UNIVERSITY OF DORTMUND

REIHE COMPUTATIONAL INTELLIGENCE

COLLABORATIVE RESEARCH CENTER 531

Design and Management of Complex Technical Processes and Systems by means of Computational Intelligence Methods

On Advantages of Scheduling using Genetic Fuzzy Systems

Carsten Franke, Joachim Lepping and Uwe Schwiegelshohn

No. CI-214/06

Technical Report ISSN 1433-3325 June 2006

Secretary of the SFB 531 · University of Dortmund · Dept. of Computer Science/XI
44221 Dortmund · Germany

This work is a product of the Collaborative Research Center 531, "Computational Intelligence," at the University of Dortmund and was printed with financial support of the Deutsche Forschungsgemeinschaft.

On Advantages of Scheduling using Genetic Fuzzy Systems

Carsten Franke*, Joachim Lepping, and Uwe Schwiegelshohn

Robotics Research Institute, Dortmund University, 44221 Dortmund, Germany
(email: {carsten.franke, joachim.lepping, uwe.schwiegelshohn}@udo.edu)

Abstract. In this paper, we present a methodology for automatically generating online scheduling strategies for a complex scheduling objective with the help of real life workload data. The scheduling problem includes independent parallel jobs and multiple identical machines. The objective is defined by the machine provider and considers different priorities of user groups. In order to allow a wide range of objective functions, we use a rule based scheduling strategy. There, a rule system classifies all possible scheduling states and assigns an appropriate scheduling strategy based on the actual state. The rule bases are developed with the help of a Genetic Fuzzy System that uses workload data obtained from real system installations. We evaluate our new scheduling strategies again on real workload data in comparison to a probability based scheduling strategy and the EASY standard scheduling algorithm. To this end, we select an exemplary objective function that prioritizes some user groups over others.

1 Introduction

In this paper, we address the development of a methodology to automatically generate scheduling strategies for Massively Parallel Processing (MPP) systems that consider the providers' preferences. The scheduling problem consists of n independent non-clairvoyant jobs that are submitted by different users over time. The scheduling strategy is responsible to assign the available processors of the MPP system to those jobs. However, the machine providers in real scenarios have different relationships to the various users or user groups. Those different relationships lead to different prioritizations of the users and their corresponding jobs. Consequently, the scheduling strategy needs to incorporate those priorities during the scheduling process.

Many installations use partitions [14] or quotas [31] to implement this kind of prioritizations of different user groups. However, those attempts result in a low system utilization in most of the cases [14]. Hence, we present the development of a rule based scheduling system that is able to generate schedules with a higher quality in terms of the provider preferences while not decreasing system utilization. To our knowledge, there is no similar work that is able to incorporate the user group prioritizations in a similar way.

The development of scheduling strategies for MPP systems is based on workload traces originating from real installations, see for example Heine et al. [16]. Such workload data include all hidden job dependencies, patterns and feedback mechanisms. For MPP systems several workloads are available, see the standard workload archive maintained by Feitelson [13], that are for instance described by Chapin [5]. Although those data are rather old they suffice for our purpose. So far, workload models are rarely used to develop scheduling algorithms as they are not able to describe workload traces with an acceptable accuracy, see Song et al. [28] and the given references there.

The online job scheduling on MPPs is usually non-clairvoyant as the processing time p_j of job j is not available at its release date r_j. However, users are often required to provide estimates $\bar{p}_j$ of the processing time that are mainly used to determine faulty jobs whose processing time exceeds the estimate. Further, parallel jobs on MPPs are typically not moldable or malleable, that is, they need concurrent and exclusive access to $m_j \leq m$ machines during the whole execution phase. The value m_j is provided at the release date r_j by the user. Finally, the completion time of job j in a schedule S is denoted by $C_j(S)$. As preemption is not allowed in many MPPs, each

* Born Carsten Ernemann.

job starts its execution at time $(C_j(S) - p_j)$. Unfortunately, the available workload data do not provide any user group information nor define any complex scheduling objective. To address the user group problem, we are using the work of Song et al. [29], who have shown that users can be reasonably well partitioned into 5 groups for all available MPP workload traces. Those groups are differentiated with respect to job characteristics and frequency of job submissions. Within this work, we will also use 5 different user groups. However, we will use the user's resource consumption as the differentiation criterion. The binary function $\varrho_i(j)$ is used to state whether job j belongs to user group i $(\varrho_i(j) = 1)$ or not $(\varrho_i(j) = 0)$.

We present a methodology to automatically generate a rule based scheduling system that is able to produce good quality schedules with respect to a given complex provider objective. Note that our methodology is not restricted to a specific user group selection.

The individual preferences of the machine providers are expressed using a complex objective function that is generated by combining well known simple basic objectives. Even if different providers use the same objective functions for the various groups, the transformation of a generic multi-objective scenario into a specific scheduling problem with a single objective depends on the actual priorities assigned to the user groups and is likely to be individual. Hence, we focus on the development of a suitable methodology and do not generate a single scheduling strategy. Without loss of generality, we exemplarily select a complex objective function to demonstrate the feasibility of our approach. Here, we present a rule based scheduling that is able to adapt to various scenarios. So far, the use of rule based systems in scheduling environments is rare. Nevertheless, first attempts [10, 4] have shown the feasibility of such an approach. However, those scheduling systems are all based on single simple objective evaluation functions that are not optimized.

The proposed scheduling process is divided into two steps. In the first step, the queue of waiting jobs is reordered according to a sorting criterion. Then an algorithm uses this order to schedule the jobs onto the available machines in the second step. Based on the present scheduling state, the rules determine the sorting criterion and the scheduling algorithm. In order to guarantee general applicability, the system classifies all possible scheduling states. This classification considers the scheduling decisions in the past, the actual schedule, and the current waiting queue. Note that we have chosen some classification features exemplarily. Other possible features can be used as well for this task. Our feature selection only serves the purpose to illustrate our methodology.

As already stated in many other publications, see for example Ernemann et al. [6, 7], a local scheduling decision influences the allocation of future jobs. Hence, the effect of a single decision cannot be determined individually. Therefore, the whole rule base is only evaluated after the complete scheduling of all jobs belonging to a workload trace. This has a significant influence on the learning method to generate this rule base as this type of evaluation prevents the application of a supervised learning algorithm, see Hoffmann [17]. Instead, the reward of a decision is delayed and determined by a critic. Furthermore, the generation of an appropriate situation classification is not known in advance and must be generated implicitly while constructing the rule based scheduling system.

The various design concepts for Fuzzy logic controllers often use Evolutionary Algorithms to adjust the membership function as well as to define the output behavior of individual rules, see, for example, Hoffmann [17]. Especially Genetic Fuzzy Systems have been proven to deal with such classification and automatic rule base generation problems in a suitable way. All those Genetic Fuzzy Systems either encode single rules (*Michigan approach*, Bonarini [3]) or complete rule bases (*Pittsburgh approach*, Smith [27]).

Within this work, the determination of a Genetic Fuzzy System is realized using the Pittsburgh approach. In this case, each individual represents a whole rule base. During the evolution, the individual rules are adjusted in order to better fit to the given situations. Furthermore, we will present a Coevolutionary approach that uses two rule bases, one for the determination of the sorting criterion and one for the scheduling algorithm that is applied. Both rule bases evolve independently with the only exception that during the quality assignment one individual from each rule base must be selected.

We use an Evolution Strategies for the optimization of the rule based scheduling system. This is in contrast to the majority of Genetic Fuzzy Systems, see Hoffmann [17]. As our membership functions include real valued parameters, Evolution Strategies are superior to Genetic Algorithms in this case, see Bäck and Schwefel [1].

To finally show the results of our approach, we use a linear priority function which favors user group 1 over user group 2 over all other user groups. The choice of another priority function may lead to different results but does not affect the feasibility of our methodology. Due to the lack of a scheduling strategy supporting priority functions, no priority functions are available in practice. Therefore, we had to define one.

For the evaluation of the derived scheduling strategy we present the distance of this schedule from the Pareto front of all feasible schedules for this workload, as generated by Ernemann et al. [8]. Although the generation of an approximate Pareto front is not subject of this paper, two restrictions must be noted:

1. For real workloads, we are only able to generate approximate Pareto fronts. Therefore, schedules of this front are not guaranteed to be lower bounds.
2. The schedules are generated off-line. On-line methods may not be able to achieve as good results due to the on-line constraints.

On purpose, we selected a criterion where user groups with a high computing demand are preferred over user groups with a low demand. Then classical scheduling algorithms will typically generate acceptable results. This is not true for a prioritization of a user group with a low resource demand. Moreover, we also show the results of the best conventional strategy that does not support priorities.

The remainder of this paper is organized as follows. In Section 2, we introduce the underlying scheduling system, Evolutionary Algorithms, and Genetic Fuzzy Systems in more detail. The scheduling objectives and features are presented in Section 3. Then the model of our approach is described in Section 4. This is followed by a detailed analysis of the system behavior and an evaluation of the results. The paper ends with a brief conclusion.

2 Background

This section introduces the main scheduling algorithms and their application within our rule based scheduling system. Furthermore, the concept of Evolution Strategies is presented. Those strategies are used to optimize the rule based scheduling system.

2.1 Scheduling Concepts

As already mentioned, scheduling strategies of high performance parallel computers need to pay more attention to certain users or user groups in order to achieve a higher degree of satisfaction for them. Priority or membership information are not available in the workloads. Hence, we use the resource consumption as a grouping criterion such that user group 1 represents all users with a higher resource consumption whereas all users in group 5 have a very low resource demand. Details of the user group definitions are provided by Ernemann et al. [8].

As already introduced, a state of a scheduling system mainly consists of the current schedule, that describes the actual allocation of processor nodes to certain jobs, the scheduling results achieved so far, and the queue of waiting jobs. This waiting queue is typically ordered.

In most cases, a static ordering like sorting by submission time or sorting by estimated runtime is applied. In some other cases, the waiting queue is dynamically reordered depending on the system state by using a more complex sorting criterion that may for instance consider limits of the waiting time.

The various scheduling algorithms mainly differ in the way they select the next job from the sorted waiting queue to insert it into the existing schedule, that is, they obey different restrictions when choosing the next job. This results in different algorithmic complexities and correspondingly different execution times for the scheduling algorithms.

In the following, we present four selected scheduling algorithms in increasing order of algorithmic complexity. Note that the first three algorithms use a statically sorted waiting queue while the last algorithm dynamically reorders this queue.

- *First Come First Serve (FCFS)* starts the first job of the waiting queue whenever enough idle resources are available. Thus, this algorithm has a constant complexity as the scheduler always only tests whether the first job can be started immediately if a job in the schedule has completed its execution or a new job has risen to the top of the waiting queue.
- *List Scheduling* as introduced by Graham [15] is not applied in this work. However, it serves as the basic template for the two backfilling variants. By applying List Scheduling, the scheduler tries to find the first job within the queue of waiting jobs, that can be started on the currently idle resources. Again, the algorithm uses the sorted queue. The complexity is higher than in the case of FCFS as in the worst case, the whole queue is tested each time the scheduling procedure is initiated.
  - *EASY Backfilling (EASY)* is similar to the original List Scheduling. However, if the first job within the waiting queue cannot be started immediately the algorithm estimates the completion time of this job. To this end, a runtime estimation provided by the user is needed. Then EASY tries to find an allocation for the following jobs of the waiting queue on the currently idle resources while ensuring that the *first job* is not further delayed. This algorithm requires more time than List Scheduling, as the scheduler needs to estimate the processing of the first job in case that it cannot be started directly.
  - *Conservative Backfilling (CONS)* extends the concept of EASY. Here, the scheduler tries to find the next job within the waiting queue, that can be started immediately while ensuring that *no previous job* within the queue is further delayed. This results in a much higher complexity of the scheduling algorithm as in the worst case, the completion time of all jobs within the waiting queue except of the last job must be estimated each time the scheduling process is initiated.
- *Greedy Scheduling (Greedy)* uses a dynamically sorted waiting queue contrary to the already introduced scheduling algorithms. To this end, the algorithm defines a complex sorting criterion. Each time, the Greedy scheduling process in started, the queue is sorted according to this criterion. Then, a simple FCFS is applied. The complexity of this algorithm is potentially high as the execution of the sorting function for each job within the waiting queue may be computationally expensive. Furthermore, the necessary sorting of all jobs must be taken into account. Greedy has the advantage to specify user or user group dependent preferences within the complex sorting criterion. In our case, the complex sorting function within the Greedy algorithm tries to schedule jobs of the user groups 1 and 2 earlier unless jobs from other user groups are already waiting for a very long time. This sorting criterion is modeled according to our scheduling objective. For more details on the used sorting criterion, see Ernemann et al. [8].

2.2 Evolution Strategies

To integrate those scheduling algorithms into an appropriate rule base system, we use Evolution Strategies, see Beyer and Schwefel [2], which are a subclass of Evolutionary Algorithms. Those algorithms are stochastic search methods that mimic the behavior of natural biological evolution. They operate on a population of μ individuals and apply genetic operators like selection, mutation and recombination to breed λ offspring individuals from those μ parent individuals. Within this paper, we do not provide a deeper insight into Evolution Strategies. Furthermore, for all details about specific genetic operators, we simply refer to references in the remainder of this paper.

2.3 Fuzzy Systems

Within this work, we aim to generate rule based scheduling systems. To this end, several approaches can be used. On the one hand, a static approach of defining strict boundaries for certain features and assigning a corresponding combination of sorting criteria and scheduling algorithm is possible. On the other hand, one may apply the more flexible approach of generating a Genetic Fuzzy System.

In our case, neither precise knowledge about the assignment of certain scheduling strategies to certain situations nor training data are available. Furthermore, individual scheduling decisions cannot be evaluated directly, but only after all jobs have been assigned to resources, see Section 1. Hence, the award for the assignment of scheduling strategies to situations is given by a critic only at the end of scheduling a whole workload trace. Furthermore, the generation of an appropriate situation classification is not known in advance and has to be generated implicitly during the generation of the rule based scheduling system.

3 Scheduling Objectives and Features

Within this section, we will introduce several simple scheduling objectives, which have been used to construct more complex evaluation functions for the whole scheduling procedure. However, our methodology is not limited to the presented objective and can be extended to any other criteria.

Furthermore, we apply several features to classify possible scheduling situations within our rule based scheduling system. The concept of this work can be extended to other features as well. Note that objectives evaluate the whole scheduling process at the end of a simulation while features only describe the current state of the system.

As mentioned in Section 1, the complex objective function of a machine provider in our case is based on individual properties of users or user groups. Therefore, both the objective and the feature set refer to those properties and to the overall performance of the whole system.

First, we introduce some definitions and notations.

- $(p_j \cdot m_j)$ as the Resource Consumption of a single job j,
- τ the set of all n jobs within our scheduling system,
- $\xi(t)$ the set of already finished jobs at time t,
- $\pi(t)$ the set of running jobs at time t, and
- $\nu(t)$ the set of waiting jobs at time t.

3.1 Scheduling Objectives

During the development of scheduling systems, an evaluation function is needed in order to describe the achieved quality. We generate our evaluation function by combining simple, commonly used scheduling objectives. Within this work, we exemplarily use 7 of those simple objectives.

Overall Utilization (U):

$$\mathrm{U} = \frac{\sum\limits_{j\in\tau} p_j \cdot m_j}{m \cdot \left(\max\limits_{j\in\tau} \{C_j(S)\} - \min\limits_{j\in\tau}\{C_j(S) - p_j\}\right)} \tag{1}$$

Average Weighted Response Time (AWRT) over all jobs of all users:

$$\mathrm{AWRT} = \frac{\sum\limits_{j\in\tau} p_j \cdot m_j \cdot (C_j(S) - r_j)}{\sum\limits_{j\in\tau} p_j \cdot m_j} \tag{2}$$

AWRT objective for user groups 1 to 5:

$$\mathrm{AWRT}_i = \frac{\sum\limits_{j\in\tau} p_j \cdot m_j \cdot (C_j(S) - r_j) \cdot \varrho_i(j)}{\sum\limits_{j\in\tau} p_j \cdot m_j \cdot \varrho_i(j)} \;,\; i \in \{1, 2, \ldots 5\} \tag{3}$$

As we have the AWRT_i for the 5 user groups, the AWRT for all users, and the utilization U the complex objective function in our system can be defined by using those 7 simple objectives.

3.2 Feature Definitions

Next, we present 7 features that are used for classification of system states within our rule base scheduling system.

In order to define our first feature, the Average Weighted Slowdown, we need to introduce the *Slowdown* (SD_j) for a single job j within schedule S:

$$\text{SD}_j = \frac{C_j(S) - r_j}{p_j} \tag{4}$$

SD_j will reach its minimum value of 1 if job j does not wait before it starts execution. Then the release date is identical with the job's start time. Normally, the range of this feature can be limited to the interval of [1,100] as values greater than 10 occur very rarely in practice.

The feature *Average Weighted Slowdown* (SD) for all already processed jobs $j \in \xi(t)$ uses the same weighting as defined for the AWRT.

$$\text{SD} = \frac{\sum\limits_{j\in\xi(t)} p_j \cdot m_j \cdot (C_j(S) - r_j)}{\sum\limits_{j\in\xi(t)} p_j^2 \cdot m_j} \tag{5}$$

This measure indicates the average delay of jobs between their release and start time for the past. Further, this feature represents the scheduling decisions in the past as only already finished jobs are used to calculate this feature. Here, we have not limited the window for SD. In practical cases, a limitation to, for instance, the last month may be appropriate.

The *Momentary Utilization* (U_m) of the whole parallel computer at time t:

$$\text{U}_m = \frac{\sum\limits_{j\in\pi(t)} m_j}{m} \tag{6}$$

The *Proportional Resource Consumption of the Waiting Queue for User Group i* (PRCWQ_i):

$$\text{PRCWQ}_i = \frac{\sum\limits_{j\in\nu(t)} \bar{p}_j \cdot m_j \cdot \varrho_i(j)}{\sum\limits_{j\in\nu(t)} \bar{p}_j \cdot m_j} \tag{7}$$

Note that the real processing time p_j is unknown for the jobs in the waiting queue. Therefore, we use the estimated processing time $\bar{p}_j$ instead. PRCWQ_i represents the relative part of the estimated resources consumption of user group i to all jobs within the waiting queue. Remember, we are using 5 user groups within our system. Hence, those five feature values represent the expected future of the system. Using these features, the scheduling system is enabled to react on a changed demand of the various user groups.

4 Rule Based Scheduling Systems

As stated in Section 1, local scheduling decisions influence the allocation of future jobs. Hence, the effect of a single decision cannot be determined individually. Therefore, the whole rule base is only evaluated after the complete scheduling of all jobs belonging to a workload trace. This has a significant influence on the learning method to generate this rule base as the evaluation prevents the application of a supervised learning algorithm. Instead, the reward of a decision is delayed and determined by a critic. Furthermore, the generation of an appropriate scheduling situation classification is not known in advance and has to be generated implicitly during the generation of the rule base scheduling system.

For a rule based scheduling approach, every possible scheduling state must be assigned to a corresponding situation class that is described using the already introduced features. A complete rule base RB consists of a set of rules R_i. Each rule R_i contains a conditional and a consequence part. The conditional part describes the conditions for the activation of the rule using the defined features. The consequence part represents the corresponding scheduling strategy recommendation.

In order to specify all scheduling states in an appropriate fashion each rule defines certain partitions of the feature space within the conditional part. The rule base system must contain at least one activated rule for each possible system state.

As already mentioned, the scheduling strategy specifies

1. a *sorting criterion* for the waiting queue $\nu(t)$ and
2. a *scheduling algorithm* that uses the order of $\nu(t)$ to schedule one or more jobs.

We use the term *strategy* to describe the whole scheduling process that consists of both steps. An *algorithm* only describes the procedure of the second step that uses the already sorted waiting queue.

First, the chosen sorting criterion is used to determine the sequence of jobs within the waiting queue. Second, the selected scheduling algorithm is used to find a processor allocation for at least one job of the sorted waiting queue. We have chosen four different sorting criteria. Those sorting criteria are only examples that are used to demonstrate our rule based scheduling approach. Other sorting criteria are possible and could easily be incorporate into the system. Our four sorting criteria are:

- *Increasing Number of Requested Processors:* Preference of jobs with little parallelism and therefore higher utilization. This sorting provides the potential gain of being able to insert many jobs into the current schedule as jobs with a smaller amount of requested processors are often easier to schedule.
- *Increasing Estimated Run Time:* Preference of short jobs and therefore higher job throughput.
- *Decreasing Waiting Time:* Preference of long waiting jobs. This sorting criterion provides a higher fairness as the jobs are processed according to their submission. Jobs with a higher waiting time are selected first.
- *Decreasing User Group Priority:* Preference of jobs from users with a higher resource demand. The sorting by user groups provides a higher ranking for all jobs of users with a higher overall resource demand according to their user group assignment. This criterion reflects our objective function.

The selected scheduling algorithm is one of the four methods presented in Section 2.1. Note that Greedy is already a complete scheduling strategy while the other scheduling algorithms of Section 2 must be supplement with a sorting criterion of $\nu(t)$. Again, the set of scheduling algorithm can be extended for other rule base systems. The general concept of the rule based scheduling approach is depicted in Figure 1.

As 4 different sorting criteria with 3 possible scheduling algorithms and the combined Greedy strategy are available, we have to chose one of 13 strategies for each possible system state. However, it is not practicable to test all possible assignments in all possible states. For example, lets assume a very coarse division of each feature into only 2 partitions. Then 13 possible strategies and 7 features result in $13^{2^7} \approx 3.84 \cdot 10^{142}$ simulations if all combinations in all possible situation states are tested. Additional problems occur during the generation of a rule based scheduling system as the number and reasonable partitions of features, that are required to describe the situation classes in an appropriate way, are generally unknown in advance.

Hence, we introduce three possible approaches to derive a rule based scheduling system using only a limited number of simulations.

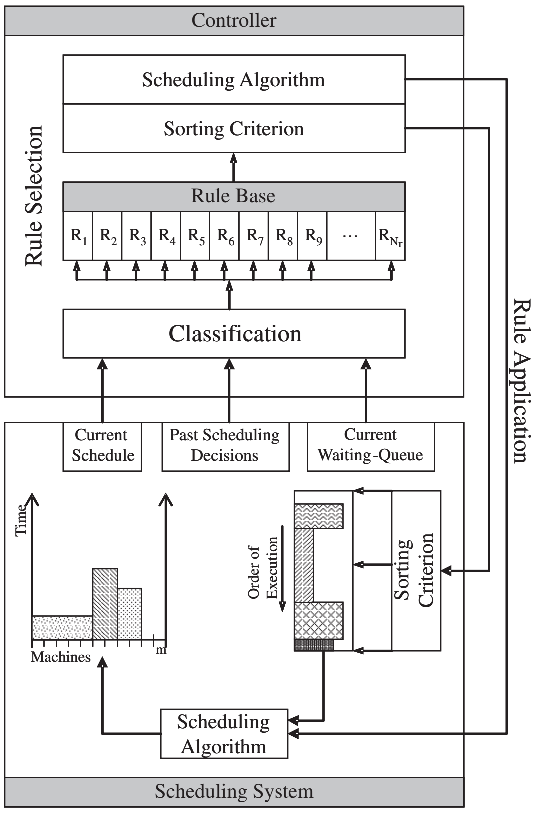

Fig. 1. General Concept of the Rule Based Scheduling Approach.

4.1 Probability Driven Rule Base Development

A rigid rule base system uses N_F features with a fixed number of intervals for each feature ω. That is, each feature ω has $(N_{part,\omega} - 1)$ static bounds, that divide the possible value range of ω into $N_{part,\omega}$ partitions. The static bounds are specified before the assignment of sorting criteria and scheduling algorithms to the various situation classes are extracted. The concept of such fixed partitions is shown in Figure 2.

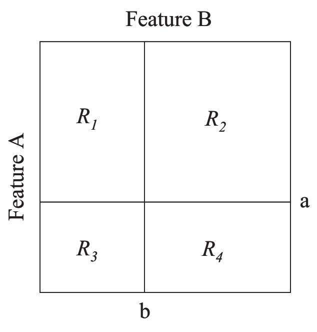

Fig. 2. Example partitioning of the feature space and the resulting set of rules $R_1 \dots R_4$.

Generally, a larger number of partitions $N_{part,\omega}$ of a feature ω potentially leads to a more accurate rule set while more situation classes must be optimized. Overall, this results in N_r situation classes that must be provided to cover all possible system states with

$$N_r = \prod_{\omega=1}^{N_F} N_{part,\omega} \ .$$

The described rigid rule based system activates only a single rule in any system state. Hence, the output recommendation of this single activated rule is the output of the whole scheduling system.

In this work, we assume one division of the intervals of SD and PRCWQ_1 to PRCWQ_5 respectively. This leads to two partitions in each case. Further, we use two divisions for the U_m feature, resulting in three partitions. Overall, this produces ($N_r = 2^6 \cdot 3 = 192$) different situation classes that are needed to build a complete rule base.

Furthermore, we have evaluated several different division values for the situation class features. The partitions which achieved the best results are used for the rigid rule base system development,see Table 1.

Feature	Intervals
SD	[1-2],]2-100]
$\text{U}_{m[\%]}$	[0-75],]75-85],]85-100]
$\text{PRCWQ}_{1[\%]}$	[0-20],]20-100]
$\text{PRCWQ}_{2[\%]}$	[0-20],]20-100]
$\text{PRCWQ}_{3[\%]}$	[0-25],]25-100]
$\text{PRCWQ}_{4[\%]}$	[0-25],]25-100]
$\text{PRCWQ}_{5[\%]}$	[0-25],]25-100]

Table 1. Feature Partitions for the Rigid Rule Based Scheduling Systems.

Such a rigid rule based scheduling system has the advantage of a simple implementation and easy interpretation. Future scheduling development may benefit from knowledge gained through this kind of interpretation. The selected scheduling algorithms and sorting criteria for a certain scheduling situation can directly be extracted from the corresponding rules without further computation.

Rule bases are generated by assigning potential scheduling strategies to rules in a random fashion such that each scheduling strategy is assigned to each rule the same number of times. Hence, not all rule bases are generated in a completely random way. Remember that the conditional part is rigid and does not vary. Thus, a single rule describes a single scheduling situation class completely.

Then we use those rule bases to produce schedules for the given workload data and evaluate those schedules with the help of the complex scheduling objective. Thus, each schedule results in a scalar objective value. The assignment of a special scheduling strategy to a rule is evaluated by adding the scalar objective values of all schedules that were generated using this assignment. Finally, we build the resulting rule base by assigning the scheduling strategies with the smallest sum of the objective values to the individual rule as we assume a minimization of the objective function. This approach is able to reduce the number of required simulations significantly as we only approximate the optimal assignments. In general, the performance can be increased by generating more rule bases. However, the trade-off between a better performance and more required simulations should be kept in mind.

A parameter p describes how often a scheduling strategy is assigned to a single rule. This parameter p influences the number of required simulations that is given by the product of the number of possible scheduling strategies (N_{Ω}) and the parameter p. In our simulations $p = 50$

turned out to be a good compromise between the required number of simulations and the scheduling quality. This results in our case in $(13 \cdot 50 = 650)$ simulations which is significantly less than the required number of simulation for all possible assignments.

Unfortunately, the fixed division of the whole feature space has a critical influence on the performance of the scheduling system. At the moment, no mechanism is available that automatically adjusts the defined partitions.

Using this approach, we avoid the excessive amount of simulations that must be performed in order to generate the rule base. Further, our approach pays attention to the cooperation aspect of the rules within the final rule base as the evaluation of the assignment of a special scheduling strategy to the consequence part of a rule is based on several simulations with varying strategy assignments for all other rules.

4.2 Scheduling Strategies Based on Genetic Fuzzy Systems

The previously presented scheduling system has several drawbacks regarding the generation of an appropriate rule based scheduling system. Mainly, the static number of feature partitions and the static pre-defined bounds for these partitions are not flexible enough and may lead to bad scheduling results. Furthermore, the whole feature space needs to be divided and appropriate scheduling strategies assigned to each individual partition. Hence, the number of rules cannot be varied.

Consequently, we need a method that automatically adjusts the partition of the feature space and assigns appropriate scheduling strategies to the resulting regions in parallel. *Genetic Fuzzy Systems*, see Hoffmann [17], provide the capabilities to solve those problems. As already mentioned within the introduction, our Genetic Fuzzy System uses the Pittsburgh approach to encode a whole rule base in a single individual. Further, we parameterize the resulting system with Evolution Strategies.

Before the different rule base encoding schemes are explained in detail, we introduce the encoding of individual rules and detail the computation of the final Fuzzy controller output.

Coding of Fuzzy Rules Our Genetic Fuzzy Systems are based on the traditional Takagi-Sugeno-Kang (TSK) model [30] for Fuzzy systems. The used coding schemes and learning techniques are adapted and slightly modified from the work of Juang et al. [21] and Jin et al. [20].

For a single rule R_i, every feature ω of all N_F features is modeled from a Gaussian Membership Function, $(\mu_i^{(\omega)}, \sigma_i^{(\omega)})$-GMF

$$g_i^{(\omega)}(x) = \frac{1}{\sigma_i^{(\omega)}\sqrt{2\pi}} \exp\left\{ \frac{-(x-\mu_i^{(\omega)})^2}{2{\sigma_i^{(\omega)}}^2} \right\} .$$

In Figure 3 a sample (5,0.75)-GMF is depicted.

A feature is then represented by a pair of real values $\mu_i^{(\omega)}$ and $\sigma_i^{(\omega)}$. The $\mu_i^{(\omega)}$ value is the center of the feature GMF that is covered by the rule R_i. Therefore, this value defines a domain in the feature space where the influence of the rule is very high. Note that, when using a so defined GMF as feature description, the condition

$$\int_{-\infty}^{\infty} g_i^{(\omega)}(z)dz = 1 \; \forall \; i \in \{1, \dots N_r\} \wedge \; \omega \in \{1, \dots N_F\}$$

always holds. In other words, for increasing $\sigma_i^{(\omega)}$ values, the peak value of the GMF decreases because the integral remains constant. Using this property of a GMF, we are able to reduce the influence of a rule for a certain feature completely by setting $\sigma_i^{(\omega)}$ to a very high value. Theoretically for $\sigma_i^{(\omega)} \rightarrow \infty$, a rule has no influence for this feature anymore. With this approach, it is also

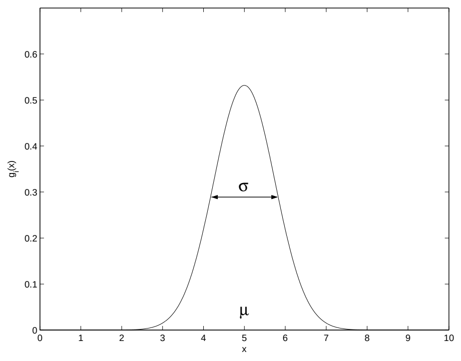

Fig. 3. Gaussian Membership Function with $\mu_i^{(\omega)} = 5$ and $\sigma_i^{(\omega)} = 0.75$.

possible to establish a kind of default value that is used if no other peaks are defined in a feature domain. Based on this feature description, a single rule can be described by

$$R_i = \left\{ g_i^{(1)}(x),\ g_i^{(2)}(x),\ \ldots\ g_i^{(N_F)}(x), \boldsymbol{\Omega}_i(R_i) \right\}.$$

The consequence part $\boldsymbol{\Omega}_i$ of every rule R_i, $i \in \{1, \ldots N_r\}$, includes a weighted recommendation for all N_Ω possible outputs. Therefore, the consequence part of rule R_i is described by a vector

$$\boldsymbol{\Omega}(R_i) = \left(w_{i1}\ w_{i2} \ldots w_{iN_\Omega} \right).$$

We restrict the possible weight values to elements of the set $\{-5,\ -1,\ 0,\ 1,\ 5\}$. The value -5 represents a *particularly unfavorable* connection while 5 is *particularly favorable* one. The other possible weights can be interpreted accordingly. We use a non-linear weight scaling in order to force distinct recommendations. When considering the superposition of those weights similar weights may lead to almost indistinguishable recommendations. Furthermore, we also include 0 as possible weight to express that a rule behaves completely neutral with respect to the recommendation of a scheduling strategy for a given situation. This may also reduce the number of overall rules.

The main advantage of using several GMFs for describing a single rule is the automatic coverage of the possible feature space. In contrast to the rigid approach, even one rule gives a scheduling strategy for all possible system states. Hence, it is the focus of this approach to find a meaningful set of N_r rules that generates a good rule base system RB. Thus,

$$RB = \{R_1, R_2,\ \ldots\ R_{N_r}\}$$

is a complete rule base consisting of N_r rules.

Computation of the Controller Decision For a given system state, we compute the superposition of the weighted output consequence parts of all rules. The system state is represented by the actual feature vector

$$\boldsymbol{x} = \left(x_1\ x_2 \ldots x_\omega \ldots x_{N_F} \right)^T$$

of N_F feature values. Then we compute the degree of membership $\phi_i(x_\omega) = g_i^{(\omega)}(x_\omega)$ of the ω-th feature of rule R_i for all N_r rules and all N_F features. The multiplicative superposition of all these values as "AND"-operation leads to an overall degree of membership

$$\phi_i(\boldsymbol{x}) = \bigwedge_{\omega=1}^{N_F} g_i^{(\omega)}(x_\omega) = \prod_{\omega=1}^{N_F} \frac{1}{\sigma_i^{(\omega)}\sqrt{2\pi}} \exp\left\{ \frac{-(x_\omega - \mu_i^{(\omega)})^2}{2\sigma_i^{(\omega)2}} \right\}$$

for rule R_i. For all N_r rules together, the corresponding values $\phi_i(\boldsymbol{x})$ are collected in a membership vector

$$\boldsymbol{\phi}(\boldsymbol{x}) = \big(\, \phi_1(\boldsymbol{x})\ \phi_2(\boldsymbol{x})\ \ldots\ \phi_{N_r}(\boldsymbol{x}) \,\big)\,.$$

Next, we construct a matrix $\underset{\sim}{C}^{N_F \times N_r}$ of the weighted consequences $\boldsymbol{\Omega}(R_i)$, $i \in \{1, \ldots N_r\}$ of all rules by using the weighted consequence vectors for all individual rules R_i. This yields

$$\underset{\sim}{C}^{N_F \times N_r} = \big[\, \boldsymbol{\Omega}(R_1)\ \boldsymbol{\Omega}(R_2)\ \ldots\ \boldsymbol{\Omega}(R_{N_r}) \,\big]\ .$$

Now, we can compute the weight vector $\boldsymbol{\Psi}$ by multiplying the membership vector $\boldsymbol{\phi}(x)$ by the transposed matrix $\underset{\sim}{C}^T$:

$$\boldsymbol{\Psi} = \boldsymbol{\phi}(\boldsymbol{x}) \cdot \underset{\sim}{C}^T = \big(\, \Psi_1\ \Psi_2\ \ldots\ \Psi_{N_\Omega} \,\big)\,.$$

The vector $\boldsymbol{\Psi}$ contains the superpositioned weight values for all N_Ω possible scheduling strategy recommendations, that is, $\boldsymbol{\Psi}$ contains 13 elements.

Finally, we choose the scheduling strategy with the highest overall value as the output of the rule base system, that is

$$\arg \max_{1 \le h \le N_\Omega} \{\boldsymbol{\Psi}_h\}\,.$$

As already mentioned, within the Pittsburgh approach, each individual represents a complete rule base. We construct such a complete rule base RB with a fixed number of rules N_r. A single rule consists of $(2 \cdot N_F)$ elements per rule within the conditional part. Furthermore, we include the vector $\boldsymbol{\Omega}(R_i)$ for the consequence part, which consists of $N_\Omega = 13$ elements. Thus, a rule based scheduling system with constant number of rules can also be modeled using the following encoding. As such,

$$\boldsymbol{o}_k = \{\overbrace{\underbrace{\mu_1^{(1)}\ \sigma_1^{(1)}}_{\text{GMF}}, \ldots, \mu_1^{(N_F)}\ \sigma_1^{(N_F)},\ \underbrace{\boldsymbol{\Omega}(R_1)}_{\Omega_1\ \ldots\ \Omega_{N_\Omega}}}^{R_1}, \overbrace{\mu_2^{(1)}\ \sigma_2^{(1)}, \ldots, \mu_{N_r}^{(N_F)}\ \sigma_{N_r}^{(N_F)}, \boldsymbol{\Omega}(R_{N_r})}^{R_2\ \ldots\ R_{N_r}}\}$$

is the coding scheme of the object parameter vector $\boldsymbol{o}_k$ of individual $\boldsymbol{a}_k$ which is a complete rule base. Hence, the number of elements u within the object parameter vector $\boldsymbol{o}_k$ of the individual $\boldsymbol{a}_k$ can be computed by

$$u = N_r \cdot (2 \cdot N_F + N_\Omega).$$

We have chosen a non-isotropic mutation, see Bäck and Schwefel [2], as this allows the individual adaptation of the mutation for the different dimensions. Therefore, each object parameter of the individuals consists of a corresponding strategy parameter that specifies its mutation strength. Further, we apply a standard Evolution Strategy with $\mu = 3$ parent and $\lambda = 21$ offspring individuals. The ratio of $1/7$ is suggested by Schwefel [26]. Further, we do not use any recombination.

Within our Evolution Strategy, we used 40 generations with a randomly initialized first generation. Our (3+21)-Evolution Strategy leads to $3 + (40 \cdot 21) = 843$ evaluations for the development of a single rule base.

We use a constant number of rules $N_r = 10$ for each rule base. This results in a constant number of object and strategy parameters within each individual. Hence, $u = N_r \cdot (2 \cdot N_F + N_\Omega) = 10 \cdot (2 \cdot 7 + 13) = 270$ parameters must be determined. Thus, the two exogenous learning rates for the non-isotropic mutation are defined as:

$$\tau_0 = \frac{1}{\sqrt{2 \cdot u}} = 0.043, \text{ and } \tau_1 = \frac{1}{\sqrt{2\sqrt{u}}} = 0.174,$$

see Bäck and Schwefel [2].

Coevolutionary Genetic Fuzzy System Development As presented in Section 4, the rule based scheduling system needs to determine for each scheduling state a corresponding sorting criterion and a scheduling algorithm. In the previously introduced rule based scheduling systems, a whole scheduling strategy, consisting of both, a sorting criterion and a scheduling algorithm, was assigned to the different scheduling states. However, this combined assignment is not necessary. Moreover, the assignment of the same sorting criterion to two scheduling states within the features space does not always lead to the assignment of the same scheduling algorithm. This motivates the usage of a Coevolutionary Algorithm as the assignment problem can easily be decomposed into two subproblems.

Concept of Cooperative Coevolutionary Algorithms Coevolutionary Algorithms potentially lead to better solutions compared with standard Evolutionary Algorithms, if the problem can be decomposed into two subproblems, see for example Jansen et. al [19]. Furthermore, Potter and De Jong [25] have proven that Coeveolutionary Algorithms achieve better results with fewer generations compared with standard Evolutionary optimization techniques.

In this work, we apply the commonly called Cooperative Coevolutionary Algorithm (CCA), see Paredis [24]. This model uses two distinct species. Both species are genetically isolated. Hence, the genetic operations are only applied to individuals of the same species. The two different species are evolved in two different populations in parallel by using standard Evolution Strategies. However, during the fitness evaluation, two individuals of each species must cooperate. In general, this concept allows a larger number of species.

First, two species with μ individuals each are randomly generated. Then, the individuals of both species are evaluated by randomly combining two individuals, one from each species. Note that other selection schemes are also possible and discussed in the literature, see for example Panait et al. [22]. However, those methods need more evaluations and additional simulations in our case. In order to avoid this effort, we use our simple heuristic. After evaluation, the genetic operators produce λ offsprings for each species separately. Then, the resulting offspring individuals are again evaluated by a randomly chosen cooperation. Finally, normal evolutionary selection determines the next parent generation.

Rule Based Scheduling Development by applying Coevolutionary Algorithms As already mentioned, our scheduling problem can be decomposed into two separate subproblems. This concept is shown in Figure 4. Contrary to the general rule based scheduling, see Figure 1, we use two separate rule bases within the same feature space. One determines the sorting criterion depending on the system state and the other calculates the scheduling algorithm. However, the partitioning of this feature space differs between the two species. To this end, the different GMF-$\mu_i^{(\omega)}$ and GMF-$\sigma_i^{(\omega)}$ values are determined separately for the two species. The resulting scheduling system is expected to react on certain system states very accurately.

Such a coevolutionary approach yields several potential advantages for the resulting scheduling system and for the extraction process of appropriate rule bases.

First, each of the two separate rule bases has fewer output recommendations. In detail, for the sorting criterion as well as for the scheduling algorithm, we have only $N_\Omega = 4$ possible output recommendations instead of 13 as in the combined scenario. This reduces the length of the individuals within the populations and enables a better and faster adaptation. However, note that the sorting criterion is redundant if the Greedy scheduling algorithm is selected since Greedy includes its own sorting. Second, as the feature space partition can be optimized for both species separately, fewer rules might be required for each species.

The Evolution Strategies for both populations are identical. We apply the Pittsburgh approach with the same genetic operators and no recombination for both populations. In detail, we use a constant number of rules $N_r = 10$ and a (3+21)-Evolution Strategy for both populations. The optimization is limited to 40 generations. Consequently, each individual within the populations consists of

$$u = N_r \cdot (2 \cdot N_F + N_\Omega) = 10 \cdot (2 \cdot 7 + 4) = 180$$

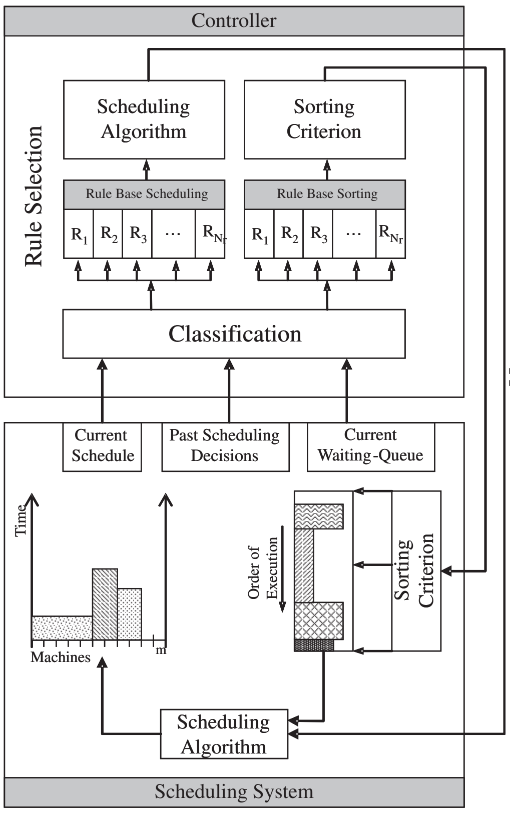

Fig. 4. General Concept of the Rule Based Scheduling Approach with Dedicated Rule Bases for Scheduling and Sorting.

object parameters. Hence, we adapt the learning rates for the non-isotropic mutation to

$$\tau_0 = \frac{1}{\sqrt{2 \cdot u}} = 0.053, \text{ and } \tau_1 = \frac{1}{\sqrt{2\sqrt{u}}} = 0.193,$$

see Bäck and Schwefel [2].

5 Evaluation

For the evaluation, we execute various discrete event simulations with real parallel computer workload traces. To this end, six well known workloads are selected. They were recorded at the Cornell Theory Center (CTC) [18], the Royal Institute of Technology (KTH) [23] in Sweden, the Los Alamos National Lab (LANL) [11] and the San Diego Supercomputer Center (SDSC 00/ SDSC 95/ SDSC 96) [12, 32]. Each of these workloads provides information about the job requests for the computational resources. In order to make those workloads comparable they are scaled to a standard machine configuration with 1024 processors as described by Ernemann et al. [9]. The characteristics of the used workloads are presented in Table 2.

As no real life objective functions are available from the workload traces, we exemplarily use the objective function ($f_{obj} = 10 \cdot \text{AWRT}_1 + 4 \cdot \text{AWRT}_2$). Clearly, this objective prioritizes user groups 1 and 2, with user group 1 having a higher priority than user group 2.

As already mentioned, we present our achieved results relative to the Pareto front (PF) of all feasible schedules for the simulated workloads. Noteworthy, the Pareto front was generated off-line and it cannot be taken for granted that this front can be reached by our proposed online scheduling

Identifier	CTC	KTH	LANL	SDSC 00	SDSC 95	SDSC 96
Machine	SP2	SP2	CM-5	SP2	SP2	SP2
Period	06/26/96 - 05/31/97	09/23/96 - 08/29/97	04/10/94 - 09/24/96	04/28/98 - 04/30/00	12/29/94 - 12/30/95	12/27/95 - 12/31/96
Processors (m)	1024	1024	1024	1024	1024	1024
Jobs (n)	136471	167375	201378	310745	131762	66185

Table 2. Scaled Workload Traces from Standard Workload Archive [13] using the Scaling Procedure by Ernemann et al. [9].

systems at all. Therefore, we refer to this front as a reference for the best achievable solution. Note that our Pareto front is only an approximation as it is derived by heuristics. Although we do not know the real Pareto front, the high density of our approximation indicates that the quality of the approximation is very good, see Figure 7.

In Table 3, the absolute results are presented. We show the AWRT values for the user groups, the objective and the overall Utilization. It is obvious that all proposed concepts achieve better

Approach	AWRT$_1$	AWRT$_2$	AWRT$_3$	AWRT$_4$	AWRT$_5$	U	f_{obj}
PF	49652.04	56330.98	60691.71	59698.30	32726.87	66.99	721844.268
EASY	59681.28	64976.07	50317.47	46120.02	31855.68	66.99	856717.0
PITTS	49639.195	56722.796	49541.757	59212.093	81268.331	66.99	723283.134
CCA	49676.087	56522.699	48723.312	57488.074	74983.133	66.99	722851.666
PROB	53780.183	59448.484	53185.9	53417.769	45390.11	66.99	775595.766

Table 3. AWRT, f_{obj} (in Seconds), and U (in %) of the Pareto Front (PF), EASY Scheduling, the Pittsburgh Approach (PITTS), the Cooperative Coevolutionary Algorithm (CCA), and the Probability Procedure (PROB) for the CTC Workload.

results than the EASY standard algorithm. We restricted the comparison to EASY as this is in most cases the best scheduling algorithm for the examined workloads with respect to the AWRT objective. Note that U remains constant and is not affected by the rule based scheduling concept although it is not explicitly included in the objective. As such we are able to prioritize different user groups without any reduction of the system utilization. Further, the results show that we are very close to the off-line generated Pareto front, see Figure 7.

In Figure 5 we presents the results for all six examined workloads. The very simple and rigid probability driven procedure is already able to improve the objective significantly. Apart from the KTH workload the rule system improves the objective value by 10 % on average compared to EASY scheduling.

However, the two Genetic Fuzzy Systems outperform this procedure. It is noteworthy that the on-line rule based scheduling systems produce schedules almost as good as those achieved in the off-line case. Despite the approximative character of the Pareto front, one can reasonably say that the results are quite close to the fronts of all workloads.

Workload	AWRT$_1$	AWRT$_2$	AWRT$_3$	AWRT$_4$	AWRT$_5$	U	f_{obj}
CTC	16.83	12.7	1.54	-28.39	-155.11	0	15.58
KTH	25.35	8.44	-57.64	-199.49	-744.53	0	19.82
LANL	19.75	14.84	-24.09	-47.2	-269.06	0	18.24
SDSC 00	60.83	42.37	-12.72	-3234.66	-14360.76	-5.57	55.79
SDSC 95	9.05	0.08	-20.7	-43.56	-38.55	0	6.37
SDSC 96	1.35	1.2	-20.03	-26.15	-4.09	0	1.31

Table 4. AWRT and Utilization Improvements Achieved with the Genetic Fuzzy System in Comparison to the EASY Scheduling Algorithm (in Percent).

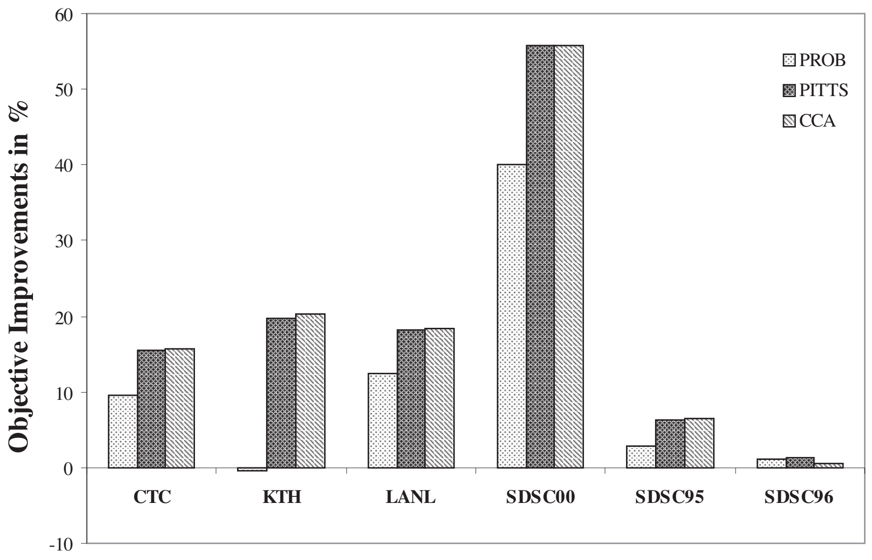

Fig. 5. Objective Improvements of all 3 Approaches in Comparison to EASY Scheduling.

The results listed in Table 4 demonstrate that the objective improvements really result in a shorter AWRT for the desired user groups. As we have already shown that the results are close to the Pareto front we now compare the Genetic Fuzzy System, created regarding to the Pittsburgh approach, with the EASY standard algorithm. The improvements of the AWRT in the gray shaded columns show that it is possible to shorten AWRT_1 and AWRT_2 significantely compared to EASY for most workloads. Apart from the SDSC 00 workload this prioritization is realized without deterioration of the utilization.

In Figure 6, we exemplarily show the achieved AWRT improvements for all 3 approaches for the CTC workload. We can realize the desired group prioritization with all proposed approaches. Note that we limited this chart at the y-axis as the AWRT values for user group 5 are extremely large. As the utilization remains constant these user groups have to pay the price for the short AWRT of the favored user groups. This is acceptable as we did not take these user groups into account for our objective formulation.

Finally, we show in Figure 7 the AWRT values of the two user groups to prioritize. This chart also depicts the Pareto front of all feasible schedules. Remember that we have 7 simple objectives. Each point within this chart represents a feasible schedule that is not dominated by any other generated feasible solution within the 7-dimensional objective space. As we show only a projection of the actual 7-dimensional Pareto front approximation, the elements cover an area in this 2-dimensional chart.

As the EASY standard algorithm does not favor any user groups, the achieved AWRT values are located in the mid of the projected front area. With the probability driven procedure, it is already possible to move the AWRT values towards the actual front. Obviously, this approach is capable to improve AWRT_2 significantly but it does only slightly improve AWRT_1. However, the two proposed Genetic Fuzzy Systems almost reach the front in our example. Thereby, the CCA leads to a little bit better results than the classic Pittsburgh approach.

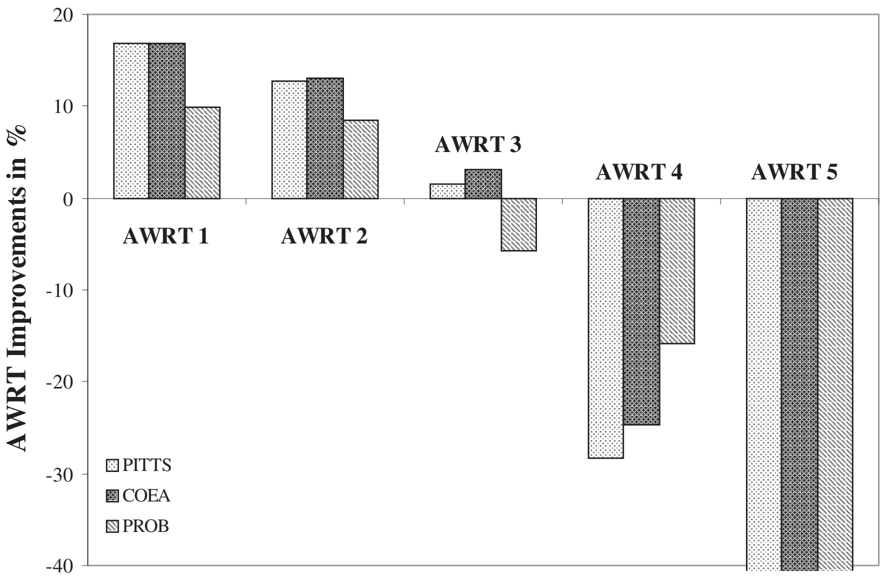

Fig. 6. AWRT Improvements of all 3 Approaches in Comparison to EASY Scheduling and the CTC Workload.

5.1 Estimation of Computational Effort to Establish the Rule Based Scheduling System

Our chosen objective is just an example and the proposed methods can be used with any other objective as well. However, we restricted our analysis to these example as this already required a high computational effort. For the probability driven procedure, we simulated $(50 \cdot 13 = 650)$ rule systems per workload. Of course this value is scalable by choosing a smaller number of guaranteed participations, but values smaller 50 did not yield good results. Nevertheless, this procedure established the rule bases with a comparatively small number of simulations.

The Genetic Fuzzy Systems are realized by an (3+21)-Evolution Strategy. In order to obtain good results, we simulated 40 evolutionary generations. Therefore, $(3 + 21 \cdot 40 = 843)$ objective evaluations per workload were necessary. Further, a single simulation of a complete workload takes about 4 hours computing time on average. For the Genetic Fuzzy System this resulted in 4 months computing time per workload and objective assuming only one available machine for the scheduling strategy generation.

Obviously, we are only able to present our results here because we used a compute cluster with 120 processors. With this installation, we can simulate all objective evaluations in parallel as they are completely independent from each other. Therefore, the simulation of one objective and one workload takes approximately one week. Furthermore, the parallel computation of the six workloads can also be realized. Despite the highly parallel execution of our simulations it still took more than 4 months to obtain the results presented in this paper.

Nevertheless, the presented effort estimates are only related to the generation of the rule bases. But remember that the execution of our scheduling algorithm in the runtime environment is not slower than the execution of a conventional scheduling algorithm.

6 Conclusion

In this paper, we have presented a novel approach to automatically generating online scheduling systems for a complex provider defined objective. The scheduling systems are based on rules that

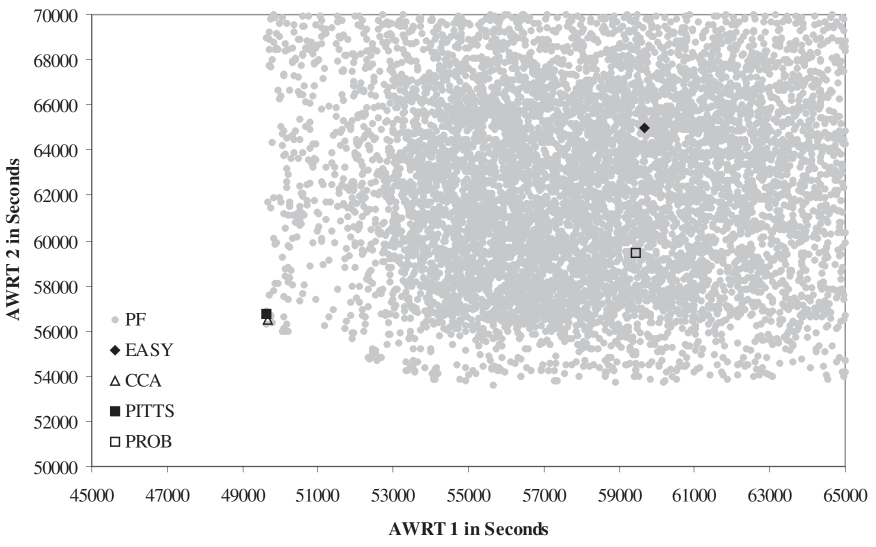

Fig. 7. $AWRT_1$ versus $AWRT_2$ of all 3 Approaches, the EASY Standard Algorithm, and the Pareto Front of all Feasible Schedules for the CTC Workload.

include standard scheduling algorithms. We used simulations with workload traces from existing installations during the development of the systems and during the evaluation process.

Even for a rather simple scheduling objective that prioritizes some user groups over others, we have demonstrated that a probability driven assignment procedure already leads to rule bases that typically produce better scheduling results than existing standard algorithms. The more sophisticated approaches using Genetic Fuzzy Systems significantly improve the achieved quality of the schedules. First, we compared our achieved results with the EASY standard scheduling algorithm. We achieved an improvement of about 10 % for our objective function with the adopted rule based scheduling system. Second, we compared our approaches with the off-line generated Pareto front of all feasible schedules. Here, we are even able to almost reach this front with the proposed Genetic Fuzzy Systems.

Acknowledgement

All authors are members of the Collaborative Research Center 531, "Computational Intelligence", at the Dortmund University with financial support of the Deutsche Forschungsgemeinschaft (DFG).

References

1. T. Bäck and H.-P. Schwefel. An Overview of Evolutionary Algorithms for Parameter Optimization. *Evolutionary Computation*, 1(1):1–23, 1993.
2. H.-G. Beyer and H.-P. Schwefel. Evolution Strategies – A Comprehensive Introduction. *Natural Computing*, 1(1):3–52, 2002.
3. A. Bonarini. Evolutionary Learning of Fuzzy rules: competition and cooperation. In W. Pedrycz, editor, *Fuzzy Modelling: Paradigms and Practice*, pages 265–284. Kluwer Academic Press, 1996.
4. R. Buyya, D. Abramson, and S. Venugopal. The Grid Economy. *Special Issue on Grid Computing, Proceedings of the IEEE*, 93(3):698–714, 2005.

5. S. J. Chapin, W. Cirne, D. G. Feitelson, J. P. Jones, S. T. Leutenegger, U. Schwiegelshohn, W. Smith, and D. Talby. Benchmarks and standards for the evaluation of parallel job schedulers. In D. G. Feitelson and L. Rudolph, editors, *Proceedings of the 5th Job Scheduling Strategies for Parallel Processing*, volume 1659 of *Lecture Notes in Computer Science (LNCS)*, pages 67–90. Springer, 1999.
6. C. Ernemann, V. Hamscher, U. Schwiegelshohn, A. Streit, and R. Yahyapour. On Advantages of Grid Computing for Parallel Job Scheduling. In *Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID2002)*, pages 39–46, Berlin, 2002. IEEE Computer Society Press.
7. C. Ernemann, V. Hamscher, and R. Yahyapour. Economic Scheduling in Grid Computing. In D. G. Feitelson, L. Rudolph, and U. Schwiegelshohn, editors, *Proceedings of the 8th Job Scheduling Strategies for Parallel Processing*, volume 2537 of *Lecture Notes in Computer Science (LNCS)*, pages 128–152. Springer, 2002.
8. C. Ernemann, U. Schwiegelshohn, N. Beume, M. Emmerich, and L. Schönemann. Scheduling Algorithm Development based on Complex Owner Defined Objectives. Technical Report CI-190/05, Dortmund University, Germany, 2005.
http://sfbci.uni-dortmund.de/Publications/Reference/Downloads/19005.pdf.
9. C. Ernemann, B. Song, and R. Yahyapour. Scaling of Workload Traces. In D. G. Feitelson, L. Rudolph, and U. Schwiegelshohn, editors, *Proceedings of the 9th Job Scheduling Strategies for Parallel Processing*, volume 2862 of *Lecture Notes in Computer Science (LNCS)*, pages 166–183. Springer, 2003.
10. C. Ernemann and R. Yahyapour. *Grid Resource Management - State of the Art and Future Trends*, chapter Applying Economic Scheduling Methods to Grid Environments, pages 491–506. Kluwer Academic Publishers, 2003.
11. D. G. Feitelson. Memory usage in the LANL CM-5 workload. In D. G. Feitelson and L. Rudolph, editors, *Proceedings of the 3rd Job Scheduling Strategies for Parallel Processing*, volume 1291 of *Lecture Notes in Computer Science (LNCS)*, pages 78–94. Springer, 1997.
12. D. G. Feitelson. Metric and Workload Effects on Computer Systems Evaluation. *Computer*, 36(9):18–25, 2003.
13. D. G. Feitelson. Parallel Workload Archive. http://www.cs.huji.ac.il/labs/parallel/workload/, March 2006.
14. D. G. Feitelson and M. A. Jette. Improved Utilization and Responsiveness with Gang Scheduling. In D. G. Feitelson and L. Rudolph, editors, *Proceedings of the 3rd Job Scheduling Strategies for Parallel Processing*, volume 1291 of *Lecture Notes in Computer Science (LNCS)*, pages 238–261. Springer, 1997.
15. R. L. Graham. Bounds for certain multiprocessor anomalies. *Bell System Technical Journal*, 45:1563–1581, 1966.
16. F. Heine, M. Hovestadt, O. Kao, and A. Streit. On the Impact of Reservations from the Grid on Planning-Based Resource Management Computational Science. In V. S. Sunderam et al., editor, *Proceedings of the 5th International Conference on Computational Science (ICCS 2005)*, volume 3516 of *Lecture Notes in Computer Science (LNCS)*, pages 155–162. Springer, 2005.
17. F. Hoffmann. Evolutionary Algorithms for Fuzzy Control System Design. *Proceedings of the IEEE*, 89(9):1318–1333, 2001.
18. S. Hotovy. Workload Evolution on the Cornell Theory Center IBM SP2. In D. G. Feitelson and L. Rudolph, editors, *Proceedings of the 2nd Job Scheduling Strategies for Parallel Processing*, volume 1162 of *Lecture Notes in Computer Science (LNCS)*, pages 27–40. Springer, 1996.
19. T. Jansen and R. P. Wiegand. Exploring the Explorative Advantage of the Cooperative Coevolutionary (1+1) EA. In E. Cantú-Paz et al., editor, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2003)*, volume 2723 of *Lecture Notes in Computer Science (LNCS)*, pages 310–321. Springer, 2003.
20. Y. Jin, W. von Seelen, and B. Sendhoff. On Generating FC^3 Fuzzy Rule Systems from Data Using Evolution Strategies. *IEEE Transactions on System, Man and Cybernetics*, 29(6):829–845, 1999.
21. 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, 2000.
22. L.Panait, R.-P. Wiegand, and S. Luke. A Sensitivity Analysis of a Cooperative Coevolutionary Algorithm Biased for Optimization. In K. Deb and R. Poli et al., editors, *Genetic and Evolutionary Computation - GECCO 2004*, volume 3102 of *Lecture Notes in Computer Science (LNCS)*, pages 573–584. Springer, 2004.
23. A. W. Mu'alem and D. G. Feitelson. Utilization, Predictability, Workloads, and User Runtime Estimates in Scheduling the IBM SP2 with Backfilling. *IEEE Transanctions on Parallel & Distributed Systems*, 12(6):529–543, 2001.

24. J. Paredis. Coevolutionary Computation. *Artificial Life*, 2(4):355–375, 1995.
25. M. A. Potter and K. De Jong. A Cooperative Coevolutionary Approach to Function Optimization. In Y. Davidor, H.-P. Schwefel, and R. Männer, editors, *Parallel Problem Solving from Nature – PPSN III*, volume 866 of *Lecture Notes in Computer Science (LNCS)*, pages 249–257. Springer, 1994.
26. H.-P. Schwefel. *Numerical Optimization of Computer Models.* John Wiley & Sons, Chichester, 1981.
27. S. F. Smith. *A Learning System Based on Genetic Adaptive Algorithms.* PhD thesis, Department of Computer Science, University of Pittsburgh, 1980.
28. B. Song, C. Ernemann, and R. Yahyapour. Parallel Computer Workload Modeling with Markov Chains. In D. G. Feitelson, L. Rudolph, and U. Schwiegelshohn, editors, *Proceedings of the 10th Job Scheduling Strategies for Parallel Processing*, volume 3277 of *Lecture Notes in Computer Science (LNCS)*, pages 47–62. Springer, 2004.
29. B. Song, C. Ernemann, and R. Yahyapour. User Group-based Workload Analysis and Modeling. In *Proceedings of the International Symposium on Cluster Computing and the Grid (CCGRID2005).* IEEE Computer Society Press, 2005. CD-ROM.
30. 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.
31. D. Talby and D. G. Feitelson. Supporting Priorities and Improving Utilization of the IBM SP Scheduler Using Slack-Based Backfilling. In *Proceedings of the 13th International Parallel Processing Symposium and 10th Symposium on Parallel and Distributed Processing*, pages 513–517. IEEE Computer Society, 1999.
32. K. Windisch, V. Lo, R. Moore, D. G. Feitelson, and B. Nitzberg. A comparison of workload traces from two production parallel machines. In *6th Symp. Frontiers Massively Parallel Computing*, pages 319–326. IEEE Computer Society Press, 1996.