HAL
open science

Knowledge discovery for scheduling in computational grids

Alexander Fölling, Joachim Lepping

To cite this version:

Alexander Fölling, Joachim Lepping. Knowledge discovery for scheduling in computational grids. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 2012, 2 (4), pp.287-297. 10.1002/widm.1060 . hal-00758208

HAL Id: hal-00758208

https://inria.hal.science/hal-00758208v1

Submitted on 29 Nov 2012

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

WILEY Interdisciplinary Reviews
DATA MINING AND KNOWLEDGE DISCOVERY

Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery

Article type: Overview

Knowledge Discovery for Scheduling in Computational Grids

DMKD-00113

Alexander Fölling

Robotics Research Institute, TU Dortmund University, 44221 Dortmund, Germany

Joachim Lepping

INRIA Rhône-Alpes, Grenoble University, 38330 Montbonnot-Saint-Martin, France

Keywords

Knowledge Discovery, Grid-computing, Computational Intelligence, Scheduling

Abstract

Scheduling in computational grids addresses the allocation of computing jobs to globally distributed compute resources. In a frequently changing resource environment, scheduling decisions have to be made rapidly. Depending on both the job properties and the current state of the resources those decisions

are different. Thus, the performance of grid-scheduling systems highly depends on their adaptivity and flexibility in changing environments. Under these conditions, methods from knowledge discovery yielded significant success to augment and substitute conventional grid-scheduling techniques. This paper presents a survey on approaches to extract, represent, and utilize knowledge to improve the grid-scheduling performance. It aims to give researchers insight into techniques used for knowledge-supported scheduling in large-scale distributed computing environments.

The ever-growing need for universally available computing and storage capacity is more and more satisfied by new architectures for networked interaction between users and providers of computing resources. The basic entity in this context is a compute site which constitutes an administrative domain that controls the access to one or more parallel computers or cluster installations. A typical example is an academic data center but also a cloud-computing provider. Today, research and industry realize *grid-computing* infrastructures for the coordinated utilization of globally distributed computing resources. Each site has its own local user community but jobs are migrated among resource providers. In such environments, submitted computing jobs are no longer bound locally. Each site has its own local user community but jobs are migrated between resource providers.

The cooperation among such resource providers requires advanced scheduling concepts. For this purpose, different architectures with special demands on information exchange between the resource providers have been developed. These architectures differ in the way in which each provider participates in grid-scheduling decisions. Grids are dynamic environments with frequently changing heterogeneous resources. Their efficient operation requires scheduling decisions that are both situation-dependent and adaptive. Considering the status of a local system is crucial whenever a decision has to made either to accept foreign workload or not. Grid-scheduling systems require computational efficiency to cope with the high throughput that involves the execution of thousands of jobs per minute to be allocated among up to tens of thousands of processors. Thus, highly complex decision-making procedures are only applicable if they are real-time capable.

Although often neglected in research, grid-scheduling is inherently an online problem as jobs are submitted over time and neither their precise runtimes nor future job submission times are known in advance. In the optimal case, user runtime estimations are available in advance but in practice these estimates are usually unreliable. As a consequence, grid-scheduling decisions are made on the basis of partial and uncertain information and the discovery and integration of the sparse knowledge available is essential.

It is obvious that all these properties make grid-scheduling a very complex problem and thus hardly manageable. To address this problem, many grid-scheduling solutions employ complex heuristics and evaluate their performance with respect to common global objectives such as the average response time over all jobs or the utilization of the machines. These heuristics often achieve

good results but they rarely consider past decisions or previous states of the scheduling process. To overcome this disadvantage and to increase the robustness and performance of grid-scheduling algorithms, more recent approaches resort to knowledge discovery techniques.

Knowledge discovery is the iterative process of selecting and pre-processing data, examining data, and interpreting extracted patterns. The goal is the extraction of knowledge to support an efficient decision-making. In the context of grid-scheduling, the knowledge discovery process results in rules defining the behavior of grid-schedulers. Such decisions depend on job properties (workload), resource configuration, previous computational demands, or other features. Metrics quantify the state of resources or the characteristics of jobs. Especially, methods summarized as *computational intelligence* are very powerful in this application area.

This survey presents the state-of-the-art in knowledge discovery for grid-scheduling and categorizes the different approaches with respect to their type of models and learning schemes. As both knowledge discovery and grid-computing in general are comprehensive topics, the paper focuses on those aspects that are most relevant to their combination. The most relevant and influential papers on knowledge discovery in the context of grid-scheduling are selected and introduced on the basis of an extensive review of publications over the past five years. They are first analyzed with respect to similarities and differences and discussed afterwards. It appears reasonable to divide this discussion into grid-scheduling models, knowledge representation, knowledge acquisition, and knowledge application.

This paper is organized as follows: First, a brief introduction to knowledge discovery is given and the main concepts of computational intelligence are described. In the core of this paper, the most prominent approaches to grid-scheduling with knowledge discovery are discussed. Four key aspects are considered: 1) the grid-scheduling model, 2) the knowledge representation schemes, 3) the knowledge acquisition schemes, and 4) the methods to apply extracted knowledge to improve future scheduling decisions. In the conclusion, important future directions of this research field are pointed out.

Knowledge Discovery with Computational Intelligence

Before the application area of grid-scheduling is discussed, some basic principles of computational intelligence are introduced. Computational intelligence in general comprises evolutionary algorithms, neural networks, and fuzzy systems. This introduction restricts itself to evolutionary algorithms and evolutionary fuzzy systems, since they assume a major role for knowledge discovery in grid-computing.

Suggested reading

R. C. Eberhart and Y. Shi, *Computational Intelligence: Concepts to Implementations*[1]: They describe the integration of various natural and engineering disciplines to establish computational intelligence. The book further rests on a foundation of evolutionary computation but has also an emphasis on practical applications and computational tools.
H.–P. Schwefel, *Evolution and Optimum Seeking*[2]: A fundamental book about evolutionary algorithms and evolution strategies.
O. Cordón et al., *Evolutionary Tuning and Learning of Fuzzy Knowledge Bases*[3]: The book summarizes and analyzes the novel field of evolutionary fuzzy systems, paying special attention to evolutionary algorithms that adapt and learn the knowledge base of a fuzzy system. It also includes a good general introduction to fuzzy systems.

Evolutionary Algorithms

Evolutionary algorithms are general purpose optimization algorithms that mimic the natural process of the Darwinian evolution and are most successfully applied to non-linear, global parameter optimization in high dimensional complex optimization problems.[4] They operate on a population of individuals, which represent solutions to the optimization problem. Typical representations encode the problem as a real-valued vector or as binary string. Genetic operators such as mutation (a random change in genome) and recombination (combining two or more parent individuals' genomes) are applied to generate offspring that inherit traits from their individuals. In global selection the individuals compete against each other to determine the new parents of the next generation. The evolutionary loop is executed either for a fixed number of generations or until the best solution surpasses a threshold on the objective function.

Fuzzy Systems

Since their conceptualization in the early 1960ies, fuzzy systems have been widely and successfully applied to modeling, pattern recognition, data analysis, decision making and control system design. Fuzzy-systems are capable of decision making under partial or incomplete knowledge. The knowledge base is composed of a set of linguistic rules that relate antecedents to consequences. The fuzzy representation offers a number of advantages: The interpolative nature of fuzzy systems allows the partial and concurrent activations of rules and gradual transitions between them. The overall decision is easily synthesized by a set of IF-THEN rules expressed in linguistic terms that reflect the expert knowledge.

Evolutionary Fuzzy Systems

Conventional fuzzy systems are designed based on expert knowledge but lack the ability to learn rules or membership function from data. Neuro fuzzy system combine the rule based representation with the supervised learning schemes of neural networks. This enables the refinement or generation of rules from data for situations in which the experts decisions are imprecise or missing. Evolutionary fuzzy systems are suitable to tune or even learn a fuzzy system in case there are no direct training examples for supervised learning. The evolutionary algorithm optimizes rules or membership functions with respect to a global performance indicator, such as the closed loop behavior in case of a fuzzy controller, or the makespan or flow time of a workload trace in case of a fuzzy grid-scheduler. Special variants of evolutionary algorithms offer the advantage of multi-objective optimization which allows the simultaneous consideration of multiple potentially conflicting objectives.

Grid-scheduling Models

Suggested reading

I. Foster and C. Kesselman, *The Grid: Blueprint for a New Computing Infrastructure*[5]: Standard textbook about the grid-computing concept. It covers the past, present, and future of network infrastructures.
M. Pinedo, *Scheduling: Theory, Algorithms, and Systems*[6]: Well established textbook on basic scheduling problems and their solutions.

Since Smarr and Catlett[7] promoted the term meta-computing in 1992, research efforts focus on the design of architectures for the shared use of globally distributed resources. Today, the term grid-computing coined by Foster and Kesselman[5] is widely used. But even this term has been interpreted differently and went through a series of metamorphoses described by Altmann et al.[8]. Any interpretation of the term is closely related to the specific requirements of alternative grid-computing infrastructures. Each community has its specific requirements on a grid-computing infrastructure. Thus, the grid-model varies significantly among different research works.
Among the different architectures of a grid (data-grid, utility-grid, etc.) a *computational grid* is most widely used. In contrast to usual parallel systems, see Bertsekas and Tsitsiklis[9], it represents a large-scale distributed system based on a merger of regional, national, and global (heterogeneous) computing systems including data centers and other resources. The participants are connected by a high performance communication network. It thus permits the aggregation of autonomous resources in a dynamic and changing environment.
Berman et al.[10] give a description of all implemented architectures in form of a taxonomy. Beyond computational grids, which are in focus of most knowledge discovery approaches for grid-scheduling, there exist other models with a strong focus on data-,

energy-, or collaboration management.[11]
The models presented in this paper are motivated from real-world installations: The diversity of today grids is reflected by the manifold of the corresponding grid-models.

Meta-scheduling Models

The most common grid-scheduling model is the *meta-scheduling model* where multiple distributed local schedulers are aggregated and supervised by a high-level scheduler instance, the meta-scheduler. Usually, the meta-scheduler also serves as a central recipient of user jobs and is responsible for their delegation to local schedulers. The meta-scheduler has complete access to the overall grid-state and allocates jobs to the location that currently provides the optimal conditions for their execution. Some models even abandon the concept of local schedulers and the meta-scheduler directly accesses the resources at the underlying computing sites. This global system view makes scheduling relatively easy as all participants are known and controllable. Due to the single submission and scheduling component, the system runs high risks of a full breakdown as it includes a single point of failure. Thus, this model is hardly scalable and unreliable.
A typical example of such an environment is described by Xhafa et al.[12]. Their model assume a central grid-scheduler instance that has exclusive access to all underlying resources. These resources are assumed to be heterogeneous in speeds while their respective performance is summarized in terms of million instructions per second (MIPS). In their grid-environment, only sequential jobs are used which are not allowed to be interrupted during execution and they consider an offline scheduling problem where all jobs are known in advance.
In another meta-scheduling model, the central scheduling component applies a runtime-prediction algorithm.[13] This model requires an open information model such that the meta-scheduler has access to information on resource states, for example, the number of queued jobs or number of running jobs. However, the authors address an online scheduling problem which makes the knowledge discovery even harder as properties of further job submissions are unknown.
Similarly, Prado et al.[14] follow a meta-scheduling approach and represent knowledge by a fuzzy system on the meta-layer. They assume homogeneous clusters in speed and other machine attributes and do evaluations with up to 3,000 parallel and sequential tasks. They also consider the online scheduling problem and apply their fuzzy approach even for previously unknown job submissions.
Zhou et al.[15] strictly stick to the grid-model supported by the common Globus Toolkit 4 (GT4) middleware[16] and develop a meta-scheduler model on this setup. They assume heterogeneous resources with local scheduler instances and choose an implementation on the basis of the additional services provided by GT4. The meta-scheduler is realized as a central resource broker and serves both as central submission component and information manager. The scheduler module steers the assignment of jobs to the different GT4 middleware installations. In another work of Yu et al.[17] the same model assumptions are made.
Finally, Prado et al.[18] use a meta-scheduling concept with multiple heterogeneous re-

source domains considering multiple virtual organizations. The internal organization of distributed resources is further specialized and adapted to real-world requirements where research groups may collaborate in virtual organization to share both knowledge and resources. However, the scheduling still follows the meta-scheduling paradigm as this component has access to internal information like used nodes or missed deadlines. The whole system is evaluated by simulations with up to 2,000 jobs from a real workload trace considering the online scheduling problem. The related works of Prado et al.[19] assume a similar model and optimize the behavior of a central scheduling component.

Decentralized Scheduling Models

The decentralized scheduling architecture[20] requires no central coordinator as the grid-schedulers directly interact and negotiate job allocations with each other. The resource sites act as a submission interface for their local users or user communities. The local resource management layer is supplemented with a grid-scheduling layer that takes over the task coordination and establishes the communication with other sites. Submitted jobs are either passed directly to the underlying local scheduling system or are migrated to other sites. The communication between sites is established via direct communication, which means that schedulers directly negotiate and exchange jobs.[21] This includes mechanisms for finding other sites and algorithms for managing lists of available exchange partners. If the migration of a job to a certain site is not possible, the scheduler decides to contact an alternative partner site for the execution of its job. This type of parallel peer-to-peer job allocation is of major importance as it scales properly even for very large world-wide grid installations. It is further more reliable than any meta-scheduling approach. Furthermore, Fölling et al.[22] assume a restrictive information policy as local system states are exclusively accessible at the local sites. In real-world installations it is observable that critical information are kept classified as they might give hints about the operation efficiency of systems.
In summary, the advantage of a distributed scheduling architecture is a much higher fault tolerance. The failure of a grid-level component or an entire site never results in a complete system breakdown. At most the overall performance of the grid decreases. Additionally, this architecture achieves a significantly better scalability for the same reasons. Large-scale grid-computing is only realizable by decentralized structures while obeying local autonomy. The disadvantage of decentralized structures originates precisely in the lack of a global grid-system view that makes scheduling more complicated. The decision to either accept or reject a foreign workload request highly depends on the status of the local scheduler and its current workload.

Other Scheduling Models

For completeness, also alternative grid-scheduling models used in the context of knowledge discovery are mentioned. In the work of Zeng et al.[23] an agent-based grid-model without any hierarchy, scheduling roles, or even layers is assumed. They address the

online scheduling problem with specific job workflows of sequential tasks and homogeneous resources. They are more interested in applying alternative knowledge discovery mechanism instead of investigating specific grid-scheduling models. Other models[24,25] assume heterogeneous grid-resources which are specified by cycles per time unit. In this way, different machine speeds are expressed. They assume further independent jobs while the processing times are given as total required cycles. Sequential jobs are scheduled in an open grid-environment without any hierarchies or site specific schedulers. Properties of different grid-scheduling models are summarized in Table 1.

Table 1 Overview: grid-scheduling models.

Model	Scheduling Design	Fault Tolerance	Scalability	Real-world Existence	Application Areas
Central	easy	low	low	frequent	prod. grids
Decentral	hard	high	high	rare	acad./research
Other	easy	specific	high	research	specific, P2P

Knowledge Representation Schemes

Suggested reading

D. Ruan (editor), *Intelligent Hybrid Systems: Fuzzy Logic, Neural Networks, and Genetic Algorithms*[26]: Collection of contributed chapters covering basic principles, methodologies, and applications of fuzzy systems, neural networks, and genetic algorithms.
A. B. Markman, *Knowledge Representation*[27]: Overview of types of knowledge representation techniques and their use in cognitive models. It is useful for students of psychology as well as researchers in related disciplines, for example, computer science.

There are several requirements for knowledge representation in the context of scheduling decisions: The representation is supposed to capture the current complex state of a scheduling system in a comprehensive manner but it should restrict the number of features to describe the state to limit the dimension of the search space in scheduler optimization. The description must be sufficient to specify scheduling decisions with respect to the underlying grid-scheduling model. Furthermore, the output decision must be computable in an efficient way as scheduling often requires many online decisions in a short amount of time. In addition, the knowledge representation should be extensible or even generalizable. It must be possible to derive scheduling decisions for foreign states from already known states.
The first and most common kind of knowledge representation is the fuzzy rulebase, that

is, a set of fuzzy rules. The rule premise defines the local region of the state space to which the rule applies. The consequent associates a scheduling decision with the states represented in the premise. To compute the overall output of the fuzzy rulebase, the membership degree of every rule (real value $\in [0, 1]$) is used as weight for each rule's output scheduling decision. Examples for this kind of representation[28] are provided in several publications by Prado et al.[14,18,19]. As they assume a meta-scheduling model with a scheduler that schedules incoming jobs on independent computing sites, the authors describe their states as tuple of seven site-specific values. Such features are, for example, the number of free processing elements or the total lateness of all executed jobs within a site. Every rule output represents a factor that influences the controller output. In the inference system, a rulebase calculates the output for every site separately. Afterwards the site with the highest output is chosen for allocation. Similar grid-scheduling approaches[15,17] use fuzzy rules which only consist of two state variables. In those cases, the system state is represented as the processor- and memory utilization.

Other fuzzy based grid-scheduling algorithms[20,21,22] make use of a Takagi-Sugeno-Kang (TSK)[29] controller. Instead of using site-specific metrics which cover a large number of system states, only two different state values are allowed. As their decentralized grid-model comprises autonomously negotiating sites, the first feature is the summed resource requests of all queued jobs at the site. It estimates the workload the site has to process in the near future. The second value expresses a single job's parallelism to indicate the work that is added when accepting the job for execution. The output of a rule is the decision whether a job should be processed locally or negotiated for remote execution.

In contrast to these approaches, Abraham et al.[25] describe a fuzzy representation with so-called particles using a direct mapping of jobs to specific resources. This is only possible since all resources and jobs are known from the beginning in an offline scheduling problem. They optimize the assignment of jobs to specific resources. In the defuzzyfication (that is, calculating the output decision assuming fuzzy input values) the resource that gets the highest output decision values is chosen. In contrast to regular fuzzy systems, they do not apply any superposition of rules.

A similar approach[24,12] omits the vague representation as fuzzy sets in favor of using direct mapping of jobs to resources. A vector of machine indexes is used where each vector component specifies the machine on which a certain job should be scheduled on. Although this representation is simple and thus easy to update during the knowledge discovery process, it lacks the possibility to optimize different permutations of jobs on a specific machine. This leads to restrictions in the search space of possible scheduling solutions.

There are two different knowledge representation schemes which are motivated by the associated learning method: In the first one of Zeng et al.[23], the scheduling decisions are optimized by learning a value function which is typically employed in reinforcement learning. According to the value function, independent job agents identify an optimal policy for choosing the most beneficial resource agent among the competing resource agents. In the second more exotic knowledge representation, Li et al.[13] store tuples of job characteristics like the user group, user name, number of requested processors, and resulting run- or wait times within a database. The knowledge base is

extended as new job execution data is collected thus providing more accurate estimates of future jobs' runtimes.
Overall, there are many representations with their respective advantages and disadvantages but all of them strongly depend on the underlying grid-model, that is, state description, variables, and calculation of scheduling decisions, see also Table 2. An offline scheduling model with a priori knowledge about all jobs and resources allows fuzzy mapping of concrete jobs to concrete resources. In contrast, online grid-scheduling problems with changing resources, unforeseeable workload bursts, or restrictions in information policy need state descriptions that characterize all future resource states and job characteristics. Knowledge representation and its learning process are closely coupled. The different knowledge acquisition schemes are presented in the next section.

Table 2 Overview: pros and cons of different knowledge representations.

Representation	Size of Input Space	Output Calculation	Update Procedure	Req. Memory	Robustness
Fuzzy system	small	easy	hard	small	high
Direct mapping	large	easy	easy	large	very low
Value function	small	easy	hard	small	low
Database	large	hard	easy	very large	high

Knowledge Acquisition Schemes

Suggested reading

O. Maimon and L. Rokach, *Data Mining and Knowledge Discovery Handbook*[30]: Provides comprehensive algorithmic descriptions of data-mining and knowledge discovery methods, including classic methods plus the extensions and novel methods developed recently. Especially part IV addressing "Soft Computing Methods" is strongly recommended.

The knowledge acquisition process is responsible for generating and updating the knowledge representation. The solution space is defined by the state variables of the representation. The knowledge acquisition process explores this space searching for representation instances that are optimal with respect to a global objective. The learning performance and the solution quality not only depend on the objective but also on the optimization algorithm for search space exploration. In some of the aforementioned publications, perceptions from the scheduling system are stored in a database.[13] In contrast to other approaches, the knowledge discovery is in such cases mere knowledge collection instead of refining knowledge representations.

Refinements are commonly realized applying methods from evolutionary optimization. Fölling et al.[20,21,22] use an evolutionary learning algorithm to learn the fuzzy rulebase. In their evolutionary fuzzy system, the shape of the membership functions and rule outputs are encoded as parameters. These parameters are evaluated for each individual in a dedicated simulation of the scheduling system. The objective expresses the achieved scheduling results.
The method of Xhafa et al.[12] is similar to the previous because they also use an approach from evolutionary computation, namely a genetic algorithm. They employ specific operators for recombination and mutation targeted to their particular offline scheduling problem. The authors basically evaluate four different crossover operators. They further try different rules for the exchange of sequences between two parent individuals. In addition, their mutation operators either switch single jobs among machines (mutually exchanging two jobs between two machines) or rebalance jobs between the machines. In this example, the objective function is a weighted combination of the two conflicting objectives makespan and flow time, while makespan is treated as main objective.
The *makespan* is equivalent to the completion time of the last job in the schedule (assuming the schedule starts at time 0). The *flow time* is defined as the span between the release time and the completion time of a job. Interchangeably, the term *response time* is used in the literature. A particular criteria is the average weighted response time (AWRT), in which a job's response time is weighted with its resource consumption, see Schwiegelshohn[31]. The wait time is the span between the submission time and the start time of the job.
Prado et al.[19] use the so-called Pittsburgh approach[32] to evolutionary fuzzy systems. Each individual represents a whole rulebase that is updated using crossover and mutation operators similar to the approach of Fölling et al.[22]. In addition, they present a concept for evolving rules with a Michigan approach[33] where single rules are evaluated with regard of their negative or positive influence on the scheduling performance. Those different approaches are then combined[14] to form a hybrid solution of updating the weights, that is, the influences of single rules. They use the Michigan approach for updating single rules and the Pittsburgh approach to update whole rulebases.
Contrary to the previous publications, Abraham et al.[25] and Prado et al.[18] optimize knowledge representations using particle swarm optimization. In this kind of optimization technique, the aforementioned individuals are replaced by particles. During the learning process, particles are neither recombined nor mutated like individuals in evolutionary algorithms. Each particle represents knowledge independently while they move through the search space. The movement is determined by a velocity matrix that has entries for each state representation of a particle. This velocity is comparable to mutation step-sizes applied in evolutionary optimization. The great advantage of this approach is that this optimization method is not generation-based anymore. All particles are simulated independently without any need for synchronization. This allows an easy parallelization and hence more evaluations per time unit are possible.
The approach of Farzi[24] is very similar to the previously described particle swarm concept. The approach employs fishes as swarm members that encode allocations of jobs to resources. The algorithm mimics the natural behavior of fishes in a swarm-oriented learning algorithm. Fishes are chasing for food that represents the objective in terms of

a combination of makespan and flow time. The fishes swim towards regions of higher food concentrations that are detected by swarm neighbors. Simultaneously, they spread across the near search space to guarantee divergence among solutions.
In another solution[23], a temporal difference reinforcement learning approach is used to optimize the value function that predicts the estimated reward of choosing a specific action in a certain situation. Different from other works, the knowledge is updated during the evaluation process. Every single chosen scheduling decision is evaluated and rewards or penalties are assigned afterwards. Hence, each evaluation influences the optimal value function directly. Nevertheless, this learning approach is an offline approach as the same workload is used in each iteration. The relevance, advantages, and disadvantages of the presented approaches are listed in Table 3.

It is important to distinguish the online/offline properties in the two domains knowledge acquisition and grid-scheduling. In offline grid-scheduling problems, all job properties are known from the beginning while in online problems future job submissions are not known in advance. In practice, all grid-scheduling problems are online problems but many offline problems are studied for algorithm development. For knowledge acquisition, offline learned knowledge might be applied to online grid-scheduling problems but only an online learning (or adjustment) approach is able to flexibly react to changing environments. To the best of our knowledge, there is no such online knowledge acquisition approach for grid-scheduling available yet.

Table 3 Overview: pros, cons, and occurrence of knowledge acquisition schemes for grid-scheduling.

Method	Real-world Existence	Method's Complexity	Learning Speed	Online Adaption
EFS (Pittsburgh)	frequent	low	slow	hard
EFS (Michigan)	rare	high	slow	hard
Particle swarm	rare	medium	fast (parallel)	possible
RL	very rare	medium	slow	possible

EFS = Evolutionary fuzzy system
RL = Reinforcement learning

Advantages of Knowledge Application

Suggested reading

J. Blazewicz et al., *Handbook on Scheduling: From Theory to Applications*[34]: Although not related to grid-scheduling, this well established textbook introduced schedul-

ing with special emphasize on practical applications. The last chapter (computer integrated production scheduling) gives examples of knowledge-based scheduling.

This section discusses how the acquired knowledge is utilized for decision making in the grid-scheduling process. Research mainly focuses on two different subjects: On the one hand, knowledge is used to design or improve grid-scheduling algorithms directly related to well-defined performance criteria. On the other hand, the interesting problem of grid-scheduling is rather used as benchmark problem while improving learning strategies is the main goal, see Table 4.

Designing Grid-scheduling Strategies

For assessing the schedule quality, it becomes necessary to refer to specific objective functions. Common examples are simple functions such as makespan or the sum of completion times[35]. In addition, practical objective functions are often hard to formulate, because they are meant to express a complex problem view[36]. Objectives might even change over time if knowledge about the maximum achievable solution quality becomes available.[37] However, to show the advantages of knowledge usage in grid-scheduling, simple objectives are mainly used.
Zeng et al.[23] minimize the makespan during the knowledge acquisition phase and measure the performance of the applied learning algorithm considering the convergence speed and makespan. They evaluate the robustness of their derived scheduling strategy on unforeseen events like machine unavailability or breakdowns. Also Xhafa et al.[12] aim to minimize the makespan or the sum of response times in their approach. The derived knowledge base is used to plan or reschedule job allocations periodically while the objective functions are always obeyed.
Other developments also take the specific job attributes into account: Li et al.[13] use the estimates of response times and runtimes to make better decisions at the meta-scheduling level. The derived knowledge is then used to choose the most appropriate execution site in the grid. For a new job, its similarity to other jobs in the database is determined and the closest match is used to estimate the wait time or runtime of the new job. The accuracy of the algorithm is evaluated with real workload traces.
The work of Fölling et al.[20] comprises both a scheduling objective related knowledge learning phase and a subsequent application phase. During the learning phase on real workload data, the scheduling objective (AWRT) is optimized and corresponding rules are extracted for the knowledge base. Dynamically, they derive rules to steer the workload distribution among grid-sites. After the learning phase, the rulebase is applied to new data from the workload traces to show that the derived knowledge is applicable for other workloads. In a later publication[22], the authors show the robustness in changing grid-environments. Setups with previously unknown partners are evaluated to show the robust interaction between the autonomous sites.

Table 4 Overview: different grid-scheduling design concepts.

Concept	Objective	Knowledge used for	Sched. Probl.	Used Traces
Zeng et al.[23]	makespan	workflow negotiation	online	synth.
Xhafa et al.[11]	makespan/RT	single domain grid-scheduling	offline	synth.
Li et al.[13]	prediction failure	predicting RT and WT in local scheduling	online	real
Fölling et al.[22]	AWRT	job negotiation	online	real
Prado et al.[18]	makespan	meta-scheduling	online	real
Prado et al.[14,19]	AWRT	meta-scheduling	online	real
Farzi[24]	makespan/RT	single domain grid-scheduling	offline	synth.
Abraham et al.[25]	makespan/RT	single domain grid-scheduling	offline	synth.

RT = total response time
WT = wait time
AWRT = average weighted response time

Using Grid-scheduling as a Test Problem

While all above described works have a scheduling related point of view, the following works turn to the learning technique itself, see Table 5. Prado et al.[14] also aim to minimize the AWRT of all jobs but they additionally investigate the performances achievable by two differently learned fuzzy systems. They compare the Michigan and Pittsburgh approach for evolutionary fuzzy rule learning and show that their hybrid approach yields superior learning result. For this investigation, they focus on a fast convergence of the learning approach instead of deriving very robust rulebases.
In another publication[18], they focus on convergence speed and show by experiments that swarm-optimization outperforms genetic algorithms. In a follow-up investigation[19], the same authors vary the number of involved sites as well as the number of overall available resources.
Similar to the previously discussed approach, Fölling et al.[21] also try to improve the learning technique itself. They define a problem specific adaptation of the learning by dividing the overall problem into multiple subproblems. They adapt a co-evolutionary algorithm for the learning of fuzzy rules and they show that it achieves nearly the same results while reducing the learning time significantly.
In other approaches, Farzi[24] and Abraham et al.[25] learn knowledge in the grid-scheduling context using techniques from computational intelligence, like special genetic algorithms and simulated annealing, to optimize both makespan and response times.

Table 5 Overview: aspects of learning technique improvements.

Aspect	Farzi[24] Abraham et al.[25]	Fölling et al.[21]	Prado et al.[14,18,19]
Fuzzy system	—	operators for Pitts. approach	comp. Michigan and Pitts. approaches
Learning alg.	PSO and AFSA	co-evolutionary algorithm	hybrid approaches
Objective	convergence speed to known optimum	learning parallelization	convergence speed

PSO = Particle swarm optimization
AFSA = Artificial fish swarm algorithm (similar to PSO)

Conclusion

The subject of scheduling in computational grids has been approached from the viewpoint of knowledge discovery. It has been shown how grid-scheduling benefits from utilization of domain knowledge acquired from historical data of the grid. The approaches differ in terms of the learning methodology, the representation of the acquired knowledge and the way in which the knowledge improves the scheduling decisions in dynamic environments. It became obvious that grid-scheduling comprises many areas: The diversity of computational grids reflects itself in the manifold of scheduling problems and approaches to their solution. Recent developments have been presented in a structured way while their specific contributions to the area of knowledge discovery and decision learning have been pointed out. The conceivable grid-models have been distinguished and analyzed. It became apparent that scheduling in the fully decentralized model is more challenging but therefore perfectly suitable for knowledge usage. State-of-the-art knowledge representation are described and the concepts for knowledge acquisition in recent grid-scheduling architectures are reviewed. Hybrid methods from computational intelligence successfully combine fuzzy representations with randomized, heuristic global search methods. Learning in frequently changing environments is difficult and grid-scheduling constitutes an inherent online problem with a partial horizon. Integrating knowledge gathered from past observations into the decision making is a means to improve scheduler performance. Experience with knowledge based grid scheduling shows that it is possible to transform and extrapolate existing knowledge to novel environments and workload scenarios. Even in a challenging application domain like grid-scheduling good decisions are made.
As knowledge discovery obviously supports future grid-scheduling developments, some further directions are conceivable. The first possible improvement is related to the grid-model itself which is still a bleeding-edge technology in which technological drifts occur frequently. Since the approaches and developments in the past decade already cover diverse architectures, it is likely that future distributed computing architecture such as cloud-computing utilize

similar techniques for knowledge acquisition. Current state of the art knowledge discovery techniques are equipped with an internal or external learning technique. As such, learning normally occurs in episodes or generations which assume a static or quasi-static environment. In the application phase, the acquired knowledge is helpful to make decisions in a changing environment. It is strongly desired to overcome this drawback and to head for the online learning of knowledge. As all grid-systems are online systems, the learning should be capable to adapt to changes in an online manner. Imagine a scheduler that starts with an initial heuristic, which is then gradually improved by learning from the experience of past decisions. In such a setup, knowledge discovery becomes much more practically feasible and relevant as real-world grid-installations are equipped with a zero-configuration scheduler. This scheduler automatically adapts to the changing environment while simultaneously obeying the user preferences and constraints. As the overall grid is far too complex to be entirely monitored and controlled by human experts, the techniques for knowledge discovery and utilization discussed in this paper may play an important role in the development of next generation computing infrastructures.

Acknowledgment

The authors would like to thank Frank Hoffmann for his valuable comments and corrections.

References

[1] R. C. Eberhart and Y. Shi. *Computational Intelligence: Concepts to Implementations*. Morgan Kaufmann, 1st edition, 2007.

[2] H.-P. Schwefel. *Evolution and Optimum Seeking*. John Wiley & Sons, 1st edition, 1995.

[3] O. Cordón, F. Herrera, F. Hoffmann, and L. Magdalena. Evolutionary tuning and learning of fuzzy knowledge bases. In *Genetic Fuzzy Systems*, volume 19 of *Advances in Fuzzy Systems - Applications and Theory*. World Scientific, Singapore, 2001.

[4] A. P. Engelbrecht. *Computational Intelligence — An Introduction*. John Wiley & Sons, 2nd edition, 2007.

[5] I. Foster and C. Kesselman. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, 2nd edition, 2003.

[6] M. Pinedo. *Scheduling: Theory, Algorithms, and Systems*. Prentice-Hall, 4th edition, 2012.

[7] L. Smarr and C. E. Catlett. Metacomputing. *Communications fo the ACM*, 35(6):44–52, 1992.

[8] J. Altmann, M. Ion, and A. A. B. Mohammed. Taxonomy of grid business models. In J. Altmann and D. Veit, editors, *Proceedings of the 4th International Workshop on Grid Economics and Business Models*, volume 4685 of *LNCS*, pages 29–43. Springer, 2007.

[9] D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Prentice-Hall, 1st edition, 1997.

[10] F. Berman, G. Fox, and A. J. G. Hey. *Grid Computing: Making The Global Infrastructure a Reality*. John Wiley & Sons, 1st edition, 2003.

[11] F. Xhafa and A. Abraham. Meta-heuristics for grid scheduling problems. In *Metaheuristics for Scheduling in Distributed Computing Environments*, volume 146 of *Studies in Computational Intelligence*, pages 1–37. Springer, 2008.

[12] F. Xhafa, J. Carretero, and A. Abraham. Genetic algorithm based schedulers for grid computing systems. *International Journal of Innovative Computing, Information and Control*, 3(6):1053–1071, 2007.

[13] H. Li, D. Groep, and L. Wolters. Mining performance data for metascheduling decision support in the grid. *Future Generation Computer Systems*, 23(1):92–99, January 2007.

[14] R. P. Prado, S. García-Galán, A. Yuste, and J. E. Munoz-Expósito. Genetic fuzzy rule-based meta-scheduler for grid computing. In *Proceedings of the 4th International Workshop on Genetic and Evolutionary Fuzzy Systems*. IEEE Computational Intelligence Society, 2010.

[15] J. Zhou, K.-M. Yu, C.-H. Chou, L.-A. Yang, and Z.-J. Luo. A dynamic resource broker and fuzzy logic based scheduling algorithm in grid environment. In B. Beliczynski, A. Dzielinski, M. Iwanowski, and B. Ribeiro, editors, *Proceedings of the 8th international conference on Adaptive and Natural Computing Algorithms*, volume 4431 of *LNCS*, pages 604–613. Springer, 2007.

[16] I. Foster. Globus toolkit version 4: Software for service-oriented systems. In H. Jin, D. A. Reed, and W. Jiang, editors, *IFIP International Conference on Network and Parallel Computing*, volume 3779 of *LNCS*, pages 2–13. Springer, 2005.

[17] K.-M. Yu, Z.-J. Luo, C.-H. Chou, C.-K. Chen, and J. Zhou. A fuzzy neural network based scheduling algorithm for job assignment on computational grids. In T. Enokido, L. Barolli, and M. Takizawa, editors, *Proceedings of the 1st international conference on Network-based information systems*, volume 4658 of *LNCS*, pages 533–542. Springer, 2007.

[18] R. P. Prado, S. García-Galán, J. E. Munoz-Expósito, and A. J. Yuste. Knowledge acquisition in fuzzy-rule-based systems with particle-swarm optimization. *IEEE Transactions on Fuzzy Systems*, 18(6):1083–1097, 2010.

[19] R. P. Prado, S. García-Galán, A. J. Yuste, and J. E. Munoz-Expósito. Genetic fuzzy rule-based scheduling system for grid computing in virtual organizations. *Soft Computing*, 15(7):1255–1271, 2011.

[20] A. Fölling, C. Grimme, J. Lepping, and A. Papaspyrou. Decentralized grid scheduling with evolutionary fuzzy systems. In E. Frachtenberg and U. Schwiegelshohn, editors, *Proceedings of the 14th Workshop on Job Scheduling Strategies for Parallel Processing*, volume 5798 of *LNCS*, pages 16–36. Springer, 2009.

[21] A. Fölling, C. Grimme, J. Lepping, A. Papaspyrou, and U. Schwiegelshohn. Competitive co-evolutionary learning of fuzzy systems for job exchange in computational grids. *Evolutionary Computation*, 17(4):545–560, 2009.

[22] A. Fölling, C. Grimme, J. Lepping, and A. Papaspyrou. Robust load delegation in service grid environments. *IEEE Transactions on Parallel and Distributed Systems*, 21(9):1304–1316, 2010.

[23] B. Zeng, J. Wei, and H. Liu. Dynamic grid resource scheduling model using learning agent. In *Proceedings of the 2009 IEEE International Conference on Networking, Architecture, and Storage*, pages 67–73. IEEE Computer Society, 2009.

[24] S. Farzi. Efficient job scheduling in grid computing with modified artificial fish swarm algorithm. *International Journal of Computer Theory and Engineering*, 1(1):13–18, April 2009.

[25] A. Abraham, H. Liu, W. Zhang, and T. Chang. Scheduling job on computational grids using fuzzy particle swarm algorithm. In B. Gabrys, R.J. Howlett, and L.C. Jain, editors, *Proceedings of the 10th International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*, volume 4252 of *LNAI*, pages 500–507. Springer, 2006.

[26] D. Ruan, editor. *Intelligent Hybrid Systems: Fuzzy Logic, Neural Networks, and Genetic Algorithms*. Springer, 1st edition, 1997.

[27] A. B. Markman. *Knowledge Representation*. Psychology Press, 2nd edition, 1998.

[28] E. H. Mamdani and S. Assilian. Application of fuzzy algorithms for control of simple dynamic plant. *Procedings of IEEE*, 121(12):1585–1588, 1974.

[29] 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, Februar 1985.

[30] O. Maimon and L. Rokach, editors. *Data Mining and Knowledge Discovery Handbook*. Springer, 1st edition, 2005.

[31] U. Schwiegelshohn. A system-centric metric for the evaluation of online job schedules. *Journal of Scheduling*, 14(6):571 –581, 2011.

[32] S. F. Smith. *A Learning System Based on Genetic Adaptive Algorithms*. PhD thesis, Department of Computer Science, University of Pittsburgh, 1980.

[33] 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.

[34] J. Blazewicz, K. H. Ecker, E. Pesch, G. Schmidt, and J. Weglarz. *Handbook on Scheduling: From Theory to Applications*. International Handbooks on Information Systems. Springer, 1st edition, 2007.

[35] P. Brucker. *Scheduling Algorithms*. Springer, 4th edition, 2004.

[36] C. Franke, J. Lepping, and U. Schwiegelshohn. Greedy scheduling with custom-made objectives. *Annals of Operations Research*, 180(1):145–164, 2010.

[37] C. Ernemann, U. Schwiegelshohn, M. Emmerich, L. Schönemann, and N. Beume. Scheduling algorithm development based on complex owner defined objectives. Technical Report 191/05, SFB 531, CI-Report, 44221 Dortmund, Germany, January 2005.