG, each with a single input task and a single output one. DAGs failing to satisfy this condition because they have more than one input or output task can be easily modified by considering two null tasks with zero processing time and communication weight [10]. Some characteristics of the DAGs are: i) $n$: number of tasks ($n \in \mathbb{N}$); ii) $I_i$: processing demand of the $i^{th}$ task, expressed as number of instructions to be processed by the task $i$ ($I_i \in \mathbb{R}_+$); iii) $B_{i,j}$: number of bits transmitted between the $i^{th}$ task and the $j^{th}$ task ($B_{i,j} \in \mathbb{R}_+$); iv) $\mathcal{D}$: set of arcs $\{ij : i < j$ and there exists an arc from vertex $i$ to vertex $j$ in the DAG$\}$. v) $s_0$: starting time of the input task. For all examples in this paper, $s_0 = 0$.

Moreover, grid resources composed of hosts and links have the following characteristics: i) $m$: number of existing hosts ($m \in \mathbb{N}$); ii) $TI_k$: time the $k^{th}$ host takes to execute instructions ($TI_k \in \mathbb{R}_+$); iii) $TB_{k,l}$: time for transmitting a bit on the link connecting the $k^{th}$ host and the $l^{th}$ host ($TB_{k,l} \in \mathbb{R}_+$); iv) $\mathcal{N}$: set $\{kl :$ host $k$ is linked to host $l\}$; v) $\delta(k)$: set of hosts linked to the $k^{th}$ host in the network.

Moreover, $T_{\max}$ is the time that the application would take to execute serially all the

tasks in the fastest host, i.e., $T_{\max} = \min TI \sum_{i=1}^{n} I_i$, where $\min TI$ is the lowest value of $TI$. $\mathcal{J} = \{1, \ldots, n\}$ is the set of existing tasks of an application and $\mathcal{H} = \{1, \ldots, m\}$ is the set of hosts.

## 2.1 LP Formulation with time as a continuous variable

This approach adopts a mixed linear programming formulation for the grid scheduling problem:

$$\text{Minimize} \quad (I_n \sum_{k=1}^{m} TI_k X_{n,k}) + s_n$$

$$\text{such that}$$

$$s_i \geq s_0 \qquad \text{for } i \in \mathcal{J}; \qquad \text{(C1)}$$

$$s_j \geq s_i + \sum_{k \in \mathcal{H}} [(I_i TI_k X_{i,k}) + \sum_{l \in \delta(k)} (B_{i,j} TB_{k,l} VA_{i,k,j,l})] \qquad \text{for } i, j \in \mathcal{J}, \; ij \in \mathcal{D}; \qquad \text{(C2)}$$

$$s_j \geq s_i + \sum_{k \in \mathcal{H}} (I_i TI_k VA_{i,k,j,k}) - y(1 - P_{i,j}) \qquad \text{for } i, j \in \mathcal{J}, \; i \neq j, \quad ij \notin \mathcal{D}, \; ji \notin \mathcal{D}; \qquad \text{(C3)}$$

$$s_i \geq s_j + \sum_{k \in \mathcal{H}} (I_j TI_k VA_{j,k,i,k}) - yP_{i,j} \qquad \text{for } i, j \in \mathcal{J}, \; i \neq j, \quad ij \notin \mathcal{D}, \; ji \notin \mathcal{D}; \qquad \text{(C4)}$$

$$\sum_{k \in \mathcal{H}} X_{i,k} = 1 \qquad \text{for } i \in \mathcal{J}; \qquad \text{(C5)}$$

$$\sum_{k \in \mathcal{H}} \sum_{l \in \delta(k)} VA_{i,k,j,l} = 1 \qquad \text{for } j \in \mathcal{J}; \qquad \text{(C6)}$$

$$2VA_{i,k,j,l} \leq X_{i,k} + X_{j,l} \qquad \text{for } ij \in \mathcal{D}, \; kl \in \mathcal{N}; \qquad \text{(C7)}$$

$$VA_{i,k,j,l} - X_{i,k} - X_{j,l} \geq -1 \qquad \text{for } ij \in \mathcal{D}, \; kl \in \mathcal{N}; \qquad \text{(C8)}$$

$$2VA_{i,k,j,k} \leq X_{i,k} + X_{j,k} \qquad \text{for } i, j \in \mathcal{J}, \quad i \neq j, \quad ij \notin \mathcal{D}, \quad ji \notin \mathcal{D}; \qquad \text{(C9)}$$

$$VA_{i,k,j,k} - X_{i,k} - X_{j,k} \geq -1 \qquad \text{for } i, j \in \mathcal{J}, \quad i \neq j, \quad ij \notin \mathcal{D}, \quad ji \notin \mathcal{D}; \qquad \text{(C10)}$$

$$VA_{i,k,j,l}, X_{i,k}, P_{i,j} \in \{0, 1\} \qquad \text{for } i, j \in \mathcal{J}, \quad k, l \in \mathcal{H}. \qquad \text{(C11)}$$

The relaxation of the above problem consists of replacing $\{0, 1\}$ in the constraints (C11) by the interval $[0, 1]$.

The constraints in (C1) state that all tasks must start after time $s_0$. The constraints in (C2) specify that a task will start only after all tasks dependent on it have been completed and the relevant data transferred. Constraints (C3) and (C4) state that if two independent tasks are scheduled to the same host, one of them will be fully executed before the start of the other. The binary variable $P_{i,j}$ has value 1 if the $i^{th}$ task is executed first (in which case constraint (C4) is satisfied) and 0 if the $j^{th}$ task is executed first (constraint (C3) is satisfied). The constant $y$ is a large positive number (e.g., $T_{max}$). Constraint (C5) states

that the tasks must be scheduled to some host ($k$). Constraint (C6) specifies that there should be a single tuple $(i, k, j, l)$ such that the $i^{th}$ and $j^{th}$ tasks are scheduled to the $k^{th}$ and to the $l^{th}$ hosts, respectively.

Constraints (C7), (C8), (C9) and (C10) determine that $VA_{i,k,j,l}$ is 1 if and only if $X_{i,k} + X_{j,l}$ is 2. The value of these two variables indicates that tasks with a dependency relationship should be mapped to interconnected hosts.

The final scheduling is established by the value of the following variables: i) $X_{i,k}$, which has the value 1 if the $i^{th}$ task is mapped to the $k^{th}$ host; otherwise it is 0 ($X_{i,k} \in \{0, 1\}$); ii) $s_i$, which sets the starting time of the $i^{th}$ task ($s_i \in \mathbb{R}_+$).

This formulation must account for a maximum of $(m^2 + 2m + 3)n^2 - (m^2 + 2m + 1)n$ constraints, and $(m^2 + 1)n^2 + (m + 1)n$ variables. The scheduler based on the exact solution of this problem involving mixed linear programming with a continuous time variable is denominated MLPCT. There are two versions of MLPCT, one involving the relaxation technique based on randomized rounding (CT-RR) and the other using the relaxation technique based on iterative randomized rounding (CT-IRR).

## 2.2 LP formulation with time as a discrete variable

This formulation considers discrete intervals of time and treats the scheduling problem as an integer linear programming problem, and is formulated as follows:

$$\text{Minimize} \quad f_n$$

$$\text{such that}$$

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

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

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

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

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

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

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

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

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

where $T_{max}$ is the longest possible finish time for a task. For convenience, the following notations is used: $\mathcal{T} = \{1, \ldots, T_{max}\}$ and $X_{i,k}$ is defined as $\sum_{t=1}^{T_{max}} x_{i,k,t}$.

The schedule is established by the value of the following variables: i) $x_{i,t,k}$: binary variable that assumes a value of 1 if the $i^{th}$ task finished at time $t$ in the host $k$; otherwise this variable assumes a value of 0; ii) $f_i$: variable that stores the time at which the execution of the $i^{th}$ task is finished ($f_i \in \mathbb{N}^*$).

The relaxation of the discrete time formulation consists of changing the set $\{0, 1\}$ of the constraints in (D9) to the interval $[0, 1]$.

The constraints in (D1) specify that a task must be executed at one time in a single host. The constraints in (D2) establish the finishing time for all the tasks, and the constraints in (D3) determine that a task ($j$) cannot terminate until it has been executed in the host $k$. The constraints in (D4) establish that if the $i^{th}$ task executes in the $l^{th}$ host before the $j^{th}$ task does, and that the $j^{th}$ task is finished at time $t$, then the time when the $i^{th}$ task finished its execution is at most $t$ minus the execution time of the $j^{th}$ task minus the time needed to transfer data between these two tasks. The constraints in (D5) establish that there is at most one task in execution at any one host at a specific time, while the constraints in (D6) guarantee that a task be scheduled to a single host at any one time, while the constraints in (D7) and (D8) determine that dependent tasks are mapped to hosts interconnected.

The accuracy of the results obtained by using this formulation depends on the interval width used in the discretization of the timeline. The wider the interval is, the faster the execution; but, the lower the accuracy.

This formulation involves a maximum of $(n^2 + n + 1)mT_{max} + (2n - 2)m + 3n - 1$ constraints and $(mT_{max} + 1)n + mn$ variables. The scheduler based on an exact solution of the integer linear programming with a discrete time variable is denominated as ILPDT. Again, two versions of schedulers with relaxation are presented, one involving the relaxation technique with randomized rounding (DT-RR) and the other using the relaxation technique with iterative randomized rounding (DT-IRR).

## 2.3 Random Draw with uniform probabilities

The seventh scheduler is based on a algorithm involving random probabilities of task scheduling to hosts. It uses an uniform probability distribution to map tasks to hosts. All the tasks have probability $1/m$ to be executed in each host.

This scheduler chooses as solution the shortest schedule among $P$ random selections. At each step of a selection, the scheduler maps a task to a host and changes the assignment probability of those tasks not mapped yet, considering the dependency relationships established in the task DAG, the network topology and resources capacity. This scheduler is denoted as RDU.

### 2.4 Drawing Using Distribution involving Grid-aware Probability values

This scheduler differs from the one in the previous subsection by the probability values used for the mapping of tasks to hosts. The following rules are considered to derive the probability values: i) the probability that a task will be executed in a given host is proportional to the processing rate of all available hosts; ii) the probability of execution of a task by a given host is proportional to the number of links connecting it to other hosts, as well as to their available bandwidth; iii) the lower the level of a task in a DAG, the higher the probability that the task will be mapped to a host with a high available processing rate; iv) the larger the number of arcs in a DAG, the higher the probability that a given task will be mapped to a host with large number of links connecting it to other hosts; v) the greater the amount of data a task needs to transfer, the higher the probability that the task will be mapped to a host with high capacity links; vi) the larger the number of instructions involved in a task, the higher the probability that the task will be mapped to a host with a large available processing rate.

The first two rules define the initial probability of mapping the $i^{th}$ task to the $k^{th}$ host, given by:

$$\begin{aligned} X_{i,k} &= \left(\frac{\frac{1}{TI_k}}{\sum_{j=0}^{m}\frac{1}{TI_j}} \times \frac{1}{3}\right) + \left(\frac{|\delta(k)| - 1}{\sum_{j=1}^{m}|\delta(j)| - m} \times \frac{1}{3}\right) \\ &+ \left(\frac{\sum_{l\in\delta(k)-\{k\}}\frac{1}{TB_{k,l}}}{\sum_{j=1}^{m}\sum_{l\in\delta(j)-\{j\}}\frac{1}{TB_{j,l}}} \times \frac{1}{3}\right) \end{aligned} \quad (1)$$

If the criteria used were limited to grid resources, hosts with greater availability of processing rates and bandwidths would be utilized all the time, whereas hosts with less capacity would be idle. To avoid such an unbalance which leads to unsatisfactory results, the characteristics of tasks also need to be considered, as done in list-based schedulers [10]. Consequently, the probability value in Equation 1 is redefined to each task considering the last four rules defined above. This scheduler is denoted DG and it also chooses the shortest schedule among $P$ random selections.

## 3 Comparison of Scheduler Efficiency

Various network topologies and tasks DAGs were used to compare schedulers proposed here. Results of the experiments involving the DAG shown in Figure 1 are representative of those obtained in other experiments and will be presented in this section. The criteria used for comparison are the speedup (the ratio between the time to a serial execution of the tasks in the processor with the greatest available processing rate and the time for task execution

using a specific schedule) and the execution time required to produce that schedule. A workstation equipped with a Pentium 4, 3.2 GHz CPU with 2GB RAM was used in the experiments. The software *xpress* was employed to solve the linear programming problems.

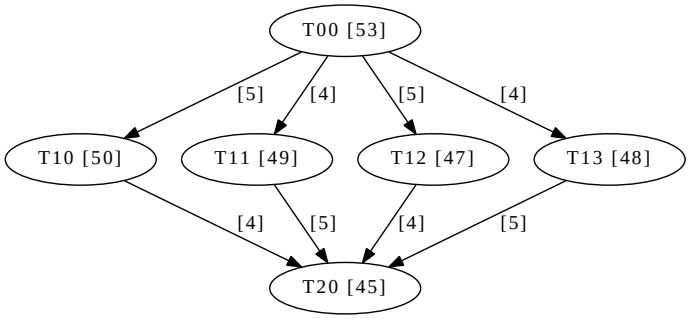


Figure 1: Tasks DAG used in the experiments.

Various topologies were generated using the Doar-Leslie method [15] by changing the number of hosts, the network connectivity (node degree) and the ratio between the longest and the shortest edge. If not stated otherwise, the network used has 50 nodes ($N$), network degree ($\beta$) 0.5 and ratio between longest and shortest edge ($\alpha$) 0.9. The processing rate of the hosts follows a uniform probability distribution function in the interval $(0.4, 2]$. The capacity of the network links varied in the interval (0, 5], according to the Doar-Leslie method. The weights of the DAG in Figure 1 were in the interval $[4, 5]$, whereas the weight of the vertices varied in the interval $[45, 53]$. Furthermore, the number of random selections ($P$) is 10,000.

Table 1 and Table 2 present the performance of the proposed schedulers as a function of the number of hosts. The performance of MLPCT is not shown since it requires much longer execution times when compared to the other schedulers. For a 40 host network, for example, MLPCT took over one hour to generate a schedule, whereas ILPDT took 12.3 seconds. The schedule producing the largest speedup for each number of hosts is written in bold. The ratio between other speedup values and the largest one ($100\% * (speedup/largest\_speedup)$) is shown as percentage in the table. ILPDT produced the largest speedup for most of the experiments, followed by CT-RR. Schedulers based on the relaxation in Algorithm 2 (CT-IRR and DT-IRR) produced the smallest speedup among the schedulers based on linear programming. This poor performance can be explained by the single random selection of the mapping probabilities in Algorithm 2. For schedulers based on random drawing DG provides better schedules than does RDU since the initial probability values of the former consider both grid and task constraints.

The execution time of schedulers based on linear programming, also portrayed in Table 2, increases as the number of hosts increases with the largest speedups achieved by schedulers which took longer to generate the schedule. Clearly, this does not happens with schedulers based on random drawing. The use of algorithms based on integrality relaxation led to lower execution times than did their exact counterparts. For a 190-host network, ILPDT took 134.03 seconds while DT-IRR took 32.05 seconds. Moreover, the execution time of

| $N$ | Speedup | | | | | | |
|---|---|---|---|---|---|---|---|
| | CT-RR | CT-IRR | DT-RR | DT-IRR | ILPDT | RDU | DG |
| 10 | 77.72% | 77.72% | 99.31% | 77.55% | 98.51% | 99.07% | **1.29** |
| 40 | 89.21% | 73.36% | 99.86% | 73.26% | **1.36** | 81.79% | 85.10% |
| 70 | **1.56** | 64.34% | 91.51% | 82.48% | 99.38% | 77.25% | 84.52% |
| 100 | **1.53** | 65.55% | 97.73% | 65.17% | 94.18% | 70.73% | 79.10% |
| 130 | 94.38% | 66.71% | 91.73% | 66.23% | **1.51** | 69.67% | 75.53% |
| 160 | 93.43% | 62.23% | 91.33% | 62.11% | **1.61** | 68.26% | 73.84% |
| 190 | 62.49% | 62.49% | 81.82% | 62.27% | **1.61** | 65.29% | 74.71% |

Table 1: Speedup of various schedules as a function of the number of hosts

| $N$ | Execution time (seconds) | | | | | | |
|---|---|---|---|---|---|---|---|
| | CT-RR | CT-IRR | DT-RR | DT-IRR | ILPDT | RDU | DG |
| 10 | 0.28 | 0.05 | 0.52 | 0.21 | 0.17 | 0.08 | **0.07** |
| 40 | 3.64 | 0.64 | 1.63 | 2.00 | **12.30** | 0.60 | 0.48 |
| 70 | **16.87** | 2.47 | 5.14 | 5.19 | 4.38 | 1.80 | 1.38 |
| 100 | **78.02** | 7.32 | 10.02 | 9.55 | 17.80 | 3.40 | 2.56 |
| 130 | 71.12 | 19.92 | 17.29 | 23.96 | **81.31** | 5.86 | 4.49 |
| 160 | 119.98 | 55.83 | 26.16 | 33.18 | **24.85** | 8.56 | 6.54 |
| 190 | 345.77 | 82.04 | 25.64 | 32.05 | **134.03** | 11.89 | 8.98 |

Table 2: Execution time of various schedules as a function of the number of hosts

schedulers based on integrality relaxation does not increase as fast as the exact ones do. Discretization of time plays a key role in decreasing the execution time as can be seen in the comparison of the time demanded by CT-RR with that required by its discrete time counterpart.

Table 3 and Table 4 show the speedup and execution time of the proposed schedules as a function of network connectivity. This connectivity is expressed as a number in the interval [0, 1], a fully-connected network having connectivity of 1.0. As in the experiments reported in Table 1 and Table 2, ILPDT and DT-IRR generally produce the best schedules. DG did generated two of the largest speedups. Again, the schedulers based on Algorithm 2 (CT-IRR and DT-IRR) provided the smallest speedup. When the connectivity increases, the execution time typically decreases more than it does when the number of hosts increases. The largest speed up is no longer associated with the longest execution time, as happened when the number of hosts was varied.

The ratio between the longest and the shortest edge was varied but no significant result different than the ones previously reported was found.

From the results found in those experiments, the scheduler which generated the largest speed up was the ILPDT but at the cost of execution time. Thus, ILPDT is appropriate to scenarios with loose requirements of running time. Moreover, schedulers based on Algorithm 1 are more appropriate to applications which can tolerate more easily a non-optimal scheduling. These schedulers produced results which were close to those given by ILPDT but the execution time required, especially for DT-RR was much less than that for ILPDT, and this savings justify eventual small losses in speedup. Whenever time requirements are too limited to wait for a schedule derived via linear programming the DG scheduler can also be used, since it produces reasonable speedup values under certain circumstances.

| $\beta$ | Speedup | | | | | | |
|---|---|---|---|---|---|---|---|
| | CT-RR | CT-IRR | DT-RR | DT-IRR | ILPDT | RDU | DG |
| 0.1 | 71.06% | 71.06% | 84.72% | 81.08% | 83.81% | 96.38% | **1.41** |
| 0.22 | 72.97% | 72.97% | 96.23% | 72.55% | 96.46% | 89.29% | **1.38** |
| 0.34 | 70.09% | 69.64% | 91.51% | 69.64% | **1.44** | 97.30% | 98.64% |
| 0.46 | 98.17% | 66.10% | **1.52** | 65.90% | 97.64% | 81.37% | 92.93% |
| 0.58 | 89.92% | 66.10% | **1.52** | 69.92% | 99.64% | 89.51% | 92.97% |
| 0.7 | 98.82% | 65.32% | 98.82% | 65.31% | **1.53** | 77.78% | 90.87% |
| 0.82 | 91.71% | 66.10% | **1.52** | 77.74% | 65.61% | 73.64% | 87.03% |

Table 3: Speedup of various schedules as a function of network connectivity

| $\beta$ | Execution time (seconds) | | | | | | |
|---|---|---|---|---|---|---|---|
| | CT-RR | CT-IRR | DT-RR | DT-IRR | ILPDT | RDU | DG |
| 0.1 | 0.82 | 0.56 | 1.69 | 0.86 | 18.84 | 1.38 | **1.10** |
| 0.22 | 5.97 | 0.71 | 1.66 | 1.33 | 12.21 | 1.38 | **1.09** |
| 0.34 | 4.56 | 1.24 | 2.33 | 1.41 | **32.28** | 1.26 | 0.99 |
| 0.46 | 4.01 | 1.24 | **3.53** | 3.08 | 7.49 | 1.08 | 0.81 |
| 0.58 | 4.85 | 0.93 | **2.35** | 1.77 | 2.22 | 0.96 | 0.73 |
| 0.7 | 7.30 | 1.73 | 3.04 | 3.69 | **3.46** | 0.38 | 0.30 |
| 0.82 | 9.98 | 1.24 | **2.60** | 3.05 | 4.29 | 0.10 | 0.09 |

Table 4: Execution time of various schedules as a function of network connectivity

# 4 Conclusions

Grid networks can accommodate a new generation of users with high computational and data transfer demands. Although several grid systems already exist, this technology is still in its infancy. One of the major challenges of grids networks is the task scheduling, which should consider the heterogeneity of resources as well as make decisions during periods of time which duration as dictated by the application needs. This paper has introduced a set of grid schedulers able to deal with heterogeneous grid resources and with temporal restrictions. The effectiveness of this set of schedulers has been illustrated in several simulation experiments involving various network topologies.

As future work, we plan to integrate these schedulers in grid environment under development by the authors.

# References

[1] I. Foster. What is the Grid? A Three Point Checklist, July 2002. `http://www-fp.mcs.anl.gov/~foster/Articles/WhatIsTheGrid.pdf`. Acessed at 20 Oct 2006.

[2] Henri Casanova. Distributed Computing Research Issues in Grid Computing. *SIGACT News*, 33(3):50–70, 2002.

[3] 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.

[4] 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 .

[5] D. M. Batista and N. L. S. da Fonseca. Scheduling Grid Tasks in Face of Uncertain Communication Demands. *IEEE Transactions on Network and Service Management*, 8(2):92–103, 2011. DOI: http://dx.doi.org/10.1109/TNSM.2011.050311.100060 .

[6] 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 .

[7] 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 .

[8] Shawn McKee. LHC Scale Physics in 2008: Grids, Networks and Petabytes, 2005. http://ciara.fiu.edu/pasi/PASI_Shawn_May_18_2005.ppt. Accessed at 20 Oct 2006.

[9] John A. Silvester. CalREN: Advanced Network(s) for Education in California, 2005. http://isd.usc.edu/~jsilvest/talks-dir/20051021-cudi-merida.pdf. Accessed at 20 Oct 2006.

[10] Dab Ma and Wei Zhang. A Static Task Scheduling Algorithm in Grid Computing. *Lecture Notes in Computer Science*, 3033:153–156, 2004.

[11] Oliver Sinnen and Leonel A. Sousa. Communication Contention in Task Scheduling. *IEEE Transactions on Parallel and Distributed Systems*, 16(6):503–515, June 2005.

[12] 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 *Proc. IEEE International Symposium on Cluster Computing and the Grid (CCGRID'05)*, volume 2, pages 759–767, May 2005.

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

[14] 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 .

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