Robust Load Delegation in Service Grid Environments
Alexander Fölling, Christian Grimme, Joachim Lepping, and Alexander Papaspyrou

Abstract—In this paper, we address the problem of finding well-performing workload exchange policies for decentralized Computational Grids using an Evolutionary Fuzzy System. To this end, we establish a non-invasive collaboration model on the Grid layer which requires minimal information about the participating High Performance and High Throughput Computing (HPC/HTC) centers and which leaves the local resource managers completely untouched. In this environment of fully autonomous sites, independent users are assumed to submit their jobs to the Grid middleware layer of their local site, which in turn decides on the delegation and execution either on the local system or on remote sites in a situation-dependent, adaptive way. We find for different scenarios that the exchange policies show good performance characteristics not only with respect to traditional metrics such as average weighted response time and utilization, but also in terms of robustness and stability in changing environments.

Index Terms—Grid Computing, Evolutionary Fuzzy Systems, Online Grid Scheduling, Performance Evaluation

1 INTRODUCTION

Today, the use of Grid Computing is not anymore limited to HPC/HTC-centric communities such as High Energy Physics, Astronomy, or Climate Research, which have a certain tradition of using such infrastructures. Other sciences—e.g. Financial Services, Construction Engineering, and even arts and humanities—also start to adopt Grid Computing as a tool for e-Science, and show an ever-increasing demand for computing power and storage space.

While well-established approaches such as the EGEE environment [1] have relied on centralized middleware infrastructures for whole e-Science communities, other—mostly emerging—efforts, such as the D-Grid Initiative 1, have chosen a Service Grid approach with smaller, more community-tailored Grids. In the latter case, however, a strong demand for enabling collaboration and cooperation on the infrastructure layer between the different communities and Grids can be observed.

A major issue in such collaborations is the possibility of inter-community resource usage: Although most communities run their own data centers, working together in an ad-hoc manner by allowing alien workload to be run on community hardware is still a tedious task and usually requires resorting to 1980s-style command line interfaces and undesirable micro-management. This is mainly due to technical issues: Many e-Science infrastructures show a lack of standardization, and therefore, collaborations between the workload gateways (usually Grid schedulers or brokers) fail on a compatibility level. There are, however, also organizational issues: Each community Grid strives for delivering the highest possible Quality of Service to its own users and, as such, is only interested in participating in joint efforts if they are beneficial for all participants likewise.

This last aspect is an open research problem in the field of Grid scheduling: algorithms for the exchange of workload between different Grid communities—with respect to common performance metrics—have to perform at least as good as in the non-cooperative case. Otherwise, the motivation for participating in a HPC/HTC federation, vanishes quickly, since one of the participating user communities will suffer from the collaboration.

Here, we can identify four important properties for such algorithms:

- **Support for environments with very strict information policies:** Although almost every Grid provides various kinds of information services, data regarding the machines themselves such as their current or overall utilization, average response times, or throughput is often kept confidential due to competition reasons.
- **Strict separation from local resource management systems (LRMS):** Machine owners usually have their own operational policies implemented on their systems and obviously are not willing to cease control over the machines they are obliged to fund.
- **Situation-dependent, adaptive decision-making:** The current state of the system is crucial when deciding on whether to accept or decline foreign workload, e.g. allowing for additional remote jobs if the local system is already highly loaded seems to be inappropriate.
- **Robustness and stability in changing environments:** Even with respect to future, still unknown (and usually unpredictable) job submissions, it is crucial that aspects such as complete site failures or even rogue

---

participants are handled gracefully with respect to the own and overall performance.

In the work at hand, we address these properties using a Fuzzy based approach for job exchange in Computational Grids, where the controller acts depending on the current system state. The states are modeled by Fuzzy sets which are represented by simple membership functions. Although the system states themselves do not contain any uncertainty the whole representation of the system is uncertain as only limited information (see above) is provided. Such Fuzzy System based scheduling techniques have been successfully applied to online scheduling problems before, see for example Franke et al. [2]. They outperform most static scheduling heuristic due to their ability to flexibly adapt decisions to changing environments. As they have proven to be a reliable concept to tackle challenging online scheduling problems, we decide to also apply them in the Grid context.

In order to establish good rules for the Fuzzy System, we furthermore use evolutionary algorithms for finding parameterization of the Fuzzy membership functions. This approach is especially suitable because of the possibility to find a simple and efficient encoding of the whole controller. This combination of Fuzzy Systems and evolutionary algorithm is commonly denoted as Evolutionary Fuzzy Systems, see Cordón et al. [3]. We show that our approach, while respecting the aforementioned requirements for Grid scheduling algorithms, shows adequate performance characteristics in real setups. Note that we do not address the scheduling of data-intensive applications, where location of data is a significant factor at the moment. Similarly, we assume no scheduling of workflows an consider only independent jobs. However, the here presented concept can easily be augmented with existing data schedulers or workflow scheduling approaches.

The remainder of the paper is organized as follows: In Section 2, we establish the basis for understanding our model, algorithm, and optimization. We then introduce our system model in Section 3 and our Fuzzy Grid Scheduling approach in Section 4. After discussing tools for performance measurement in Section 5, we depict the evolutionary learning of rule sets in Section 6. Next, we evaluate our approach with respect to adaptiveness in a Grid federation in Section 7 and robustness in unknown environments in Section 8. Along with a review of the current state of the art in Section 9, we conclude our work in Section 10.

2 Background

This section briefly introduces to the basic of job scheduling on Massively Parallel Processing (MPP) systems, Evolutionary Fuzzy Systems, and evolutionary algorithms. These definitions and tools are applied throughout the paper to easily describe the used Grid architecture as well as the proposed approach for realizing job migration.

2.1 Job Scheduling for MPP Systems

The scheduling of MPP systems is an online problem as jobs are submitted over time and the precise processing times of those jobs is unknown in advance. Furthermore, information about future jobs are not available. We assume independent rigid parallel batch jobs for our analysis, which are dominant on most parallel computer systems. Those jobs are neither moldable nor malleable and require concurrent and exclusive access to the requested resources. Formally, each job \( j \) is characterized by its degree of parallelism \( m_j \) and its processing time \( p_j \). Although many additional criteria are conceivable, see Feitelson et al. [4], we restrict ourselves to only those two required job properties.

During the execution phase, job \( j \) requires the concurrent and exclusive access to \( m_j \leq m_k \) processing nodes with \( m_k \) being the total number of nodes on the MPP system at site \( k \). The number of required processing nodes \( m_j \) is available at the release date \( r_j \) of job \( j \) and does not change during the execution. As the network does not favor any subset of the nodes and all nodes of a parallel computer system are either identical or very similar, we assume that a job \( j \) can be processed on any subset of \( m_j \) nodes of the system.

Further, most current real installations of parallel computers do not use preemption but let all jobs run to completion. The completion time of job \( j \) within the schedule \( S \) is denoted by \( C_j(S) \).

2.2 Evolutionary Algorithms

Optimization algorithms that mimic the natural process of Darwinian evolution are widespread in computer science and often applied for parameter optimization when the fitness landscape of the optimization problem is unknown.

A specific type of these algorithms—Evolution Strategies [5]—operate on a population of \( \mu \) individuals, where each individual represents a real-coded solution to the given optimization problem. These approaches apply variation operators like mutation (a random change in genome) and recombination (combining two or more parent individuals’ genomes) to breed \( \lambda \) offspring individuals from those \( \mu \) parental individuals, followed by a global selection process in which the individuals compete against each other to form the new \( \mu \) parents for the next generation. The above described evolutionary loop is executed until a given termination criterion, like a fixed number of generations or a quality level within the objective space, is satisfied. Two versions of Evolution Strategies are distinguishable: In the \( (\mu, \lambda) \)-strategy, the next parent generation is selected just from the offspring individuals while the \( (\mu + \lambda) \)-strategy selects the best individuals of both the parent and offspring generations. All other individuals are removed from the system and the next loop iteration starts.
2.3 Evolutionary Fuzzy Systems

Since their conceptualization in the early 1960ies Fuzzy Systems have been widely and successfully applied to various areas like for example control systems or classification. Especially in control systems, they are particularly suited for the representation of problem specific knowledge, as imprecision or vague descriptions are common properties of expertise. Currently, many decision making methods (e.g. in the fields of resource management or robot behavior) solve problems in a heuristic fashion. They give advice for actions in certain—often fuzzy described—situations that have turned out to be profitable with respect to a given objective. Such a collection of situation-dependent expertise is called a knowledge base.

There are several advantages to represent a knowledge base by Fuzzy logic within a Fuzzy System: The interpolative nature of Fuzzy Systems has the ability to express partial and concurrent activations of behaviors and gradual transitions between them. Further, the behavior can be conveniently synthesized by a set of IF-THEN rules using linguistic terms to encode the expert knowledge. Finally, due to its approximate reasoning capabilities, Fuzzy logic produces controllers that are robust to uncertainty and imprecision. Especially, the latter property is of great importance for the problem addressed in this paper, as we aim to produce robust exchange mechanisms within changing environments.

However, one of the major drawbacks of classic Fuzzy Systems is their missing learning ability. They always require a existing knowledge base that has to be derived from experts knowledge which is often called training data. In many cases, that data is not available and the design of Fuzzy Systems is not possible at all. Also for the problem at hand we cannot revert to any kind of training data. Therefore, we employ an evolutionary learning process to automate the Fuzzy System design.

Evolutionary Fuzzy Systems are Fuzzy Systems derived and optimized by an evolutionary learning process. For these systems an evolutionary algorithm is employed to learn or tune different components. They are always applied, if neither expert knowledge nor training data is available or cannot be transformed directly into corresponding rules. Those algorithms do not require particular knowledge about the problem structure and can be applied to various systems.

3 System Model

The problem of job distribution between federated computer clusters has been continuously studied since the emergence of Grid computing in the beginning of the 1990ies. Early approaches favor a hierarchical scheduling structure, where a central scheduler instance—often called Meta-Scheduler, Grid Scheduler, or Broker—delegates submitted jobs to subordinated partner sites [6]. The most profound problem of this scheduling structure is its bad fault-tolerance and lack of scalability.

Such a centralized model implies full knowledge of the Grid sites’ state and exclusive control over the local resource management system (LRMS) to facilitate efficient job scheduling. The local user community is often not allowed to bypass the central scheduler for submitting jobs.

The afore described approach is contrasted by a decentralized structure, in which local sites can act as autonomous peers and share jobs in an equitable fashion. Thus, the process of job interchange is deferred to the competence of each local site scheduler and puts special emphasis on the decision making process of accepting remote and dispensing local jobs.

With respect to the basic parameters of modern e-Infrastructures regarding organizational autonomy and equity, we assume our Computational Grid as a loose cooperation between different HPC centers—further referred to as sites—and consider Massively Parallel Processing (MPP) systems as their basic entities. For every MPP entity we assume an own local user demand for computational resources which is reflected by the sites’ originating workload. This includes the submission characteristics, but also the adaptation of the submitted jobs’ resource demand to the local configuration. This scenario is based on the perception that, as a general rule, Grid environments are not build from scratch, but emerge from collaborations between different organizational domains, each of which already operating one or more MPP systems for internal purposes, in order to serve a prescribed, project-driven community of users.

Although Kee et al. [7] present an analysis of existing resource configurations and proposes a Grid platform generator that synthesizes realistic configurations of both computing and communication resources, we follow a much simpler model for our first analysis. Existing models can be used to test the here derived strategies in a larger context, but for the first development we rather assume interconnected parallel computers as Grid entities.

More formally, a Computational Grid consists here of $|K|$ independent sites. Each site $k \in K$ is modeled by $m_k$ parallel processors which are identical such that a parallel job can be allocated on any subset of these machines. Note that we do not assume heterogeneous machines as such a model would require, that some Job dependent speed-up factor must be added. It is widely agreed that a simple linear speed-up model does not fit reality as the speed-up is dependent on both the processor architecture and the structure of a program. It would be therefore necessary to introduce benchmarks that cover most of the submitted applications, but as no details about the programs are included in the workload traces this is not applicable. As long as there is no comprehensive speed-up model available it is better to assume all machines with approximately the same speed.
Further, splitting jobs over multiple sites (multi-site computation) is not allowed. Moreover, we assume that all sites only differ in the number of available processors, but not in their speed: As we focus on the job exchange algorithms, the differences in execution speeds can be neglected, see Schwiegelshohn et al. [8].

The workload management within the infrastructure is conducted by a two-tier middleware, see Figure 1, comprising a Local Resource Management System (LRMS) and a Grid Resource Management System (GRMS) on each site. While the LRMS takes care of assigning workload to resources for the local site only, the GRMS decides on the delegation of jobs from and to the site. Users submit their workload to the local site in the same manner as on classic LRMS systems; a small submission component intercepts those and forwards them to the local GRMS for further inspection.

That is, jobs that are submitted to the local site scheduler may not be accepted for execution elsewhere because of their resource demand being oversized for some or all of the other sites. Ignoring the inter-site collaboration for a moment, we describe the local scheduling problem on MPP systems in the next paragraph.

3.1 LRMS Layer

The Local Resource Management System (LRMS) layer consists of a waiting queue and a scheduler. The waiting queue stores all locally submitted jobs while the scheduler executes a specific scheduling strategy in order to assign jobs from the waiting queue onto the available local resources. On MPP system layer, this approach allows the realization of priorities for jobs of different user groups. Usually, the scheduling strategies are formulated by the system provider to fulfill the users’ needs. Although many special-purpose algorithms exist that are tailored for certain MPP system owner priorities, we use the basic and simple First-Come-First-Serve (FCFS) algorithm as an example on LRMS. This heuristic starts the first job of the waiting queue whenever enough idle resources are available. Despite the very low utilization that is produced in the worst case this heuristic works well in practice [9]. In general, our methodology is not restricted to any special LRMS scheduling algorithm and can be trained for arbitrary locally customized systems. Here, FCFS serves only as a simple and still widely used example that can easily be substituted by backfilling alternatives or even more advanced strategies.

In the remainder of this paper, we assume a LRMS with one dedicated waiting queue for all incoming jobs and FCFS as local scheduling algorithm. An advantage, besides its fairness and easy implementation, is the sparse amount of required information. As we develop Grid job exchange policies within information restrictive scenarios (e.g. no runtime estimations etc.), an LRMS algorithm is required that works fine with a very basic job description. Thus, we chose FCFS as it solely requires a job’s degree of parallelism to make its scheduling decision. However note that our general methodology for learning of rule based exchange policies, see Section 4, is not restricted to any special LRMS algorithm. This concept can be applied with arbitrary LRMS configuration and FCFS serves only as a most common example.

3.2 GRMS Layer

The Grid Scheduling Resource Management System (GRMS) extends every site by an additional layer on top of the LRMS, see Figure 1. The GRMS accepts locally submitted jobs on behalf of the underlying LRMS. The actual exchange behavior is realized exclusively by the GRMS and due to this strict layered architecture the LRMS is kept completely unmodified. Both removal of jobs from LRMS queues as well as any kind of intervention in the local scheduling process is prohibited. Furthermore, the GRMS is transparent to local users and the LRMS. From the users point of view, all submitted jobs are executed on the local site, whereas each LRMS considers every job as a locally submitted independent of its origin. Decisions about a job’s delegation to another GRMS or local scheduling is made by a deployed exchange policy.

This exchange policy can be differentiated into two independent policies:

Location Policy

This policy becomes relevant if more than one exchange partner is available in the Grid. Thus, there exists more than one possibility to delegate a job to a remote Grid participant. For such scenarios, the location policy determines as a first step the sorted subset of possible delegation targets Ω, see Figure 2.
The big advantage of the TSK model is its representative power. It is capable of describing a highly nonlinear system using a small number of rules. Furthermore, since the output of the model has an explicit functional expression form, it is conventional to identify its parameters using some learning algorithms as e.g. evolutionary algorithms. Exactly this combination will be used later to optimize the Fuzzy model, see Section 6.

In this paper, the Fuzzy GRMS decision policy is founded on a set of rules. Each specific rule describes a system state in which decisions about the acceptance or refusal of jobs must be made. Thus, each system state is described by a set of features. From the different parts of the overall system various state describing features are conceivable. They might be related to the current state of the LRMS layer or to the currently job to decide, see Figure 3. Please note that information about remote sites’ systems states is assumed strictly classified. Following

The Fuzzy rule concepts, a rule consists of a feature describing conditional part and a consequence part that decides on the acceptance or decline of an offered job. An exemplary set of rules might consist of the following reasonable statement expressing some sort of experts’ knowledge or experience:

\[
\begin{align*}
\text{IF } & \text{ queue is long and job highly parallel} & \text{ THEN decline job} \\
\text{IF } & \text{ queue is empty and job not parallel} & \text{ THEN accept job}
\end{align*}
\]

A complete rule base now constitutes the core of the rule system that can therefore be considered as a controller. The current system is checked whenever a new job has been submitted to the local system or has been offered from remote sites. In all those cases the current system state might change and the controller output has to be changed if necessary. The controller concept is described in the next paragraph.
4.1 Fuzzy System for Decision Making

More formally, the general TSK model consists of $N_r$ IF-THEN rules $R_i$ such that

$$R_i := \text{IF } x_1 \text{ is } g_i^{(1)} \text{ and } \ldots \text{ and } x_{N_f} \text{ is } g_i^{(N_f)} \text{ THEN } y_i = b_{i0} + b_{i1}x_1 + \ldots + b_{iN_f}x_{N_f}$$

where $x_1, x_2, \ldots, x_{N_f}$ are input variables and elements of a vector $\vec{x}$, and $y_i$ are local output variables. Further, $g_i^{(h)}$ is the $h$-th input Fuzzy set that describes the membership for a feature $h$. Thus, system state is described by a number of $N_f$ features. The actual degree of membership is computed as function value of an input Fuzzy set which is characterized for example by a Gaussian Membership Function (GMF). The here used Fuzzy sets are explained in the next section. Furthermore, $b_{ih}$ are real valued parameters that specify the local output variable $y_i$ as a linear combination of the input variables $\vec{x}$. The overall output of the system $y_D(\vec{x})$ is computed by Equation 2.

$$y_D(\vec{x}) = \frac{\sum_{i=1}^{N_r} \phi_i(\vec{x})y_i}{\sum_{i=1}^{N_r} \phi_i(\vec{x})}$$

where $\phi_i(\vec{x})$ is the degree of membership of rule $R_i$ for a given input vector $\vec{x}$, which is defined as

$$\phi_i(\vec{x}) = g_i^{(1)}(x_1) \land g_i^{(2)}(x_2) \land \ldots \land g_i^{(N_f)}(x_{N_f})$$

Each rule’s recommendation is weighted by its degree of membership with respect to the input vector $\vec{x}$. The corresponding output value of the TSK-System is then computed by the weighted average output recommendation over all rules. In the following, we explain how this very general model is adapted to the problem of decision making addressed here. The specific coding of rules and the output computation will be detailed in the following paragraphs.

4.2 Encoding of Rules

The TSK-model allows to describe the degree of membership by any sort of membership function. However, as the proposed Fuzzy-Systems is going to be optimized with an evolutionary algorithm the number of parameters for a rule system must be kept as small as possible as every additional parameter increases the search space of the optimization problem and might deteriorate the solution quality. Because of their smoothness and concise notation, the Gaussian membership function has become a very popular for specifying Fuzzy-Sets. It only requires two parameters (mean and variance) to specify its shape which makes it particularly suited to represent a rule base with a minimum number of parameters.

This, for a single rule $R_i$ every feature $h$ of all $N_f$ features is now modeled by a $(\gamma_i^{(h)}, \sigma_i^{(h)})$-Gaussian Membership Function (GMF)$^2$ with no normalization as shown in Equation 4.

$$g_i^{(h)}(x) = \exp \left\{ -\frac{(x - \gamma_i^{(h)})^2}{\sigma_i^{(h)} \gamma} \right\}$$

This function is completely described by defining the $\gamma_i^{(h)}$ and $\sigma_i^{(h)}$ values. The $\gamma_i^{(h)}$-value adjusts the center of the feature value, while $\sigma_i^{(h)}$ models the region of influence for this rule in the feature domain. In other words, for increasing $\sigma_i^{(h)}$ values the GMF becomes wider, while the peak value remains constant at 1. Using this property of a GMF we are able to steer the influence of a rule for a certain feature by $\sigma_i^{(h)}$.

![Fig. 4. Encoding pattern for single rules and construction concept for a whole rule base using concatenation.](image)

Using this GMF as membership function a feature can be coded as a pair of real values $\gamma_i^{(h)}$ and $\sigma_i^{(h)}$ following the approach of Juang et al. [11] and Jin et al. [12]. Using this feature description, a single rule’s conditional part is composed as shown in Figure 4. For the consequence part, the general model in Equation 2, can be simplified as we have to deal with binary decisions only. Dependent on the current system state, the Fuzzy decision maker has to decide whether to accept an offered job or not. Thus, we represent the acceptance of a job by an output value of 1 and the corresponding refusal of a job by -1. With this binary decision concept, all weights except $b_{i0}$ in Equation 2 are set to 0 and the TSK model output becomes $y_i = b_{i0}$. As we have to decide between the acceptance/decline of a job offer, the output values

2. Different from the common notation we denote the mean of the GMF by $\gamma$ to avoid conflicts with the parental population size of Evolution Strategies which is in this paper denoted by $\mu$. 
for a rule $R_i$ can be chosen as

$$y_i = \begin{cases} 
1, & \text{if job is accepted} \\
-1, & \text{otherwise} 
\end{cases} \quad (5)$$

Exactly this encoding concept is now used to build an individual within an evolutionary algorithm. This scheme allows the encoding of a single rule by a string of $2 \cdot N_f$ real-valued and one integer variable, see Figure 4.1. The whole rule base is encoded by concatenation of single rules. A whole rule base consisting of $N_r$ rules is therefore entirely described by a set of $l = N_r \cdot (2 \cdot N_f + 1)$ parameters. This encoding scheme is perfectly suited as individual representation within an evolutionary algorithm where individuals have the length $l$. As objective function for the evolutionary algorithm we use the AWRT, see Section 5.1. Therefore, the optimization problem is to find suitable parameter settings for the rules in order to minimize the AWRT for all participating sites in the Grid.

### 4.3 Computation of the Controller Decision

To determine the actual controller output for a set of input states $\vec{x}$ the superposition of all degrees of memberships for a single rule $R_i$ is computed first. For each rule $R_i$ a degree of membership $g_i^{(h)}(x_h)$ of the $h$-th of all $N_f$ features is determined for all $h$. This value is computed as the function value of the $h$-th GMF for the given input feature value $x_h$. According to the general model, see Equation 3, the multiplicative superposition of all these values as “AND”-operation leads to an overall degree of membership $\phi_i(\vec{x})$ for rule $R_i$ as shown in Equation 6.

$$\phi_i(\vec{x}) = \prod_{h=1}^{N_f} g_i^{(h)}(x_h) = \prod_{h=1}^{N_f} \exp \left\{ -\frac{(x_h - \gamma_i^{(h)})^2}{\sigma_i^{(h)} x} \right\} \quad (6)$$

Further, the final controller output $Y_D$ can be computed by considering the leading sign only, see Equation 7.

$$Y_D = \text{sgn}(y_D(\vec{x})) \quad (7)$$

where a positive number again represents the acceptance of the job and a negative values the decline. Note that the value zero corresponds to a decline as well. The TSK-model allows including an arbitrary number of features as controller input. Thus, it is possible to achieve a preferably accurate state description. However, this would increase the number of adjustable system parameters drastically as each feature requires an additional $(\gamma, \sigma)$-pair per rule. As the proposed Fuzzy system is going to be optimized with an evolutionary algorithm the number of system describing parameters must be kept as small as possible as every additional parameter increases the search space of the optimization problem and might deteriorate the solution quality. Thus, we restrict ourselves to only two features for the system state description and detail them in the next paragraph.

### 4.4 Feature Selection for System State Description

For the description of the current system state we rely on only $N_f = 2$ different features that will constitute the conditional part of a rule. We denote jobs that have been inserted into the waiting queue $\nu$ at site $k$ as $j \in \nu_k$. In order to cover comprehensive system information with only a single feature we consider the Normalized Waiting Parallelism at site $k$ $(\text{NWP}_k)$ as the first feature, see Equation 8.

$$\text{NWP}_k = \frac{1}{m_k} \sum_{j \in \nu_k} m_j \quad (8)$$

This feature indicates how many processors are expected to be occupied by all submitted jobs (note that the number of requires processors $m_j$ is known at release time) related to the maximum number of available processors $m_k$ at site $k$. It reflects the efficiency of the currently running LRMS and measures the near future expected load of the machine.

The second feature focuses on the actual job that has to be decided. The ratio of a job’s resource requirements $m_j$ and the maximum number of available resources $m_k$ at the job’s submission site $k$ is expressed by the Normalized Job Parallelism $(\text{NJP})$ in percent, see Equation 9.

$$\text{NJP}_j = \frac{m_j}{m_k} \cdot 100 \quad (9)$$

With those two selected features we approximate every possible system state. Note that the NWP is not limited in range; however, values greater than 10 occur very rarely in practice.

### 4.5 Configuration of the Evolutionary Fuzzy System

Before we present the evaluation results the configuration of the Evolutionary Fuzzy System and the further evaluation circumstances are detailed. We generate our Evolutionary Fuzzy Systems with a fixed number of $N_r = 10$ rules. Previous studies of Franke et al. [13] revealed that rule bases consisting of five to ten rules yield good results. As we encode the whole rule base in one individual, we have to optimize a problem with $N_r \cdot (N_f \cdot 2 + 1) = 10 \cdot (2 \cdot 2 + 1) = 50$ parameters.

For the tuning of the Fuzzy System we apply a $(\mu + \lambda)$-Evolution Strategy. During the run of 150 generation a continuous progress in fitness improvement is observable. As recommended by Schwefel [5], the ratio of $\mu/\lambda = 1/7$ should be used for Evolution Strategies. We created a parent population of $\mu = 13$ individuals which results in a children population of $\lambda = 91$ individuals. Hence, 91 individuals must be evaluated within each generation. As objective function we use the AWRT, see Section 5.1.

For the variation operators we used further the following configurations: The mutation is performed with an individual mutation step-size for each feature. As the two features vary in their possible value range by a ratio of 1:10, see Figure 4.4, we used a mutation step-size of...
0.01 for NWP and 0.1 for NJP respectively. This mutation is applied for the conditional part of the rule as they are real values. For the binary consequence part we mutate values by flips from -1 to 1 or vice versa. Further, we apply discrete recombination in each reproduction step.

The population is uniformly initialized within the ranges $[0, 10]$ for the $(\gamma, \sigma)$-values of NWP and $[0, 100]$ for NJP respectively. As the fitness evaluation of an individual is quite time consuming (from several minutes up to half an hour) we evaluated the whole population in parallel on a 200 node cluster with Pentium IV, 2.4Ghz machines. To train a basic rule set, it takes on the cluster computer between five and ten hours depending on the simulated setups, see Table 2, as for $K$ sites ($K^2 - K$) pairings have to be simulated. However, for the actual application of the Fuzzy-Systems only basic mathematical operations are required which can be performed within a negligible amount of time.

5 Performance Evaluation

In order to evaluate the performance of our approach, we define several well-known performance indicators for job scheduling in the context of Grid computing, both from the users’ and the providers’ point of view. Additionally, we discuss the workload traces we use as input data that are derived from real-world setups.

5.1 Average Weighted Response Time

This objective is computed for all jobs $j \in \pi_k$ that have been initially submitted to site $k$, see Equation 10. In the following we further denote by $j \in \pi_k$ all jobs that have been actually executed on site $k$. It is widely agreed that a short AWRT is the best way to describe that on average users do not wait long for their jobs to complete. Following Schwiegelshohn and Yahyapour [9], we use the resource consumption $(p_j, m_j)$ of each job as weight. This ensures that neither splitting nor combination of jobs can influence the objective function in a beneficial way.

$$\text{AWRT}_k = \frac{\sum_{j \in \pi_k} p_j \cdot m_j \cdot (C_j(S) - r_j)}{\sum_{j \in \pi_k} p_j \cdot m_j}$$  

(10)

Note that this also respects the execution on remote sites and, as such, the completion time $C_j(S)$ refers to the site that executed job $j$.

5.2 Squashed Area and Utilization

The first two objectives are Squashed Area $SA_k$ and Utilization $U_k$, both specific to a certain site $k$. They are measured from the start of the schedule $S_k$, that is minimal $j \in \pi_k \{C_j(S_k) - p_j\}$ as the earliest job start time, up to its makespan $C_{max,k} = \max_{j \in \pi_k} \{C_j(S_k)\}$, that is the latest job completion time and thus the schedule’s length. $SA_k$ denotes the overall resource usage of all jobs that have been executed on site $k$, see Equation 11.

$$SA_k = \sum_{j \in \pi_k} p_j \cdot m_j$$  

(11)

$U_k$ describes the ratio between overall resource usage and available resources after the completion of all jobs $j \in \pi_k$, see Equation 12.

$$U_k = \frac{SA_k}{m_k \cdot \left(C_{max,k} - \min_{j \in \pi_k} \{C_j(S_k) - p_j\}\right)}$$  

(12)

$U_k$ describes the usage efficiency of the site’s available machines. Therefore, it is often serving as a schedule quality metric from the site provider’s point of view.

However, comparing single-site and multi-site utilization values is not recommended: since the calculation of $U_k$ depends on $C_{max,k}$, valid comparisons are only admissible if $C_{max,k}$ is approximately equal between the single-site and multi-site scenario. Otherwise, high utilizations may indicate good usage efficiency, although the corresponding $C_{max,k}$ value is very small and shows that only few jobs have been computed locally while many have been delegated to other sites for remote execution.

As such, we additionally introduce the Change of Squashed Area $\Delta SA_k$, which provides a makespan-independent view on the utilization’s alteration, see Equation 13.

$$\Delta SA_k = \frac{SA_k}{\sum_{j \in \pi_k} p_j \cdot m_j}$$  

(13)

From the system provider’s point of view this objective reflects the real change of the utilization when jobs are shared between site compared to the local execution.

5.3 Input Data

The Parallel Workloads Archive\(^3\) provides job submission and execution traces recorded on real-world MPP system sites, each of which containing information on relevant job characteristics. Four well known workloads have been selected for evaluation: The KTH trace which contains records from a 100 processor IBM RS/6000 SP system at the Swedish Royal Institute of Technology in Stockholm, the CTC trace from a 430 processor IBM RS/6000 SP system at the Cornell Theory Center in Ithaca, NY, and two logs recorded at the San Diego Supercomputer Center (SDSC) in La Jolla, CA. On the one hand, the SDSC05 "DataStar" trace recorded on a IBM eServer pSeries 655/690 system with a total of 1664 processors and the SDSC00 trace containing submissions to a 128 processor IBM RS/6000 SP system. Relevant details of the examined cleaned traces are given in Table 1.

The original workloads record time periods of different length. In order to be able to combine different

\(^3\) http://www.cs.huji.ac.il/labs/parallel/workload/.
workloads in a multi-site simulations and to have a validation set available, we shortened and separated the workloads to set of five and six month respectively. In the remainder of this work, the five month sequence will serve as training sequences for the Evolutionary Fuzzy Systems. The six month sequences are then used for application tests. In this context, the SDSC00-6 trace will only be used to investigate the behavior of the trained system when an previously unknown partner participates in the system. Thus, we created no five month training sequence of the SDSC00 workload. It is important to mention that the here described data only serve as simulation input and do not contain any additional information concerning Fuzzy Systems (e.g. knowledge base, see Section 2.3).

Further, we do not shift the traces regarding their originating timezones and restrict our study to workloads which are all submitted within the same timezone. Therefore, the known diurnal rhythm of job submission is similar for all sites in our scenario and time shifts cannot be availed to improve scheduling. In a global grid the different timezones even benefit the job scheduling cannot be availed to improve scheduling. In a global grid is similar for all sites in our scenario and time shifts Therefore, the known diurnal rhythm of job submission.

We simulated the workload on their original machines with a LRMS that applied FCFS, see Section 3.1 for the local scheduling. The results for the above described performance metrics as well as other relevant job characteristics are listed in Table 1. We will refer to this non-cooperative case for the matter of comparison in the rest of this paper.

### TABLE 1

<table>
<thead>
<tr>
<th>Identifier</th>
<th>#jobs</th>
<th>( m_k )</th>
<th>AWRT</th>
<th>U</th>
<th>( C_{\text{max}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>KTH-5</td>
<td>11780</td>
<td>100</td>
<td>48885.49</td>
<td>64.84</td>
<td>13765972</td>
</tr>
<tr>
<td>KTH-6</td>
<td>16009</td>
<td>100</td>
<td>99236.27</td>
<td>68.32</td>
<td>16420762</td>
</tr>
<tr>
<td>CTC-5</td>
<td>35360</td>
<td>430</td>
<td>57897.77</td>
<td>63.74</td>
<td>13009718</td>
</tr>
<tr>
<td>CTC-6</td>
<td>41839</td>
<td>430</td>
<td>59118.15</td>
<td>67.05</td>
<td>16316403</td>
</tr>
<tr>
<td>SDSC05-5</td>
<td>28184</td>
<td>1664</td>
<td>36925.10</td>
<td>45.94</td>
<td>13078215</td>
</tr>
<tr>
<td>SDSC05-6</td>
<td>46019</td>
<td>1664</td>
<td>77463.52</td>
<td>70.97</td>
<td>16419459</td>
</tr>
<tr>
<td>SDSC06-6</td>
<td>16316</td>
<td>128</td>
<td>41397.04</td>
<td>73.38</td>
<td>17022360</td>
</tr>
</tbody>
</table>

Workload characteristics of the used input data, including AWRT in seconds, U in %, and \( C_{\text{max}} \) in seconds for single site execution with FCFS.

6.1 Results for Training Sequences

The training results are listed in Table 2; gray-shaded lines indicate the evolved site while the other lines indicate the static site as described above. As expected, the optimization leads to significant improvements of the AWRT in all examined setups.

This results in larger AWRT for the partner site that does not adapt its behavior. For instance in Setup I, the AWRT improves by 86.47% compared to FCFS, see Table 1, while the AWRT for the CTC worsens for almost 10%. Note that this corresponds to a strong shift of work as the Squashed Area (ASA) is 48.56% lower for the KTH and approximately 12% higher on the CTC site. However, when the focus is changed, see Setup II, and CTC is optimized we achieve also improvements of 5.44% for AWRT and slight load relief for the CTC site. Besides that, the AWRT is still significantly improved in Setup II although we do not focus on the KTH. This is due to the worse performance in the non-cooperative case and indicates that Grid computing for this site is very advantageous.

Furthermore, in Setup III and IV the small KTH interacts with the very large SDSC05 compute center and naturally the KTH benefits from more available resources. It is remarkable that also the SDSC05 can improve its AWRT for more than 3%, see Setup IV. At the same time, the Squashed Area is slightly increased which indicated that an improvement in AWRT is not necessarily caused by smaller utilization.

When the CTC interacts with a large compute center, see Setup V, the CTC also strongly benefits as its AWRT is decreased by more than 20%. Likewise, the SDSC05 can benefits from the cooperation with a medium size compute installation like the CTC, see Setup VI.
In order to understand the optimized transfer behavior in more detail, we show in Figure 5 the set of characteristic curves for the different Grid pairs. There, the system states are shown on the $x$- and $y$-axis for NJP and NWP respectively. The border between acceptance of jobs ($>0$) and refusal of jobs ($\leq 0$) is marked by a grid at level zero on the $z$-axis. In Figure 5(a) and Figure 5(b) one can see that the small KTH site has a quite restrictive exchange behavior as only small jobs are accepted and almost all remote jobs are declined. As local and remote jobs are treated similarly, such a behavior tries to offer all jobs to remote sites first. The black dots indicate the areas in the decision space that are activated during the controller application. It becomes obvious that mainly small jobs are accepted. In Figure 5(b) also larger jobs from the SDSC05 are offered as almost the whole range for NJP is activated. However, the exchange policy for the KTH refuses them in any situation.

The situation is different for the CTC that does not only accept its own large jobs but also remote jobs that require many processors. Here, the small jobs are offered to the remote site first, see Figure 5(c). Finally, the large compute center, see Figure 5(e) and (f), accepts almost any kind of jobs as long as the waiting queue is not too full. If the NWP is larger that 4 the transfer policy switches from acceptance to refusal. This is reasonable as it prevents the starvation of jobs in the queue and protect the Grid participant for overuse of resources. Summarizing it becomes apparent from these figures that job exchange among grid participants is only beneficial if

### Table 2

<table>
<thead>
<tr>
<th>Setup</th>
<th>Site</th>
<th>AWRT$_{K}$</th>
<th>$U_k$</th>
<th>$\Delta$AWRT$_{K}$</th>
<th>$\Delta U_k$</th>
<th>$\Delta$SA$_k$</th>
<th>$\Delta$C$_{max,k}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>KTH-5</td>
<td>66100.77 sec</td>
<td>35.33%</td>
<td>86.47%</td>
<td>-45.51%</td>
<td>-48.56%</td>
<td>5.60%</td>
</tr>
<tr>
<td></td>
<td>CTC-5</td>
<td>63534.29 sec</td>
<td>71.40%</td>
<td>-9.74%</td>
<td>12.01%</td>
<td>12.15%</td>
<td>-0.14%</td>
</tr>
<tr>
<td>II</td>
<td>KTH-5</td>
<td>62884.33 sec</td>
<td>73.00%</td>
<td>67.12%</td>
<td>12.58%</td>
<td>6.40%</td>
<td>5.49%</td>
</tr>
<tr>
<td></td>
<td>CTC-5</td>
<td>54735.53 sec</td>
<td>62.70%</td>
<td>5.44%</td>
<td>-1.64%</td>
<td>-1.60%</td>
<td>-0.05%</td>
</tr>
<tr>
<td>III</td>
<td>KTH-5</td>
<td>59409.60 sec</td>
<td>45.15%</td>
<td>87.84%</td>
<td>-30.36%</td>
<td>-34.04%</td>
<td>5.26%</td>
</tr>
<tr>
<td></td>
<td>SDSC05-5</td>
<td>58799.15 sec</td>
<td>47.34%</td>
<td>-3.29%</td>
<td>3.06%</td>
<td>3.04%</td>
<td>0.01%</td>
</tr>
<tr>
<td>IV</td>
<td>KTH-5</td>
<td>70561.62 sec</td>
<td>33.39%</td>
<td>85.35%</td>
<td>-16.60%</td>
<td>-21.75%</td>
<td>4.9%</td>
</tr>
<tr>
<td></td>
<td>SDSC05-5</td>
<td>54733.39 sec</td>
<td>46.85%</td>
<td>3.78%</td>
<td>1.98%</td>
<td>1.94%</td>
<td>0.02%</td>
</tr>
<tr>
<td>V</td>
<td>CTC-5</td>
<td>45997.73 sec</td>
<td>58.55%</td>
<td>20.55%</td>
<td>-8.14%</td>
<td>-8.15%</td>
<td>0.01%</td>
</tr>
<tr>
<td></td>
<td>SDSC05-5</td>
<td>57013.44 sec</td>
<td>47.27%</td>
<td>-0.16%</td>
<td>2.90%</td>
<td>2.91%</td>
<td>-0.02%</td>
</tr>
<tr>
<td>VI</td>
<td>CTC-5</td>
<td>57069.59 sec</td>
<td>63.30%</td>
<td>1.43%</td>
<td>-0.69%</td>
<td>-0.51%</td>
<td>-0.20%</td>
</tr>
<tr>
<td></td>
<td>SDSC05-5</td>
<td>49916.01 sec</td>
<td>46.04%</td>
<td>12.31%</td>
<td>0.22%</td>
<td>0.18%</td>
<td>0.02%</td>
</tr>
</tbody>
</table>

Results for the pair-wise rule base training. The gray shaded rows indicated the optimized rule base.

![Figure 5](image-url)

(a) KTH vs. CTC  
(b) KTH vs. SDSC05  
(c) CTC vs. KTH  
(d) CTC vs. SDSC05  
(e) SDSC05 vs. KTH  
(f) SDSC05 vs. CTC
the partners show some sort of cooperation; that is they have to accept also alien jobs in suitable situations.

6.2 Robustness of Trained Rule Sets
To test the robustness of the pair-wise learned rule bases, we apply them to the 6 month workloads within the same setups. To this end, only two site Grids are considered and every partner applies its egoistically learned rule base. Note that AWF is not used in these scenarios anymore.

In Figure 6, the changes in AWRT and SA are depicted when both partners apply their learned rule bases to previously unknown job submissions.

![Fig. 6. AWRT and U improvements for the optimized rules on non-trained data sets.](image)

Obviously, the Evolutionary Fuzzy Systems still decrease the AWRT significantly in all cases. This indicates a high robustness with respect to submission changes.

Further, we show in Figure 7 the AWRT improvements in comparison to the AWF transfer policy and additionally to near optimal solutions. Although AWF is a quite naive exchange method it is sufficient for the training purpose. However, for the matter of comparison it is more meaningful to compare our approach against the best achievable solutions. Unfortunately, there exists no such algorithm that can generate optimal solutions under the given information restrictions. Thus, we apply an request based exchange method (OPT) that actually relaxes all information restrictions and allows also backfilling of jobs in the queues. This method was detailed and extensively evaluated by Grimme et al. [15] and may serve—by neglecting the given information restrictions—as best achievable solution.

In the figure it becomes obvious that the rule based approach achieves results which are approximately half as good as the optimal results (OPT). This shows that even under the strong information restriction the proposed algorithm does not perform too bad as the optimal results would be never achievable under these circumstances. Although AWF performs good for KTH and leads to slight improvements for SDSC05, it completely fails for the CTC workload trace. However, the rule based transfer policy outperforms AWF in all cases and leads even to shorter AWRT values for the SDSC05 together with the much smaller KTH.

7 COPING WITH MORE THAN ONE PARTNER
After setting up the basic rule sets for job exchange in a controlled environment with a single partner, we now focus on the applicability of the rule bases in a Grid scenario with more participants. To this end, KTH, CTC, and SDSC05 are combined and the unknown submissions from the remaining six month of the traces are used. This time, however, a location policy needs to be applied in order to prioritize the options of delivering jobs to another participant.

7.1 An AWRT-based Location Policy
In order to create a prioritization of the available potential delegation targets we follow a two-step approach. As first step, we generate the subset of sites that in total provide enough machines to execute the job. That is, we sort out all sites with $m_k < m_j \forall k \in K$.

As second step the generated subset is sorted according to their former achievement with respect to a delegation source. Good achievements can be measured by short AWRT of jobs on the corresponding sites. For this end, only jobs are considered that have been delivered to the corresponding partner. The AWRT indicates how long the delegation source had to wait for the completion of its delivered jobs in the past. This metric is based on the assumption that a short AWRT for delivered jobs in the past is expected to yield also short AWRT values for future delegated job.

7.2 Results for Multiple Partners
The results in Figure 8 clearly indicate that the AWRT is still significantly improved for all sites while the
utilization decreases for the small partner. However, although the CTC and SDSC05 are slightly more utilized it does again improve their objective values.

8 Coping with Alien Partners

Until now, our learning approach was suited to generate a pool of rule bases for partners that are known in advance. This, however, requires knowledge about the submitted workload in order to tune the transfer policies. With respect to the robustness requirement, we therefore extend our approach to being able to perform well in an environment with previously unknown Grid participants. This requires the automatic adjusting of transfer behavior to partners that were not part of a training scenario.

8.1 Selection of Rule Base

As mentioned in Section 3.2, the rule based transfer policy is applied to each partner site separately. If a new partner arises a transfer policy has to be selected from the pool of all learned transfer policies. To identify the best suitable transfer policy we assume a correlation between delegation targets’ maximum amount of available resources and their transfer behavior. We conjecture that the behavior within the grid mainly depends on a site’s resource number. Thus, we categorize the various trained rule bases by the machine sizes they belong to. Among the whole trained pool of transfer policies the best fitting one, with respect to the number of maximum available resources, is selected to make the decision for a submitted job.

8.2 Results for Alien Partners

Finally, we investigate the performance of the rule base selection concept and add the SDSC00 as a site with $m_k = 128$ processors to the Grid. Following the rule base selection concept, every site uses the KTH learned rule base for the interaction with SDSC00 as it has the greatest similarity with respect to the machine size. The SDSC00 site, in turn, uses AWF for exchange purpose.

In Figure 9 the results for interaction with CTC and SDSC05 are depicted and again we observe strong AWRT improvements. Similarly to the KTH, also SDSC00 shows a poor performance for exclusive single site execution. Therefore, there is a high potential to improve the AWRT. However, it is important to see that not only this partner can improve its AWRT but also other participants are able to improve their AWRT for at least 8%.

The same holds for a four-site grid with one unknown partner, which results are presented in Figure 10. Again, AWRT improvements are achieved for all site for at least 10% while the utilization decreases on the smaller sites and increases on the larger site.

Summarizing, the learned Evolutionary Fuzzy Systems realize a beneficial job exchange for several cooperative computing environments. The examined Grid sizes range from two to four sites and include unknown job submissions as well as previously unknown Grid participant. In all cases, the AWRT can be significantly decreased which results in improvements of about 10%-
20% for large sites and 40%-80% for larger sites. It has been shown, that the job exchange policies show a strong robustness with respect to both new sort of job submissions and environmental changes.

9 Related Work

Although decentralized scenarios are supposed to be widely-used in future, todays development of Grid infrastructure focuses on the application of centralized Grid scheduling services [1] and yields only limited innovation regarding efficient algorithms for resource allocation. Ernemann et al. [16] point out advantages of hierarchical scheduling in general by considering the AWRT objective. In another work, they propose an adaptive algorithm which decides based on the overall system state whether to split up a job to execute it on different Grid sites. Further, Kurowski et al. [17] identify multiple objectives for efficient job scheduling in Grids and propose a strategy based on prediction mechanisms and resource reservation. Further Hong and Prasanna [18], model a heterogeneous grid as a graph, which is capable of representing an arbitrary network topology. They focus on the maximization of the steady state throughput of such systems and show that the throughput maximization problem can be solved through a linear programming formulation. Recently, Iosup et al. [19] proposed a delegated matchmaking method, which temporarily binds resources from remote sites to the local environment. As such, they strive to realize also Grid-interaction on the Meta-Level, but they assume a hybrid structure and augment a hierarchical structure by P2P architectures.

Of course, due to the highly dynamical character of the problem, nature inspired techniques for adaptive scheduling have also been proposed during the last years: besides GA-driven Meta-schedulers [20] hybrid heuristics have been applied [21].

For the decentralized scenario, only few results that support the application of job delegation in Grids have been published so far. England and Weissman [22] give an estimation of benefits and costs but base their results on synthetic workloads and review the average slowdown only. A more real-world centered investigation has been made by Hamscher et al. [23]. They apply different job sharing strategies to two real workloads and analyze the resulting behavior. However, they give no quantification of the actual benefit compared to non Grid-cooperative set-ups. Grimme et al. [24] analyze the prospects of collaborative job sharing in Grids. They compare their results to the non-cooperative scenario of the same machines but do not give a quantitative estimation of possible collaborative benefits. A follow-up to this initial work [25] focuses on the maximum achievable improvements for the AWRT objective considering Grid participants with mutually conflicting performance goals. They consider the job exchange problem in Grids as multi-objective optimization where the resulting Pareto front shows high potential for achievable AWRT improvements within Grids, motivating the design of more advanced exchange policies.

Fuzzy Systems have only been partially applied to scheduling in Computational Grid. Huang et al. [26] determine a knowledge base for Grid computing by transforming Grid monitoring data into a performance data set. They extract the association patterns of performance data through fuzzy association rule in order to optimize the Grid scheduling. This approach however requires the availability of various monitoring data and does not work within information restricted environments. Further, Fayad et al. [27] address a Grid scheduling problem where processing times underlay uncertainty. They use Fuzzy sets to model the uncertain processing times and apply tabu search to maximize the number of scheduled jobs. Their approach is evaluated with workloads recorded at real-world installations. However, they consider sequential jobs only, i.e. non-parallel jobs see Section 2.1, and assume artificial due dates that have to be met. Further, they allow direct access of the Grid scheduler to local resources.

The capabilities of Evolutionary Fuzzy Systems have only recently been applied to scheduling. So far—at least to the authors’ knowledge—no endeavors have been made to apply them to Grid scheduling problems. However, some promising results have been obtained for parallel computer scheduling with Evolutionary Fuzzy Systems: Franke et al. [28] derived a methodology to genetically learn a Fuzzy System from real workload data to establish user group prioritizations. With such a system, the owner of a parallel computer is able to flexibly realize prioritizations for certain user groups without worsening the overall system utilization. After a training phase, the controller realizes the desired priorities while still delivering significantly shorter response times for the favored user groups.

10 Conclusion and Future Work

We presented an Evolutionary Fuzzy System approach to finding non-invasive, situation-adaptive, and robust algorithms for workload distribution in decentralized Computational Grids. Such environments assume full autonomy of the participating HPC/HTC centers and strict confidentiality of dynamic system information and demand Grid middlewares that do not interfere with the running LRMS.

In our model, we introduced a decoupled GRMS layer on top of the available systems, which decides upon execution on the local system or delegation to a remote site for user-submitted jobs in an online, non-clairvoyant manner. The decision mechanism is established by using a Fuzzy controller system with flexible rule sets that are optimized using evolutionary computation, using a pairwise training approach and performance metric-based rule base selection.

The presented system shows that—using real-world data—it is possible to establish job exchange policies...
which lead to significantly improved performance for all user communities in terms of response time and utilization. We further find that our approach behaves robustly with respect to fluctuations in the workload pattern and shows situational adaptiveness even under circumstances of unknown submission characteristics. Overall, we think that the derived controllers provide a stable basis for workload distribution and interchange in Computational Grids, and may qualify as a promising technology for future Service Grid-based e-Science infrastructures.

For future work it is essential to consider a more fine grained machine an Grid environment model. For example, it is necessary to include also data intensive applications. The here proposed method can be extended to data-aware scheduling where the replication of the accessibility of data is considered as well. Further, the training of the system should be made online, during scheduling in form of an self-learning system. Learning scheduling decision with machine learning techniques would lead to a zero-configuration scheduling system which can be easily integrated into existing middleware solutions.

References


