Genetic Fuzzy Systems applied to Online Job Scheduling

Carsten Franke, Joachim Lepping, and Uwe Schwiegelshohn

Abstract— **This paper presents a comparison of three different design concepts for Genetic Fuzzy systems. We apply a Symbiotic Evolution that uses the Michigan approach and two approaches that are based on the Pittsburgh approach: a complete optimization of the problem and a Cooperative Coevolutionary algorithm. The three different Genetic Fuzzy systems are applied to a real-world online problem, the generation of scheduling strategies for Massively Parallel Processing systems. The Genetic Fuzzy systems must classify different scheduling states and decide about a corresponding scheduling strategy within each scheduling state. The main challenge arise in the delayed reward given by a critic. Therefore, it is impossible to directly evaluate the assignment of scheduling strategies to scheduling states. In our paper, the three design concepts are evaluated with real workload traces considering result quality, computational effort, convergence behavior, and robustness.**

I. Introduction

Fuzzy systems are usually designed by modeling implicit knowledge of an expert within a set of linguistic variables and Fuzzy rules, see Hoffmann [1]. They have been applied successfully to many real-world problems. Especially Genetic Fuzzy systems are well suited to address classification and automatic rule base generation. They provide mechanism to automatically extract expert knowledge from existing data by utilizing Evolutionary Algorithms to adjust membership functions and to define the output behavior of individual rules. The different Genetic Fuzzy systems encode either single rules (*Michigan approach*, Bonarini [2]) or complete rule bases (*Pittsburgh approach*, Smith [3]).

The Michigan approach is mainly associated with continuous learning in non-inductive problems, see Cordón et al. [4]. During the evolutionary optimization it uses a set of individuals each representing only a single rule. This set is evolved during the optimization process. Hence, all rules compete to participate in the next generation. However, they also must *cooperate* to establish rule bases that perform well with respect to the given objective function. This cooperation-competition dualism must be well balanced to obtain good results and is therefore a drawback of this approach. In this paper, we realize the Michigan approach as a Symbiotic Evolutionary Algorithm as introduced by Juang et al. [5]. Therefore, we will use the terms *Michigan approach* and *Symbiotic Evolution* interchangeably in the following.

The Pittsburgh approach has been developed with a focus on generalization. To address the representation of knowledge in a single entity it encodes a whole rule base in each individual. Therefore, an composition of individuals to rule bases is not necessary as each individual can be evaluated independently as an entire Fuzzy system. However, the main drawback of the Pittsburgh approach in comparison to the Michigan approach is the potentionally large size of the individuals. This involves a large search space for the evolutionary optimization and may reduce the efficiency of various genetic operators.

To reduce the search space we also evaluate a Cooperative Coevolutionary algorithm (CCA) [6] where the whole problem can be partitioned into different subdomains and each subpopulation is based on the Pittsburgh approach. For this concept, individuals are assigned to certain species, and the CCA architecture therefore models an ecosystem that consists of two or more species. As in nature, the species are genetically isolated, that is, individuals only mate with other members of their species. Such mating restrictions are simply enforced by evolving the species in separate populations where they try to adapt the environment by the repeated application of a classic Evolutionary Algorithm. Furthermore, the species interact with one another within a shared problem domain model and have a cooperative relationship. In order to evaluate individuals from one species, they have to form collaborations with representatives from each of the other species.

In this paper, we utilize all three design concepts for the generation of rule bases that are applied to a large real-world online problem which will be described in the next section. For this online problem there exists a large amount of realistic data as a record of real-world problem instances that are used for our evaluation. The actual evaluation process compares the three approaches in terms of absolute and relative solution quality, the computational effort, and the convergence behavior. Further, we also consider some aspects of robustness.

II. Background

As a large online problem we select parallel job scheduling at Massively Parallel Processing (MPP) systems where independent computational jobs are submitted over time. This scheduling problem is an online problem as neither the real processing time nor the release time of the jobs are

Manuscript received on February 12, 2007. This work was financially supported by the Deutsche Forschungsgemeinschaft (DFG) as a subproject of the Collaborative Research Center 531, "Computational Intelligence", at the Dortmund University.

Carsten Franke was a member of the Robotics Research Institute at Dortmund University and he is now with the SAP Research CEC Belfast, TEIC Building, University of Ulster, Newtownabbey BT37 0QB, UK (email: carsten.franke@sap.com). Joachim Lepping and Uwe Schwiegelshohn are with the Robotics Research Institute, Section Information Technology (IRF-IT), 44221 Dortmund, Germany (email: {joachim.lepping, uwe.schwiegelshohn}@udo.edu).

known in advance. The users only provide estimates of the processing time at the submission time of a job. We have chosen this type of online problem as it is rather simple due to the absence of any precedence constraints between jobs. Furthermore, it is of practical relevance and there exist comprehensive and representative workload traces from real MPP systems that can be used for training and evaluation. However, note that those traces only provide realistic input data for the online problem but there exist no separate training data as a knowledge base for the actual rule base generation.

Each MPP system consists of multiple processing nodes, and in all such systems every job requires the concurrent and exclusive access to a fixed number of machines during the whole execution phase. If a job has been submitted but cannot be started immediately due to an insufficient number of free processing nodes it is sent to the queue of waiting jobs. Therefore in MPP systems, a scheduling state mainly consists of the current schedule that describes the actual allocation of processing nodes to certain jobs, the queue of waiting jobs, and the characteristics of the resource allocations of already finished jobs. The various scheduling strategies mainly differ in the way they sort the waiting queue and how they select the next job from the waiting queue to insert it into the current schedule, that is, they obey different restrictions when choosing the next job. As the focus of this paper is on Genetic Fuzzy systems, we refer the interested reader to Franke et al. [7], [8] for further details on the scheduling problem.

For this problem, individual scheduling decisions can only be evaluated after all jobs completed processing. Hence, the award for the assignment of scheduling strategies to states is given by a critic only at the end of scheduling a whole workload trace. Furthermore, an appropriate system state classification is not known in advance and is implicit part of the generation of the rule based scheduling system.

In order to learn our rule bases, we must combine the Fuzzy systems with Evolutionary Algorithms. In this work, we apply Evolution Strategies, see Bäck and Schwefel [9], which are a subclass of Evolutionary Algorithms. Evolution Strategies are stochastic search methods that mimic the behavior of biological evolution. They operate on a parental population of size μ and apply evolutionary operators like selection, mutation, and recombination to breed λ offspring individuals from parent individuals. Based upon the evaluation results, a new population of individuals is selected. In detail, we use a $(\mu + \lambda)$-Evolution Strategy. For details see Bäck and Schwefel [9]. The detailed configurations of our Evolution Strategies are also given in other publications [7], [8].

III. Rule based Scheduling Concept

A rule based scheduling system for our problem uses a set of rules where every single rule consists of a conditional and a consequence part. The conditional part describes the system state in which this rule may be applied. Each individual rule R_i $(i \in \{1, \cdots, N_r\})$ assigns a consequence part Ω_i to its conditional part. Each system state at time t consists of the current Schedule $S(t)$, the current waiting queue $\nu(t)$, and the set of already finished jobs $\xi(t)$. It is described by a set of features which will be detailed later on. The scheduling strategy must determine:

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

Many existing scheduling algorithms for MPP systems can be applied to this online problem. We have chosen the Conservative Backfilling (CONS), the EASY Backfilling (EASY) [10], First-Come-First-Serve (FCFS), and a Greedy algorithm [11] which are the most popular ones. The Greedy algorithm is already a complete scheduling strategy while the former three algorithms must be supplemented with a sorting criterion of $\nu(t)$ [7]. For this sorting criterion, we use one of four different simple properties of a job:

1) the degree of parallelism
2) the current waiting time
3) the estimated runtime
4) the user group

This results in 13 different options for the scheduling strategies. In Figure 1, the rule based scheduling concept is depicted. The controller first determines the actual system state and selects a corresponding sorting of the job waiting queue and a scheduling algorithm. Both are applied next to the underlying scheduling system. As already mentioned, we

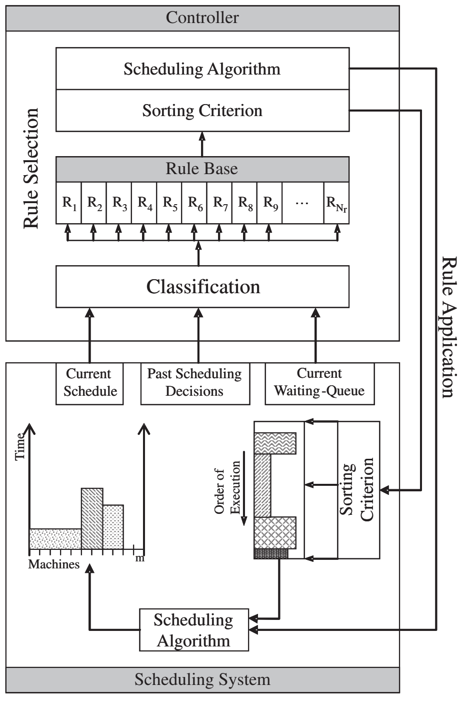

Fig. 1. Schematic Depiction of the Rule Based Scheduling Concept.

use several simple features to describe a system state. The first feature is the *Average Weighted Slowdown* (SD(t)) over all jobs that have already completed their processing [7]. This feature is the ratio between the response and the waiting time of those jobs [8]. The lower the value of SD(t) the better. The *Momentary Utilization* (U$_m(t)$), see Franke et al. [7], of

the whole system at time t represents the second feature. The last 5 features refer to the waiting jobs $j \in \nu(t)$ of each user group separately. We have chosen 5 user groups as previous studies have indicated that this complies with the workload structure in real installations, see Song et al. [12]. For more details on our user group selection see Franke et al.[7]. The *Proportional Resource Consumption of Waiting Queue for User Group* i ($\mathrm{PRCWQ}_i(t)$) represents the percentage of the estimated resource consumption of the waiting jobs of the specified user group in relation to the estimated resource consumption of all waiting jobs.
Finally, we exemplarily define a scheduling objective that considers user priorities. It is based on the *Overall Utilization* (U), the *Average Weighted Response Time* (AWRT) over all jobs of all users, and the AWRT for each user group separately, see Franke et al. [7]. The machine owner is able to flexibly define an individual objective function, based on these elements.

IV. Scheduling Strategies Based on Genetic Fuzzy Systems

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

A. Coding of Fuzzy Rules

Our Genetic Fuzzy Systems are based on the traditional Takagi-Sugeno-Kang (TSK) model [13] of Fuzzy systems. The coding schemes and learning techniques are adapted and slightly modified from the work of Juang et al. [5] and Jin et al. [14].
For a single rule R_i, $i \in \{1, \dots N_r\}$, every feature ω of all N_F features is modeled using a Gaussian Membership Function $(\mu_i^{(\omega)}, \sigma_i^{(\omega)})$-GMF

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

A feature is represented by a pair of real values $\mu_i^{(\omega)}$ and $\sigma_i^{(\omega)}$. The $\mu_i^{(\omega)}$ value is the center of the 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 for increasing $\sigma_i^{(\omega)}$ values, the peak value of the GMF decreases while the enclosed area remains constant. Using this property of a GMF, we are able to reduce the influence of a feature within a rule completely by increasing $\sigma_i^{(\omega)}$. Theoretically for $\sigma_i^{(\omega)} \to \infty$, a feature has no influence for this rule anymore. With this approach, it is also 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

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

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

$$\vec{\Omega}(R_i) = \left(\ w_{i1}\ w_{i2} \dots w_{iN_\Omega} \ \right)^T .$$

We restrict the possible weight values to elements of the set $\{-5,\ -1,\ 0,\ 1,\ 5\}$. The value -5 represents a *particularly unfavorable* influence 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. We also include 0 as possible weight to express that a rule behaves completely neutral with respect to the recommendation of the corresponding scheduling strategy for a given system state. Here,

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

is a complete rule base consisting of N_r rules.

B. 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

$$\vec{x} = \left(\ x_1 \quad x_2 \quad \dots \quad x_\omega \quad \dots \quad x_{N_F} \ \right)^T$$

of N_F feature values. 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

$$\begin{aligned} \phi_i(\vec{x}) &= \bigwedge_{\omega=1}^{N_F} g_i^{(\omega)}(x_\omega) \\ &= \prod_{\omega=1}^{N_F} \frac{1}{\sigma_i^{(\omega)}\sqrt{2\pi}} \exp\left\{ \frac{-(x_\omega-\mu_i^{(\omega)})^2}{2{\sigma_i^{(\omega)}}^2} \right\} \end{aligned} \quad (1)$$

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

$$\vec{\phi}(\vec{x}) \ = \ \left(\ \phi_1(\vec{x}) \quad \phi_2(\vec{x}) \quad \dots \quad \phi_{N_r}(\vec{x}) \ \right) .$$

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

$$\underset{\sim}{C}^{N_\Omega \times N_r} \ = \ \left[\ \vec{\Omega}(R_1) \quad \vec{\Omega}(R_2) \quad \dots \quad \vec{\Omega}(R_{N_r}) \ \right] .$$

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

$$\vec{\Psi} = \vec{\phi}(\vec{x}) \cdot \underset{\sim}{C}^T \ = \ \left(\ \Psi_1 \quad \Psi_2 \quad \dots \quad \Psi_{N_\Omega} \ \right) .$$

The vector $\vec{\Psi}$ contains the superpositioned weight values for all N_Ω possible scheduling strategy recommendations, that

is, $\vec{\Psi}$ contains 13 elements.
Finally, we choose the scheduling strategy with the highest overall value as the output of the rule base system, that is

$$\arg \max_{1 \le h \le N_\Omega} \left\{ \vec{\Psi}_h \right\}.$$

C. Encoding Schemes for Different Design Concepts

For the Michigan approach, we apply the following encoding scheme of Equation 2 where each individual represents a single rule

$$\overbrace{[\underbrace{\mu_i^{(1)}\ \sigma_i^{(1)}}_{\text{GMF}},\ \mu_i^{(2)}\ \sigma_i^{(2)}, \dots \mu_i^{(N_F)}\ \sigma_i^{(N_F)},\ \underbrace{\vec{\Omega}(R_i)}_{\Omega_1\ \dots\ \Omega_{N_\Omega}}\ ]}^{R_i} \quad (2)$$

This results in a fixed length for each single rule. But for interaction, the number of rules per systems must be specified in advance. The number of elements in the object parameter vector l of an individual is determined by the number of features, output decisions and number of rules. It can be computed by $l = N_r\,(2 \cdot N_F + N_\Omega)$. For the Michigan approach, where $N_r = 1$ holds, this results in a fixed number of $2 \cdot 7 + 13 = 27$ parameters per individual.
Contrary, the Pittsburgh approach encodes a whole rule base in each individual, see the representation in Equation 3.

$$\overbrace{[\underbrace{\mu_1^{(1)}\ \sigma_1^{(1)}}_{\text{GMF}},\ \mu_1^{(2)}\ \sigma_1^{(2)}, \dots \mu_1^{(N_F)}\ \sigma_1^{(N_F)},\ \underbrace{\vec{\Omega}(R_1)}_{\Omega_1\ \dots\ \Omega_{N_\Omega}}}^{R_1}, \dots$$

$$\dots\ \overbrace{\mu_2^{(1)}\ \sigma_2^{(1)}, \dots \mu_{N_r}^{(N_F)}\ \sigma_{N_r}^{(N_F)}, \vec{\Omega}(R_{N_r})}^{R_2\ \dots\ R_{N_r}}] \quad (3)$$

If we assume the same configuration as for the Michigan approach we must consider $N_r = 50$ rules which would result in $50 \cdot (2 \cdot 7 + 13) = 1350$ parameters. However, after different experiments [7] we also found good results for the Pittsburgh approach with $N_r = 10$.
The CCA uses the same encoding scheme as the Pittsburgh approach but consists of two distinct species. We use one species for the sorting of the waiting queue and a second species for the determination of the scheduling algorithm [7].

V. Evaluation and Discussion

To compare the three different design concepts we use some workloads from real installations as input data [7] and apply a user group definition based on resource consumption, which distinguishes 5 different user groups, as explained by Ernemann et al. [15]. For our analysis, we exemplarily define 2 different priority schemes that are represented by the following 2 objective functions:

$$\begin{aligned} f_1 &= 10 \cdot \text{AWRT}_1 + 4 \cdot \text{AWRT}_2 \\ f_2 &= 2 \cdot \text{AWRT}_1 + 10 \cdot \text{AWRT}_3 \end{aligned}$$

A short AWRT value for a user group corresponds to a good schedule quality for this group. Here, f_1 prioritizes user groups 1 and 2, with user group 1 having a higher priority than user group 2 and f_2 prioritizes user group 3 over group 1 over all other user groups.

A. Relative Comparison

First, we show the achieved objective value for all three design concepts compared to the commonly used EASY standard strategy [10]. We restrict our comparison to the EASY algorithm as this method performs best for the examined workloads. However, note that EASY does not support any priorities. In Table I, we show the results of the Symbiotic evolution (SYMB), the Pittsburgh approach (PITTS), and the Cooperative Coevolutionary algorithm (CCA) compared to EASY. We train all systems with the Cornell Theory Center (CTC) workload trace which is available from the Standard Workload Archive [16].
For f_1 the Genetic Fuzzy systems achieve the desired prioritization of user groups 1 and 2, as the AWRT values are significantly shorter for these user groups. Obviously, the non-preferred user groups have to pay the price with their increasing response times while the overall utilization (U) remains unchanged. Objective f_1 uses a prioritization scheme that corresponds to the resource consumption of the user groups. It seems to support easy optimization as all systems yield almost similar optimization results, see Table II. However, this is not true for objective f_2 where the low-consuming user group 3 has a higher priority than user group 2 requiring more total resources, see Table II. While there are improvements for all approaches CCA clearly outperforms SYMB and PITTS.

Method	AWRT_1	AWRT_2	AWRT_3	AWRT_4	AWRT_5	U
SYMB	16.64	12.7	2	-26.35	-153.62	0
PITTS	16.83	12.70	1.54	-28.3	-155.1	0
CCA	16.76	13	3.17	-24,65	-135.38	0

TABLE I
Relative Improvements of AWRT (in %) and UTIL (in %) for the CTC Workload and f_1 for Symbiotic/Michigan (SYMB), Pittsburgh (PITTS), and CCA compared to EASY.

	SYMB	PITTS	CCA
Objective f_1	15.45%	15.57%	15.62%
Objective f_2	7.44%	7.36%	9.1%

TABLE II
Relative Objective Improvements for the CTC Workload, Symbiotic/Michigan (SYMB), Pittsburgh (PITTS), and CCA compared to EASY.

B. Comparison to Pareto Front

After addressing the relative improvements that are possible with the three design concepts, we focus on the comparison relative to the best achievable solutions. To this end, we represent the distance of the resulting schedules from the approximated Pareto front of all feasible schedules for a simulated workload, as generated by Ernemann et al. [15]. As this is only an approximation of the Pareto front, schedules of this front are not guaranteed to represent the real bounds.

On the other hand, the approximations are generated offline (using knowledge of the future). Online methods (as applied here) may not be able to achieve as good results. Ignoring these restrictions we refer to this front as the best achievable solutions for our scheduling problem. In Figure 2, the results for objective f_1 are depicted and it becomes obvious that it is possible to come very close to the approximated Pareto front. This is different for the objective f_2 as shown in Figure 3. It

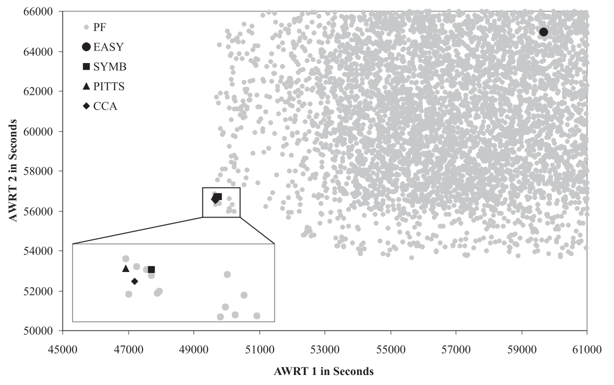

Fig. 2. AWRT$_1$ and AWRT$_2$ Compared to Pareto Front Approximation of the CTC workload for EASY, SYMB, PITTS, and CCA.

is still possible to come closer to the Pareto front than EASY with all approaches. All genetic Fuzzy systems outperform iterative rule learning methods, see Franke et al.[7], [8].

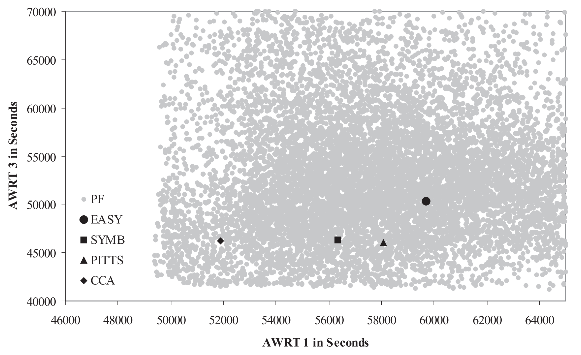

Fig. 3. AWRT$_1$ and AWRT$_3$ Compared to Pareto Front Approximation of the CTC workload for EASY, SYMB, PITTS, and CCA.

C. Computational Effort

It is important to realize that the optimization and adaptation processes of all three concepts is time consuming. But the runtime of the resulting scheduling algorithm is negligibly small compared to the computational effort that has to be spend for the optimization. We realize PITTS and CCA with a (3+21)-Evolution Strategy and run 40 generations in order to obtain good results. The resulting number of $(3 + 21 \cdot 40) = 843$ objective evaluations is relatively small for an evolutionary optimization. However, if we consider the long evaluation times of approximately 4 hours per simulation the complete optimization takes about 5 months for a single objective function and workload. For SYMB the computational effort is even higher than the PITTS and CCA as we use a (50+350)-Evolution Strategy. The large population size is required to guarantee a sufficient number of participations for each individual within a temporally composed Fuzzy system, see Juang et al. [5]. In order to ensure this number of participations, we create 70 Fuzzy systems per generation [8]. This results in $70 \cdot 40 = 2800$ simulations that are required to evolve 40 generations. Thus, it takes approximately 15 months to optimize one objective function for a single workload using the SYMB approach on a single machine. The computational effort is unacceptably high compared to the Pittsburgh based concepts. Only with a parallel execution it is possible to obtain all results presented in this paper within a reasonable time. Therefore, we have executed the simulations of the population evaluations in parallel on a 120 node cluster.

D. Convergence Behavior

As the computational effort is relatively high for all approaches a fast convergence to a good solution is highly desirable. In Figure 4, the development of the objective function

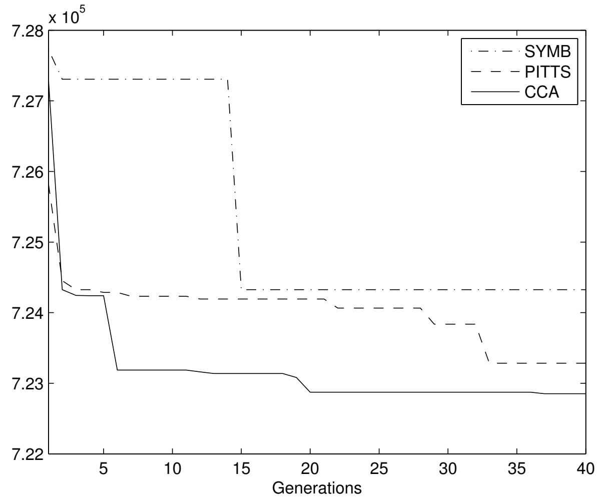

Fig. 4. Fitness Development During the Optimization for All Three Approaches, Objective f_1, and the CTC Workload Trace.

is depicted versus the number of generations. Apparently, SYMB converges slowly and, as already discussed, the final solution quality is worse than PITTS or CCA. Furthermore, the progress to the final solution is characterized by only a few large improvement steps. Contrary, both Pittsburgh based approaches converge in smaller steps towards the final result. Those concepts seems to adapt better to our problem as also intermediate solutions are found during optimization. The best final result is achieved by CCA which also performs best in terms of convergence.

E. Robustness

As a final aspect, we focus on the robustness of the derived rule bases. In Figure 5, we display the objective improvements of all used approaches compared with EASY by optimizing the different strategies again for the CTC workload and applying the resulting scheduling strategies to three different but similar workloads. Those were recorded at the Swedish Royal Institute of Technology (KTH), the Los Alamos National Lab (LANL), and the San Diego Supercomputer Center (SDSC00). The results clearly indicate that only

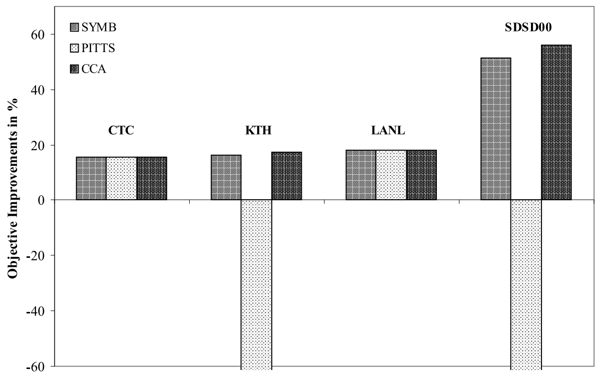

Fig. 5. Relative improvements of objective f_1 for all design methods related to EASY, and all workloads using the strategies optimized for the CTC workload trace.

the SYMB and CCA should be used in such scenarios. The PITTS approaches fails for the KTH and the SDSC00 workload traces. Note that we limited the y-axis to -60% but the real objective deterioration is about -150%. We assume that SYMB is better suited to adapt to the different scheduling scenarios but it requires 50 rules per rule base to achieve these results [8]. As already shown, 10 rules are enough for PITTS to generate very good schedules for a certain workload individually. However, it seems that 10 rules are not enough to generate robust scheduling strategies. The CCA uses 10 rules for each subpopulation. The achieved results indicate that this case is not too specialized on a specific workload as it is also able to generate good schedules for the other workloads. The quality of the generated schedules for SYMB and CCA are very similar. Thus, we recommend the CCA approach for a robust rule base generation as the number of required simulations is in this case much smaller compared to SYMB.

VI. Conclusion

In this paper, we compared three design concepts for Genetic Fuzzy Systems applied to a real-world online scheduling. We have defined two objectives that express different user group prioritization for the underlying scheduling problem. Within this study it becomes apparent that the Fuzzy system created with a Cooperative Coevolutionary algorithm based on the Pittsburgh approach (CCA) performs best. The CCA outperforms the Symbiotic Evolution (SYMB) for the second examined objective, convergence behavior, and the computational effort which is very high due to the large number of required simulations while the robustness analysis shows almost equal results for both approaches. On the other hand, the traditional Pittsburgh approach (PITTS) shows worse results than CCA for the second objective, the convergence behavior, and the robustness while the computational effort is identical. Especially, for robustness where the rule base is applied to three similar workloads, PITTS fails completely for two workloads. Therefore, the decomposition of the whole problem into two subproblems and the application of the CCA approach is recommended for the examined problem.

References

[1] F. Hoffmann, "Evolutionary algorithms for fuzzy control system design," *Proceedings of the IEEE*, vol. 89, no. 9, pp. 1318–1333, September 2001.
[2] A. Bonarini, *Fuzzy Modelling: Paradigms and Practice*. Kluwer Academic Press, 1996, ch. Evolutionary Learning of Fuzzy rules: competition and cooperation, pp. 265–284.
[3] S. F. Smith, "A learning system based on genetic adaptive algorithms," Ph.D. dissertation, Department of Computer Science, University of Pittsburgh, 1980.
[4] O. Cordón, F. Herrera, F. Hoffmann, and L. Magdalena, *GENETIC FUZZY SYSTEMS - Evolutionary Tuning and Learning of Fuzzy Knowledge Bases*, 1st ed., ser. Advances in Fuzzy Systems - Applications and Theory, L. A. Zadeh, Ed. Singapore: World Scientific, July 2001, vol. 19.
[5] 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*, vol. 30, no. 2, pp. 290–302, April 2000.
[6] J. Paredis, "Coevolutionary computation," *Artificial Life*, vol. 2, no. 4, pp. 355–375, 1995.
[7] C. Franke, J. Lepping, and U. Schwiegelshohn, "On Advantages of Scheduling Using Genetic Fuzzy Systems," in *Proceedings of the 12th Job Scheduling Strategies for Parallel Processing*, ser. Lecture Notes in Computer Science (LNCS), E. Frachtenberg and U. Schwiegelshohn, Eds., vol. 4376. Springer, January 2007, pp. 68–93.
[8] C. Franke, J. Lepping, F. Hoffmann, and U. Schwiegelshohn, "Development of Scheduling Strategies with Genetic Fuzzy Systems," University Dortmund, Tech. Rep. 0106, May 29 2006, iSSN 0941-4169.
[9] T. Bäck and H.-P. Schwefel, "An overview of evolutionary algorithms for parameter optimization," *Evolutionary Computation*, vol. 1, no. 1, pp. 1–23, 1993.
[10] D. G. Feitelson and A. M. Weil, "Utilization and Predictability in Scheduling the IBM SP2 with Backfilling," in *Proceedings of the 12th International Parallel Processing Symposium and the 9th Symposium on Parallel and Distributed Processing*. IEEE Computer Society Press, 1998, pp. 542–547.
[11] C. Franke, J. Lepping, and U. Schwiegelshohn, "Greedy Scheduling with Complex Objectives," in *Proceedings of the 2007 IEEE Symposium Series in Computational Intelligence*. IEEE Press, 2007, to appear.
[12] B. Song, C. Ernemann, and R. Yahyapour, "User group-based workload analysis and modeling," in *Proceedings of Cluster Computing and Grid (CCGRID05)*. IEEE Press, 2005, CD-ROM.
[13] T. Takagi and M. Sugeno, "Fuzzy Identification of Systems and Its Applications to Modeling and Control," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-15, no. 1, pp. 116–132, 1985.
[14] 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*, vol. 29, no. 6, pp. 829–845, December 1999.
[15] C. Ernemann, U. Schwiegelshohn, N. Beume, M. Emmerich, and L. Schnemann, "Scheduling Algorithm Development based on Complex Owner Defined Objectives," Dortmund University, Germany, Tech. Rep. CI-190/05, 2005, http://sfbci.uni-dortmund.de/Publications/Reference/Downloads/19005.pdf.
[16] D. G. Feitelson, "Parallel Workload Archive," http://www.cs.huji.ac.il/labs/parallel/workload/, April 2007.