Journal of Grid Computing, vol. 4, no. 4, 2006, pp. 373-394.
The original publication is available at www.springerlink.com

Entropic Grid Scheduling

Youcef Derbal
School of Information Technology Management
Ryerson University
350 Victoria Street, Toronto, ON, M5B 2K3, Canada
yderbal@ryerson.ca

Abstract – *Computational Grids (CGs) are large scale dynamical networks of geographically distributed peer resource clusters. These clusters are independent but cooperating computing systems bound by a management framework for the provision of computing services, called Grid Services. In its basic form, the grid scheduling problem consists in finding at least one cluster that has the capacity to handle, within the constraints of a specified quality of service, a user service request submitted to the CG. Since CGs span distinct management domains, the scheduling process has to be decentralized. Furthermore, it has to account for the ubiquitous uncertainty on the state of the CG. In this paper, we propose a scalable distributed Entropy-based scheduling approach that utilizes a Markov chain model to capture the dynamics of the service capacity state. An entropy-based quantification of the uncertainty on the service capacity information is developed and explicitly integrated within the proposed grid scheduling approach. The performance of the proposed scheduling strategy is validated, through simulation, against a random delegation scheme and a load balancing-based scheduling strategy with respect to throughput, exploitation and convergence speed respectively.*

Keywords: Computational Grid, Decision Making, Scheduling, Entropy, Markov Chain.

1 Introduction

Computational Grids (CGs) are large scale dynamical networks of geographically distributed peer resource clusters made up of computing hosts such as desktops, instruments, high performance computing clusters, supercomputers, and special computing devices. These clusters are independent but cooperating computing systems bound by a management framework for the provision of computing services, called Grid Services. The evolution of CGs is fueled by the vision of computing as a utility to be sold to interested customers; including firms that maintain complex and costly yet underutilized computing infrastructures. With the maturity of CGs, these firms "will outsource their computing to specialists (IBM, HP, etc.) and pay for it as they use it, just as they now pay for their electricity, gas and water"[1] . In addition to their relevance to business operations, CGs have an enormous potential utility for the Grand Challenge Applications such as pharmaceutical drug discovery, oil reservoir modeling, and large scale environmental modeling [2-4].

In a CG, the resource clusters are usually contributed by distinct providers which have full control over the exploitation of the associated computing resources. We will assume that these resources are exposed for consumption through a uniform service oriented interface similar to the one specified by the Open Grid Service Infrastructure (OGSI) recommendations of the Global Grid Forum (GGF). For non-commercial grids, the providers aggregate their resources for the purpose of efficient sharing. This is the case for scientific grids such as Grid3 (http://www.ivdgl.org/grid3). In the case of

commercial grids, the providers may enter into agreements to organize their computing resources along an economy model to provide commercially viable business services. In both cases, the management of a CG spans multiple administrative domains and is vulnerable to the ubiquitous uncertainty on the knowledge of the state of resources. This uncertainty is induced by a number of environmental factors; including intermittent resource participation, dynamic load, network latency and processing delays, and random system failures. In this respect, the grid decision-making processes such as scheduling have to be decentralized and ought to account for the uncertainty on the state of the grid.

In its basic form, the grid scheduling problem consists in finding at least one cluster that has the appropriate service capacity to handle, within the constraints of a specified Quality of Service (QoS), a user service request submitted to the CG. This formulation is similar to the definition of the scheduling problem for processor-array-based computing systems [5]. However, there are many important distinctions; including the lack of central control, the unbounded number of computing hosts, the uncertainty on the knowledge of resource state, and the large scale geographic distribution of CGs [6]. The response to these challenges may take advantage of the rich pool of research on traditional scheduling in various application domains such as traditional high performance computing (HPC), and distributed computing systems [7-9]. In traditional HPC, scheduling may either be approached from the perspective of the application program or from the view of the underlying processing system [5]. The perspective of the application program divides the scheduling techniques into static and dynamic categories [10-12]. Static scheduling assumes the knowledge, at compile time, of many properties of the program, including data dependencies and task processing times [13-16]. The job execution process can therefore be modelled using a Direct Acyclic Graph (DAG), where the node weights represent the task processing times and the edge weights represent the inter-task dependencies and communication times [11]. Dynamic scheduling techniques on the other hand are devised to handle programs for which the above assumptions may not always be appropriate [17]. A significant number of the these techniques rely on dynamic processor load balancing through “idle-cycle stealing” so as to minimize the overall execution time of the parallel program as well as the scheduling overhead. The scheduling strategies may also be categorized in relation to the architecture of the system. The Bounded Number of Processors category applies to a fully connected cluster of processors where the underlying physical network is fast enough to neglect the communication overhead [17]. Efficient scheduling can in this case be achieved since the scheduling problem is reduced to the allocation of tasks to one of the available processors irrespective of their network location. On the other hand, the Arbitrary Number of Processor (APN) scheduling category is more complex. In this case the scheduling performance is significantly affected by the underlying topology of the inter-processor connectivity and the inter-task communication and synchronization delays[11, 18]. The grid scheduling problem share many similarities with dynamic scheduling for the APN category. However, while the emphasis in traditional HPC scheduling is on the minimization of the execution time for a parallel program, in grid scheduling the focus is on the handling of user requests in compliance with their specified QoS in addition to the maximization of grid resource exploitation and the system throughput. For most of the above mentioned scheduling strategies, the difficulty is well illustrated by the proven NP-completeness of the DAG scheduling variants, to the exception of some special cases [7, 11, 14]. The assumptions behind many of the proposed heuristics with polynomial-time complexity are in practice not feasible for the open grid environment [19]. In order to address the dynamic load and varying resource performance in a CG, many adaptive scheduling methods have been proposed [19-35]. For many of these approaches, the resource allocation is based on a prediction model of resource performance, application execution behaviour, and load conditions. Often, the focus of the prediction is centred around the task start time and task makespan (completion time) [19, 24, 25]. In

AppleS [20], performance driven scheduling strategies based on predefined templates are synthesized for specific categories of applications. This custom approach is particularly suited for parameter sweep applications such as MCell [36]. In He et al [19] quality of service is used to formulate a strategy that relies on a prediction of the task completion time as proposed in [25]. The assumptions made for the estimation of the model's parameters do not reflect the reality of the grid dynamics. Yang et al [37] proposed a work allocation scheme whereby less work is assigned to resources with higher expected load variance so as to achieve the completion of all the tasks for a given job at roughly the same time for all the resources involved in the processing. A time-series based predictor provides the expected load and variance [29]. Spooner et al [28] suggests a multi-tier scheduling algorithm where a performance model is used to manage the delegation of jobs between tiers.

Among the works cited above there is a subset that is closely related to the proposed approach [38, 39]**.** These strategies are similarly framed within a multi-step scheduling approach and address the delegation and the local scheduling phases using different methods. In [39] a hierarchical scheduling mechanism views the grid as a collection of independent resource clusters each equipped with a local scheduler. Tasks that can not be locally scheduled at the cluster level are either migrated to neighbouring clusters or delegated to a grid scheduler where an appropriate cluster with the required resources is sought for the execution of the tasks. The inter-cluster interaction underlying the migration function is performed without the involvement of the grid scheduler and as result may lead to improved scalability compared to [38]. However, the need for a central grid scheduler is detrimental to the overall scalability of the scheduling approach. In [38], execution time predictions are used in the selection of a target execution cluster for parallel jobs. At the cluster level, a genetic algorithm is utilized to carry out resource allocation and job assignment so as to minimize a comprehensive performance metric that involves makespan, over-deadline and resource idle-time. In addition, a cluster load metric is used by the scheduler at the grid layer to balance the load among the various clusters. This metric is based on job centric parameters such as the number of running jobs, the sum of their execution time, and the total job size. Other scheduling strategies that attempt to deal with the dynamic grid environment include those that are economy-centric. These are not the focus of this paper, however, for completeness it may be of value to mention that they are built around economic considerations such as resource price, utilization budget, and the overall economic utility of resources to the consumer [40-43].

Many of the surveyed strategies use performance prediction models to account for the dynamic grid environmental factors such as the varying resource performance, and the dynamic load conditions. However, they do not include an explicit quantification of the uncertainty on the resource availability and load information used by the prediction models and the scheduling decision mechanism. Furthermore, most of the surveyed works use a grid scheduling layer to coordinate the scheduling operation across management domains. This may be detrimental to the scalability of the associated scheduling strategies. In this paper we explore a grid scheduling approach that accounts for the dynamical nature of the resource behavior and the effect of the uncertainty associated with the state of the grid service capacity information. Motivated by the distributed topology of CGs, we view the grid scheduling problem as a set of two sub-problems; namely: (1) a local scheduling sub-problem; and (2) a delegation sub-problem. For the local scheduling sub-problem, the cluster that receives the service request performs a comparative assessment of local handling versus delegation to a remote cluster. The scheduling decision is then synthesized through the determination of the option that would lead to a higher likelihood of success in the presence of a quantified uncertainty on the estimated state of the grid service capacity. The state estimation and prediction is provided by a Markov chain model, while the state uncertainty is quantified using an entropy function. In the above formulation it is assumed

that the tasks associated with any service request can be executed on any cluster, provided that the required resources are available. The grid architecture adopted in this work is coherent with the desirable organization of the grid as a decentralized system of independent providers with distinct management domains and usage policies [44]. The resulting scheduling strategy is fully decentralized and posses an inherent scalability advantage over many of the surveyed strategies since no scheduler is required outside the confine of the independent management domain of a cluster. In addition, the proposed approach includes a direct account of the uncertainty on the dynamic state of the grid service capacity. To the best of our knowledge we are not aware of any similar treatment of the subject of resource state uncertainty in grid scheduling. However, in addition to the cited adaptive strategies proposed to deal with the grid environmental factor causing this uncertainty, there are some attempts to account for such uncertainty through advanced reservation schemes [45, 46]. These schemes often rely on centralized repositories of resource descriptions and state such as Globus MDS [47]. The scalability problem associated with a centralized directory and the insufficient attention given to the dynamics of the resource state render these deterministic reservation schemes ineffective for the open and dynamic grid environment.

The paper is organized as follows: Section 2 describes the grid architectural framework. The details of the proposed grid scheduling strategy are given in section 3. Section 4 presents the simulation results, while section 5 includes a short discussion on open issues and future works. The paper is concluded in section 6.

2 Grid Architectural Framework

The framework under consideration views the grid as a dynamic federation of resource clusters contributed by various organizations. Each cluster constitutes a private management domain. It provides a set of grid services assumed to be exposed in a fashion that reflects the basic outlines of the OGSI recommendations [48]. Clusters may join or leave the grid at any time without any disruption to the grid operation. The effect of this dynamic membership is limited to the configuration of neighboring clusters. Each cluster includes a set of agents that host the offered services (see figure 1). One agent, labeled as the Principal, is designated to coordinate inter-cluster operations such as dissemination of the availability state of service capacity and the dispatching of delegated service requests. The lines linking the Principals define the topology of the grid, i.e. the pathways for the information exchange about the capacity of hosted services. This in turn defines the notion of a neighbor. Hence, two clusters are considered to be neighbors if and only if there is an information pathway linking them as defined above. The resulting topology, has the robustness properties of a Power Law network where most nodes have few links and few nodes have numerous links [49, 50]. In this topology, that we call Grid Neighborhood (GN) topology, there are no restrictions on the IP connectivity between any pair of grid clusters. In fact, it is essential that such connectivity be available for the implementation of grid-wide scheduling strategies and service request delegations among peer clusters. Furthermore, we will assume that each resource cluster has the capability to handle service requests for which it has the required resources. Clusters are also assumed to have the capability to delegate the handling of service requests to other clusters in function of some inter-cluster Service Level Agreements (SLAs) [51]. While SLA negotiations and enforcements are not addressed in this paper, we will assume the existence of some agreements that govern inter-cluster interactions such as the regular exchange of resource state information. Future works may be focused on the integration of SLA negotiation as part of the delegation step of the proposed strategy especially as such integration is expected to be further facilitated with the upcoming GGF WS-Agreements standard.

The proposed scheduling approach does not make any assumptions on the internal architecture of a cluster. However, in order to provide a context for the arguments that follow we will assume that at any given time there is one agent configured as the Principal of the cluster (see figure 2). This configuration may vary dynamically to provide a higher degree of cluster reliability. The Principal is associated, at a minimum, with a scheduler, a user request manager and a service manager. The responsibilities of the Principal include the management and allocation of resources within the cluster and the discovery of grid services hosted by peer clusters. The Principal is also responsible for the dispatching of delegated user service requests to peer clusters. Each agent is equipped with a local resource manager and a task execution manager. The resource manager keeps a current account of resource load and availability, while the execution manager is responsible for the execution of the tasks assigned to the agent.

3 Entropic Grid Scheduling

In light of the emerging Service Oriented Computing (SOC) paradigm [52] and the increasing acceptance of the Service Oriented Architecture (SOA) approach to the organization of CGs, a grid may be viewed as a large scale service provisioning environment. The provided grid services may include compute cycle services, data mining services, data storage services, network bandwidth provision, and specialized application services. For the exploitation of grid resources, users submit service requests that may require one or more grid services to be available with a sufficient capacity. Hence we define a User Service Request (USR) as follows:

$$\begin{aligned} &usr = (\Omega, R, \Phi, Q) \\ &\Omega = \{s_0, s_1, ..., s_{m-1}\} \\ &R = \{c_0, c_1, ..., c_{m-1}\} \end{aligned} \qquad (1)$$

$s_i, i = 0,...,m-1$ is a required grid service and $c_i, i = 0,...,m-1$ is the service capacity required for service s_i. The service capacity is expressed as the number of servslots, where a servslot is defined as the unit capacity of the hosting environment to run a single instance of the service in question [53]. It represents the aggregated capacity of the collection of software and hardware resources required for the successful operation of a single service instance. The resources in question may include CPU slots, RAM, special hardware devices, disk space, cache size as well as any required licenses of utility software that the service instance may need for its successful operation. If the service requires for its execution a specific Operating System, some processor architecture, or the presence of a Java Virtual Machine and possibly a required heap size, then these would be part of the resources attached to a servslot. The service handling flow Φ, which may be specified using a state machine, defines the execution sequence of the various tasks associated with the handling of the USR. Q is a set of user defined performance metrics which make up the user desired QoS. These metrics may include, for example, the maximum wait time before scheduling, and the number of required restart of failed tasks. In this paper, the Time-To-Schedule (TTS) is the only considered QoS parameter. It is defined as the maximum allowed discrete time interval separating the events of USR submission and the scheduling of the last task of the USR respectively.

The grid scheduling problem can be formulated along the following scenario. A USR is submitted to a given cluster (submission cluster), the question then is: what is the collection of clusters that have the required service capacity to execute the tasks associated with the USR in compliance with some user specified QoS? Our proposed solution is a multi-step scheduling approach applied independently

to each grid service required by the USR. Let the sequence of scheduling decisions needed to find a cluster that hosts the required grid service s_j with a sufficient available capacity be defined as follows:

$$D_j = \left\{ d_{ij} \right\}_{i=0}^{n_D}, \ j = 0,...,m-1 \qquad (2)$$

n_D is a non-negative integer. Each decision d_{ij} is bound to the target cluster x where it is performed. The scheduling decision d_{ij} has one of the following outcomes: (O1) the TTS is exceeded; (O2) cluster x is deemed to be the best cluster that can provide the required service s_j; (O3) the scheduling task is delegated to the best neighboring cluster that can provide the required service s_j. The "best" qualifier is assumed in a sense to be defined later in the section. The scheduling process is terminated when either (O1) or (O2) is reached. The TTS could be set by the user as part of the USR QoS specification. The first outcome implies that the USR can not be scheduled in compliance with the required QoS and the entire scheduling operation is aborted. The second outcome implies that the component of the USR is deemed to be scheduled and the corresponding cluster is added to the solution set. For the third outcome, the scheduling step results in the delegation of the task handling to a peer cluster.

In this paper, we assume that the required services of the USR can be independently scheduled. In other words the selections of the potential providers for the required services of the USR are assumed to be performed separately. Furthermore, we assume that all service requests arriving at any cluster comply with the authentication and authorization requirements that might be associated with the required services in question. In the absence of such assumption, the proposed model may be expanded to account for the security concerns. However, such expansion would need to address the issue of service denial. This may be accomplished through a service discovery phase that precedes the scheduling operation. The discovery process would compile a list of providers for which the given service request is authorized to be handled. This information may then be used to inform the scheduling process so as to avoid delegations that otherwise would result in a denial of service.

3.1 Grid Service Capacity Model

The interplay of intermittent resource participation, resource load dynamics, network latency and processing delay and random subsystems' failures create a ubiquitous uncertainty on the state of the grid capacity to handle user requests. In order to account for its dynamics as well as its state uncertainty, the capacity of a given service is modeled as a stochastic process $\{X_n, n = 0,1,2,...\}$ that takes on a finite or countable number of possible values of servslots. The set of possible values of the process is the set of non-negative integers $\{0,1,2,...\}$. The index n represents discrete time. The process is said to be in state i at time n if $X_n = i$. Let us assume that whenever the process is in state i, there exists a constant probability P_{ij} that the process will next be in state j such that:

$$P_{ij} = P\left\{ X_{n+1} = j \middle| X_n = i, X_{n-1} = i_{n-1},..., X_0 = i_0 \right\} \qquad (3)$$

Assuming that this is true for all $n \geq 0$ and all states $i_0, i_1, ..., i_{n-1}, i, j$, the process under consideration is known as a Markov chain [54]. The probabilities P_{ij} are also called the one-step transition

probabilities since they are associated with a single increment of the time index variable. Then we can define the matrix of one-step transition probabilities P_{ij} as follows:

$$\mathrm{P} = \begin{pmatrix} P_{00} & P_{01} & \ldots \\ P_{10} & P_{11} & \ldots \\ . & . & . \end{pmatrix} \qquad (4)$$

Similarly, we can define the matrix of the n-step transition probabilities as follows:

$$\mathrm{P}^{(n)} = \begin{pmatrix} P_{00}^{(n)} & P_{01}^{(n)} & \ldots \\ P_{10}^{(n)} & P_{11}^{(n)} & \ldots \\ . & . & . \end{pmatrix} \qquad (5)$$

The element $P_{ij}^{(n)} = P\left\{X_{n+m} = j \middle| X_m = i\right\}$ represents the probability that the service capacity takes a value equal to j at time $n+m$ knowing that it took a value of i at time m. Each cluster maintains a registry of its hosted services. Let s be one such service, and let $l_{ij}^{(n)}$ be the total number of events, recorded in the time window $(0 \quad n]$, where the capacity of s has changed from a value i to a value j according to the cluster's resource accounting and management process. Then we can estimate the one-step transition probabilities at time n as follows:

$$\hat{P}_{ij}(n) = \frac{l_{ij}^{(n)}}{n} \qquad (6)$$

These transition probabilities are maintained in the service registry and updated at every discrete moment of time. The above equation can be rewritten in the following recursive form:

$$\hat{P}_{ij}(n) = (1-\lambda).\hat{P}_{ij}(n-1) + \lambda.n_0 \qquad (7)$$

$\lambda = 1/n$, $n_0 = 1$ if a transition $i \to j$ occurs at time n, otherwise $n_0 = 0$. For the implementation of the above relation we start with $\hat{P}_{ij}(0) = 0$ for all the element of the transition matrix except for $\hat{P}_{ww}(0)$, which is set to 1, where w is the state of high service capacity. Given the estimates of one-step transition probabilities, the model can provide an n-step-ahead prediction of a specific service capacity state such that:

$$\mathrm{P}^{(n)} = \mathrm{P}^n \qquad (8)$$

The above relation is a direct result of the Chapman-Kolmogorov equations [54]. Using (5), (6), (7), and (8) it is now possible to predict the service capacity state for any arbitrary future time-step. This capability will be used in the next subsection to complete the formulation of the proposed scheduling strategy.

The cluster bound service registry holds the service capacity information (service capacity state and one-step transition probabilities) for the services hosted by the cluster in question and its

neighbors respectively. In the case of services hosted by neighboring clusters, the value of n_0 is computed based on the information assumed to be regularly disseminated by theses neighbors about the current state of their service capacity. Since neighbors disseminate the capacity information only about their hosted services, the registry size for the $i-th$ cluster is equal to:

$$RSZ_i = \left(N_i + \sum_{j=1}^{M_i} N_j \right) u \qquad (9)$$

Where u is the required storage space associated with a single service entry which includes the current service capacity level and state, the one-step-ahead transition probability matrix, and other identifying fields such as the service name, the cluster ID, and the IP address of the cluster's principal. N_i is the number of services hosted by the $i-th$ cluster, and M_i is the number of its peer neighbors. Given the above described fields of a service entry, u is estimated to be equal to $(64+n_s^2)\,Bytes$, where n_s is the number of states of the Markov chain model. Figure 3 illustrates the required storage size of the registry as a function of the number of states n_s for a cluster with seven neighbors. In practical terms, the registry would be implemented using a relational database or an LDAP directory bound to the home cluster. Judicious caching of the registry in the Principal's run-time memory would ensure a faster information retrieval. However, since the registry is local to the cluster and exclusively accessible to its management mechanisms, even in the absence of caching the queering performance would not degrade with the increase of users or gird size as is the case for Globus MDS2 GRIS and GIIS [55].

3.2 Entropic Scheduling Strategy

Let us assume that at time m a cluster h hosting the service s has a number of queued requests for this service with a required cumulative capacity of $c_{queue} > 0$. In addition to the already queued requests, an additional request is received for the service s with a required capacity $c_{new} > 0$. In order to assess the ability of the cluster to handle the service request before the TTS expiry, we have to estimate the probability that the service capacity enters the state j (corresponding to the $c_{queue} + c_{new}$ value) at some time $n \in [m \quad m+T]$ starting from the current state i_0 at time m, where $i_0 \neq j$. T is a non-negative integer which denotes the actual value of the TTS discrete time interval. This probability is denoted as follows:

$$P\left\{ \bigcup_{n=0}^{T} \{ X_{m+n} = j \} \mid X_m = i_0 \right\} \triangleq P_{i_0 \xrightarrow{\leq T} j} \qquad (10)$$

$P_{i_0 \xrightarrow{\leq T} j}$ being the probability that the capacity enters the state j in T or less steps is not equal to $P_{i_0 j}^{(T)}$ which represents the probability that the capacity enters the state j in exactly T steps. This last may be denoted as $P_{i_0 \xrightarrow{T} j}$. Using this observation, we can compute $P_{i_0 \xrightarrow{\leq T} j}$ as follows:

$$
\begin{aligned}
P_{i_0 \xrightarrow{\leq T} j} &= P_{i_0 \xrightarrow{0} j} + \\
&P_{i_0 \xrightarrow{1} j} \cdot \left(1 - P_{i_0 \xrightarrow{0} j}\right) + \\
&P_{i_0 \xrightarrow{2} j} \cdot \left(1 - P_{i_0 \xrightarrow{1} j}\right) \cdot \left(1 - P_{i_0 \xrightarrow{0} j}\right) + \\
&\ldots. \\
&P_{i_0 \xrightarrow{T} j} \cdot \prod_{\xi=0}^{T-1} \left(1 - P_{i_0 \xrightarrow{\xi} j}\right)
\end{aligned}
\qquad (11)
$$

Every term of the above relation represents the probability that the capacity enters the state j in the corresponding step on condition that it has not entered such state in all the previous steps. Relation (11) can be written in the following compact form:

$$
\begin{aligned}
P_{i_0 \xrightarrow{\leq T} j} &= \sum_{n=1}^{T} P_{i_0 \xrightarrow{n} j} \cdot \prod_{\xi=0}^{n-1} \left(1 - P_{i_0 \xrightarrow{\xi} j}\right) \\
&= \sum_{n=1}^{T} P_{i_0 j}^{(n)} \cdot \prod_{\xi=0}^{n-1} \left(1 - P_{i_0 j}^{(\xi)}\right)
\end{aligned}
\qquad (12)
$$

The term $P_{i_0 j}^{(0)}$ is omitted since it is equal to zero for $i_0 \neq j$. Note that a level of service capacity greater than $c_{queue} + c_{new}$, hosted by a cluster h, would also satisfy the requirement of the service request. Therefore, the likelihood that there will be a sufficient capacity to handle the request should hence be given by the probability $P_{i_0 \xrightarrow{\leq T} \Sigma_j^{(h)}}$ that the capacity enters any state $\Sigma_j^{(h)} \subset \Sigma$ within T steps starting in i_0. $\Sigma_j^{(h)}$ is the set of states associated with a capacity greater than or equal to that associated with state j. Σ is the set of possible states of the service capacity. For an observer cluster h, we define $P_{h,h,i_0 \xrightarrow{\leq T} \Sigma_j^{(h)}}$ as the probability that the capacity of service s (that it hosts) enters any state $k \in \Sigma_j^{(h)}$ on condition that it has not entered any of the other states of the set $\Sigma_j^{(h)}$ in the previous steps. It follows that:

$$
\begin{aligned}
&P_{h,h,i_0 \xrightarrow{\leq T} \Sigma_j^{(h)}} = \\
&\sum_{k \in \Sigma_j^{(h)}} P_{i_0 \xrightarrow{\leq T} k} \cdot \prod_{\zeta \in \left(\Sigma_j^{(h)} - \{k\}\right)} \left(1 - P_{i_0 \xrightarrow{\leq T} \xi}\right)
\end{aligned}
\qquad (13)
$$

The first index h refers to the observer cluster where the computation is made, while the second index h refers to the service home cluster. With the assumption made earlier that neighboring clusters disseminate the capacities of their hosted services, cluster h can estimate, as an observer, the likelihood that a neighboring cluster x would have a sufficient capacity to handle a delegated request. Hence we can generalize relation (13) to quantify the estimated probability by an observer cluster h, that a peer cluster $x \in \bar{\aleph}_h$ (neighbor or itself) has a sufficient capacity to handle a service request as follows:

$$P_{h,x,i_0 \xrightarrow{\leq T} \Sigma_j^{(x)}} = \sum_{k \in \Sigma_j^{(x)}} P_{i_0 \xrightarrow{\leq T} j} \cdot \prod_{\zeta \in \left(\Sigma_j^{(x)} - \{k\}\right)} \left(1 - P_{i_0 \xrightarrow{\leq T} \zeta}\right) \quad (14)$$

$\aleph_h$ is the set of clusters neighboring h in the sense defined in section 2, and $\bar{\aleph}_h = \aleph_h \cup \{h\}$. $\Sigma_j^{(x)}$ is the set of states associated with a capacity equal or greater than that of state j for the x- hosted service. Note that when estimating the neighbors' capabilities, the observer cluster, i.e. h in this case, is only aware of the capacity information disseminated by its neighbors. As a result the state j corresponds to a capacity $c_0 + c_{new}$, where c_0 is the service capacity of the neighbor at the time of the estimation. The explicit mention of the service s under consideration is omitted in (11)-(14) to reduce cluttering. Furthermore, note that the basic transition probabilities $P_{ij}^{(n)}$ used to evaluate (11)-(14) are always computed from the perspective of the observer cluster. Given the above discussion, the question then is: *(Q1) given a service request submitted to a cluster h, should such request be queued for handling in the receiving cluster h or should it be delegated to a neighboring cluster?* It is important to remember that such choice is to be performed in the context of a geographically distributed system that spans multiple administrative domains. Hence, it should in principal take in consideration the cost of network bandwidth and congestion in addition to the lower assurance of QoS that may be expected from a remote cluster. These are important issues; however, our primary focus is the effect of the service capacity state uncertainty on the grid scheduling process. An expanded elaboration that includes the above mentioned issues of network bandwidth and QoS is deferred to another venue.

Part of our approach to question *Q1* is the use of entropy [56] as a performance measure that reflect the quality of a scheduling decision that is based on uncertain information about the state of service capacity. There is ample evidence in control theory as well as information theory about the general applicability of entropy to dynamical systems [57, 58]. In control theory, equivalence was shown between a generalized energy function as a measure of performance of dynamical systems and entropy as defined by the second law of thermodynamics [59]. For instance, it was noted that since decision making is related to information processing one can make the assumption that it contribute to the generation of entropy in the information theoretic sense [57, 60]. Entropy as a performance measure was also shown to be applicable and common to both information and feedback control theories [57]. In this respect, let $E_n, n = 1, 2, .., n_h$ be the set of events associated with the transitions $i_0 \to k \in \Sigma_j^{(x)}$, where $x \in \aleph_h$, and n_h is the cardinality of $\aleph_h$. Assuming that these events are mutually exclusive, we can define an entropy function as a measure of uncertainty on the collection $\Omega = \bigcup_{n=1}^{n_h} \{E_n\}$ of these events as follows:

$$H_{\aleph_h}^{(h)} = -\sum_{x \in \aleph_h} P_{h,x,i_0 \xrightarrow{\leq T} \Sigma_j^{(x)}} . \log\left(P_{h,x,i_0 \xrightarrow{\leq T} \Sigma_j^{(x)}}\right) \quad (15)$$

The superscript in $H_{\aleph_h}^{(h)}$ indicates that the uncertainty is estimated from the perspective of cluster h. The probabilities are computed using the state information supplied by the neighboring clusters about the capacity associated with the hosted service s in question. Similarly we define a measure of

uncertainty on the occurrence or nonoccurrence of the transition event $i_0 \to \Sigma_j^{(h)}$ for the cluster h as follows:

$$H_h^{(h)} = -P_{h,h,i_0 \xrightarrow{\leq T} \Sigma_j^{(h)}} . \log\left(P_{h,h,i_0 \xrightarrow{\leq T} \Sigma_j^{(h)}} \right) \qquad (16)$$

Now let us consider the subset $\aleph_h^* \subset \aleph_h$ such that for all $z \in \aleph_h^*$ we have:

$$P_{h,z,i_0 \xrightarrow{\leq T} \Sigma_j^{(z)}} = \max_{x \in \aleph_h} \left(P_{h,x,i_0 \xrightarrow{\leq T} \Sigma_j^{(x)}} \right) \qquad (17)$$

Each element $z \in \aleph_h^*$ is associated with the maximum probability that the hosted service in question will have a capacity that enters one of the states of the set $\Sigma_j^{(z)}$. If we let $z^* \in \aleph_h^*$ be a randomly chosen cluster then the proposed delegation heuristic can be described as follows. A delegation to the neighboring cluster z^* is performed if only if the following relations are satisfied:

$$P_{h,h,i_0 \xrightarrow{\leq T} \Sigma_j^{(h)}} < P_\alpha \leq P_{h,z*,i_0 \xrightarrow{\leq T} \Sigma_j^{(z*)}} \qquad (18)$$

$$H_{\aleph_h}^{(h)} < H_h^{(h)} \qquad (19)$$

In all other cases, the service request is queued locally. Relation (18) ensures that the delegation takes place only if the transition to the desired set of solution states is most likely to happen compared to a local queuing of the service request. The second relation ensures that such predictive assessment is not done in the presence of a higher uncertainty. Indeed, given the shape of the entropy function, relation (18) would not be sufficient to yield a delegation decision that is necessarily better than local queuing (see figure 4). For example a probability of 0.5 for delegation is not necessarily better for the scheduling performance than a probability of 0.25 for local queuing given the higher uncertainty associated with the 0.5 probability. In order to avoid the region of high uncertainty, we introduced the threshold P_α which should preferably be chosen to be greater than 0.5.

For the considered single service USR, the scheduling sequence given in (2) reduces to $D = \{d_i\}_{i=0}^{n_D}$. Given the above delegation strategy, the decisions d_i are synthesized using the proposed entropic grid scheduling heuristic illustrated using the UML activity diagram of figure 5. Because a delegation is more costly (network bandwidth, congestion etc…), it is allowed only if: (1) it is estimated that the target neighboring cluster is more likely to have a sufficient service capacity compared to the receiving cluster where the scheduling decision is being made; (2) the uncertainty associated with the information used to make the determination in (1) is lower than that associated with the prediction information about the future state of the locally hosted instances of the service. Note that due to the use of the TTS limit to terminate the scheduling process, the sequence D is always finite. If the TTS limit is not exceeded, the scheduling sequence would consist of n_D delegations and a single final decision to locally schedule the service request.

4 Simulation Results

The Midland Grid Emulator developed by the author was used for the simulation of the proposed scheduling approach (see figure 6). The emulated grid may be configured for an arbitrary number of clusters. The inter-arrival time of USRs at the various clusters is simulated using a Poisson process for the first set of simulations while a Pareto distribution is used to simulate a bursty USR load for the second set of simulations. For the reported simulation results, the USRs consist of a single required grid service with a randomly generated required capacity. The duration of the execution time necessary for the handling of a USR is also simulated using a Poisson process.

One of the potential commercial advantages of a grid is the ability to construct an open computing system where new resources can be added as needed in order to handle increased user load or more computationally demanding applications. The realization of this advantage relies critically on the scalability of the decision-making mechanisms such as service discovery, scheduling and load balancing. In concrete terms, the scheduling strategy is said to be scalable if an increase in the grid size does not result in a drastic degradation of the grid performance as embodied by the most pivotal performance indicators such as response time, throughput, and resource exploitation. In order to quantify these indicators we define three corresponding performance metrics; namely:

$$v = \frac{1}{n_R} \sum_{i=1}^{n_R} nhops_i \qquad (20)$$

$$\mu = \frac{n_H}{n_R} \qquad (21)$$

$$\varphi = \mu . \frac{n_{US}}{n_{DS}} \qquad (22)$$

v is the grid-wide average number of hops per successful scheduling decision. μ is the scheduling throughput rate, and φ is the resource exploitation rate. n_R is the total grid-wide number of service requests, and n_H is the number of handled service requests. $nhops_i$ is the number of hops before the $i-th$ service request was successfully scheduled. n_{DS} is the grid-wide total number of deployed instances for a given service, and n_{US} is the grid-wide total number of used instances of the same service. The exploitation ratio is normalized using the throughput in order to reflect the reality that for the same simulation time there are more service requests generated in the case of random delegation because of the smaller processing overhead of the Midland Grid Emulator. Using these defined performance metrics, the performance of the proposed scheduling strategy is illustrated through a comparison against two different scheduling strategy; namely: (1) a scheduling scheme based on random delegation (see figure 7-9); and an adaptive scheme based on dynamic load balancing (see figure 10-15).

In the first set of simulations, the delegation for the random scheduling scheme is made to a randomly selected cluster that is believed to have the required service capacity at the time of the delegation decision according to the capacity information disseminated among neighbors. The simulation results clearly illustrate that the proposed entropy-based scheduling outperforms the random-based delegation. The simulated grid environment had the exact same configuration for both scheduling

strategies. This includes the service request distributions as well as the hosted resource distributions. Both strategies are multi-tier scheduling approaches where the service request is either handled at the submission cluster or delegated to a peer cluster if the service capacity of the submission cluster is inadequate. The delegation process is recursively applied until a cluster with the sufficient service capacity is found or the TTS timeout is reached. The main distinction between the proposed entropy-based scheduling algorithm and the random delegation is in the choice of the next peer cluster designated to handle the service request. In this respect we attribute the relative superior performance of the proposed entropic scheduling approach to the quantification of the uncertainty on the information about the state of service capacity. With the use of the quantified uncertainty, the rate of rejection of delegated service requests due to exceeded TTS limit is reduced. Hence the throughput for the entropic scheduling is found to be about 100% higher than that for random delegation (see figure 7). For the same service request distributions, and grid resource distributions, the resource exploitation of the proposed algorithm not only outperforms the random delegation, but was also maintained around the 80% level as the grid size is increased to 1000 clusters (see figure 9). Under the same grid environmental conditions, the random delegation resulted in a dramatic decrease of the resource exploitation (see figure 9). The convergence speed was also far better for the proposed algorithm compared to the random delegation (see figure 8). In fact, the simulation results of figure 8 do not express the full story because the number of hops is computed only for the delegations that resulted in a service request handling within the TTS time limit. Indeed, given the high rejection rate (low throughput) for the random delegation, the performance differential between the entropic scheduling and random delegation would in reality be more significant if we included the delegations that never converged within the TTS limit. The other clear advantage of the proposed entropic scheduling algorithm is its scalability. All three defined performance metrics show a relatively stable level as the grid size is increased from 25 to 1000 clusters (figures 7-9).

It is important to note that what we have labeled as a random delegation is in fact a scheduling strategy that relies on the choice of a target cluster from a legitimate set of clusters that are all deemed adequate for the handling of the service request as far as it can be determined from the propagated resource state information at the time of the delegation decision. In fact this is one of the most widely used scheduling approaches in commercial schedulers such as LSF [61]. Hence, we consider such strategy as a legitimate representative of any scheduling scheme that utilizes the currently observed grid resource state information without any consideration for the associated uncertainty. However, in order to further illustrate the performance of the proposed strategy, we run a second set of simulations driven by a bursty request submission process, where the request inter-arrival time follows a Poisson process while the number of simultaneously arriving service requests is governed by a Pareto process. The arrival Poisson rate is different for each cluster and is randomly chosen. The range and shape of the Pareto process are set to 10 and 2 respectively for all clusters so as to provide an equalized input load from all submission points of the grid. For this simulation, the proposed strategy is compared to an adaptive scheduling scheme that we call Adaptive Load Scheduling (ALS). Similarly to the proposed strategy, ALS uses a multi-step scheduling framework where a service request is scheduled locally if a sufficient service capacity is available otherwise it is delegated to the neighboring peer cluster with the lowest load among all neighbors of the cluster currently processing the service request. The load is defined here as the ratio between the used and the total capacity for the hosted service in question. The ALS strategy has much in common with the load balancing approach used in the adaptive scheduling scheme proposed in [38]. However, instead of making use of a grid scheduler to perform the load balancing, ALS utilizes a delegation strategy based on disseminated load information among neighbors. The common use of the peer-to-peer delegation framework allows a more appropriate comparison between the Entropic and the ALS strategies. The simulations are

conduct for balanced and unbalanced grid resource distributions. For an unbalanced distribution, the grid clusters are configured to host randomly chosen set of services with randomly generated values of respective service capacities. For a balanced distribution all grid clusters are configured to have the same services with the same service capacity levels. The results of the simulations show that all three metrics maintain a relatively stable level as the grid size increases for both compared strategies (see figures 10-15). This illustrates the expected good scalability of both methods since they share the same decentralized and neighborhood-based delegation mechanism. Both strategies yield a comparable good performance with respect to throughput and exploitation levels for both balanced and unbalanced grid resource distributions. This similar performance may be explained by the fact that the compared strategies utilize, within the same neighborhood delegation framework, two "conjugate" exploitation state indicators; namely: available service capacity (supply) and service load (demand). The implied equivalence between these two exploitation variables is meant only in the sense that both load and capacity availability may be used to build an effective resource management mechanism that maximizes exploitation and throughput. The two strategies diverge in their performance vis-à-vis the average number of hops, where the entropic scheduling approach outperforms ALS (see figures 10 and 13). While both strategies yield a satisfactory resource exploitation, the entropic approach achieves this goal at a lower cost (number of hops) than ALS. This may be explained by the fact that the prediction of future service capacity and the quantified uncertainty are used in the proposed approach with a bias favoring local queuing so as to encourage the consumption of resources that are closer to the submission cluster. As result, the cumulative effect of this bias throughout the delegation chain leads to a provider cluster that is closer to the submission cluster than would otherwise be possible. On the other hand, the delegation in ALS relies on a one step-ahead prediction of the loads associated with the cluster where the scheduling step is being taken and its neighbors respectively. The request is delegated to a peer whenever the current scheduling cluster is estimated to have a higher load than its neighbors, resulting hence in a higher number of average hops. Equipping ALS with a different load prediction may reduce the average number of hops. However, the entropic approach would still have an advantage, at least conceptually, with respect to the fact that a many-step-ahead prediction of service capacity is provided along with a quantification of the uncertainty associated with the very information being utilized to make the prediction. The lower average number of hops for the balanced resource configuration compared to the unbalanced configuration is expected (figures 10 and 13). This is because the uniformly available service capacity throughout the grid translates into a diminished incentive for request delegations since the local and remote exploitation patterns are likely to converge to similar equilibrium points given the similar USR load distribution used for all clusters. It should be mentioned that a lower average number of hops may not necessarily translate in a higher throughput. Indeed, a service request may be handled after few hops but separated by longer pending periods of time between delegations, while another service request may be handled after many hops with smaller pending periods of time. This may be the reason behind the difference in the average number of hops of the entropic and the ALS strategies while yielding similar resource exploitation results.

The use of a bursty USR load distribution resulted in an increase of the average number of hops for the proposed scheduling strategy (figures 8, 10, and 13). This is expected given the highly dynamic nature of service capacity induced by the combination of a Poisson distribution for the request inter-arrival time and the Pareto distribution for the simultaneously arriving number of service requests. Nevertheless, the scalability advantage illustrated for the Poisson USR load still holds for the Pareto USR load. Furthermore, the overall performance of the entropic scheduling approach is illustrated to perform equally well or better compared to the load balancing ALS approach. This is facilitated through an experimental widening of the capacity intervals associated with the states of the Markov chain so as to filter the high frequency changes of the service capacity levels while maintaining an

acceptable prediction. While we have illustrated the encouraging performance under a bursty user load, other related issues may be considered in future works; including the finite size of the submission queues and the associated issue of service request loss that may result from bursty USR load distributions.

5 Discussion and Future Works

The scientific and commercial viability of the grid computing paradigm requires a comprehensive approach to a number of critical issues, in particular: the ubiquitous uncertainty, the distribution of the grid management framework, and the scalability of the exploitation and control mechanisms. In this respect, the synthesis of the proposed scheduling approach is systematically developed to take into account these critical issues. Starting from the necessity for a distributed management and control framework, the proposed scheduling approach is designed to be fully decentralized. The delegation to neighboring clusters encourages the use of the grid resources that are closer to the submission point. This has been suggested to result in a more efficient and cost effective use of the underlying network infrastructure [61]. With the assumed topology of the grid, the exchange of service capacity information is limited to inter-neighbor pathways. This, we suggest, induces an improved scalability of the scheduling approach since immediate neighbors are privileged with the knowledge of the service capacity state information to the exclusion of other peers. Such advantage is highly desirable since scalability is a critical performance indicator that ranks high in importance for the practical feasibility of grid decision making strategies; including scheduling. The network of distributed service registries needed to support the proposed scheduling approach is more scalable compared to Globus-MDS2. In fact, it was shown that both MDS2 GRIS and GIIS of the Globus Toolkit may maintain a good scalability only if data caching is used, otherwise the performance degrades dramatically with the increase in the number of users [55]. Such caching share significant similarities with the distributed cluster bound service registries considered in this paper. The second contribution of this work is the proposed theoretical formulation of an uncertainty model of the service capacity state. This uncertainty model was illustrated to yield a very encouraging performance with respect to throughput, exploitation and convergence speed when compared to random delegation and adaptive load balancing approaches. We believe that the development of this uncertainty model on the service capacity and its integration in the scheduling strategy is not only novel but is also of critical importance to the grid computing paradigm. This is because addressing the fundamental issue of uncertainty is by far one of the most decisive challenges to the success of any management or control framework applied to large scale dynamical systems such as computational grids.

The extra storage overhead necessary for the maintenance of the transition probabilities associated with the Markov chain model may require future attention. As shown in section 3.1, the size of the matrices of the one-step transition probabilities is dependent on the number of states associated with the service capacity. One possible approach to limit the size of these matrices is to associate the states with disjoint intervals of capacity values. The width of these intervals may be appropriately chosen to minimize the registry storage size and achieve an appropriate filtering of the fast changes that may be caused by bursty request arrivals. However, such reduction in resolution of the model would have to be evaluated in the context of a tradeoff between the storage size of the service registry, the sensitivity to the capacity changes and the performance of the scheduling process. Another issue that may be considered in future works is the assurance of request handling within the specified TTS. The proposed strategy has a built in process of compliance with the TTS requirement through the selection of a cluster that is predicted to have a sufficient capacity within the specified TTS. However, a more

dynamic assurance may be considered in future works whereby requests that are estimated at risk of violating the TTS requirement are given a higher priority of handling. Such assurance would have to consider the tradeoff between network congestion and request blocking since a long TTS may lead to less blocking of service requests but more network traffic while a low TTS may lead to more blocking but less congestion.

6 Conclusions

The paper presents a decentralized grid scheduling approach that relies on a Markov chain based estimation model of the service state capacity and a novel entropy-based quantification model of the related uncertainty. The proposed approach is illustrated to have a scalability advantage in the sense that an increase in the size of the grid does not negatively impact the scheduling performance in any significant fashion. Furthermore, the performance of the entropy-based scheduling approach is shown to perform well compared to two different scheduling strategies with respect to throughput, exploitation and convergence speed. The proposed approach does however require an extra storage capacity at the cluster level to maintain the dynamical model of service capacity. This, we conjecture, can in practice be satisfactorily addressed through a discretization of the service capacity range into a sufficiently small set of states.

Journal of Grid Computing, vol. 4, no. 4, 2006, pp. 373-394.
The original publication is available at www.springerlink.com

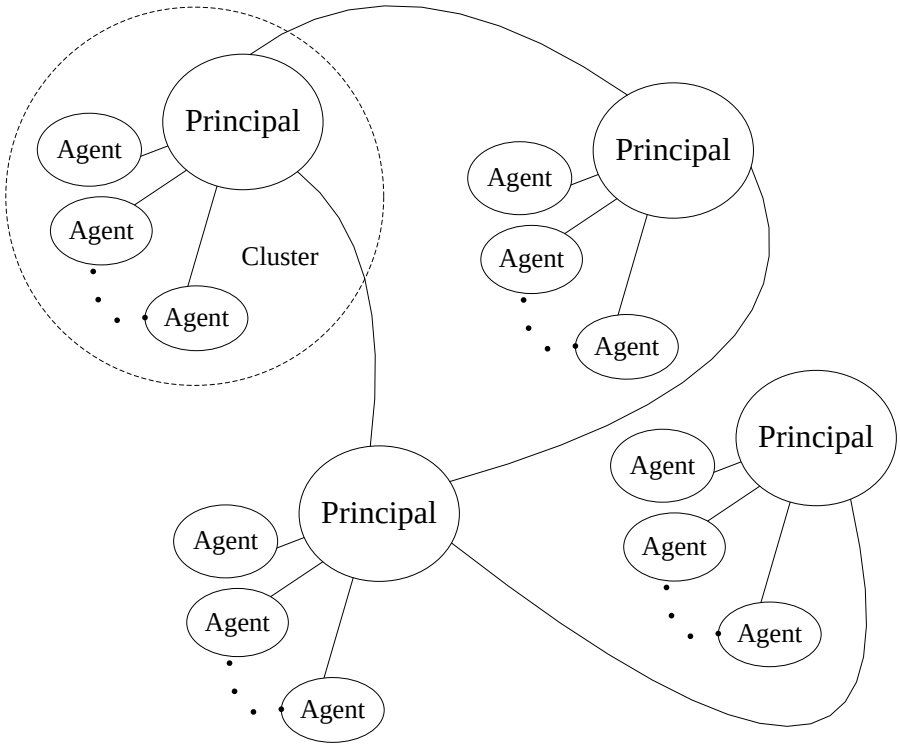

Figure 1: The grid as a federation of resource clusters.

Journal of Grid Computing, vol. 4, no. 4, 2006, pp. 373-394.
The original publication is available at www.springerlink.com

Figure 2: Cluster Infrastructure.

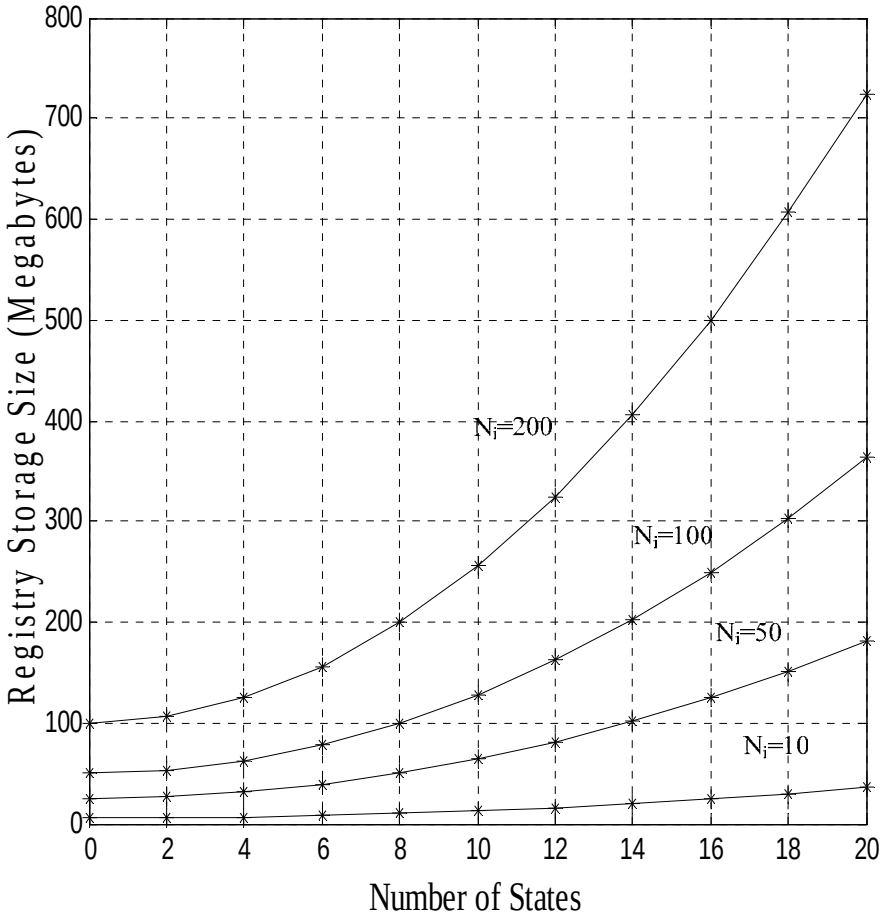

Figure 3: Storage size of the service registry versus the number of capacity states.

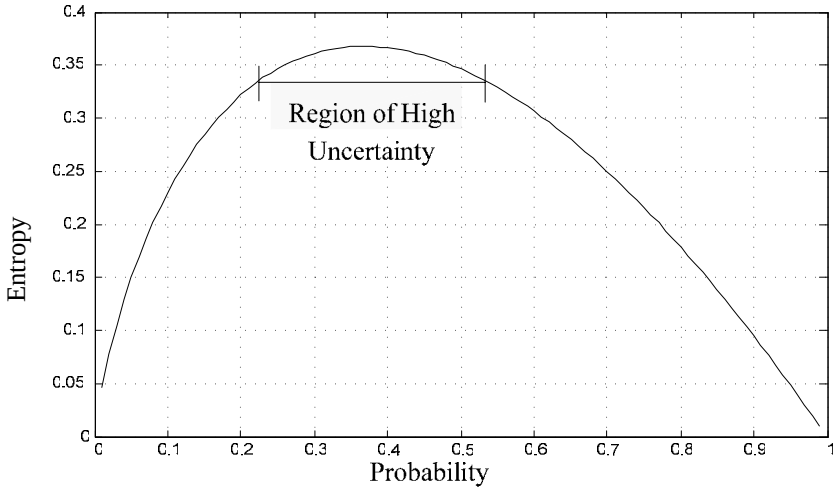

Figure 4: Profile of the Entropy function.

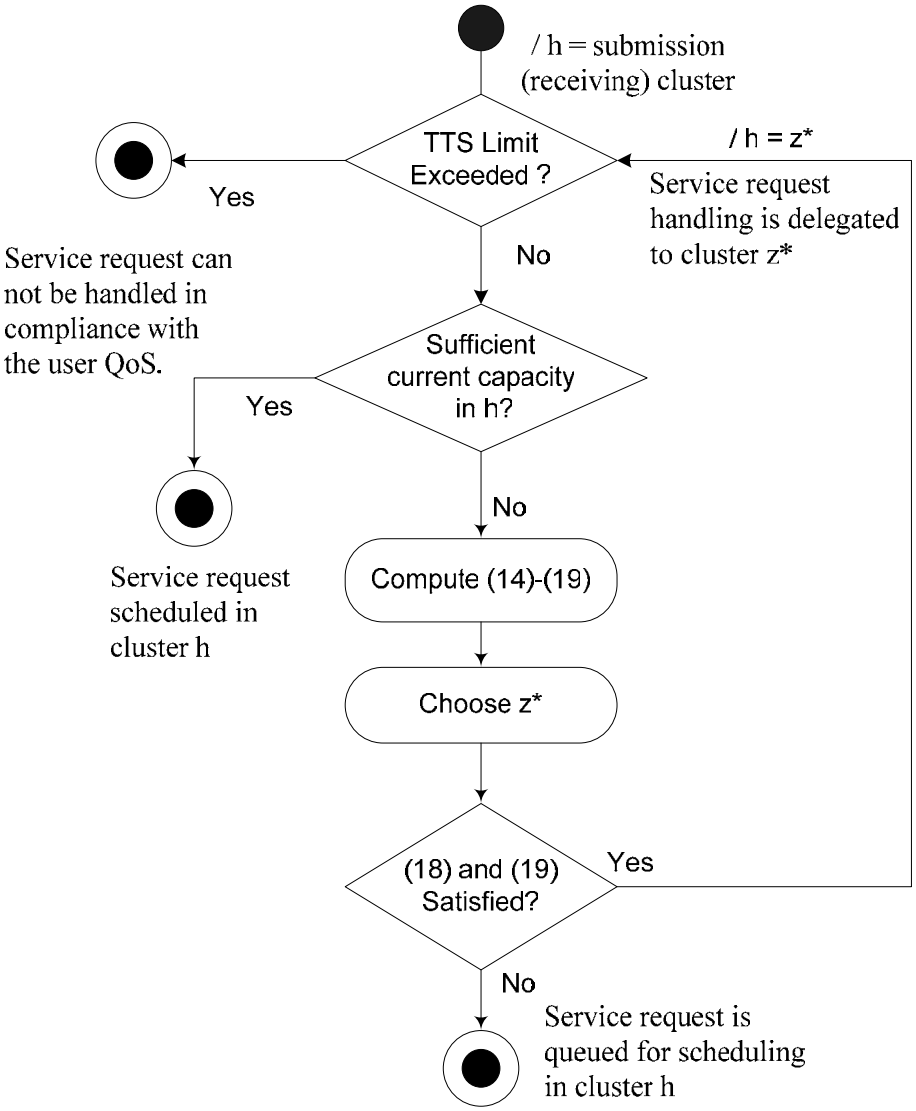

Figure 5: Summary of the Entropic grid scheduling heuristic.

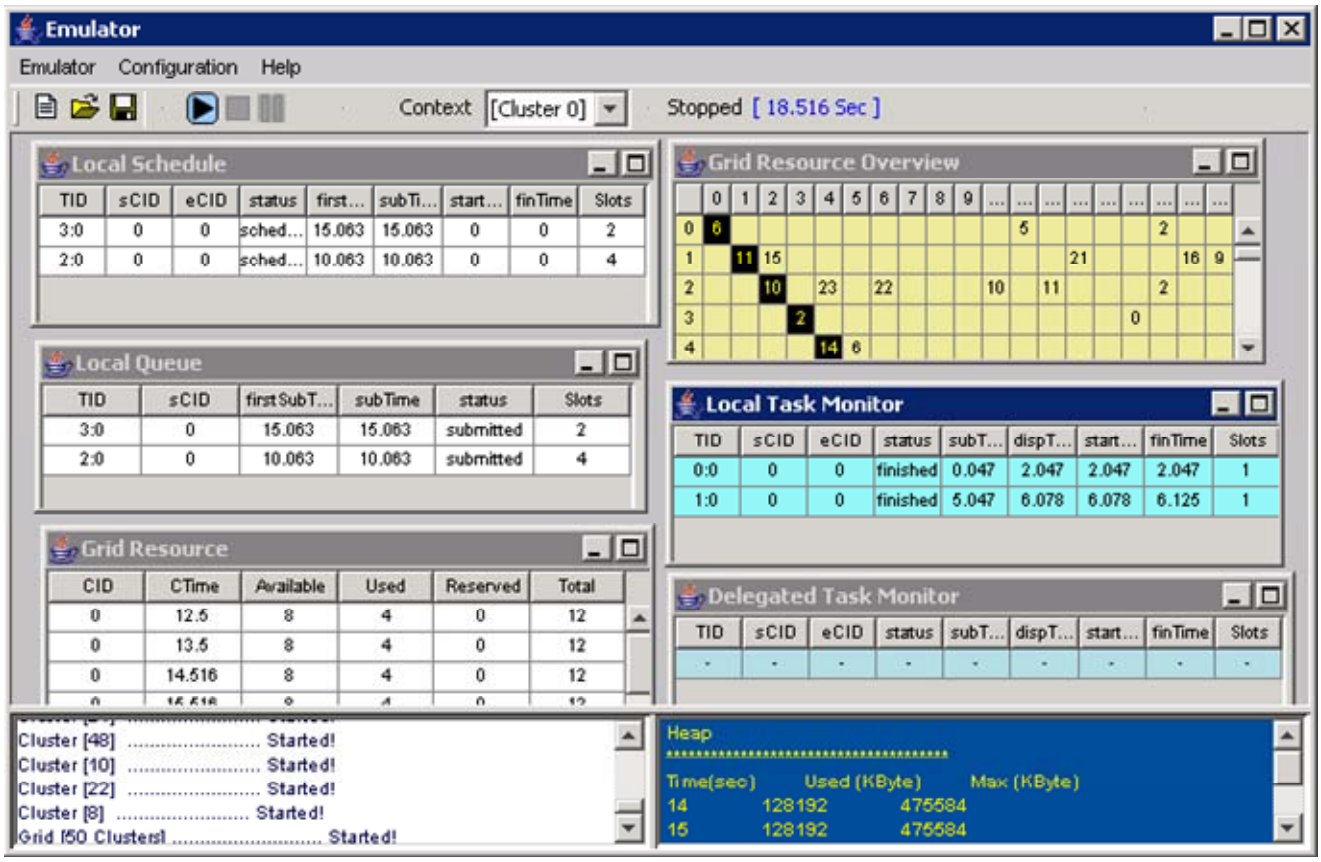

Figure 6: The Midland Grid Emulator.

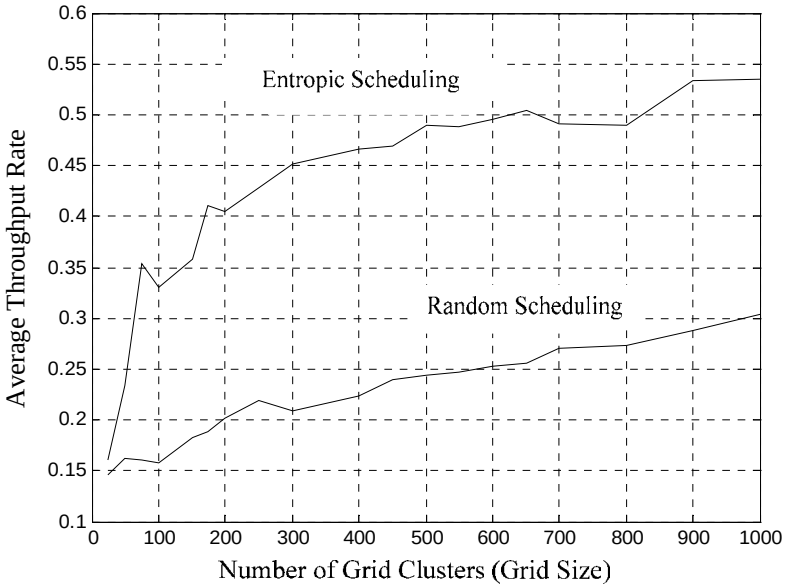

Figure 7: Average throughput rate as a function of the grid size for the random delegation and entropic scheduling strategies respectively.

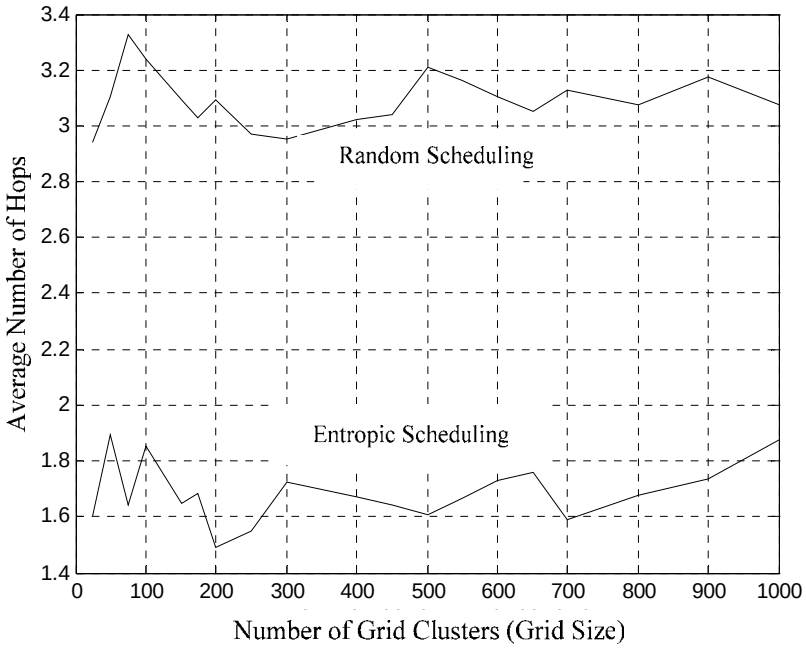

Figure 8: Average number of hops as a function of the grid size for the random delegation and entropic scheduling strategies respectively.

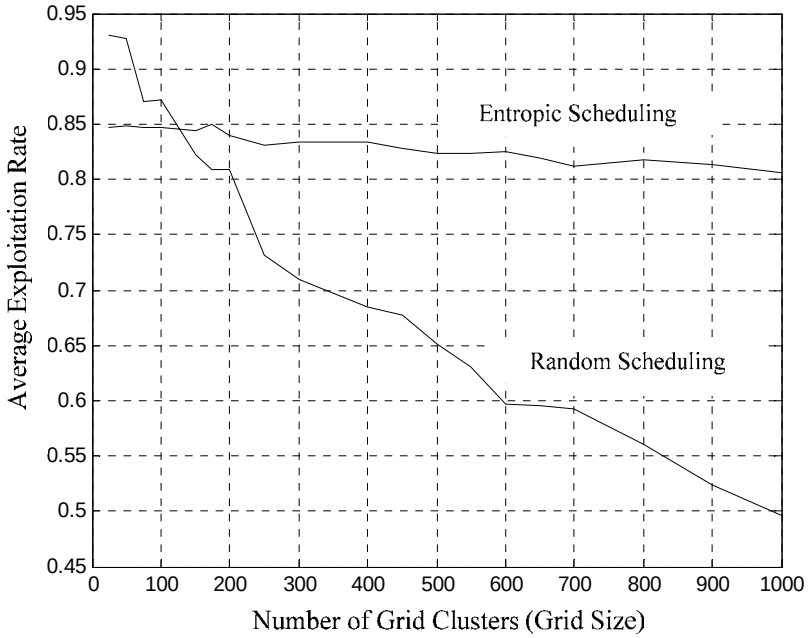

Figure 9: Average exploitation as a function of the grid size for the random delegation and entropic scheduling strategies respectively.

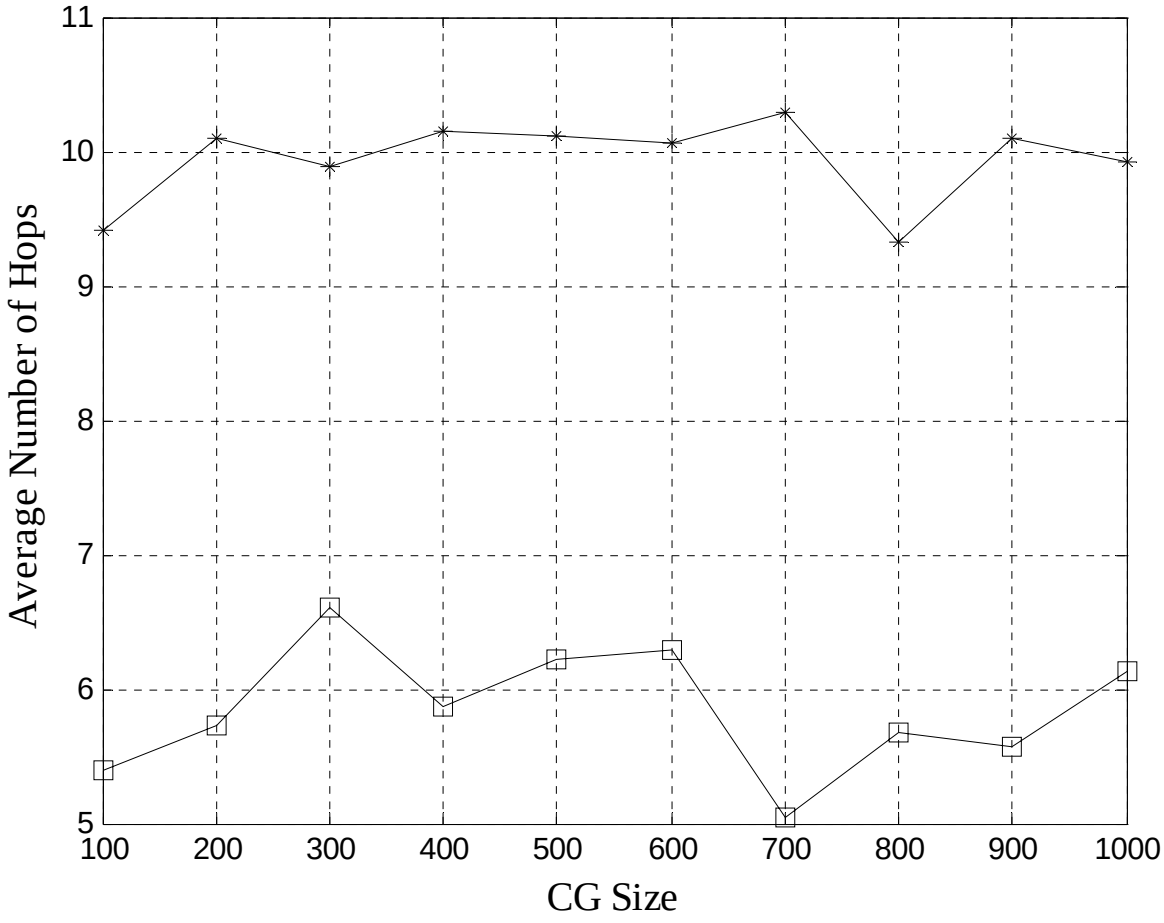

Figure 10: Average number of hops as a function of the grid size for the ALS (*) and entropic scheduling strategies respectively in the case of unbalanced grid resource distribution.

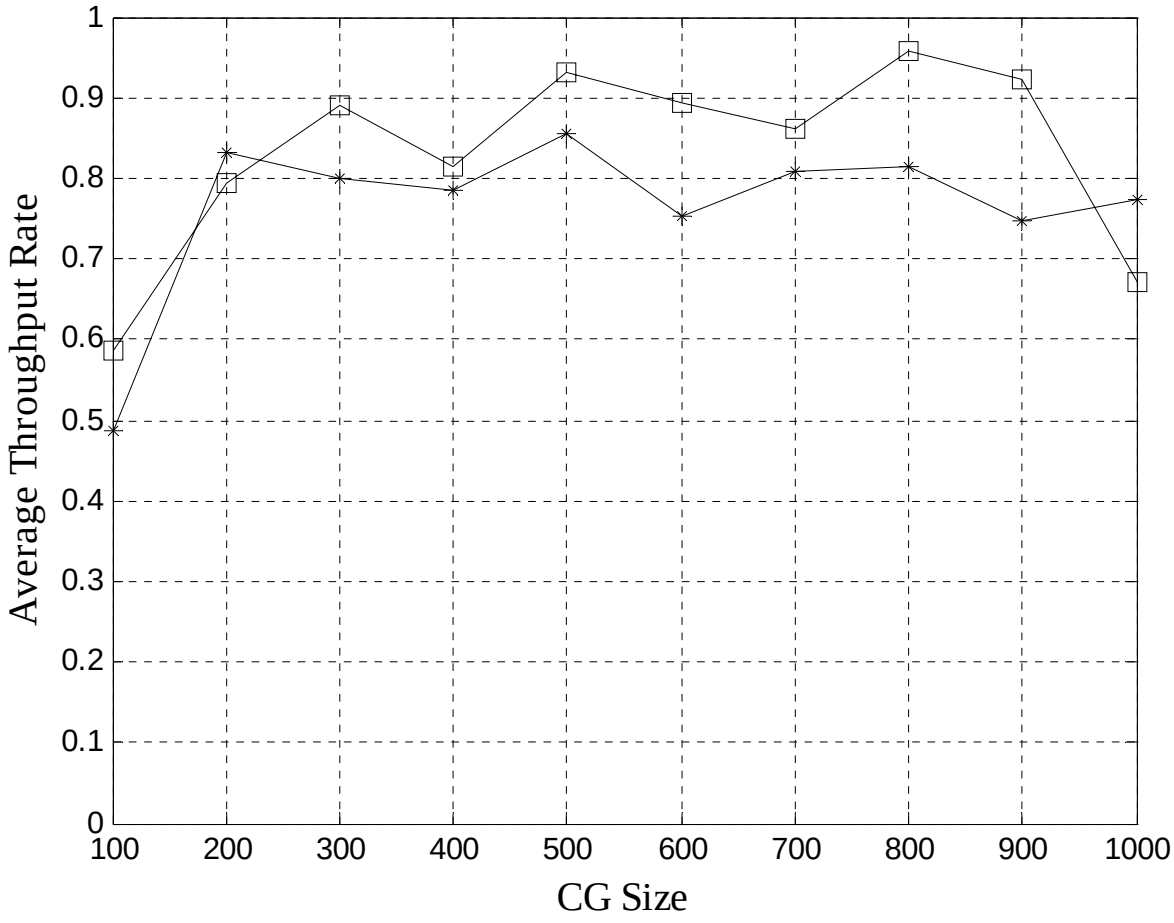

Figure 11: Average throughput rate as a function of the grid size for the ALS (*) and entropic scheduling strategies respectively in the case of unbalanced grid resource distribution.

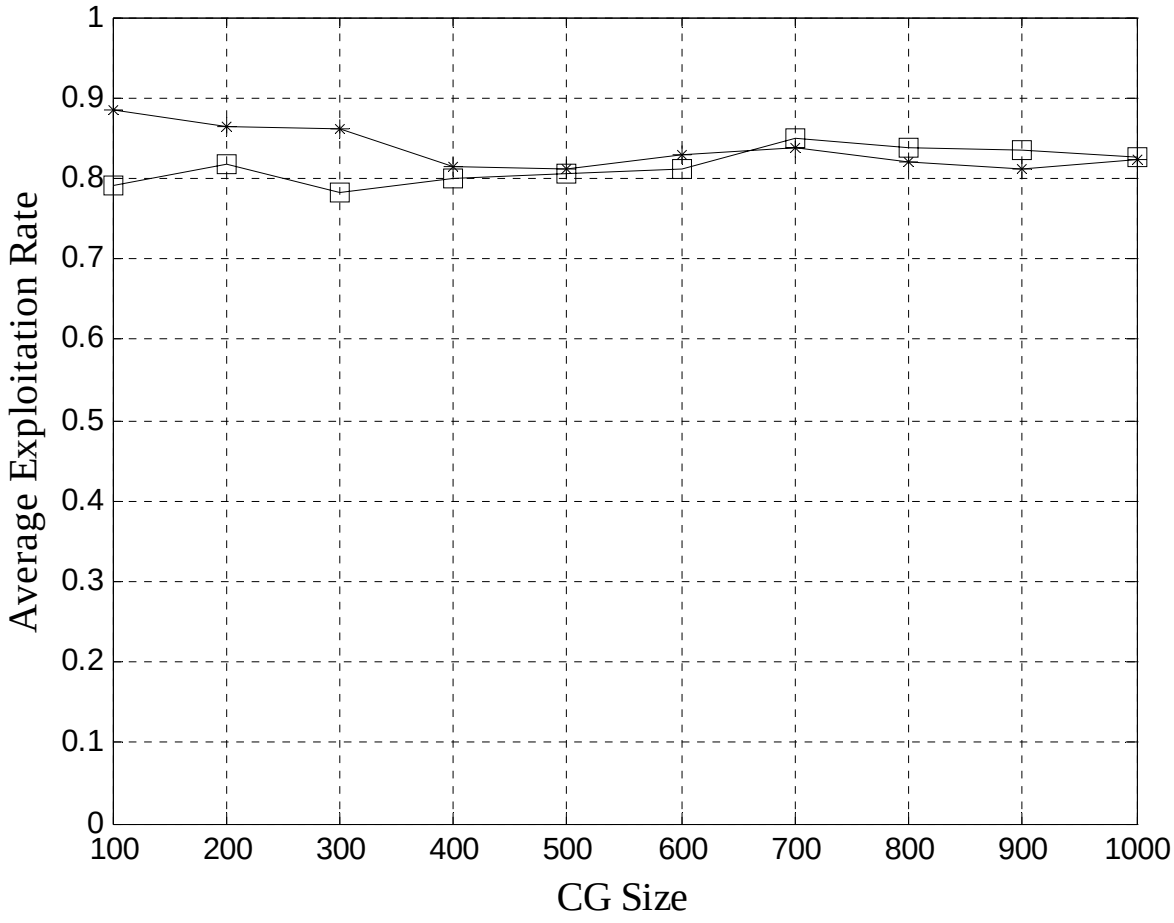

Figure 12: Average exploitation as a function of the grid size for the ALS (*) and entropic scheduling strategies respectively in the case of unbalanced grid resource distribution.

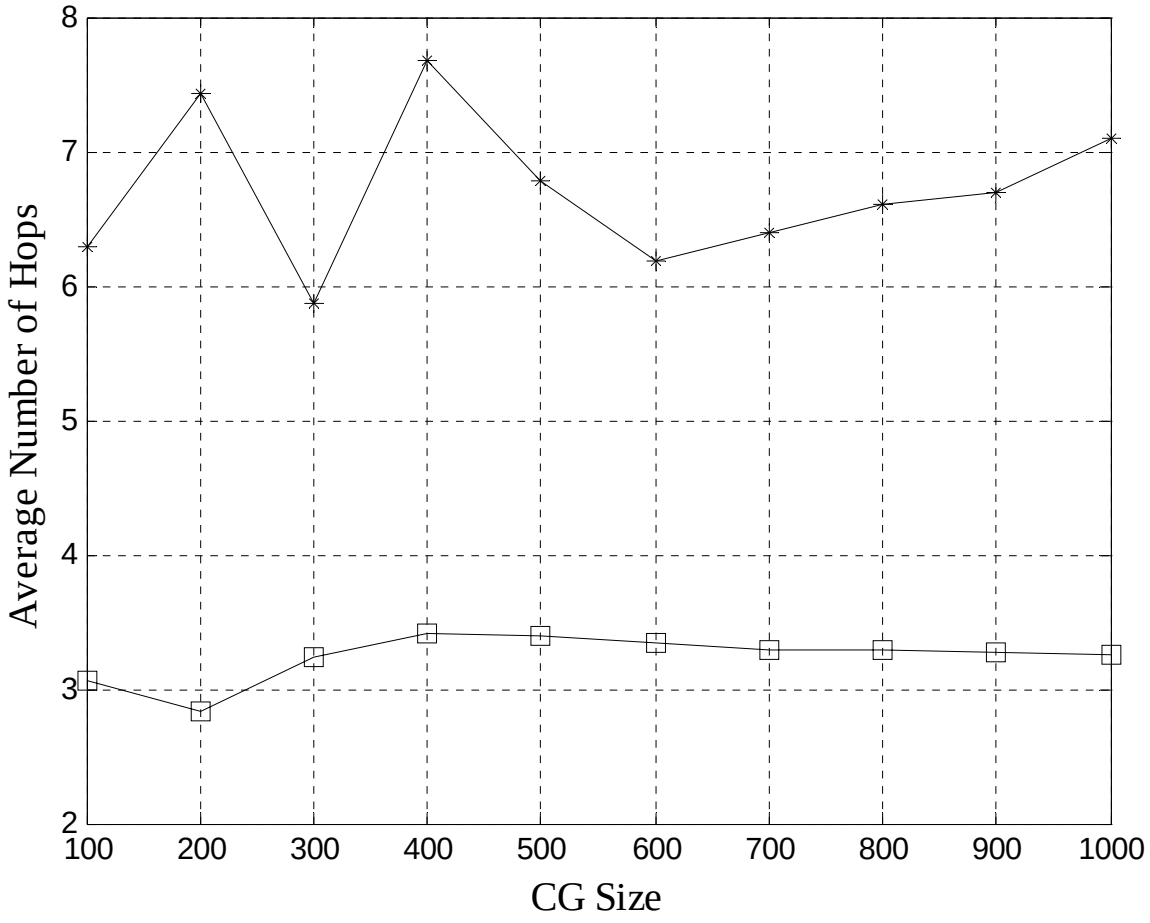

Figure 13: Average number of hops as a function of the grid size for the ALS (*) and entropic scheduling strategies respectively in the case of balanced grid resource distribution.

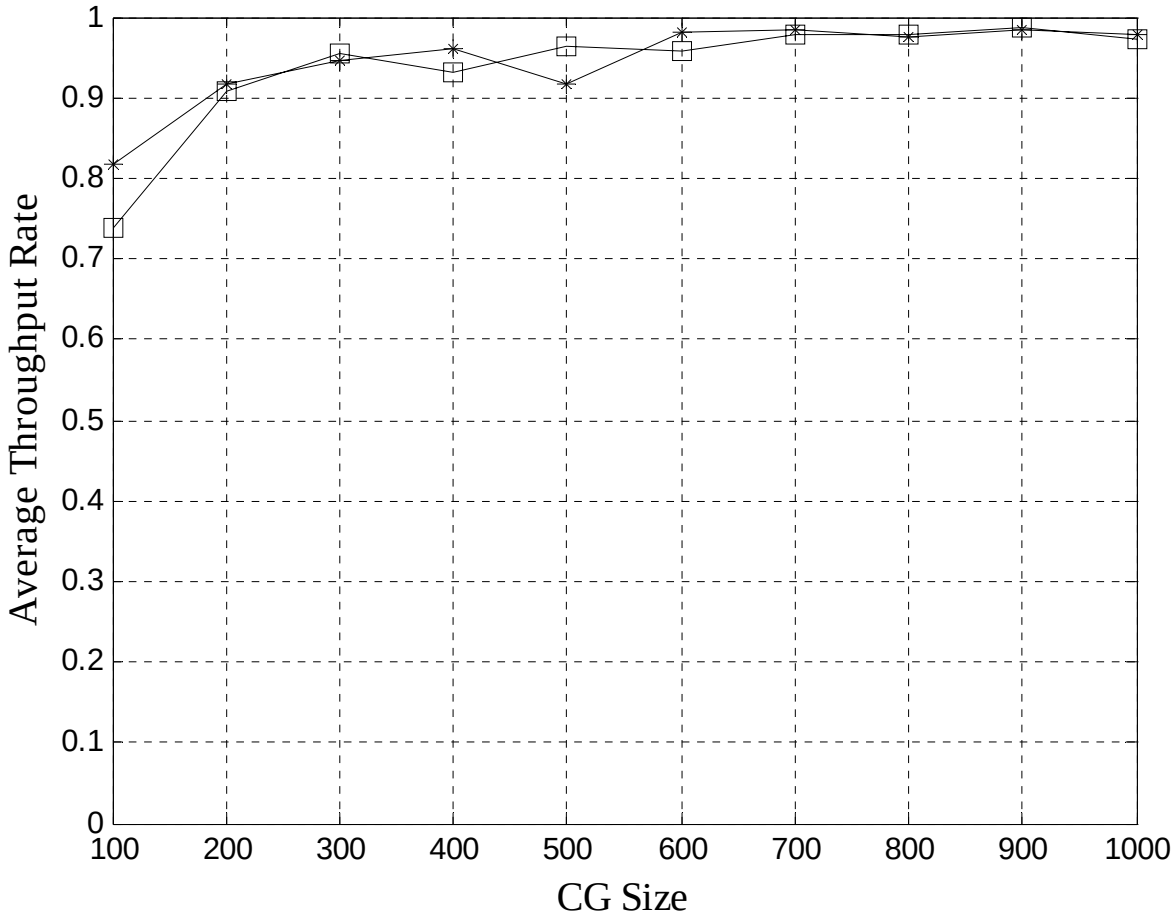

Figure 14: Average throughput rate as a function of the grid size for the ALS(*) and entropic scheduling strategies respectively in the case of balanced grid resource distribution.

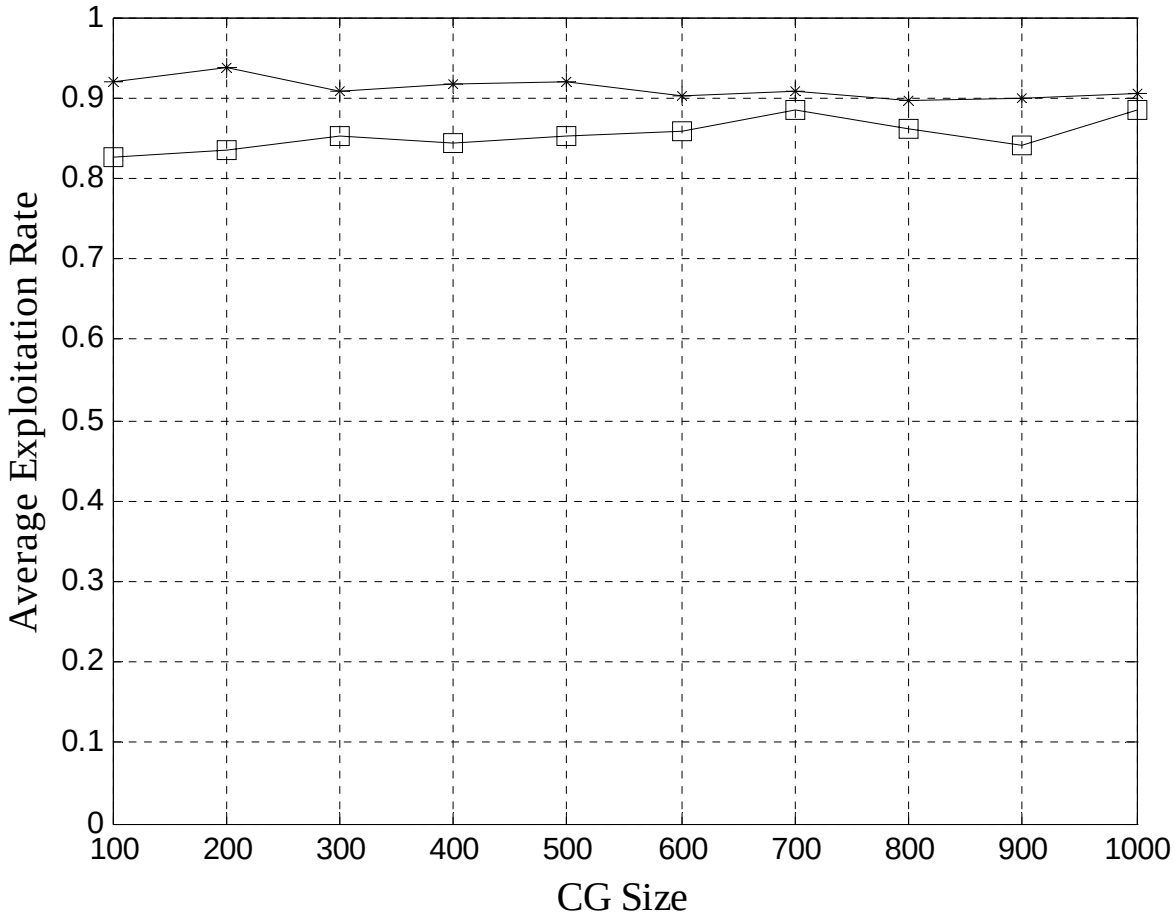

Figure 15: Average exploitation as a function of the grid size for the ALS (*) and entropic scheduling strategies respectively in the case of balanced grid resource distribution.

Acknowledgments

I am grateful to Professors Aziz Guergachi and Ojelanki Ngenyama for gracefully sharing the computing resources used to run the simulations reported in this paper.

References

[1] The Economist, "One grid to rule them all", *The Economist*, vol. 373, pp. 94, 2004.
[2] J. Gustafson, "Program of grand challenge problems: expectations and results", in *Aizu International Symposium on Parallel Algorithms/Architecture Synthesis*, 1997, pp. 2-7.
[3] R. Buyya, K. Branson, J. Giddy, and D. Abramson, "The Virtual Laboratory: A toolset to enable distributed molecular modelling for drug design on the world-wide grid", *Concurrency Computation Practice and Experience*, vol. 15, pp. 1-25, 2003.
[4] E. Tantoso, H. A. Wahab, and H. Y. Chan, "Molecular docking: An example of grid enabled applications", *New Generation Computing*, vol. 22, pp. 189-190, 2004.
[5] I. Ahmad and Y.-K. Kwok, "On parallelizing the multiprocessor scheduling problem", *IEEE Transactions on Parallel and Distributed Systems*, vol. 10, pp. 414-432, 1999.
[6] I. Foster and C. Kesselman, *The grid: blueprint for a new computing infrastructure*. Elsevier Science:San Francisco, 2004.
[7] T. L. Casavant and J. G. Kuhl, "A Taxonomy of Scheduling in General-Purpose Distributed Computing Systems", *IEEE Transactions on Software Engineering*, vol. 14, pp. 141-155, 1988.
[8] R. Buyya, *High performance cluster computing*. Prentice Hall PTR:Upper Saddle River, N.J., 1999.
[9] T. D. Braun, H. J. Siegel, N. Beck, L. Boloni, M. Maheswaran, A. I. Reuther, J. P. Robertson, M. D. Theys, and B. Yao, "Taxonomy for describing matching and scheduling heuristics for mixed-machine heterogeneous computing systems", in *Proceedings of the IEEE Symposium on Reliable Distributed Systems*, 1998, pp. 330-335.
[10] M. A. Al-Mouhamed, "Lower bound on the number of processors and time for scheduling precedence graphs with communication costs", *IEEE Transactions on Software Engineering*, vol. 16, pp. 1317-1322, 1990.
[11] H. El-Rewini, T. G. Lewis, and H. H. Ali, *Task scheduling in parallel and distributed systems*. Prentice Hall:Englewood Cliffs, N.J., 1994.
[12] J.-J. Hwang, Y.-C. Chow, F. D. Anger, and C.-Y. Lee, "Scheduling precedence graphs in systems with interprocessor communication times", *SIAM Journal on Computing*, vol. 18, pp. 244-257, 1989.
[13] M. Cosnard and M. Loi, "Automatic Task Graph Generation Techniques", *Prallel Processing Letters*, vol. 54, pp. 527-538, 1995.
[14] Y.-K. Kwok and I. Ahmad, "Benchmarking the task graph scheduling algorithms", in *Proceedings of the International Parallel Processing Symposium, IPPS*, 1998, pp. 531.
[15] Y.-K. Kwok and I. Ahmad, "Static Scheduling Algorithms for Allocating Directed Task Graphs to Multiprocessors", *ACM Computing Surveys*, vol. 31, pp. 406-471, 1999.
[16] Y.-K. Kwok and I. Ahmad, "Dynamic critical-path scheduling: an effective technique for allocating task graphs to multiprocessors", *IEEE Transactions on Parallel and Distributed Systems*, vol. 7, pp. 506-521, 1996.
[17] I. Ahmad and Y.-K. Kwok, "Parallel Program Scheduling Techniques", in R. Buyya (eds), *High Performance Cluster Computing*, Prentice Hall PTR:New Jersey, pp. 553-578,1999.
[18] R. Gary and D. Johnson, *Computers and Intractability - A Guide to the Theory of NP-Completeness*. Freeman:San Francisco, 1979.

[19] X. He, X. Sun, and G. Von Laszewski, "QoS guided Min-Min heuristic for grid task scheduling", *Journal of Computer Science and Technology*, vol. 18, pp. 442-451, 2003.

[20] F. Berman, R. Wolski, H. Casanova, W. Cirne, H. Dail, M. Faerman, S. Figueira, J. Hayes, G. Obertelli, J. Schopf, G. Shao, S. Smallen, N. Spring, A. Su, and D. Zagorodnov, "Adaptive computing on the grid using AppLeS", *IEEE Transactions on Parallel and Distributed Systems*, vol. 14, pp. 369-382, 2003.

[21] C. Weng and X. Lu, "Heuristic scheduling for bag-of-tasks applications in combination with QoS in the computational grid", *Future Generation Computer Systems*, vol. 21, pp. 271-280, 2005.

[22] F. Berman, H. Casanova, A. Chien, K. Cooper, H. Dail, A. Dasgupta, W. Deng, J. Dongarra, L. Johnsson, K. Kennedy, C. Koelbel, B. Liu, X. Liu, A. Mandal, G. Marin, M. Mazina, J. Mellor-Crummey, C. Mendes, A. Olugbile, M. Patel, D. Reed, Z. Shi, O. Sievert, H. Xia, and A. Yarkhan, "New grid scheduling and rescheduling methods in the GrADS project", *International Journal of Parallel Programming*, vol. 33, pp. 209-229, 2005.

[23] H. Nakada, M. Sato, and S. Sekiguchi, "Design and implementations of Ninf: towards a global computing infrastructure", *Future Generation Computer Systems*, vol. 15, pp. 649-658, 1999.

[24] J. Cao, S. A. Jarvis, S. Saini, D. J. Kerbyson, and G. R. Nudd, "ARMS: An agent-based resource management system for grid computing", *Scientific Programming*, vol. 10, pp. 135-148, 2002.

[25] X.-H. Sun and M. Wu, "Grid Harvest Service: a system for long-term, application-level task scheduling", in *Parallel and Distributed Processing Symposium*, 2003, pp. 25-33.

[26] Y. Gao, H. Rong, and J. Z. Huang, "Adaptive grid job scheduling with genetic algorithms", *Future Generation Computer Systems*, vol. 21, pp. 151-161, 2005.

[27] J. Cao, D. P. Spooner, S. A. Jarvis, and G. R. Nudd, "Grid load balancing using intelligent agents", *Future Generation Computer Systems*, vol. 21, pp. 135-149, 2005.

[28] D. P. Spooner, S. A. Jarvis, J. Cao, S. Saini, and G. R. Nudd, "Local grid scheduling techniques using performance prediction", *IEE Proceedings: Computers and Digital Techniques*, vol. 150, pp. 87-96, 2003.

[29] L. Yang, J. M. Schopf, and I. Foster, "Conservative Scheduling: Using Predicted Variance to Improve Scheduling Decisions in Dynamic Environments", in *Proceedings of the 2003 ACM/IEEE conference on Supercomputing*, 2003, pp. 31-47.

[30] N. Krothapalli and A. V. Deshmukh, "Dynamic allocation of communicating tasks in computational grids", *IIE Transactions (Institute of Industrial Engineers)*, vol. 36, pp. 1037-4053, 2004.

[31] K. Krauter, R. Buyya, and M. Maheswaran, "A taxonomy and survey of grid resource management systems for distributed computing", *Software - Practice and Experience*, vol. 32, pp. 135-164, 2002.

[32] M. Maheswaran, S. Ali, H. J. Siegel, D. Hensgen, and R. F. Freund, "Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing systems", *Proceedings of the Heterogeneous Computing Workshop, HCW*, pp. 30-44, 1999.

[33] V. Hamscher, U. Schwiegelshohn, A. Streit, and R. Yahyapour, "Evaluation of Job-Scheduling Strategies for Grid Computing", in *Proceedings of the 1st IEEE/ACM International Workshop on Grid Computing (Grid 2000)*, 2000, pp. 191-202.

[34] L. Adzigogov, J. Soldatos, and L. Polymenakos, "EMPEROR: An OGSA Grid Meta-Scheduler Based on Dynamic Resource Predictions", *Journal of Grid Computing*, vol. 3, pp. 19-37, 2005.

[35] S. Sanyal and S. K. Das, "MaTCH: Mapping Data-Parallel Tasks on a Heterogeneous Computing Platform Using the Cross Entropy Heuristic", in *Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium*, 2005, pp. 64-74.

[36] H. Casanova, T. M. Bartol, J. Stiles, and F. Berman, "Distributing MCell simulations on the grid", *International Journal of High Performance Computing Applications*, vol. 15, pp. 243-257, 2001.
[37] L. Yang, I. Foster, and J. M. Schopf, "Homeostatic and Tendency-based CPU Load Predictions", in *Procedeeings of the Parallel and Distributed Processing Symposium*, 2003, pp.
[38] L. He, S. A. Jarvis, D. P. Spooner, X. Chen, and G. R. Nudd, "Hybrid performance-based workload management for multiclusters and grids", *IEE Proceedings: Software*, vol. 151, pp. 224-231, 2004.
[39] A. Goldman and C. Queiroz, "A model for parallel job scheduling on dynamical computer Grids", *Concurrency Computation Practice and Experience*, vol. 16, pp. 461-468, 2004.
[40] D. Abramson, R. Buyya, and J. Giddy, "A computational economy for grid computing and its implementation in the Nimrod-G resource broker", *Future Generation Computer Systems*, vol. 18, pp. 1061-1074, 2002.
[41] L. Chunlin and L. Layuan, "A distributed utility-based two level market solution for optimal resource scheduling in computational grid", *Parallel Computing*, vol. 31, pp. 332-351, 2005.
[42] R. Buyya, D. Abramson, and S. Venugopal, "The grid economy", *Proceedings of the IEEE*, vol. 93, pp. 698-714, 2005.
[43] R. Buyya, D. Abramson, J. Giddy, and H. Stockinger, "Economic models for resource management and scheduling in grid computing", *Concurrency Computation Practice and Experience*, vol. 14, pp. 1507-1542, 2002.
[44] K. Czajkowski, I. Foster, and C. Kesselman, "Agreement-based resource management", *Proceedings of the IEEE*, vol. 93, pp. 631-643, 2005.
[45] W. Smith, I. Foster, and V. Taylor, "Scheduling with advanced reservations", *Proceedings of the International Parallel Processing Symposium, IPPS*, pp. 127-132, 2000.
[46] A. S. McGough, A. Afzal, J. Darlington, N. Furmento, A. Mayer, and L. Young, "Making the grid predictable through reservations and performance modelling", *Computer Journal*, vol. 48, pp. 358-368, 2005.
[47] K. Czajkowski, S. Fitzgerald, I. Foster, and C. Kesselman, "Grid information services for distributed resource sharing", in *IEEE International Symposium on High Performance Distributed Computing, Proceedings*, 2001, pp. 181-194.
[48] S. Tuecke, K. Czajkowski, I. Foster, J. Frey, S. Graham, C. Kesselman, T. Maquire, T. Sandholm, D. Snelling, and P. Vanderbilt, "Open Grid Services Infrastructure (OGSI)," Global Grid Forum, 2003.
[49] M. Faloutsos, P. Faloutsos, and C. Faloutsos, "On power-law relationships of the Internet topology", *Computer Communication Review*, vol. 29, pp. 251-262, 1999.
[50] A. Barabási and R. Albert, "Emergence of Scaling in Random Networks", *Science*, vol. 286, pp. 509-512, 1999.
[51] R. Al-Ali, A. Hafid, O. Rana, and D. Walker, "An approach for quality of service adaptation in service-oriented Grids", *Concurrency Computation Practice and Experience*, vol. 16, pp. 401-412, 2004.
[52] M. P. Singh and M. N. Huhns, *Service-Oriented Computing: Semantics, Processes, Agents*. John Wiley & Sons, 2005.
[53] Y. Derbal, "Service oriented grid resource modeling and management", in *Proceedings of the 1st International Conference on Web Information Systems and Technologies (WEBIST 2005)*, Miami, Florida, 2005, pp. 146-153.
[54] S. M. Ross, *Introduction to Probability Models*. Academic Press, Inc.:San Diego, CA, 1989.

[55] X. Zhang and J. M. Schopf, "Performance analysis of the globus toolkit monitoring and discovery service, MDS2", in *Proceedins of the IEEE International Performance, Computing and Communications Conference*, 2004, pp. 843-849.
[56] J. D. Fast, *Entropy. The significance of the concept of entropy and its applications in science and technology. [Translated by M.E. Mulder-Woolcock]*. Macmillan:[London], 1970.
[57] G. N. Saridis, "Analytic formulation of the principle of increasing precision with decreasing intelligence for intelligent machines", *Automatica*, vol. 25, pp. 461-467, 1989.
[58] C. E. Shannon and W. Weaver, *The mathematical theory of communication*. University of Chicago Press:Urbana, 1978.
[59] G. N. Saridis, "Entropy formulation of optimal and adaptive control", *IEEE Transactions on Automatic Control*, vol. 33, pp. 713-721, 1988.
[60] R. C. Conant, "Laws of information which govern systems", *IEEE Transactions on Systems, Man and Cybernetics*, vol. SMC-6, pp. 240-255, 1976.
[61] J. Gray, *Distributed Computing Economics*. Microsoft Research, 2003.