Decentralized Grid Scheduling with Evolutionary Fuzzy Systems

Alexander Fölling, Christian Grimme,
Joachim Lepping, and Alexander Papaspyrou

Robotics Research Institute, TU Dortmund University, 44221 Dortmund, Germany
email: {*firstname.lastname*}@udo.edu

Abstract. In this paper, we address the problem of finding 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.

1 Introduction

Modern science more and more relies on experimental scientific discovery made with extensive simulations, and during the last decade, Grid Computing has become the key infrastructure in academia to support this development. The use of Grid Computing, however, 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 [6] have relied on centralized middleware infrastructures for whole e-Science communities, other—mostly emerging—efforts 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, see Grimme et al. [7, 8]: 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. Such Fuzzy System based scheduling techniques have been successfully applied to online scheduling problems before, see for example Franke et al. [4]. 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. [1]. We show that our approach, while respecting the aforementioned requirements for Grid scheduling algorithms, shows adequate performance characteristics in real setups.

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 and conclude our work Section 9.

2 Background

This section briefly introduces the basics 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. [3], 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 [11]—operate on a population of μ 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 λ offspring individuals from those μ parental individuals, followed by a global selection process in which the individuals compete against each other to form the new μ 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 (μ, λ)-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 1960s 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 advices 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 without any required a priori knowledge. 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 compute clusters has been continuously studied since the emergence of Grid computing in the beginning of the 1990s. 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 [10]. The most profound problem of this scheduling structure is its bad fault-tolerance and lack of scalability.

With respect to the basic parameters of modern e-Infrastructures regarding organizational autonomy and equity, we therefore 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.

More formally, a Computational Grid consists 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. 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. [12].

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

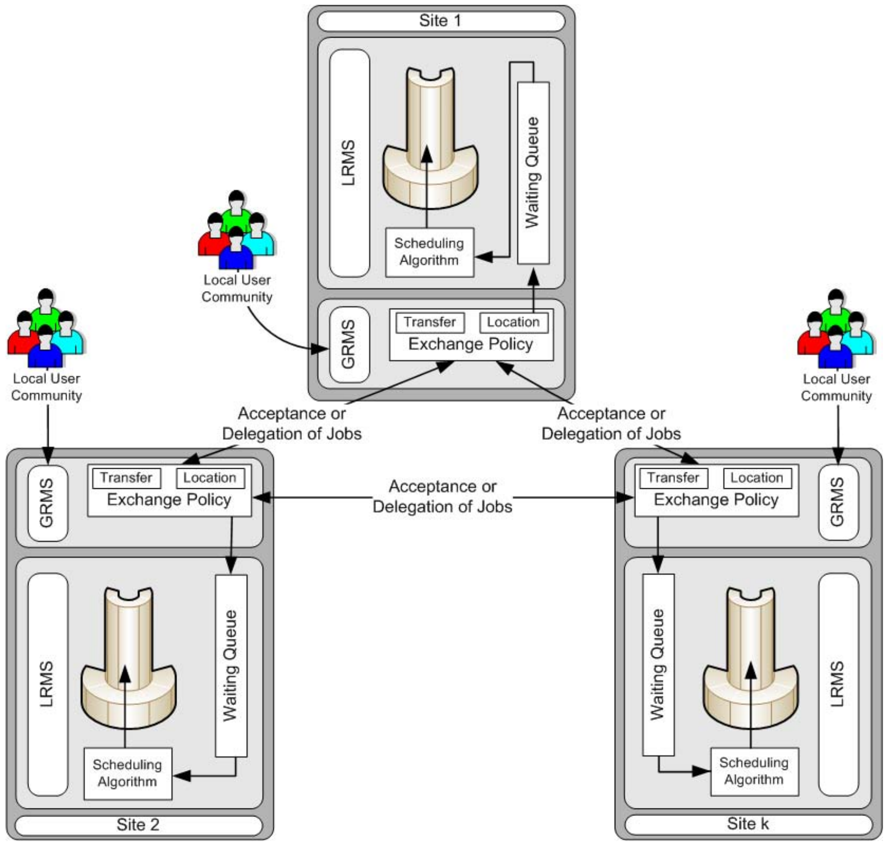


Fig. 1. Computational Grid scenario with independent sites in a federated environment

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 exists 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. The 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 [13]. Please note that our job exchange methodology is not restricted to any kind of local scheduling algorithm but it serves for any arbitrary scheduling algorithm on the LRMS layer.

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 ①, see Figure 2.

Transfer Policy

After the location policy has been applied the transfer policy specifies whether a job should be delegated to a certain partner or not. For this purpose the policy is applied separately on each partner in an redetermined order. Every time the transfer policy is consulted it decides whether the job should be executed locally ② or delegated to the considered partner ③. In the first case, the job is sent to the remote LRMS ④ and in the other case the considered partner is requested for a job's acceptance. A request can be replied in two different ways:

1. The job is accepted by the remote partner ⑤. In this case, the job is delegated to this partner and no other further delegation attempts have to be made.
2. If the acceptance is declined the transfer policy is applied for another partner in the Grid ⑥. This iterative procedure is continued until all partners have been requested. If none of the Grid participants is willing to accept the job, the requesting site must execute it locally ⑦, ⑧, and ④.

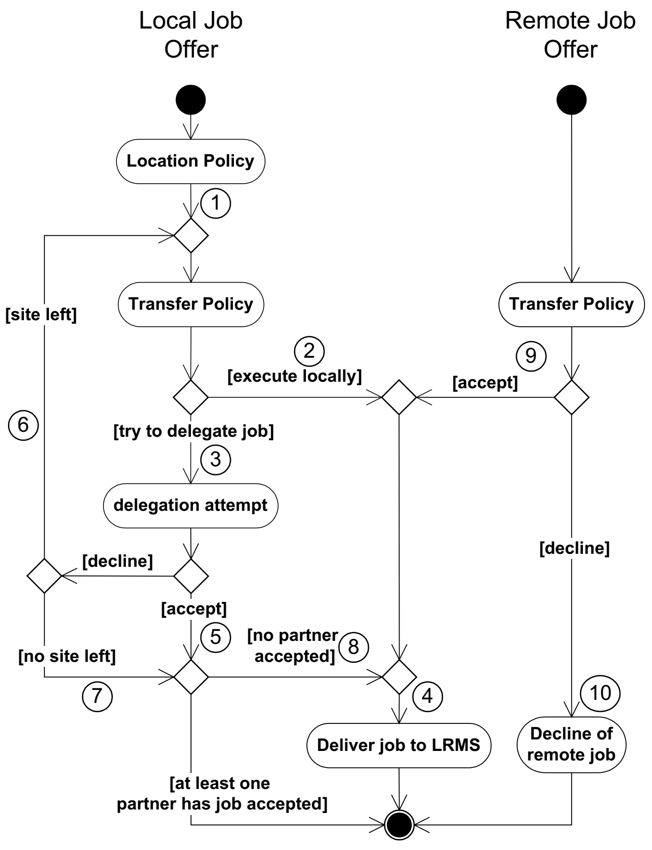


Fig. 2. Decision making concept at GRMS layer

Further, the transfer policy has to decide about jobs that are offered from remote sites and can choose between accepting or declining a job offer. In the former case, the job is immediately forwarded to the LRMS ⑨, while it is rejected in the latter case ⑩.

4 Fuzzy System Based Grid Scheduling Approach

For the design of our GRMS decision policy we apply the method of Fuzzy inference proposed by Takagi, Sugeno and Kang, which is known as the Takagi-Sugeno-Kang (TSK) model in Fuzzy Systems literature [14].

Such a 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. 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. The so composed rule base 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

The general TSK model consists of N_r IF-THEN rules R_i such that

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

where $x_1, x_2, \ldots, x_{N_f}$ are input variables and elements of a vector $\boldsymbol{x}$, 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 $\boldsymbol{x}$. The overall output of the system $y_D(\boldsymbol{x})$ is computed by Equation 2.

$$y_D(\boldsymbol{x}) \quad = \quad \frac{\sum_{i=1}^{N_r} \phi_i(\boldsymbol{x}) y_i}{\sum_{i=1}^{N_r} \phi_i(\boldsymbol{x})} = \frac{\sum_{i=1}^{N_r} \phi_i(\boldsymbol{x})(b_{i0} + b_{i1}x_1 + \ldots + b_{iN_f}x_{N_f})}{\sum_{i=1}^{N_r} \phi_i(\boldsymbol{x})} \tag{2}$$

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

$$\phi_i(\boldsymbol{x}) = g_i^{(1)}(x_1) \wedge g_i^{(2)}(x_2) \wedge \ldots \wedge g_i^{(N_f)}(x_{N_f}) \tag{3}$$

Each rule's recommendation is weighted by its degree of membership with respect to the input vector $\boldsymbol{x}$. 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 here addressed problem of decision making. The specific coding of rules and the output computation will be detailed in the following paragraphs.

4.2 Encoding of Rules

For a single rule R_i every feature h of all N_f features is modeled by a $(\gamma_i^{(h)}, \sigma_i^{(h)})$-Gaussian Membership Function (GMF)[1] with no normalization as shown in Equation 4.

$$g_i^{(h)}(x) = \exp\left\{\frac{-(x-\gamma_i^{(h)})^2}{{\sigma_i^{(h)}}^2}\right\} \tag{4}$$

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 $\sigma_i^{(h)}$.

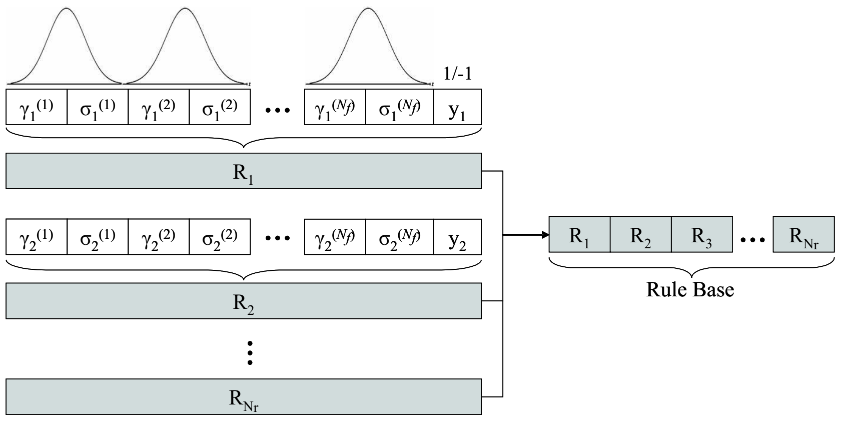

1/-1
γ1(1) σ1(1) γ1(2) σ1(2) ... γ1(Nf) σ1(Nf) y1
R1
γ2(1) σ2(1) γ2(2) σ2(2) ... γ2(Nf) σ2(Nf) y2
R2
RNr
R1 R2 R3 ... RNr
Rule Base

Fig. 3. Encoding pattern for single rules and construction concept for a whole rule base using concatenation.

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. [9].Using this feature description, a single rule's conditional part is composed as shown in Figure 3. 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

[1] Different from the common notation we denote the mean of the GMF by γ to avoid conflicts with the parental population size of Evolution Strategies which is in this paper denoted by μ.

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 for a rule R_i can be chosen as

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

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 3. 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

$$l = N_r(2 \cdot N_f + 1) \tag{6}$$

parameters, see Equation 6. This encoding scheme is perfectly suited as individual representation within an evolutionary algorithm where individuals have the length l.

4.3 Computation of the Controller Decision

To determine the actual controller output for a set of input states $\boldsymbol{x}$ 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 $\phi_i(\boldsymbol{x})$ for rule R_i as shown in Equation 7.

$$\phi_i(\boldsymbol{x}) \quad = \quad \bigwedge_{h=1}^{N_f} g_i^{(h)}(x_h) = \prod_{h=1}^{N_f} \exp\left\{ -\frac{(x_h - \gamma_i^{(h)})^2}{\sigma_i^{(h)2}} \right\} \tag{7}$$

Further, the final controller output Y_D can be computed by considering the leading sign only, see Equation 8,

$$Y_D = \operatorname{sgn}(y_D(\boldsymbol{x})) \tag{8}$$

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 (γ, σ)-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 ν 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 (NWP$_k$) as the first feature, see Equation 9.

$$\text{NWP}_k = \frac{1}{m_k} \sum_{j \in \nu_k} m_j \tag{9}$$

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 features 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 (NJP), see Equation 10.

$$\text{NJP}_j = \frac{m_j}{m_k} \tag{10}$$

With those two selected features we approximate every possible system state.

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. [5] 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, see Equation 6.

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 [11], 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.

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 they possible value range by a ratio of 1:10, see Section 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 (γ, σ)-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.

5 Performance Evaluation

In order to evaluate the performance of our approach in the given scenario, we introduce the tools we use for assessing the optimized exchange policies against a realistic background. To this end, 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 $j \in \tau_k$ that have been initially submitted to site k, see Equation 11. 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 [13], we use the resource consumption $(p_j \cdot m_j)$ of each job as weight. This ensures that neither splitting nor combination of jobs can influence the objective function in a beneficial way.

$$\mathrm{AWRT}_k = \frac{\sum\limits_{j \in \tau_k} p_j \cdot m_j \cdot (C_j(S) - r_j)}{\sum\limits_{j \in \tau_k} p_j \cdot m_j} \tag{11}$$

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 $\min_{j \in \pi_k} \{C_j(S_k) - p_j\}$ 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 12.

$$\mathrm{SA}_k = \sum_{j \in \pi_k} p_j \cdot m_j \tag{12}$$

U_k describes the ratio between overall resource usage and available resources after the completion of all jobs $j \in \pi_k$, see Equation 13.

$$\mathrm{U}_k = \frac{\mathrm{SA}_k}{m_k \cdot \left(C_{max,k} - \min\limits_{j \in \pi_k}\{C_j(S_k) - p_j\}\right)} \tag{13}$$

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 forbidden: 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* $\Delta\mathrm{SA}_k$, which provides a makespan-independent view on the utilization's alteration, see Equation 14.

$$\Delta\mathrm{SA}_k = \frac{\mathrm{SA}_k}{\sum\limits_{j \in \tau_k} p_j \cdot m_j} \tag{14}$$

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[2] provides job submission and execution traces recorded on real-world MPP system sites.Relevant details of the examined cleaned traces are given in Table 1.

Naturally, the total number of available processors differs in workloads which makes it possible to model unequally sized site configurations. Further, the original workloads record time periods of different length. In order to be able to combine different 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. To reflect different site configurations (e.g. small machine and large machine) we combine only workloads that represent a long record period to obtain meaningful results. Therefore, we created shortened versions of

[2] http://www.cs.huji.ac.il/labs/parallel/workload/.

Identifier	#Jobs	m_k	AWRT	U	C_{max}
KTH-5	11780	100	488387.49	64.84	13765377
KTH-6	16699	100	99236.27	68.52	16420782
CTC-5	35360	430	57897.77	63.74	13009718
CTC-6	41839	430	59118.15	67.05	16346403
SDSC05-5	28184	1664	56925.10	45.94	13078215
SDSC05-6	46719	1664	77463.52	70.97	16419455
SDSC00-6	16316	128	413957.04	73.38	17002360

Table 1. Workload characteristics of the used input data, including AWRT in seconds, U in %, and C_{max} in seconds for single site execution with FCFS.

the KTH, CTC, and SDSC05 workloads[3]. 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).

We do not shift the traces regarding their originating timezones. We 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 as idle machines at night can be used by jobs from peak loaded sites at noon, see Ernemann at al. [2]. As we cannot benefit from timezone shifts the presented results might be even better in a global grid.

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.

6 Learning a Basic Rule Set

So far, we presented our Fuzzy controller concept at the GRMS layer, explained how the complex decision making process can be adjusted by a set of parameters, and discussed metrics and test data for the system's performance evaluation. Now, we will introduce the learning process of rule sets for the Fuzzy system, detail the evolution-driven optimization procedure, and present corresponding results.

[3] http://www.it.irf.uni-dortmund.de/~lepping/traces/.

Starting with no rule set at all, we need to bootstrap the system: A first, basic set of rules has to be learned. Although it is generally necessary to create rule sets for both location and transfer policy, see Section 3.2, we start with a pair-wise training approach in order to reduce other partners' influences as much as possible. To this end, we limit job exchange to a single partner only—thus needing no location policy at all—and concentrate on the optimization of the transfer policy. Furthermore, we evolve only one site at a time while applying a static transfer policy on the other site. This policy realizes an *Accept When Fit (AWF)* behavior, which accepts all jobs offered to the GRMS layer for local execution if they do not require more than the currently free resources. Otherwise, the job is offered to the other participants.

The motivation for using a static transfer policy for the training partner and refraining from developing both sites together lies in the application of evolutionary optimization methods: A simultaneous training of two rule bases would lead to a mutual adaption of both training partners. This however, would result in an environment subject to continuous change, making an evolutionary-guided adaptation very difficult: A rule base that leads to good results during one generation might fail completely in the next generation if the partner site changes its behavior completely, too. The aspired robustness in changing environments will be achieved by additional refinements of the concept in the next sections.

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.

Setup	Site	AWRT$_k$	U$_k$	ΔAWRT$_k$	ΔU$_k$	ΔSA$_k$	ΔC$_{max,k}$
I	KTH-5	66100.77 sec.	35.33%	86.47%	-45.51%	-48.56%	5.60%
	CTC-5	63534.29 sec.	71.40%	-9.74%	12.01%	12.15%	-0.14%
II	KTH-5	62884.33 sec.	73.00%	87.12%	12.58%	6.40%	5.49%
	CTC-5	54745.53 sec.	62.70%	5.44%	-1.64%	-1.60%	-0.05%
III	KTH-5	59409.60 sec.	45.15%	87.84%	-30.36%	-34.04%	5.28%
	SDSC05-5	58799.15 sec.	47.34%	-3.29%	3.06%	3.04%	0.01%
IV	KTH-5	70561.62 sec.	53.39%	85.55%	-17.66%	-21.75%	4.97%
	SDSC05-5	54773.39 sec.	46.85%	3.78%	1.98%	1.94%	0.02%
V	CTC-5	45997.73 sec.	58.55%	20.55%	-8.14%	-8.15%	0.01%
	SDSC05-5	57013.44 sec.	47.27%	-0.16%	2.90%	2.91%	-0.02%
VI	CTC-5	57069.59 sec.	63.30%	1.43%	-0.69%	-0.51%	-0.20%
	SDSC05-5	49916.01 sec.	46.04%	12.31%	0.22%	0.18%	0.02%

Table 2. Results for the pair-wise rule base training. The gray shaded rows indicated the optimized rule base.

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 (ΔSA) 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 is for this site very advantageous.

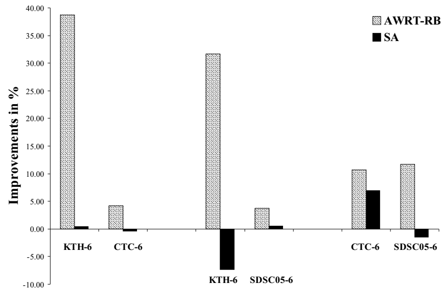
Furthermore, in Setup III and IV the small KTH interacts with the very large SDSC05 compute center and naturally the KTH benefits from more available resources. It is remarkable that also the SDSC05 can improve its AWRT for more than 3%, see Setup IV. At the same time, the Squashed Area is slightly increased 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.

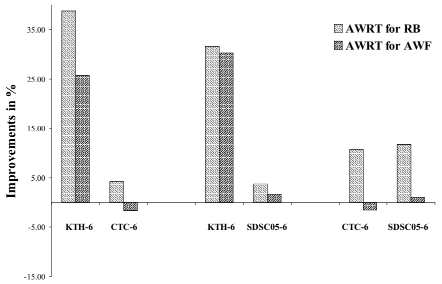
6.2 Robustness of Trained Rule Sets

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 4(a), the changes in AWRT and SA are depicted when both partners apply their learned rule bases to previously unknown job submissions.



(a) Application to non-trained data sets.



(b) RB approach compared to AWF.

Fig. 4. AWRT and SA improvements for the optimized rule sets.

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 4(b) the AWRT improvements in comparison to the AWF transfer policy. 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 the here proposed location policy, a site calculates the AWRT metric with respect to every exchange partner. To 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 5(a) 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

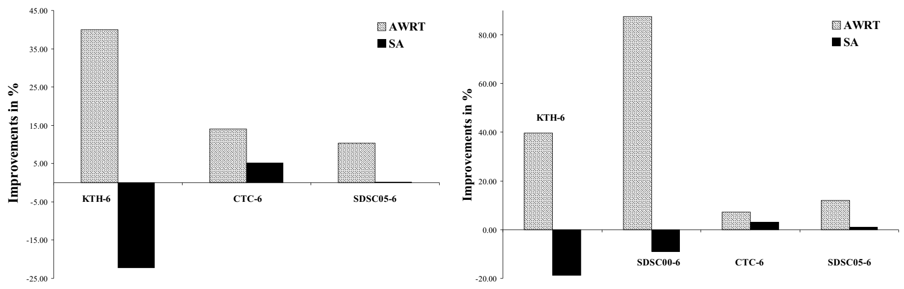
As mentioned in Section 3.2, the rule based transfer policy is applied to each partner site separately. If a new partner arises a transfer policy has to be selected from the pool of all learned transfer policies. To identify the best suitable transfer policy we assume a correlation between delegation 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 machines 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 5(b) the results for interaction with three other partners 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 10%.

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.



(a) Three-site Grid scenario (b) Four-site Grid scenario

Fig. 5. AWRT and SA improvements for the optimized rules in a three-site Grid scenario on non-trained data sets (a) and in a four-site Grid scenario with SDSC00 as unknown participant (b).

9 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 pair-wise 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.

References

1. O. Cordón, F. Herrera, F. Hoffmann, and L. Magdalena. Evolutionary Tuning and Learning of Fuzzy Knowledge Bases. In *GENETIC FUZZY SYSTEMS*, volume 19 of *Advances in Fuzzy Systems - Applications and Theory*. World Scientific, Singapore, July 2001.

2. Carsten Ernemann, Volker Hamscher, and Ramin Yahyapour. Benefits of global grid computing for job scheduling. In *Proceedings of the Fifth IEEE/ACM International Workshop on Grid Computing (GRID'04)*, pages 374–379. IEEE Computer Society, 2004.
3. D. G. Feitelson and B. Nitzberg. Job characteristics of a production parallel scientific workload on the NASA ames iPSC/860. In D. G. Feitelson and L. Rudolph, editors, *Proceedings of the 1st Job Scheduling Strategies for Parallel Processing*, volume 949 of *Lecture Notes in Computer Science (LNCS)*, pages 337–360. Springer, 1995.
4. Carsten Franke, Frank Hoffmann, Joachim Lepping, and Uwe Schwiegelshohn. Development of Scheduling Strategies with Genetic Fuzzy Systems. *Applied Soft Computing*, 8(1):706–721, January 2008.
5. Carsten Franke, Joachim Lepping, and Uwe Schwiegelshohn. Genetic Fuzzy Systems applied to Online Job Scheduling. In *Proceedings of the 2007 IEEE International Conference on Fuzzy Systems*, pages 1573–1578, London, June 2007. IEEE Press.
6. Fabrizio Gagliardi, Bob Jones, Francois Grey, Marc-Elian Begin, and Matti Heikkurinen. Building an infrastructure for scientific grid computing: status and goals of the egee project. *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences*, 363(1833):1729–1742, 2005.
7. Christian Grimme, Joachim Lepping, and Alexander Papaspyrou. Prospects of Collaboration between Compute Providers by means of Job Interchange. In Eitan Frachtenberg and Uwe Schwiegelshohn, editors, *Proceedings of Job Scheduling Strategies for Parallel Processing*, volume 4942 of *Lecture Notes in Computer Science (LNCS)*, pages 132–151. Springer, June 2007.
8. Christian Grimme, Joachim Lepping, and Alexander Papaspyrou. Discovering Performance Bounds for Grid Scheduling by using Evolutionary Multiobjective Optimization. In M. Keijzer et al., editors, *Prococeedings of the Genetic and Evolutionary Computation Conference (GECCO 2008)*, pages 1491–1498, Atlanta, Georgia, USA, July 2008. ACM, ACM Press.
9. C.-F. Juang, J.-Y. Lin, and C.-T. Lin. Genetic Reinforcement Learning through Symbiotic Evolution for Fuzzy Controller Design. *IEEE Transactions on System, Man and Cybernetics*, 30(2):290–302, April 2000.
10. Dan C. Marinescu, Ladislau Boloni, Ruibing Hao, and Kyung koo Jun. An alternative model for scheduling on a computational grid. In *Proceedings of ISCIS'98, the Thirteenth International Symposium on Computer and Information Sciences, Antalya*, pages 473–480. IOP Press, 1998.
11. H.-P. Schwefel. *Evolution and Optimum Seeking*. John Wiley & Sons, New York, 1995.
12. Uwe Schwiegelshohn, Andrei Tchernykh, and Ramin Yahyapour. Online scheduling in grids. In *22nd IEEE International Parallel and Distributed Processing Symposium (IPDPS 2008)*. IEEE Press, April 2008. CD-ROM.
13. Uwe Schwiegelshohn and Ramin Yahyapour. Fairness in parallel job scheduling. *Journal of Scheduling*, 3(5):297–320, 2000.
14. T. Takagi and M. Sugeno. Fuzzy Identification of Systems and Its Applications to Modeling and Control. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-15(1):116–132, 1985.