Future Generation Computer Systems 25 (2009) 237–256

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Towards a general model of the multi-criteria workflow scheduling on the grid

Marek Wieczorek [a,*], Andreas Hoheisel [b,1], Radu Prodan [a,2]

[a] Institute of Computer Science, University of Innsbruck, Technikerstraße 21a, A-6020 Innsbruck, Austria

[b] Fraunhofer FIRST, Kekuléstraße 7, D-12489 Berlin, Germany

ARTICLE INFO

Article history:
Received 5 June 2008
Received in revised form
28 August 2008
Accepted 1 September 2008
Available online 16 September 2008

Keywords:
Grid computing
Workflow
Multi-criteria scheduling
Taxonomy

ABSTRACT

Workflow scheduling on the Grid becomes more challenging when multiple scheduling criteria are considered. Existing studies provide different approaches to the multi-criteria Grid workflow scheduling problem, and address different variants of the problem. A profound understanding of the problem's nature can be an important step towards more generic scheduling approaches. Based on the related work and on our own experience, we propose several novel taxonomies of the problem, considering five facets: workflow model, scheduling criteria, scheduling process, resource model, and task model. We make a survey of the existing related work, and classify it according to the proposed taxonomies, identifying the most common use cases and the areas that have not been sufficiently explored yet.

1. Introduction

Executing complex processes in heterogeneous distributed computing environments is a major goal in applying information and communication technologies in industry and science. Workflows are commonly used for automating the data and control flow of distributed IT processes. The term "workflow scheduling" refers to the resource planning, i.e., the spatial and temporal mapping of workflow tasks onto resources.

Scheduling of computational tasks and workflows on the Grid is a complex optimization problem, that often requires several scheduling criteria to be considered. In recent years, many theoretical studies on distributed and Grid systems have been realized, that schedule tasks and workflows for more than one scheduling criterion. Many of these approaches address only specific criteria — usually execution time and economic cost. Some of them also aim at increasing the Grid's utility, by considering criteria such as execution fairness, resource usage, and global efficiency (job throughput). Finally, some approaches try to address a more generic model and schedule tasks and workflows with respect to user-defined scheduling criteria. However, in a multi-dimensional parameter space, it is in general not possible to find a solution that is "best" with respect to all the metrics at the same time.

There are several existing approaches to the problem of multi-criteria scheduling. One possibility, is to extend the definition of optimality to *Pareto optimality* [1], and therefore assume all solutions that are *not dominated* by any other solution to be optimal. Pareto optimality does not seem to be a proper optimization goal, as Pareto sets are usually very large, and include also degenerated solutions such as solutions that are optimal for one criterion only. Concerning the existing bi-criteria scheduling approaches, they are usually limited to optimizing two specific criteria [2–7], which is a consequence of the difficulties with the development of a general multi-criteria scheduling approach. Many existing bi-criteria approaches impose some fixed constraints on one criterion (e.g., deadline or budget limitation), and try to optimize the schedule for one criterion only. Another common solution, is to optimize a linear combination of multiple scheduling criteria with a different weight value assigned to each of them. The approaches that try to optimize a linear combination of multiple criteria [8,9] assume that the user is able to specify the requirements in such a model, which is not always the case. Moreover, the workflow structure and the quantitative effects of multiple task scheduling decisions made within a single workflow have to be taken into consideration when scheduling a workflow. As a consequence, the scheduling techniques that do not implement any specific mechanisms to address these issues have to handle the workflow as a "black box" and evaluate only the complete workflow schedule, which may have a negative impact on the scheduling effectiveness and on the scheduling time. A careful analysis of different types of criteria is required in order to be able to better address the problem.

There are several other aspects of the multi-criteria Grid workflow scheduling problem, which make this problem challenging,

* Corresponding author. Tel.: +43 512 507 96831; fax: +43 512 507 2758.
E-mail addresses: marek@dps.uibk.ac.at (M. Wieczorek), andreas.hoheisel@first.fraunhofer.de (A. Hoheisel), radu@dps.uibk.ac.at (R. Prodan).
1 Tel.: +49 30 6392 1819; fax: +49 30 6392 1805.
2 Tel.: +43 512 507 6445; fax: +43 512 507 2758.

0167-739X/$ – see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.future.2008.09.002

and prevent researchers from developing more general scheduling approaches. For instance, many static full-graph scheduling approaches originate from cluster systems that are usually smaller, more centralized, and less dynamic systems than contemporary Grids. When applied to the Grid, the same approaches may appear to be much less efficient, due to the higher level of complexity of Grid systems. As the applications that can be executed on the Grid may come from many different domains, the characteristics of the workflows may also differ considerably. For instance, workflows can be based on different basic structures (e.g., parallel section, tree) and different processing principles (e.g., pipelined workflows), can be dynamically adjustable, and also the workflow tasks may have a different nature (e.g., moldable and malleable tasks). Scheduling for each of these types of workflows can be considered as a separate problem class. There has been much scientific effort that tried to address some of the aforementioned classes of the problem. The experience coming from all this work should be analyzed and classified, in order to support the development of more general scheduling models. Recently, several extensions to the classical resource management model have been proposed and investigated, including the concepts of advance reservation and Grid economy. How these concepts can be incorporated in workflow scheduling is still an open research problem. Finally, in the domain of scheduling, multiple terms have sometimes been invented to describe the same concept. For instance, the concepts of multiprogrammed and non-multiprogrammed tasks are often referred to as "cumulative" and "disjunctive" tasks, respectively, and the concept of full-graph scheduling is often referred to as "global" scheduling (as opposed to more dynamic "local" scheduling). The existence of a common standardized terminology would be an important factor stimulating the further progress in the domain of workflow scheduling.

The goal of the current work, is to analyze the general problem of Grid workflow scheduling, mostly for multiple scheduling criteria, by reviewing the related work and discovering regularities and irregularities between different problem classes. Rather than focusing on the solutions applied, we try to recognize the complexity of the problem itself. In our research, we use the related work as a guideline, but, in addition, we also introduce ideas coming from our own experience and analysis. Another goal of this article is to propose a uniform terminology to describe different scheduling-related concepts. Our main contribution is a set of five novel taxonomies of the workflow scheduling problem that embrace five different *facets* (aspects): workflow model, scheduling criteria, scheduling process, resource model, and task model. We try to answer the question of how generic we can be in developing multi-criteria scheduling approaches, and which classes of the general workflow scheduling problem could be addressed by a single generic approach. We also summarize the presented survey of the related work in several tables, where different scheduling systems are classified according to the proposed taxonomies. Finally, we try to illustrate the usefulness of our classification effort by presenting a generic bi-criteria scheduling approach [10] developed by us — that is based on the presented taxonomies.

The rest of the article is organized as follows. In Section 2, we formally describe the problem which we address in this article. Section 3 gives a selective overview of the existing classification efforts in the area of Grid computing. Section 4 gives a short introduction to the taxonomies introduced by us for the workflow scheduling problem, that are subsequently described in detail in Sections 5–9. The taxonomies consider both different problem variants and different approaches used to solve the problem. Section 10 analyzes the existing work in the context of the introduced taxonomies, by identifying the problem classes that have been most commonly addressed and those that have not been explored yet. Section 11 presents an example application of the introduced taxonomies — a bi-criteria workflow scheduling approach designed for a generic criteria model, based on the taxonomy of scheduling criteria. Finally, Section 12 concludes the article and provides a short roadmap for the future work.

2. Grid workflow scheduling problem

This article focuses on workflows modeled as *graphs* that consist of a set of *nodes* and a set of *edges*, where nodes represent computational *tasks* (either single or multiple) or *data transfers*, and edges represent control and data *dependencies* (multiple data transfers can be associated with a single edge). In addition, some workflow representations applied in the related work consider special semantics assigned to the workflow tasks, such as composite sequential and parallel loops, or `if`/`switch` conditions. Examples of such representations are Petri Nets [11–13] applied in the K-WfGrid project [14], Abstract Grid Workflow Language (AGWL) [15] applied in ASKALON [16], and Business Process Execution Language (BPEL) [17]. In Section 5 we analyze the workflow representations more in detail. In the scheduling process, workflow tasks are mapped to *services* that are available in the Grid and *implement* these tasks. One task can be mapped to many alternative services, and one service can implement many tasks. *Scheduling* is defined as a function that maps each task in a workflow to a service deployed on the Grid that implements this task. The *cost model* for workflows is described by multiple *scheduling criteria*, for instance, by execution time, economic cost, and data quality. The *partial cost functions* defined for each scheduling criterion assign to each service deployed on the Grid its *partial cost* (e.g., "execution time of 5 min", "economic cost of 5€", "data quality of 100%"). In the remainder of this article, we will sometimes refer to the cost of a service s implementing a task τ as the *cost of the task* τ. Similarly to the partial cost function, the *total cost function* assigns to a workflow scheduled by a scheduling function the *total cost* of the workflow that is calculated based on the partial costs of the services mapped to the workflow tasks. The optimization goal is to find for a workflow a schedule that provides the *best possible* total cost. As we describe in Section 6, the total costs can be evaluated in different ways.

3. Overview of the existing classification studies for the Grid

There has been much scientific effort in the past that aimed at classifying the existing Grid research based on certain taxonomies. The reason for creating taxonomies is to organize the knowledge in a certain domain, by finding regularities in all the research. As a primary goal, taxonomies should facilitate the understanding of the classified knowledge and what follows, they should stimulate the further research in the area.

For instance, a taxonomy of Grid resource management systems has been proposed by Krauter et al. [18], a taxonomy of data Grids and distributed data systems has been proposed by Venugopal et al. [19], and a taxonomy of Grid monitoring systems has been proposed by Zanikolas and Sakellariou [20]. Different Grid economy-based systems have been classified by Yeo and Buyya [21], with respect to the market models applied in them. Plankensteiner et al. [22] performed a survey by classifying and identifying faults that can be detected, recovered, and prevented by existing Grid workflow management systems. Truong et al. [23] classified performance metrics that performance monitoring and analysis tools should provide during the evaluation of the performance of Grid workflows, and introduced an ontology for describing performance data of workflows. Concerning the research done on workflow models, Van der Aalst et al. [24] compiled the experience coming from

different workflow description languages, and proposed a set of 43 *workflow patterns* [25] that systematically describe different types of workflow constructors that can be supported by different workflow representation languages. The studies that mostly resemble the current work have been presented by Yu and Buyya [26] who classified different Grid workflow management systems, and by Casavant and Kuhl [27] and by Dong and Akl [28], who made an extensive survey and classified different scheduling approaches in distributed systems.

Our current work is focused on a specific problem that is multi-criteria Grid workflow scheduling, and our scientific effort is to introduce taxonomies that classify the existing and future research from this particular perspective. Our main goal is to investigate the question of how generic we can be in developing multi-criteria scheduling algorithms, which classes of the problem can be approached in a generic way, and which ones require specialized methods.

4. Principles of the taxonomies

When analyzing the problem of workflow scheduling, several important *facets* (e.g., resource model, criteria model) of the problem have to be considered, as they may strongly influence the decision as to which scheduling approach is most appropriate in the given case. Each facet describes the scheduling problem from a different perspective. In this section, we analyze in detail five different facets of the problem:

- workflow model
- scheduling criteria
- scheduling process
- resource model
- task model.

For every facet, we propose a certain *taxonomy* that introduces different possible *classes* distinguished according to different *aspects* of the problem. The classes are distinguished either with respect to different variants of the scheduling problem (e.g., multiple workflows, user-oriented scheduling), or with respect to the way the problem is approached (e.g., full-ahead planning, advance reservation based). We describe the classes, using the RDF [29] notation *subject-predicate-object*, that we extend in some cases to distinguish between different *sub-classes* of the problem. The proposed taxonomies can by no means be considered exhaustive, as our attempt is to create a model only for a certain subset of the general workflow scheduling problem, i.e., for the multi-criteria workflow scheduling on the Grid. We illustrate the derived taxonomies by providing examples of approaches for different classes that partially come from the related work. Some of those examples are taken from a more complete analysis of the scheduling problem on the Grid presented by Dong and Akl [28].

5. Taxonomy of workflow models

5.1. Introduction

Workflows may vary with respect to their level of complexity, to the semantics of their components, and to the dynamics of their execution model. All these aspects should be taken into consideration when scheduling workflows, as the execution of a workflow which is based on a specific model may require a specific scheduling approach, or may simply perform better when such an approach is applied. For instance, there exist many scheduling algorithms dedicated for parameter sweep workflows, even though more general DAG scheduling algorithms would also work for this workflow model. The taxonomy depicted in Fig. 1 differentiates workflows with respect to their representation and behavior.

5.2. Component model

In the literature [30,31], a workflow model is usually described by applying four different perspectives: *control-flow* perspective, *data* perspective, *resource* perspective, and *operational* perspective. The control-flow (or process) perspective describes tasks and their execution ordering through different workflow constructors, such as sequence, choice, parallelism, and synchronization. The data perspective defines the flow of data between tasks in the workflow. The resource perspective defines the responsibilities of resources for executing tasks (i.e., scheduling of tasks). Finally, the operational perspective describes the elementary actions executed by tasks (i.e., task implementation). The workflow model defined by us applies the *scheduling* perspective. We combine some of the four aforementioned perspectives, and focus on tasks and data transfers which are atomic workflow components being subject to scheduling (i.e., "schedulable units"). We distinguish two classes of workflow models:

- *Task oriented*. Tasks are represented as graph nodes. Data transfers or control preconditions are represented as graph edges.
- *Task and data transfer oriented*. Both tasks and data transfers are represented as graph nodes.

Business workflows often consider only the control-flow of the processes, while scientific IT workflows mostly imply data-flows together with the execution of tasks. As our main focus is on scientific IT workflows, we treat control workflows as a special case of task oriented workflows, where the edges just specify the flow of boolean tokens representing the pre-conditions of the control-flow.

The existing Grid workflow scheduling approaches are based predominantly on the task oriented model. There are only few workflow representations which support the task and data transfer oriented model (e.g., Karajan [32] and Stork [33]). In Vienna Grid Environment [8], the low level workflow representation denotes both tasks and data transfers as workflow nodes. However, in the high level representation used for requirement specification and scheduling, there are no separate nodes representing data transfers. The distributed query workflows considered by Gounaris et al. [34] include also special workflow nodes called *exchange operators* that involve communication between other workflow nodes.

5.3. Structure

Different workflow scheduling methods consider different workflow structures, varying with respect to their level of generality and designed for different specific domains. The most common workflow model is the *Directed Acyclic Graph* (*DAG*), where dependencies between workflow nodes can be organized in any way that does not cause any cycles in the graph to be created. Some existing works extend this model by allowing cycles (loops) and other workflow constructors, such as parallel loops and conditionals (`if` or `switch`). We will refer to these models as *extended digraphs*. Some other models are based on a simpler structure than the general DAG model, for instance, workflow applications based on a single parallel section, pipelined applications, or special tree-like structures used in certain application domains. We will distinguish the following three workflow models:

- *DAG*. The workflow is a Directed Acyclic Graph.
- *Extended digraph*. The workflow structure extends the DAG model, by allowing some additional structures, such as loops and conditionals.

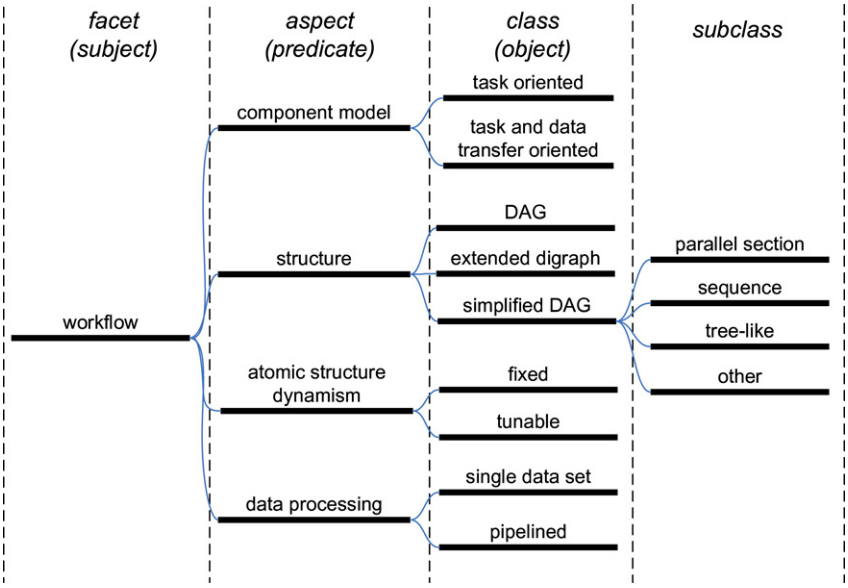

Fig. 1. Taxonomy of workflow model.

- *Simplified DAG*. The workflow structure has certain regularities, so it can be described by a well-defined subset of the DAG model.

Concerning the most common simplified DAG models that are encountered in the literature, we distinguish the following subclasses:

- *Parallel section*. The main computation is done in a single parallel section, where independent computations are distributed among multiple workers.
- *Sequence*. The workflow is a single sequence, for instance, a sequential pipelined application.
- *Tree-like*. The workflow graph is a tree.
- *Other*. Some other specific workflow structures such as Fast Fourier Transformation, Strassen algorithm, or fork-join DAGs.

In the Abstract Grid Workflow Language (AGWL) [15] used in ASKALON [16], workflows are expressed by means of hierarchical embedded structures (loops, parallel loop, conditionals, etc.), which is appropriate for a broad range of scientific workflows. For scheduling purposes, the workflows expressed in AGWL are converted to the DAG model [35]. The workflow representation used in the P-GRADE project [36] is based on the general DAG model; this model can be extended by using some specific components called *generator* and *collector*, used to construct parameter study workflows. Other scheduling approaches that consider the general DAG model were presented, for instance, by Sakellariou and Zhao [37,38], Tsiakkouri et al. [2], Mika et al. [39], and Yu et al. [3].

Many existing approaches are based on a specific workflow model. The most typical example of parallel section workflows are *parameter sweeps*, where all computations in the parallel section are of the same type, and each of them is executed for a different parameter value. The dynamic scheduling of parameter sweep applications considered by Ma and Buyya [40] is approached by a special prioritization policy that gives higher priority to the tasks whose so-called *children's ancestors* have already been finished. Scheduling of parameter sweep application is addressed also in the Nimrod/G system [41]. Casanova et al. [42] compare several heuristics for dynamic scheduling of parameter sweep applications (Min–min, Max–min, Suffrage), and propose a new heuristic called XSuffrage. The work presented by Subhlok and Vondran [43] considers a pipelined workflow model based on a sequence of data parallel tasks. The workflows used to model distributed database queries addressed by Gounaris et al. [34] are based on a special tree-like structure, constructed according to certain restricted composition rules. The workflows considered in Pegasus [44] also have a regular structure; the regular structure of these workflows was the motivation for introducing in Pegasus the idea of workflow partitioning that consists in converting the workflow to a sequence of subworkflows. Dynamic Data Mining workflows addressed by Luo et al. [45,46] are also based on a specific structure, corresponding to the data mining process.

5.4. Atomic structure dynamism

Apart from task mapping, also changing the basic workflow structure can be considered as a scheduling method. Workflow nodes (atomic workflow elements) can be added to or removed from a workflow, or can be grouped together to form new atomic elements, with the aim of increasing the profit of the user or of the Grid. We will say that an approach is designed for workflows with a *tunable* atomic structure, if it modifies the workflow structure (for optimization purposes) within the scheduling process, in contrast to the approaches that modify the workflow structure only as a consequence of a normal workflow execution (e.g., through loop unrolling or user interactions). We also impose an additional restriction on this group, by assuming that it contains only those approaches which add/remove/modify nodes, not those which just add/remove/modify dependencies. The reason for this is to exclude the approaches based on workflow clustering (i.e., on an auxiliary partition of the workflow to a set of non-atomic subworkflows), which is a standard scheduling approach. We introduce the following two workflow classes:

- *Fixed*. The atomic workflow structure is not changed during the scheduling process (some additional dependencies can be added or removed).
- *Tunable*. Atomic nodes can be added, removed, or modified during the scheduling process.

In K-WfGrid [14], workflows are created on demand and semantically tuned by the components called Workflow Composition Tool (WCT) [47] and Automatic Application Builder (AAB) [48]

before the tasks are mapped to services. Also in Pegasus [44], workflows are first converted from an *abstract* to a *concrete* form. Three different restructuring techniques are involved in this process. First, data sets that are produced by workflows running in the Grid can be reused in the subsequent workflow executions, which makes the execution of some workflow tasks unnecessary. Second, the *granularity* of a workflow is increased by combining (clustering) several tasks and treating the result as a single unit for mapping and scheduling. The third restructuring technique consists in clustering together several tasks scheduled to multiprocessor systems, and running them together as one schedulable unit, possibly in a master/slave fashion. The last two approaches aim at decreasing the scheduling overheads. In the approaches designed for pipelined workflows (e.g., in the work by Subhlok and Vondran [43]), tasks in the original sequence can be *replicated* (several instances of the same task may process different data sets in parallel) in order to increase the overall throughput.

5.5. Data processing

In this classification, we distinguish two different types of workflow processing that are addressed in different scheduling approaches. When considering the amount of data processed by an individual workflow, we can identify the following two workflow models:

- *Single input*. The workflow is executed once, for a single input that may consist of a single or multiple data elements.
- *Pipelined*. The workflow is executed many times, for multiple data inputs that are processed by the workflow as a stream.

Most of the existing Grid approaches address the first of the aforementioned classes. The second class is common in several application domains, including digital signal processing, image processing, and computer vision. The approach presented by Subhlok and Vondran [43] addresses the problem of scheduling of pipelined computations with the goal of optimizing the latency and the throughput of execution. The applications consist of a sequence of data parallel tasks that can be mapped onto a parallel machine in a variety of ways, employing different combinations of task and data parallelism. Tian and Chandy [49] analyze the problem of scheduling of pipelined (*streaming*) applications, and give several reasons why the classical scheduling algorithms are not well suited to the problem addressed by them. Although they define the problem for workflows, they provide only a solution for scheduling of single processing units. Very specific pipelined workflows used for data stream processing are considered by Schmidt [50]. Pipelined workflows are also supported in the P-GRADE Grid environment [36].

5.6. Summary

We distinguished four different aspects of the workflow model, based on the structure and the behavior of the workflow. In the context of the related work, the classification based on component model does not seem to be of major relevance, as almost all scheduling approaches are based of the task-oriented model. However, the task and data oriented model may show some potential benefits, in particular in the systems focused on data transfer optimization. The classification based on workflow structure seems to be important, since many scheduling approaches have been dedicated for different workflow structures distinguished by us. Several scheduling approaches consider tunable workflow, however this class still shows some research potential. Finally, the classification based on data processing introduces two different problem classes that clearly need to be approached in different ways.

6. Taxonomy of scheduling criteria

6.1. Introduction

The scheduling criteria may be characterized by various properties (e.g., workflow structure dependence, calculation method) that determine the optimization goal, and the way in which the total cost of a workflow is calculated for the given criterion. When scheduling workflows on the Grid, it is always important to take into consideration the type of criteria used as the optimization objectives in the given case. For instance, one scheduling algorithm will be applied when minimizing the execution time of a workflow, and another one will be applied when maximizing the quality of the results produced by a workflow. The scheduling criteria may also vary for different Grid actors (e.g., resource consumer, resource provider, environment) for whom the optimization goal is defined. The proposed taxonomy of scheduling criteria, considering both the properties of a single criterion and the joint properties of groups of criteria, is depicted in Fig. 2.

6.2. Optimization model

Considering workflow scheduling as an optimization process, we can distinguish two different perspectives from which the criteria can be defined:

- *Workflow-oriented*. The optimization criterion is defined for the user who executes the workflow (e.g., execution time, economic cost).
- *Grid-oriented*. The optimization criterion is defined for the Grid, usually for Grid sites, Virtual Organizations, or resource providers (e.g., resource usage, fairness of execution, economic profit).

Most of the related work proposes approaches based on the former perspective. The latter perspective is common for local resource management systems (e.g., PBS [51], Grid Engine [52], LSF [53], Maui [54]), and is also applied for workflow scheduling, for instance, in the work by Zhao and Sakellariou [38] where fairness of multiple workflow executions is considered as one of the optimization goals. Other Grid systems that consider Grid-oriented optimization include ASKALON [55] (resource usage, fairness), UNICORE 6 / Chemomentum [56] (statistical job turnaround time, resource usage), and Instant-Grid [57] (a special *resource quality* metric). Grid-oriented scheduling criteria (different resource consumption metrics) are considered also in the work on data stream processing workflows presented by Schmidt [50]. Dynamic cost models, based on Grid economy and on other negotiation-based strategies, that are described more in detail later in this section, can be used to equilibrate between the requirements of the resource consumer, the resource producer, and the Grid environment. Some research was done by Li and Yahyapour [58–60] to compare the influence that different negotiation strategies have on resource utilization on the Grid.

6.3. Workflow structure dependence

Whereas a task batch consists of independent tasks, workflows also contain dependencies that determine a certain *workflow structure*. For some scheduling criteria (e.g., for execution time), the structure has to be considered when calculating the total cost, while for some others (e.g., for economic cost) the structure can be neglected. This leads us to two distinct classes of criteria:

- *Structure dependent* (e.g., execution time).
- *Structure independent* (e.g., economic cost).

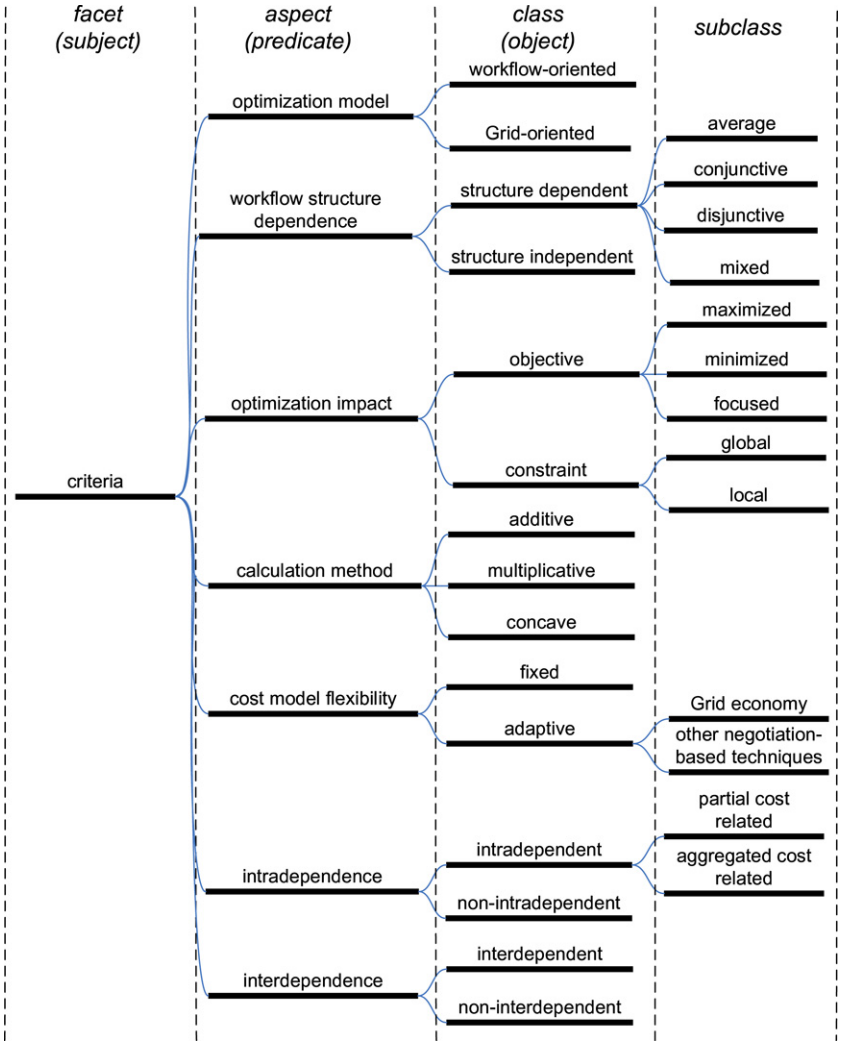

Fig. 2. Taxonomy of workflow scheduling criteria.

Most of the existing workflow scheduling approaches only optimize execution time that is a structure dependent criterion. Some multi-criteria workflow scheduling approaches (e.g., Tsiakkouri et al. [2], Yu and Buyya [3–5], and Brandic et al. [8]) also consider economic cost that is structure independent. Some other scheduling criteria can belong to either of the two classes, depending on the way the user defines them. Reliability (sometimes referred to as fault tolerance) is an example of structure independent criterion; in context of workflow scheduling, reliability was considered, for instance, by Doğan and Özgüner [6], and by Qin and Jiang [7].

Let us denote by *data quality*, any kind of qualitative description (for instance, expressed in percentage) of the results produced by alternative services (e.g., the quality will usually be higher for an expensive commercial application than for its open-source equivalent). The general aspect of data quality has been analyzed by Pipino et al. [61], where 16 different *quality dimensions* have been distinguished. Some other work focuses on more specific types of data quality. Data quality of data stream processing in tree-like workflows is the topic of the doctoral thesis presented by Schmidt [50]; the doctoral thesis presented by Fleury [62] is focused on data quality in image processing.

Within the class of structure dependent criteria, we can distinguish several sub-classes, depending on the way in which the partial costs are aggregated in the workflow. In the research focused on data quality, these different cost aggregation methods are sometimes referred to as *data quality propagation* models. The work presented by Ballou and Pazer [63] proposes a general propagation model in multi-input multi-output information systems modeled as DAGs, based on four distinguished data quality dimensions: accuracy, timeliness, completeness, and consistency. Data quality is expressed there by means of errors that are propagated and altered according to the functions assigned to individual processing activities. The functions calculate the output error of the corresponding processing activity, based on the input data and on the characteristics of the activity. Other data quality propagation schemes are proposed by some more specific data-quality-based scheduling approaches. In the simple tree-based workflow applications addressed by Schmidt [50], data quality is propagated according to the non-functional description of different types of operators. In the image processing workflows addressed by Fleury [62], data quality is propagated, based on prediction methods that use neural networks and polynomial approximations. In the current study, we apply a general cost aggregation model, based on the concept of *aggregation functions*.

Let us consider a common example of execution time calculation. In order to calculate the total execution time, we calculate the *aggregated costs* (execution times) for all tasks in a workflow, and use the maximum aggregated cost as the total cost (execution time) of the workflow. The calculation scheme for such a structure dependent criterion is depicted in Fig. 3, where the aggregated cost for the task γ is calculated based on the partial cost of the task γ and on the aggregated costs of the tasks β_i, $1 \leq i \leq n$. The aggregated costs are calculated recursively, so the same scheme would

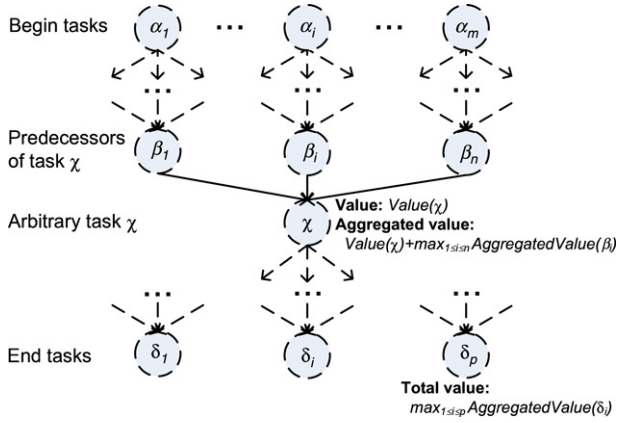

Fig. 3. Recursive calculation of aggregated costs for a structure dependent criterion.

also apply for the tasks β_i, $1 \leq i \leq n$. The *aggregated cost function* is a function that assigns to a workflow its aggregated cost.

In case of execution time, the aggregated costs of the predecessors are aggregated by taking the *maximum* cost among them. This type of aggregation function is called *disjunctive function*, as it simulates the logical OR operation and gives outputs no smaller than the largest argument. For some other criteria (e.g., for data quality), the aggregation function can calculate the mean (or weighted mean) over the arguments; such a function is called *averaging function*. When referring to any aggregation function, we will use the term *aggregation operator*. Many different aggregation operators are proposed in the literature [64,65]. For our taxonomy, we chose four aggregation operators that seem to be most relevant from the point of view of workflow scheduling:

- *Averaging*. *Averaging functions* give outputs that lie between the greatest and the smallest element of the input (e.g., mean, weighted mean).
- *Conjunctive*. *Conjunctive functions* simulate the logical AND and give outputs no greater than the smallest element of the input (e.g., minimum).
- *Disjunctive*. *Disjunctive functions* simulate the logical OR and give outputs no smaller than the largest element of input (e.g., maximum). Example: execution time.
- *Mixed*. *Mixed aggregation functions* exhibit different behavior in different regions of the workflow (e.g., maximum for the end tasks, average for the other tasks).

This classification shows some similarities to the classification of calculation methods that is introduced later in this section. However, an aggregation operator can only be defined for a structure dependent criterion, and it applies only to a part of the cost calculation procedure (i.e., to the aggregation of the predecessor costs).

6.4. Optimization impact

Scheduling criteria may have different impact on the optimization process. If the goal of the process is to find the best possible cost for a certain criterion (e.g., to minimize the total cost), then we can say that the criterion has an *optimization objective*. If the optimization process is constrained by a constant limit imposed on a certain criterion (e.g., a budget limit or a deadline), then we can say that there is an *optimization constraint* assigned to the criterion. Obviously, there may exist a constraint (or multiple constraints) defined for a certain criterion that has an optimization objective. Therefore, the optimization impact of workflow scheduling criteria can be divided into two classes:

- *Objective*. An optimization goal to find the best possible cost for the given criterion (e.g., to minimize the execution time).
- *Constraint*. A restriction imposed on the results of an optimization process (e.g., a time deadline, a budget limit).

In most of the existing workflow scheduling approaches, there is an optimization objective defined for execution time (time minimization). A common way to deal with a multi-criteria scheduling [1] is to define an optimization objective for one criterion, and to establish constraints for all the other criteria. The scheduling techniques presented by Yu and Buyya [3–5] and Tsiakkouri et al. [2] apply this approach to the problem of bi-criteria scheduling, by defining a constraint for one of the two scheduling criteria (either execution time or economic cost), and by minimizing the other one.

When considering a criterion for which an optimization objective is defined, we should also consider the optimization goal associated with the objective. For instance, when optimizing the execution time of a workflow, the goal is to *minimize* the total time. On the other hand, when optimizing data quality or when optimizing security or reliability of the execution, the goal is to *maximize* the total "cost". We can also imagine that the scheduling criterion is the ratio between the costs for two contradicting criteria (e.g., between the memory usage and the execution time). In such a case, the goal will be to obtain a total cost that is possibly close to a certain goal value (i.e., the optimization objective is *focused* on a certain goal cost). We will distinguish three different subclasses of scheduling objectives:

- *Maximized*. The optimization goal is to maximize the total cost (e.g., for quality of results).
- *Minimized*. The optimization goal is to minimize the total cost (e.g., for economic cost).
- *Focused*. The optimization goal is to achieve a certain total cost (e.g., for memory usage/execution time ratio).

Some works (e.g., Brandic et al. [8]) distinguish between *global constraints* and *local constraints*:

- *Global constraint*. A constraint defined for the whole workflow.
- *Local constraint*. A constraint defined for a single workflow task (e.g., an execution time limit for a single task imposed by the local resource manager).

6.5. Calculation method

Another classification can be done with respect to the operation used for cost calculation. For instance, addition is performed to combine the individual economic costs of tasks, when calculating the total workflow cost. The same operation is used to calculate the total execution time of a workflow, with the difference that the partial costs are added up taking into consideration also the structure of the workflow (see Fig. 3). For many other criteria (e.g., data quality, probability of failure, availability rate, security) it is convenient to express costs as real numbers from the range [0, 1]. For these criteria, we usually multiply the partial costs of the workflow tasks to calculate the total cost of the workflow. To make the picture more complete, we also mention the class of *concave* criteria proposed by Xiao and Ni [66]. The total cost of a concave criterion is equal to the minimal cost among all the individual costs (e.g., bandwidth in pipelined execution or in networks). Therefore, at least three important classes of criteria can be distinguished:

- *Additive* (e.g., economic cost, execution time).
- *Multiplicative* (e.g., data quality).
- *Concave* (e.g., bandwidth).

6.6. Cost model flexibility

A simple cost model assumes that the partial costs of services are a fixed input for scheduling and cannot be changed. This model is widely accepted in the Grid, so it is applied in most of the existing Grid workflow systems. However, there is an increasing interest in more *adaptive* flexible cost models, where the costs can be negotiated or established through some economy-based mechanisms before the application is executed. From this point of view, we distinguish the following two cost models for scheduling criteria:

- *Fixed*. The partial costs of services are given as a fixed input for scheduling.
- *Adaptive*. The partial costs of services are dynamically adjusted through certain mechanisms (e.g., auctions or negotiations).

This classification is similar to the classification based on *intradependence*, that is introduced later in this section. The difference is that for the intradependent criteria costs are calculated internally by the scheduler, using some deterministic functions, while in case of the adaptive cost models discussed here, costs are either determined externally by a Grid broker or they result from negotiations between different actors of the Grid.

Adaptive pricing has been extensively studied in the past (although usually not in context of workflow scheduling), and different models have been proposed. An important class of such model originates from human economy, so the common name to refer to them is Grid economy. Other negotiation-based techniques are common for *agent systems*. The automatic negotiation techniques introduced in such systems are developed especially for computer environments, rather then originate from human economy. Therefore, we distinguish two following subclasses of the adaptive cost model:

- *Grid economy*. Service costs are determined by applying mechanisms originating from human economy (e.g., commodities market, auctions).
- *Other negotiation-based models*. Service costs are determined as a result of negotiations between different actors in the Grid (e.g., between the resource consumer and the resource manager), based on automatic negotiation models specific for computational environments.

Many Grid economy models have been reviewed and discussed by Buyya et al. [21,67,68], and a Grid architecture realizing them has also been proposed. In the *commodities market model*, prices are established in a central way based on the current demand and supply rate, with the goal of achieving *market equilibrium*. In the *tender/contract-net model*, the consumer announces its requirements, and the service providers respond with their offers. *Auction models* support one-to-many negotiation between a service provider and many consumers. Different auction models (English auction, first-price auction, Vickrey auction, Dutch auction) are known in the literature. The other economic models mentioned by Buyya et al. [67] include the *posted price model*, the *bargaining model*, the *bid-based proportional resource sharing model*, the *community/coalition/bartering/share holders model*, and the *monopoly/oligarchy model*.

The Grid economy models are usually applied to determine the economic cost of services or resources, where the cost can either represent real money or be applied just as a useful abstraction introduced, for instance, for the sake of a fair balance between the demands of different users of the Grid. Different types of resources are treated as individual and interchangeable commodities. The scheduling approach proposed by Tian and Chandy [49] uses the commodities market model to determine the cost of resource usage in the context of non-workflow streaming applications. The approaches based on a single market and on multiple markets are compared in this work. Wolski et al. [69] compare the economic model based on the commodities market with the one based on the second-price Vickrey auctions, claiming the superiority of the former approach in terms of the economic factors such as price stability, market equilibrium, consumer efficiency, and producer efficiency. The introduced market model called "The First Bank of the G" is an extension of the Scarf's algorithm known in economy. In the Nimrod/G system [41], the commodities market model is applied to support resource allocation for scheduling of parameter sweep applications. A real workflow scheduling approach based on an economic model is introduced by Chien et al. [70], where first-price auctions are applied. Workflows are scheduled in a full-ahead manner, and scheduling is performed together with bidding for resources. The distance of an individual task from the end of the workflow (its *bottom level*) determines how *urgent* the task is; more urgent tasks are given higher prices during the auction, in order to increase the possibility of meeting the deadline defined for the workflow.

An extensive description of the problem of automatic negotiation is given by Jennings et al. [71]. According to this work, a negotiation strategy can be described by the *negotiation protocol*, *negotiation objects* (objectives for which the negotiation is performed), and the *decision making model* (the negotiation strategy). Three groups of negotiation strategies are distinguished: the *game theoretic techniques* based on the extensively studied strategies known in game theory, the *heuristics* based on more intuitive techniques which lack solid theoretical grounds, and the *argumentation-based techniques* in which the negotiating parties can exchange between each other any kind of *feedback* rather than only simple *counter-proposals*. The work presented by Li and Yahyapour [58–60] proposes non-workflow scheduling techniques using heuristic-based negotiation strategies. The heuristics are implemented through special *utility functions* that determine the behavior of the negotiating parties. For instance, some utility functions can make a negotiator "tough" (i.e., unwilling to change its initial proposals), while some other functions can make it "conceding" (i.e., apt to accept counter-proposal). The authors examine different scenarios in which *job users* and *resource providers* apply different negotiation strategies, comparing the ratio of agreements successfully created within a limited time, the achieved *utility value*, and the duration of the negotiation process.

The workflow scheduling approach for Distributed Data Mining workflows presented by Luo et al. [45] employs sealed-bid auctions to gain information about the possible costs of executing workflows in the Grid. However, this technique is applied by the scheduler to get information truthfully declared by the external schedulers (i.e., no real negotiation takes place), which means that the cost model is not really adaptive.

6.7. Intradependence

The notion of intradependence of scheduling criteria has a major impact on workflow scheduling. For some criteria, scheduling decisions made for some workflow tasks may change the costs of some other tasks. A good example of such a criterion is the economic cost in a special progressive price model. A common practice in the market is to introduce a dependence between the size of an order, and the price for an individual item (usually, the larger the order, the lower the price). If this is the case, then the scheduling decisions depend on one another within a scheduling criterion (i.e., are "intradependent"). Also, for execution time, the scheduling decisions made for some tasks may influence the aggregated costs of some other tasks, because these tasks consume resources whose amount is limited. On the other hand, the scheduling decisions made for reliability, data quality, or the

economic cost calculated in a simple price model do not seem to show any intradependence. From this point of view, we distinguish two classes of criteria:

- *Intradependent* (e.g., economic cost in a progressive price model, execution time).
- *Non-intradependent* (e.g., data quality, reliability, economic cost in a simple price model).

Workflow scheduling with respect to intradependent criteria is usually an NP-complete problem that cannot be optimally solved in an efficient way. On the other hand, a workflow schedule that is optimal with respect to a single non-intradependent criterion can usually be found in an efficient way by applying a greedy algorithm. Within the class of intradependent criteria – which is the most difficult one for scheduling – we can also distinguish two subclasses. For instance, in the aforementioned progressive price economic cost, decisions made for individual workflow tasks may influence the *partial costs* of some other tasks. For a change in case of execution time, a scheduling decision made for a workflow task does not usually change the execution times of other tasks, however it influences the way in which the *aggregated costs* are calculated. In this way, we can distinguish two types of intradependence:

- *Partial cost related*. The partial costs of workflow tasks are influenced by the scheduling decisions made for some other workflow tasks (e.g., economic cost in a progressive price model).
- *Aggregated cost related*. The aggregated costs of workflow tasks are influenced by the scheduling decisions made for some other workflow tasks (e.g., execution time).

6.8. Interdependence

When considering multiple scheduling criteria, we may observe that some of them strongly depend on others, while some others are mutually independent. For example, when optimizing the execution time of a workflow, also the availability and the reliability of services should be taken into consideration, as highly unstable resources on which a service is deployed may provide longer execution times than their more reliable counterparts. On the other hand, the economic cost of a service often does not have any influence on the execution time, so it can be considered irrelevant from the point of view of this criterion. This observation is of major importance for scheduling, since when considering a group of criteria where some criteria depend on some other ones, the multi-criteria optimization problem can often be reduced to the optimization of a single goal function. Therefore, when considering groups of criteria, we will distinguish the following two disjoint classes:

- *Interdependent* (e.g., execution time and availability).
- *Non-interdependent* (e.g., execution time and economic cost).

An interesting workflow scheduling problem that considers multiple interdependent criteria is presented by Luo et al. [45]. The goal of the first scheduling phase (called *external scheduling*), is to minimize the cost for a special criterion called Estimate Task Response Time (ETRT). This cost for a task i executed in an "*IntraGrid*" (subgrid) j is calculated according to the formula:

$$\mathrm{ETRT}_{i,j} = \left(\mathrm{RTRT}_i + \frac{\mathit{Task\_Data\_Size}_i}{\mathrm{NTR}_{k,j}}\right) \cdot \mathrm{AIC}_j$$

where RTRT_i is the requested task response time (an approximation of the execution time) of the task i, $\mathit{Task\_Data\_Size}_i$ is the data size of the task i, $\mathrm{NTR}_{k,j}$ is the network transfer rate between two IntraGrids j and k, and AIC_j is the *average IntraGrid credibility* (a measure of reliability) of the IntraGrid j. Another workflow scheduling approach based on the idea of interdependent criteria reduction was proposed in the Instant-Grid [57]. The two criteria (number of CPUs and the last known load respective the last known number of waiting or running jobs in the corresponding queues) are used to calculate a special *quality* value for each resource, based on which the scheduler selects the most appropriate mapping for each workflow task (the Grid-oriented optimization perspective applied).

6.9. Summary

We presented a taxonomy of scheduling criteria, that shows seven different aspects with respect to which the criteria may differ from one another. This taxonomy seems to be of major importance from the point of view of the multi-criteria workflow scheduling problem. The two optimization models (workflow-oriented and Grid-oriented), represent two different optimization perspectives that are usually applied in two disjoint application areas: user-oriented scheduling, and Grid-oriented resource management, respectively. Objectives and constraints are two different ways in which the user may specify the requirements, and they should be used complementarily when defining a particular scheduling problem. Structure dependence and intradependence are two aspects which make the scheduling problem really challenging, and which are major obstacles to development of generic scheduling approaches. In particular, execution time is an example of a widely used scheduling criterion, which is both structure dependent and intradependent. Both the optimization goal of an objective and the cost calculation method are two aspects where different problem classes can be treated in a similar way, after applying some mathematical transformations between criteria from different classes (for example, after transforming maximized objectives to minimized objectives, or multiplicative criteria to additive criteria, see Section 11). However, few scheduling approaches apply such transformations. Adaptive cost models gained recently a lot of interest, however, they are only scarcely considered in the context of workflow scheduling. Finally, interdependence is an interesting technique, which allows to reduce the complexity of the multi-criteria scheduling problem, and which has already been applied in several works.

7. Taxonomy of scheduling process

7.1. Introduction

The workflow scheduling process can be performed in different ways, depending on the problem definition, on the optimization principles, and on the environment in which the workflows are executed. Scheduling should also be considered as a part of the whole workflow processing. For instance, the scheduling process will be different for different numbers of criteria, and it may also take advantage of the properties of the environment such as advance reservation of resources. In this section, we will show different aspects that may have a major impact on the workflow scheduling process, and distinguish different problem classes that are relevant from this point of view (see Fig. 4).

7.2. Criteria multiplicity

Criteria multiplicity is essential from the point of view of the current work. Multiple criteria make the scheduling process much more difficult, as they represent multiple and often contradicting optimization goals that require multi-objective scheduling techniques. From this point of view, the scheduling processes can be divided into two classes:

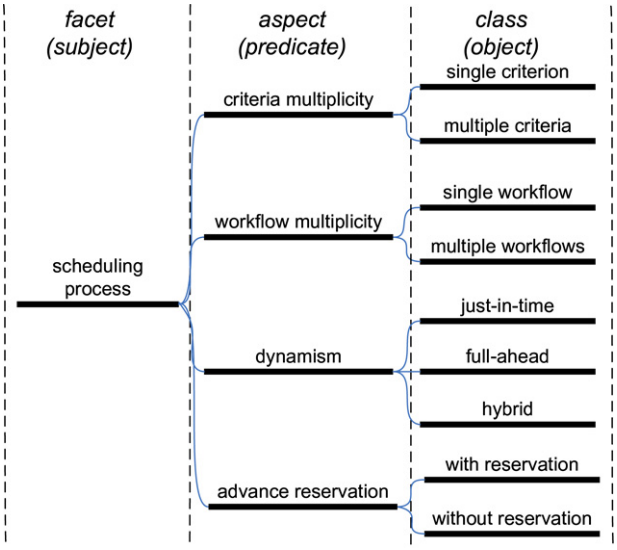

Fig. 4. Taxonomy of workflow scheduling process.

- *Single criterion*. Optimization is done for one criterion only (usually, for execution time).
- *Multiple criteria*. The scheduler tries to optimize multiple scheduling criteria.

There exist several workflow scheduling approaches that consider more than one criterion (e.g., Tsiakkouri et al. [2], Yu and Buyya [3–5], Subhlok and Vondran [43], Doğan and Özgüner [6], Qin and Jiang [7], Brandic et al. [8], and Singh et al. [72]), and many of them consider the trade-off between execution time and economic cost. The scheduling approaches presented by Doğan and Özgüner [6] and by Qin and Jiang [7] address the trade-off between execution time and reliability (data transfer time is considered in both studies). The work presented by Doğan and Özgüner [6] introduces two scheduling algorithms; one of them is called Biobjective Dynamic Level Scheduling (BDLS) and is an extension of the classical Dynamic-Level Scheduling (DLS) algorithm [73], and the second one is called Biobjective Genetic Algorithm (BGA). The second of the aforementioned works [7] is approaching the problem by task duplication, focusing mostly on the cases when some processors fail (other existing models usually assume that the processors are fault-free). A general bi-criteria scheduling algorithm called Dynamic Constraint Algorithm (DCA) [10] has been implemented in the ASKALON Grid environment. The user requirements are specified in the DCA, using a so-called *sliding constraint*, the problem is modeled as a modification of the Multiple-Choice Knapsack Problem [74], and the algorithm applied is based on the concept of dynamic programming [75]. Vienna Grid Environment [8,9] addresses multiple scheduling criteria (usually execution time and economic cost) by applying a general multi-criteria scheduling approach. An optimization technique based on integer programming [76] is used to optimize a weighted goal function combining different criteria (*Quality of Service parameters*). In K-WfGrid [14] a general multi-criteria scheduling approach is applied when workflows are initially scheduled based on user-defined criteria, by a semantic-based application builder called Automatic Application Builder (AAB) [48].

Some other criteria are the main focus for the Grid-oriented optimization (see Section 6.2) and for the pipelined workflows (see Section 5.5). In Instant Grid [57], a simple resource ranking based on the number of CPUs and the last known load is created dynamically in order to optimize the profit of the Grid. Subhlok and Vondran [43] schedule pipelined workflows with respect to the throughput and the latency of workflow execution which are both execution time-related criteria.

7.3. Workflow multiplicity

The optimization process performed by a workflow scheduler usually considers a single workflow only, but it can also attempt to optimize the execution of multiple workflows at a time, where these multiple workflows are *independent* (i.e., are not subworkflows of a single workflow). Therefore, we can distinguish the following two classes of workflow scheduling processes:

- *Single workflow*. The execution of a single workflow is optimized within a single scheduling process.
- *Multiple workflows*. The execution of multiple workflows can be optimized within a single scheduling process.

Only few existing scheduling approaches schedule more than one workflow at a time. Zhao and Sakellariou [38] distinguish three different approaches to the problem, the first one based on a sequential scheduling of multiple graphs (DAGs), the second one that incorporates also backfilling to fill gaps in the schedule, and the third one based on an initial merging of multiple DAGs into a single DAG. The article concentrates on the third approach, and distinguishes four different merging schemes. It also proposes an approach to increase fairness of scheduling, by trying to equalize the slowdown of different DAGs being scheduled (the slowdown is defined as the difference in the expected execution time for the same DAG when scheduled together with other workflows and when scheduled alone). The work presented by Luo et al. [45,46] focuses on scheduling of Distributed Data Mining (DDM) workflows with a regular structure, that are split in a preprocessing phase into a set of subworkflows. The subworkflows are scheduled independently in a highly decentralized agent-based environment, and so the scheduling problem addressed is referred to as *dynamic scheduling for competitive DAGs*, even though the ultimate scheduling objective is to minimize the execution time of a whole DDM workflow.

7.4. Dynamism

Workflow scheduling is a process of preparing workflows for an actual execution. Therefore, scheduling and execution should be considered together, and the time relation between them may differ in different approaches. Deelman et al. [77] distinguish three different types of workflow scheduling: *full-plan-ahead*, *in-time local scheduling*, and *in-time global scheduling*. The first approach is fully static, as it schedules the whole workflow before the actual execution starts. On the other extreme, the second approach can be considered as dynamic, as tasks are scheduled dynamically only when they are ready for execution. The in-time local scheduling can be considered as the approach that better suits the dynamic nature of the Grid. On the other hand, the full-plan-ahead approach enables a deeper analysis of the whole workflow structure, which may bring some benefits in terms of scheduling accuracy. The in-time global scheduling combines the two former approaches by performing full-ahead planning every time a new scheduling decision needs to be made (for instance, when some resources go down). We modified this classification by extending the category of in-time global scheduling to all *hybrid* approaches that combine the fully static and the fully dynamic approach, and by omitting the ambiguous words "local" and "global". We distinguish the following three classes of scheduling processes:

- *Just-in-time scheduling*. The scheduling decision for an individual task is postponed as long as possible, and performed before the task is ready to start (fully dynamic approach).

- *Full-ahead planning*. The whole workflow is scheduled before its execution starts (fully static approach).
- *Hybrid*. The scheduling approach combines the two aforementioned approaches.

Simple scheduling heuristics such as Min–min, Max–min, Suffrage, and XSuffrage are usually applied to make just-in-time scheduling, for example to schedule parameter sweep workflows on the Grid [42]. Typical approaches which fall into the second class are presented, for instance, by Topcuouglu et al. (HEFT) [78], Sakellariou et al. [37], Sih and Lee (DLS) [73], and Yu and Buyya [4]. Jugravu and Fahringer [79] applied Min–min and Max–min in a full-ahead manner (and also applied the HEFT algorithm) to schedule workflows in a Grid system called JavaSymphony. In Vienna Grid Environment [8], both a full-ahead scheduling approach and a just-in-time scheduling approach are applied (referred to as *static planning* and *dynamic planning*, respectively). Static planning can be applied only if the *meta data* for performance prediction is known in advance. The hybrid approach proposed in Pegasus [44] combines the just-in-time scheduling and the full-ahead planning by partitioning the workflow into subworkflows and by performing full-graph scheduling of the individual subworkflows in a just-in-time manner. Nimrod/G [41] extends static planning by adaptive resource allocation based on dynamic resource discovery, and by applying rescheduling when the scheduled tasks fail to start execution. Another hybrid approach, presented by Yu and Shi [80], achieves the same goal by triggering rescheduling when the state of the Grid changes (i.e., when some resources appear or disappear). Rescheduling of applications is the most widely used method to make full-ahead planning more dynamic. To trigger rescheduling of an application, certain acceptance criteria defined for the application execution are required, as well as a monitoring system to control the fulfillment of these criteria. An example of such acceptance criteria are the *performance contracts* proposed by Vraalsen et al. [81], that define the expectation concerning the execution time of the applications, and that are applied in the GrADS system [82,83]. The work presented by Luo et al. [45,46] considers workflow scheduling in large Grid environments (referred to as *IntraGrids*) that consist of multiple *InterGrids*; an approach called "task migration" is applied to dynamically reschedule jobs between different *IntraGrids*, if the decisions made in the initial "external scheduling" appear not to be optimal. This approach should not be confused with the notion of *task migration* introduced later in Section 9.3, as in the aforementioned approach only the jobs that have not started yet can be migrated (i.e., this approach can be described as rescheduling rather than task migration).

7.5. Advance reservation

When scheduling a workflow, we should take into consideration the environment where the workflow will be executed. Most of the Grid environments are based on local resource managements with standard queuing systems that can give only a guarantee that a task submitted to the Grid will be executed sometime (starvation-free property). Many of the systems (e.g., Pegasus [44]) process workflows by simply sending workflow tasks in an established order to local queuing systems. This simple model can be extended by applying *advance reservation*, that is a limited or restricted delegation of a particular resource capability over a certain time interval to a certain user. If an environment supports advance reservation, then the user can know in advance when a task may start, not relying on the best-effort policy of the local queuing system. Therefore, we can distinguish the following two types of scheduling:

- *With reservation*. Advance reservation is supported and considered by the scheduler.
- *Without reservation*. Advance reservation is not considered by the scheduler or not supported by the environment.

Different advance reservation models for Grid workflow scheduling are proposed by Singh et al. [72], Wieczorek et al. [55], and Zhao and Sakellariou [84]. Singh et al. [72] propose different algorithms for resource provisioning, which reserve time slots on resources based on the economic cost and the execution time criteria. The approach implemented in ASKALON [55] proposes workflow scheduling based on so-called *progressive* reservation. The introduced approach optimizes the profit both of the user (minimal execution time) and of the environment (best possible resource usage and fairness), by imposing some limitations on the amount of resources reserved for a single user at a time, and shows some advantage over the approach based on simple *attentive* reservations that do not impose any fairness policy. Zhao and Sakellariou [84] propose an advance reservation model based on the concept of *Application Spare Time*. The spare time is assigned to every workflow task based on the deadline defined by the user for the whole workflow, in order to guarantee the feasibility of the workflow execution when the actual task execution times differ to a certain extent from the predicted times. Two different approaches for spare time allocation are proposed: *recursive allocation* and *critical-path-based allocation*.

7.6. Summary

We distinguished four aspects of the scheduling process, based on the variants of the problem (single-/multiple-criteria scheduling, single-/multiple-workflow scheduling), on the scheduling principles (static/dynamic scheduling), and on the special features of the Grid environment (advance reservations). We can see that there are many approaches dedicated to the problem of multi-criteria scheduling, although there is still a lot to be done in this area. For instance, only few approaches are designed for a generic model of criteria. Only few works tackle the problem of multi-workflow scheduling; however, one may argue that decentralized scheduling approaches that deal with only one workflow at a time are more appropriate for the Grid than real multi-workflow approaches. Concerning the dynamism of scheduling process, we can see that there are both many fully dynamic and many fully static approaches, although most of the recent work tends towards more hybrid approaches. Finally, advance reservation in context of workflow scheduling is still a challenge that should be explored in the future.

8. Taxonomy of Grid resources

8.1. Introduction

Characteristics of the resources on which tasks are executed are especially important from the point of view of *performance-oriented scheduling*, in which the scheduling goal is to optimize the amount of useful work compared with the time and resources used (usually, the execution time or the job throughput optimization). The scheduler has to take into consideration the type of resources used for execution, and also the way in which the resources handle the execution of tasks. Note that when referring to task execution, we do not only mean purely computational tasks, but also tasks that perform other types of activity, such as database access or invocation of external services. The proposed taxonomy of Grid resources from the point of view of workflow scheduling is shown in Fig. 5.

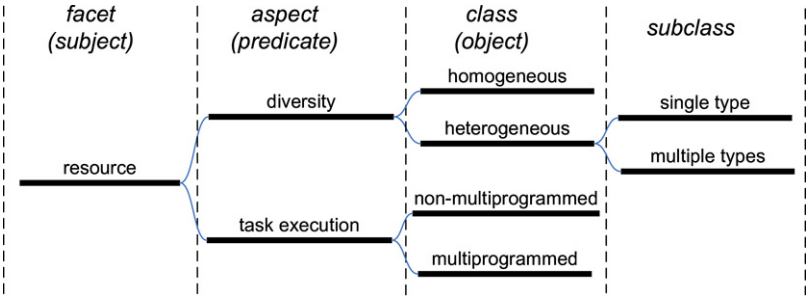

Fig. 5. Taxonomy of Grid resources.

8.2. Diversity

One of the main characteristics of the Grid resources is their *heterogeneity*. Therefore, most of the existing Grid environments belong to the second of the following two classes:

- *Homogeneous*. Multiple resources have identical static and dynamic characteristics (i.e., same type, same performance, same load, etc.).
- *Heterogeneous*. Multiple resources have diverse characteristics (i.e., different type, different performance, different load, etc.).

Heterogeneity can be understood as the existence of diverse characteristics (e.g., processor speed, memory size) within a group of resources of the same type (e.g., computational resources). At the extreme, we can take into consideration even the dynamic resource characteristics, and also call the identical resources with different CPU loads or different amounts of free memory heterogeneous. On the other hand, heterogeneity can be considered only as the distinction between different resource types (e.g., computational resources, network resources, storage resources). We will distinguish two types of heterogeneity:

- *Single type*. Resources of the same type (e.g., computational resources) differ with respect to their characteristics (e.g., CPU speed, RAM size).
- *Multiple types*. Resources differ with respect to their types (e.g., computational, storage, network).

The existing workflow scheduling approaches we are aware of are focused on the single type heterogeneity, and consider only computational resources. However, the description of computational resources includes sometimes [34] also characteristics of some other types of resources (e.g., network bandwidth, storage size).

Much effort has been put into addressing multiple types of resources on the Grid. Stork [33] aims at "making data placement a first class citizen in the Grid", by handling data transfers tasks in a similar way as computational tasks. The concept of Open Grid Service Architecture (OGSA) [85] has been introduced to describe the Grid as a service-oriented environment where heterogeneous resources are treated in a uniform way as so-called *Grid Services*. The MetaScheduling-Service (MSS) [86] developed within the VIOLA project aims at co-allocation of different types of resources (currently, computational and network resources) in multiple administrative domains.

8.3. Task execution

Computational resources can be divided into two categories, according to the way they can be used by multiple tasks:

- *Non-multiprogrammed*. The scheduler can schedule, at most, a single task at a time to a single computational resource.
- *Multiprogrammed*. The scheduler can schedule multiple tasks at a time to a single computational resource.

The computational resources from these two classes are sometimes referred to also as *disjunctive* and *cumulative*, respectively [87]. Most of the existing Grid environments consist of parallel machines being managed by local resource managers that allow only for disjunctive access to the resources (external load on the resources can always be the case). Therefore, all the Grid workflow scheduling approaches that we are aware of address the non-multiprogrammed resource model. Rodriguez Moreno et al. [88] propose a scheduler called O-OSKAR that schedules workflows of general (not necessarily computational) activities to multiprogrammed resources. The problem is approached as a Meta-CSP (Meta-Constraint Satisfaction Problem), and solved using an algorithm called ISES [89]. The project called SCOJO [90] aims at exploring the benefits coming from the execution of tasks in the multiprogrammed resource model, however workflows are not covered by this work. Instant-Grid [57] supports multiprogrammed resources for local resource managers based on *fork*, such as interactive Globus Toolkit nodes. The maximum number of tasks on one resource is determined on basis of the current CPU load.

8.4. Summary

We distinguished two aspects of Grid resources that can be meaningful from the point of view of Grid scheduling. Heterogeneity of resources is one of the most important characteristics of the Grid, and so heterogeneous resources are considered in almost all existing Grid scheduling studies. The classification based on task execution model distinguishes between two alternative execution approaches, one of which (non-multiprogrammed) is by far more widely present in the existing research. The multiprogrammed task execution model is an interesting topic for the further research.

9. Taxonomy of workflow tasks

9.1. Introduction

Workflow tasks may differ with respect to their requirements and characteristics, that have to be taken into consideration when scheduling a workflow. The classical model, in which a task has fixed resource requirements and should be scheduled, started, and executed without interruption is often extended by allowing more "elastic" scheduling and execution. For instance, scheduling of moldable MPI tasks is often considered as a separate research area that requires different approaches than in the case of standard rigid tasks. The proposed taxonomy of tasks is depicted in Fig. 6.

9.2. Resource mapping

In a similar way as a single resource can be used by multiple tasks at a time (see Section 8.3), also a single task may require multiple resources to be used (e.g., parallel MPI and OpenMP programs). Based on the classification proposed by Feitelson and Rudolph [91], we distinguish three classes of task, with respect to their resource mapping requirements:

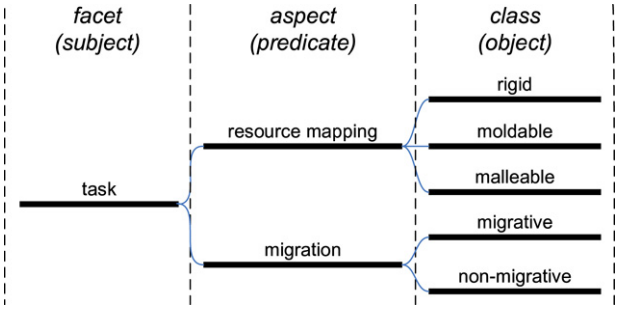

Fig. 6. Taxonomy of workflow tasks.

- *Rigid*. A task uses a fixed number of resources (usually, one resource).
- *Moldable*. A task may use a variable number of Grid resources, and this number is not known *a priori* but determined before the execution starts.
- *Malleable*. A task may use a variable number of Grid resources which may be added or withdrawn according to the task requirements and to the current system state.

Feitelson and Rudolph distinguish also a fourth class of *evolving* tasks. The difference between evolving and malleable tasks is that for the former the number of resources assigned to a task is determined by different requirements of the task at different execution stages, while for the latter, this number is determined arbitrarily by the scheduler. As we did not find this distinction either relevant or common in the existing research on workflow scheduling, we decided to merge these two classes to one class of malleable tasks.

Most of the existing workflow scheduling approaches assume that tasks belong to the first of the aforementioned classes. The other two classes are much more difficult for scheduling, as a new dimension is added to the task allocation problem. Many of the existing algorithms for moldable and malleable tasks proceed in two steps [92]: the first step aims at finding an optimal *allocation* for each task, and the second step determines the *placement* for the allocated tasks, that is the actual processor set to execute each task that minimizes the total completion time. *Mixed task and data parallel application* are considered often as cases of moldable and malleable tasks (e.g., Radulescu et al. [93], N'Takpé and Suter [92], Casanova et al. [94], Subhlok and Vondran [43], and Gounaris et al. [34]). The work by Aida and Casanova [95] addresses the problem of scheduling of such applications with advance reservation.

A typical algorithm that deals with the problem of workflow scheduling of moldable tasks in homogeneous environments is the Critical Path and Area-based algorithm (CPA) [93]. This algorithm aims at finding the best compromise between the length of the critical path and the *average area* that measures the mean processor-time area required by the application. N'Takpé and Suter [92] extend the CPA by proposing the Heterogeneous Critical Path and Area-based algorithm (HCPA) designed for heterogeneous environments. To adapt the algorithms to the heterogeneous environments, the following two modifications are introduced: (i) a novel "virtual" cluster methodology for handling platform heterogeneity is applied in the allocation step; (ii) a novel task placement step is introduced to determine whether the placement step of heuristics for homogeneous platforms is adapted to the heterogeneous case.

Another approach to the problem of scheduling of moldable tasks in workflows is proposed by Casanova et al. [94]. The authors show a way in which a typical list scheduling algorithm for heterogeneous environment can be adjusted for moldable tasks. The authors propose a new M-HEFT algorithm that extends the Heterogeneous Earliest Finish Time (HEFT) algorithm [78] with respect to the way in which the *cost values* (expected execution times) for different tasks are calculated. The cost values are used in the algorithm to determine the scheduling order, and to find the best mapping for each task. Since a single task may use different numbers of CPUs of a compound Grid site, the values are estimated for different *configurations* of different Grid sites (e.g., for different numbers of CPUs of a cluster). In the simplest version of the proposed algorithm (called M-HEFT1), the cost values are estimated for a 1-processor configuration of each site. Gounaris et al. [34] address the problem of distributed database query scheduling on the Grid. The authors enumerate three common approaches to the problem based on three different kinds of parallelism: *independent*, *pipelined*, and *partitioned* (or *intra-operator*). In context of the taxonomies proposed by us, the first type of parallelism assumes that all tasks are rigid, the second type is related to the *pipelined workflows* (see Section 5.5), and the third type, that is addressed in the proposed approach, assumes that all tasks are moldable. Distributed queries in the problem under consideration are defined as tree-like DAGs consisting of different basic tasks (*operators*), that are originally described as *single-node plans* (where "node" refers to a computational node), and that are subsequently converted to *multi-node plans* (where individual operators can be mapped to multiple computational nodes) by the proposed algorithm. The parallelization of single-node plans is done by incrementally increasing the number of computational nodes mapped to the *costliest* (i.e., most time consuming) parallelizable operators.

The problem of scheduling of malleable tasks in a parallel environment is addressed by Błażewicz et al. [96]. The authors provide a theoretical analysis of the problem of scheduling of independent tasks, and propose a scheduling algorithm that solves the problem in linear time when all the processing speed functions are convex, and in polynomial time when the speed functions are concave.

The GrADS project [83] applies a dynamic performance tuning of malleable tasks, by applying so-called *MPI Swapping*. In this approach, the resources are grouped into two sets, the *active* set and the *inactive* set, where only the first set contains resources which can be used by applications. During the execution, the resources are systematically moved between the sets, depending on the current performance measurements.

9.3. Migration

Dynamic scheduling can be implemented more effectively in environments where preemption and migration are enabled. With respect to these properties, we will distinguish two classes of tasks:

- *Migrative*. Task execution can be checkpointed at a certain resource, preempted, migrated, and resumed on another resource (assuming that the operating systems on the resources support migration).
- *Non-migrative*. Task migration is not supported.

Task migration is rarely applied in the real Grid, due to well-known problems with the implementation of reliable and effective task checkpointing. All existing implementations are restricted only to specific platforms, and impose strict prerequisites on the tasks that can be migrated [97]. Some performance study on *blocking and non-blocking* coordinated checkpointing approaches for MPI applications, also in the context of Grid systems, was presented by Buntinas et al. [98]. In this work, it was shown that the blocking checkpointing approach gives the best performance in high-speed networks, while the non-blocking approach seems

to work more efficiently in clusters of workstations and computational Grids, due to the high cost of network synchronization to produce the *checkpointing wave* of the blocking protocol in such environments. The only Grid workflow system we are aware of that supports task migration is GrADS [83]. Also the INFN/CNAF Workflow Management System [99] claims to target task migration, however, this system is still in a development stage.

9.4. Summary

We presented a taxonomy of workflow tasks, showing two aspects that are significant from the point of view of Grid scheduling: resource mapping and migration. In the classification based on resource mapping, we can see that most of the existing scheduling approaches focus on the rigid task model; two other models (in particular, malleable tasks) are considered much less frequently, although they are also very common in the Grid. Task migration still remains as a future work for Grid scheduling, mostly due to the existing implementation problems.

10. Survey and analysis of existing scheduling approaches

In Tables 1–4 we summarize different theoretical approaches and distributed environments that deal with workflow scheduling, and classify them with respect to the taxonomies introduced in Sections 5–9. For the sake of presentation clarity, we decided to present the analysis in the following way. Firstly, we divided all the related works into two categories:

(1) Theoretical approaches and special-purpose distributed systems (Table 1– 2);
(2) General-purpose Grid systems (Table 3– 4).

The reason for making such a distinction was to differentiate between specific methods that are more research-oriented, and usually focused on a certain class of problems, and the approaches that aim to provide complete and general solutions to the scientific and business community. From our own experience with the K-WfGrid [14] and ASKALON [16] projects, we think that this distinction is important, as we recognize the specific characteristics of such systems: all the scientific effort is conformed to a superior goal of having a functional system, which persuades researchers to lean towards less theoretical and more generic and pragmatic solutions. Another decision made by us, was to apply in our survey a simplified scheduling criteria taxonomy and classify the scheduling approaches with respect to the concrete scheduling criteria that are most common in the related work. Finally, in several cases we grouped multiple approaches into a single table row when the approaches were proposed by the same authors and were logically related (e.g., they were direct follow-ups, or were developed within the same project).

Tables 1 and 3 classify the characteristics of the aforementioned groups of approaches, and Tables 2 and 4 summarize the Tables 1 and 3, respectively, by showing for each pair of problem classes how many times the given combination was addressed among the classified approaches. Note that by grouping together different approaches in one row in Tables 1 and 3 we also cumulate the problem classes addressed, which is shown in Tables 2 and 4.

As we can see in these tables, almost all the studies considered by us use the task and data-oriented workflow model, which means that this model is commonly considered to be most appropriate to represent workflows. We are not aware of any theoretical approach that addresses a more complex workflow model than the general DAG. The existing systems that are based on an extended digraph model (for example ASKALON [16], and K-WfGrid [14]) usually use standard DAG scheduling algorithms, for instance by applying some kind of workflow conversion before scheduling the workflow.

We encounter only few approaches that address pipelined workflows, and none of them considers the scheduling criteria such as economic cost or reliability. This could also mean some potential for research, however, we realize that the pipelined workflow model is not very common and its usage is restricted to some specific problem domains only (e.g., multimedia or signal processing).

Execution time is the most common scheduling criterion that is considered in all the scheduling approaches under consideration. Concerning other specific criteria (economic cost, reliability, data quality), they are scarcely considered by theoretic work and specific systems. The general-purpose Grid systems are more apt to apply a generic criteria model, for instance, the bi-criteria scheduling applied in ASKALON [10], the semantic application builder of K-WfGrid [48], and the linear optimization model of VGE [8].

There are few scheduling approaches dedicated for multiple workflows, in particular, we are not aware of any such approach that addresses economic cost and data quality. However, multiple workflows can often be modeled as a single *disconnected workflow*, and the multi-workflow scheduling problem seems to be of higher importance for Grid-oriented optimization (e.g., fairness and resource usage in a multi-user environment).

We have found no approaches that consider full-ahead or hybrid scheduling for data quality, and also no purely theoretical work that considers a generic criteria model. In our opinion, the generic approaches proposed, for instance, in ASKALON [10] and VGE [8] and the existing specific bi-criteria scheduling approaches can be used as a basis for a more profound theoretical analysis of the general multi-criteria scheduling problem. The lack of just-in-time scheduling techniques that address economy cost and reliability is probably caused by the fact that in order to make optimization for these criteria, a more detailed full-graph analysis is required.

There are not many approaches that consider advance reservation, in particular for moldable or malleable tasks. We are only aware of the work of Aida and Casanova [95] that addresses the problem of scheduling of moldable mixed-parallel applications with advance reservation. Grid-oriented scheduling approaches are more common in general-purpose Grid environments; in fact, they seem to be a common requirement for any such environment. We have found no Grid-oriented approaches that consider tunable workflows. Workflow tuning seems to be a good technique to decrease Grid load, for instance, by eliminating the redundant workflow nodes, as proposed, for instance, in Pegasus [44].

There is a definite shortage of workflow scheduling approaches based on adaptive cost models (e.g., based on Grid economy techniques), which means that more research is still needed in this area. A similar situation is with the approaches based on the multiprogrammed resource model; the experience with the systems like SCOJO [90] shows some research potential related with the scheduling techniques based on such a resource model.

Finally, an unexplored area is scheduling of workflows based on the malleable task model, as well as those based on the moldable task model in combination with most of the scheduling criteria under consideration (reliability, data quality, economic cost). Task migration is barely touched in the existing workflow scheduling research, probably because of implementation and portability problems.

In our opinion, the scientific effort towards a generic multi-criteria workflow scheduling approach should focus mostly on the workflow-oriented criteria, leaving the responsibility for Grid-oriented optimization to the Grid environment. This would result in a flexible layered system architecture, that could be

Table 1
Survey of theoretical workflow scheduling approaches and specialized systems

	Workflow model								Criteria								Scheduling process					Resour.		Task		
Paper/System	task oriented	task and data transfer oriented	DAG	extended digraph	simplified DAG	tunable workflows	single input workflows	pipelined workflows	multiple criteria	execution time	economic cost	reliability	data quality	generic criteria model	Grid-oriented criteria	adaptive cost model	multiple workflows	just-in-time	full-ahead	hybrid	advance reservation	heterogeneous resources	multiprogrammed resources	moldable tasks	malleable tasks	migrative tasks
Casanova et al. [94]	+	-	+	-	-	-	+	-	-	+	-	-	-	-	-	-	-	-	+	-	-	+	-	+	-	-
Casanova et al. [42]	+	-	-	-	+	-	+	-	-	+	-	-	-	-	-	-	-	+	-	-	-	+	-	-	-	-
Chien et al. [70]	+	-	+	-	-	-	+	-	-	+	-	-	-	-	-	+	-	-	+	-	-	+	-	-	-	-
Singh et al. [72]	+	-	+	-	-	-	+	-	+	+	+	-	-	-	-	-	-	-	+	-	+	+	-	+	-	-
Doğan, Özgüner [6]	+	-	+	-	-	-	+	-	+	+	-	+	-	-	-	-	-	-	+	-	-	+	-	-	-	-
Tsiakkouri et al. [2]	+	-	+	-	-	-	+	-	+	+	+	-	-	-	-	-	-	-	+	-	-	+	-	-	-	-
Sakellariou, Zhao [37,84,38]	+	-	+	-	-	-	+	-	-	+	-	-	-	-	+	-	+	-	+	-	+	+	-	-	-	-
Gounaris [34]	-	+	-	-	+	-	+	-	-	+	-	-	-	-	-	-	-	-	+	-	-	+	-	+	-	-
Luo et al. [45,46]	+	-	-	-	+	-	+	-	+	+	-	+	-	-	-	-	-	-	+	+	-	+	-	-	-	-
Ma, Buyya [40]	+	-	-	-	+	-	+	-	-	+	-	-	-	-	-	-	-	+	+	-	-	+	-	-	-	-
Yu, Buyya, et al. [4,5,3]	+	-	+	-	-	-	+	-	+	+	+	-	-	-	-	-	-	-	+	-	+	+	-	-	-	-
Mika et al. [39]	+	-	+	-	-	-	+	-	-	+	-	-	-	-	-	-	-	-	+	-	-	+	-	-	-	-
N'Takpé, Suter [92]	+	-	+	-	-	-	+	-	-	+	-	-	-	-	-	-	-	-	+	-	-	+	-	+	-	-
Qin, Jiang [7]	+	-	+	-	-	-	+	-	+	+	-	+	-	-	-	-	-	-	+	-	-	+	-	-	-	-
Radulescu et al. [93]	+	-	+	-	-	-	+	-	-	+	-	-	-	-	-	-	-	-	+	-	-	-	-	+	-	-
Schmidt [50]	+	-	-	-	+	-	-	+	+	+	-	-	+	-	+	-	-	+	-	-	-	-	-	-	-	-
Subhlok, Vondran [43]	+	-	-	-	+	+	-	+	+	+	-	-	-	-	-	-	-	-	+	-	-	-	-	+	-	-
Yu, Shi [80]	+	-	+	-	-	-	+	-	-	+	-	-	-	-	-	-	-	-	+	+	+	+	-	-	-	-

easily extended by the user. Advance reservation, Service Level Agreements, and Grid economy techniques can be applied to implement such an environment. We believe that the taxonomy of scheduling criteria introduced by us will facilitate the development of scheduling approaches that cover a possibly broad range of workflow-oriented scheduling criteria. The case with multiple workflows does not seem to be relevant in the aforementioned Grid model, as the workflows submitted by a single user can be treated as a single disconnected workflow, and all the issues related with a fair execution of multiple workflows submitted by multiple users can be addressed by the environment, by applying some fairness policy. The existing scheduling approaches make a clear distinction between the rigid and the moldable and malleable tasks; a real challenge would be to develop a scheduling strategy that can cover in an efficient way all these classes at the same time. A generic scheduling approach should be based on a general DAG model, although there will always exist some specific application domains, where a more specific workflow model would be more appropriate (e.g., signal processing pipelined workflows). We believe that pipelined workflows form a separate class that requires different scheduling approaches than single input workflows. Finally, we think that a hybrid scheduling model is the most appropriate for multi-criteria workflow scheduling in a dynamic and heterogeneous Grid environment. Just-in-time models can be really successful when optimizing execution time, and full-ahead scheduling would work best for the other criteria; assuming the dynamic and unpredictable nature of the Grid, a solution that combines the advantages of these two models would provide best expected results.

11. Case study: Dynamic constraint algorithm

An example of a work inspired by the introduced taxonomies is the scheduling approach based on a generic bi-criteria (bi-objective) scheduling algorithm called Dynamic Constraint Algorithm (DCA) [10], developed by us in the ASKALON Grid environment. The development of this approach was preceded by a meticulous study of the taxonomy of scheduling criteria. We focused on the workflow-oriented criteria. As input, the user has to specify which criterion should be the *preliminary criterion*, and which one should be the *secondary criterion*. The goal of DCA is to find a full-graph workflow schedule with the "*best*" possible secondary criterion cost and with the primary criterion cost that is not "*worse*" than the estimated best possible cost by more than a certain *sliding constraint* which is a value chosen by the user. In the *preliminary scheduling* phase, the algorithm tries to find the schedule that is optimal for the preliminary criterion only (*preliminary schedule*). In the *secondary scheduling* phase, the algorithm is traversing the search space delimited by the sliding constraint by applying different modifications to the *intermediate schedules* that are kept in a special table (at the beginning, the table contains only the preliminary solution). The best solution found by the secondary scheduling is the *final schedule*. The computational cost of DCA is strongly dependent on careful selection of the intermediate solutions to be stored in the table, as a proper selection may considerably delimit the search space to be traversed. The optimization problem addressed in the secondary scheduling is modeled as a modification of the Multiple-Choice Knapsack Problem [74], and the algorithm applied is based on the concept of dynamic programming [75].

We figured out that in order to be able to adjust the approach for different criteria (objectives), we have to customize two operations performed by the algorithm: (1) calculation of the total workflow cost, and (2) evaluation of the cost for a criterion. The evaluation of the criterion cost means that for any criterion it should be possible to say which one of any two schedules is "better" and which one is "worse" (also, which schedule is the "best" one in a set). First, we decided to consider scheduling *objectives* with different *optimization impacts*. The information whether the given

Table 2
Problem classes combinations addressed by theoretical workflow scheduling approaches and specialized systems

		Workflow model								Criteria								Scheduling process					Resour.		Task		
		task oriented	task and data transfer oriented	DAG	extended digraph	simplified DAG	tunable workflows	single input workflows	pipelined workflows	multiple criteria	execution time	economic cost	reliability	data quality	generic criteria model	Grid-oriented criteria	adaptive cost model	multiple workflows	just-in-time	full-ahead	hybrid	advance reservation	heterogeneous resources	multiprogrammed resources	moldable tasks	malleable tasks	migrative tasks
Workflow model	task oriented		-	12	-	5	1	15	2	8	17	3	3	1	-	2	1	1	3	15	2	4	14	-	5	-	-
	task and data transfer oriented	-		-	-	1	-	1	-	-	1	-	-	-	-	-	-	-	-	1	-	-	1	-	1	-	-
	DAG	12	-		-	-	-	12	-	5	12	3	2	-	-	1	1	1	-	12	1	4	11	-	4	-	-
	extended digraph	-	-	-		-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	simplified DAG	5	1	-	-		1	4	2	3	6	-	1	1	-	1	-	-	3	4	1	-	4	-	2	-	-
	tunable workflows	1	-	-	-	1		-	1	1	1	-	-	-	-	-	-	-	-	1	-	-	-	-	1	-	-
	single input workflows	15	1	12	-	4	-		-	6	16	3	3	-	-	1	1	1	2	15	2	4	15	-	5	-	-
	pipelined workflows	2	-	-	-	2	1	-		2	2	-	-	1	-	1	-	-	1	1	-	-	-	-	1	-	-
Criteria	multiple criteria	8	-	5	-	3	1	6	2		8	3	3	1	-	1	-	-	1	7	1	2	6	-	2	-	-
	execution time	17	1	12	-	6	1	16	2	8		3	3	1	-	2	1	1	3	16	2	4	15	-	6	-	-
	economic cost	3	-	3	-	-	-	3	-	3	3		-	-	-	-	-	-	-	3	-	2	3	-	1	-	-
	reliability	3	-	2	-	1	-	3	-	3	3	-		-	-	-	-	-	-	3	1	-	3	-	-	-	-
	data quality	1	-	-	-	1	-	-	1	1	1	-	-		-	1	-	-	1	-	-	-	-	-	-	-	-
	generic criteria model	-	-	-	-	-	-	-	-	-	-	-	-	-		-	-	-	-	-	-	-	-	-	-	-	-
	Grid-oriented criteria	2	-	1	-	1	-	1	1	1	2	-	-	1	-		-	1	1	1	-	1	1	-	-	-	-
	adaptive cost model	1	-	1	-	-	-	1	-	-	1	-	-	-	-	-		-	-	1	-	-	1	-	-	-	-
Sched. proc.	multiple workflows	1	-	1	-	-	-	1	-	-	1	-	-	-	-	1	-		-	1	-	1	1	-	-	-	-
	just-in-time	3	-	-	-	3	-	2	1	1	3	-	-	1	-	1	-	-		1	-	-	2	-	-	-	-
	full-ahead	15	1	12	-	4	1	15	1	7	16	3	3	-	-	1	1	1	1		2	4	14	-	6	-	-
	hybrid	2	-	1	-	1	-	2	-	1	2	-	1	-	-	-	-	-	-	2		1	2	-	-	-	-
	advance reservation	4	-	4	-	-	-	4	-	2	4	2	-	-	-	1	-	1	-	4	1		4	-	1	-	-
Res.	heterogeneous resources	14	1	11	-	4	-	15	-	6	15	3	3	-	-	1	1	1	2	14	2	4		-	4	-	-
	multiprogrammed resources	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-		-	-	-
Task	moldable tasks	5	1	4	-	2	1	5	1	2	6	1	-	-	-	-	-	-	-	6	-	1	4	-		-	-
	malleable tasks	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-		-
	migrative tasks	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	

Table 3
Survey of general-purpose Grid workflow systems

Paper/System	Workflow model								Criteria								Scheduling process					Resour.		Task		
	task oriented	task and data transfer oriented	DAG	extended digraph	simplified DAG	tunable workflows	single input workflows	pipelined workflows	multiple criteria	execution time	economic cost	reliability	data quality	generic criteria model	Grid-oriented criteria	adaptive cost model	multiple workflows	just-in-time	full-ahead	hybrid	advance reservation	heterogeneous resources	multiprogrammed resources	moldable tasks	malleable tasks	migrative tasks
Nimrod/G [41]	+	-	-	-	+	-	+	-	+	+	+	-	-	-	-	+	-	-	-	+	-	+	-	-	-	-
GrADS [83,82]	+	-	+	-	-	-	+	-	-	+	-	-	-	-	-	-	-	-	+	+	-	+	-	+	+	+
VGE [8,9]	+	-	+	-	-	+	+	-	+	+	-	-	-	+	-	-	-	+	+	-	+	+	-	+	-	-
PEGASUS [77,44]	+	-	+	-	-	+	+	-	-	+	-	-	-	-	-	-	+	-	+	+	-	+	-	+	-	-
ASKALON [16,35,55,15]	+	-	-	+	-	-	+	-	+	+	-	-	-	+	+	-	+	-	+	-	+	+	-	-	-	-
K-WfGrid [13,14,48]	+	-	-	+	-	+	+	-	+	+	-	-	-	+	-	-	-	-	+	-	-	+	-	-	-	-
Instant-Grid [57]	+	-	-	+	-	-	+	-	+	-	-	-	-	-	+	-	+	+	-	-	-	+	+	-	-	-
P-GRADE [36]	+	-	+	-	+	-	+	+	+	+	-	-	-	-	+	-	+	+	-	-	-	+	-	-	-	-
UNICORE6/Chemomentum [56]	+	-	+	-	-	-	+	-	+	+	-	-	-	-	+	-	-	+	-	-	-	+	-	-	-	-
Knowledge Grid [100]	+	-	+	-	-	-	+	-	+	+	-	-	-	+	-	-	-	-	+	-	-	+	-	-	-	-

objective is *minimized* or *maximized* is a part of the definition of any objective, and is taken into consideration whenever two different schedules are compared with respect to this particular objective. In a similar way, the *focused* objectives can also be supported. Second, we decided to consider two *calculation methods*: *additive* and *multiplicative*. We implemented the cost calculation and schedule evaluation techniques customized for the additive criteria; if the given criterion is multiplicative, then we first transform the cost c of each service to its natural logarithm ($c \rightarrow \ln(c)$). After this transformation, the criterion can be treated as an additive criterion, which is a consequence of the well-known mathematical formula:

$$\forall c_1, \ldots, c_n \in \mathbb{R}^+ : c_1 \cdot \cdots \cdot c_n = \exp(\ln(c_1) + \cdots + \ln(c_n)).$$

Table 4
Problem classes combinations addressed by general-purpose Grid workflow systems

		Workflow model								Criteria								Scheduling process					Resour.		Task		
		task oriented	task and data transfer oriented	DAG	extended digraph	simplified DAG	tunable workflows	single input workflows	pipelined workflows	multiple criteria	execution time	economic cost	reliability	data quality	generic criteria model	Grid-oriented criteria	adaptive cost model	multiple workflows	just-in-time	full-ahead	hybrid	advance reservation	heterogeneous resources	multiprogrammed resources	moldable tasks	malleable tasks	migrative tasks
Workflow model	task oriented		-	6	3	2	3	10	1	8	9	1	-	-	4	4	1	4	4	6	3	2	10	1	3	1	1
	task and data transfer oriented	-		-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	DAG	6	-		-	1	2	6	1	4	6	-	-	-	2	2	-	2	3	4	2	1	6	-	3	1	1
	extended digraph	3	-	-		-	1	3	-	3	2	-	-	-	2	2	-	2	1	2	-	1	3	1	-	-	-
	simplified DAG	2	-	1	-		-	2	1	2	2	1	-	-	-	1	1	1	1	-	1	-	2	-	-	-	-
	tunable workflows	3	-	2	1	-		3	-	2	3	-	-	-	2	-	-	1	1	3	1	1	3	-	2	-	-
	single input workflows	10	-	6	3	2	3		1	8	9	1	-	-	4	4	1	4	4	6	3	2	10	1	3	1	1
	pipelined workflows	1	-	1	-	1	-	1		1	1	-	-	-	-	1	-	1	1	-	-	-	1	-	-	-	-
Criteria	multiple criteria	8	-	4	3	2	2	8	1		7	1	-	-	4	4	1	3	4	4	1	2	8	1	1	-	-
	execution time	9	-	6	2	2	3	9	1	7		1	-	-	4	3	1	3	3	6	3	2	9	-	3	1	1
	economic cost	1	-	-	-	1	-	1	-	1	1		-	-	-	-	1	-	-	-	1	-	1	-	-	-	-
	reliability	-	-	-	-	-	-	-	-	-	-	-		-	-	-	-	-	-	-	-	-	-	-	-	-	-
	data quality	-	-	-	-	-	-	-	-	-	-	-	-		-	-	-	-	-	-	-	-	-	-	-	-	-
	generic criteria model	4	-	2	2	-	2	4	-	4	4	-	-	-		1	-	1	1	4	-	2	4	-	1	-	-
	Grid-oriented criteria	4	-	2	2	1	-	4	1	4	3	-	-	-	1		-	3	3	1	-	1	4	1	-	-	-
	adaptive cost model	1	-	-	-	1	-	1	-	1	1	1	-	-	-	-		-	-	-	1	-	1	-	-	-	-
Sched. proc.	multiple workflows	4	-	2	2	1	1	4	1	3	3	-	-	-	1	3	-		2	2	1	1	4	1	1	-	-
	just-in-time	4	-	3	1	1	1	4	1	4	3	-	-	-	1	3	-	2		1	-	1	4	1	1	-	-
	full-ahead	6	-	4	2	-	3	6	-	4	6	-	-	-	4	1	-	2	1		2	2	6	-	3	1	1
	hybrid	3	-	2	-	1	1	3	-	1	3	1	-	-	-	-	1	1	-	2		-	3	-	2	1	1
	advance reservation	2	-	1	1	-	1	2	-	2	2	-	-	-	2	1	-	1	1	2	-		2	-	1	-	-
Res.	heterogeneous resources	10	-	6	3	2	3	10	1	8	9	1	-	-	4	4	1	4	4	6	3	2		1	3	1	1
	multiprogrammed resources	1	-	-	1	-	-	1	-	1	-	-	-	-	-	1	-	1	1	-	-	-	1		-	-	-
Task	moldable tasks	3	-	3	-	-	2	3	-	1	3	-	-	-	1	-	-	1	1	3	2	1	3	-		1	1
	malleable tasks	1	-	1	-	-	-	1	-	-	1	-	-	-	-	-	-	-	-	1	1	-	1	-	1		1
	migrative tasks	1	-	1	-	-	-	1	-	-	1	-	-	-	-	-	-	-	-	1	1	-	1	-	1	1	

Third, we decided to consider both *structure dependent* and *structure independent* criteria. When calculating the total cost for a structure dependent criterion, we have to take into consideration the workflow structure and also the *aggregation operator* characterizing the given criterion. The evaluation of any structure dependent criterion is particularly difficult to perform, as two intermediate schedules may often have the same total cost with respect to this criterion, while one of them may in the optimization process lead to potentially better schedules (it means, one of the solutions *dominates* the other one). By storing too many solutions in the table that are dominated, we may risk an excessively long scheduling time. We developed a special schedule comparison method based on the concept of *scheduling wavefront* [10], which is able to filter out dominated solutions unless the criterion is non-intradependent. Finally, we considered also a special case of *intradependent* criterion which is execution time. Intradependence is the factor that may have a major impact on the scheduling model, as the number of possible intradependence schemes is potentially unlimited. For example, the resource provider may introduce any arbitrary pricing model for resources, based on some quantitative discounts. For execution time, we customized the total cost calculation method, by taking into consideration the limited number of resources, but we did not change the criterion evaluation method. We proved by experiment, that our approach is able to produce good results also in case of the execution time-based scheduling. Concerning the other classes of criteria, we decided to limit our approach to *workflow oriented* criteria with a *fixed* cost model, and we did not consider *interdependence* of criteria.

In the experimental study [10], we showed that the proposed scheduling algorithm provides better results than two existing bi-criteria scheduling algorithms, and that the time spent on scheduling is reasonable for medium-scale workflows. By applying a generic criteria model, we were able to apply our approach for different scheduling criteria; in our experiments, we considered four different criteria: execution time, execution cost, reliability, and data quality. We believe that based on the taxonomies introduced in the current work, it will be possible to develop other generic scheduling algorithms, and to adjust the existing algorithms to different classes of the scheduling problem.

12. Conclusions

In the current work, we analyzed the problem of multi-criteria Grid workflow scheduling. Based on a careful selection of the relevant work in this area and on our own experience, we identified the main aspects of the problem that have to be taken into consideration when scheduling workflows on the Grid. As the main contribution, we introduced five taxonomies of the addressed problem, with the goal of creating a solid basis for more generic scheduling approaches that can be developed in the future. We made a survey of the related works, classified them with respect to the introduced taxonomies, and identified the problem classes that have been most commonly addressed and those that present the most promising research potential. We provided some guidelines for development of a generic multi-criteria scheduling approach, pointing at the problem classes that such an approach should address. As a case study, we presented a bi-criteria workflow

scheduling approach developed by us based on the taxonomy of scheduling criteria.

Our choice to distinguish between different problem classes, and to divide the related work into two groups were not always straightforward and can be subject to some further analysis and discussion. Some of the classes introduced by us seem to be scarcely addressed in the related work (e.g., task-and-data-transfer-oriented workflows), which may indicate their low relevance, but may also mean that they have not been sufficiently studied in the past. On the other hand, the multitude of the existing workflow models may imply the need to introduce some other specific classes into our taxonomy. We leave all these questions for a further analysis that may improve the proposed taxonomies. We believe that the presented work can be an inspiration to develop some generic scheduling techniques, as well as some more specialized techniques.

Acknowledgments

This work is supported in part by the European Union through the IST-2002-004265 Network of Excellence CoreGRID.

References

[1] V. T'kindt, J. Billaut, Multicriteria Scheduling, Springer Verlag, Berlin, 2002.
[2] E. Tsiakkouri, R. Sakellariou, H. Zhao, M.D. Dikaiakos, Scheduling workflows with budget constraints, in: S. Gorlatch, M. Danelutto (Eds.), Proceedings of the CoreGRID Workshop "Integrated research in Grid Computing", 2005, pp. 347–357.
[3] J. Yu, R. Buyya, C.K. Tham, QoS-based scheduling of workflow applications on service Grids, in: Proceedings of the 1st IEEE International Conference on e-Science and Grid Computing, e-Science 2005, IEEE, IEEE CS Press, Melbourne, Australia, 2005.
[4] J. Yu, R. Buyya, A budget constrained scheduling of workflow applications on utility Grids using genetic algorithms, in: Proceedings of the 15th IEEE International Symposium on High Performance Distributed Computing, HPDC 2006, IEEE, IEEE CS Press, Paris, France, 2006.
[5] J. Yu, R. Buyya, Scheduling scientific workflow applications with deadline and budget constraints using genetic algorithms, in: Scientific Programming, IOS Press, 2006, pp. 14–217.
[6] A. Doğan, F. Özgüner, Biobjective scheduling algorithms for execution time-reliability trade-off in heterogeneous computing systems, Comput. J. 48 (3) (2005) 300–314.
[7] X. Qin, H. Jiang, A novel fault-tolerant scheduling algorithm for precedence constrained tasks in real-time heterogeneous systems, Parallel Comput. 32 (5) (2006) 331–356.
[8] I. Brandic, S. Benkner, G. Engelbrecht, R. Schmidt, QoS support for time-critical Grid workflow applications, in: E-SCIENCE'05: Proceedings of the First International Conference on e-Science and Grid Computing, IEEE Computer Society, Washington, DC, USA, 2005, pp. 108–115.
[9] I. Brandic, S. Pllana, S. Benkner, Amadeus: A holistic service-oriented environment for Grid workflows, in: GCCW'06: Proceedings of the Fifth International Conference on Grid and Cooperative Computing Workshops, IEEE Computer Society, Washington, DC, USA, 2006, pp. 259–266.
[10] M. Wieczorek, S. Podlipnig, R. Prodan, T. Fahringer, Bi-criteria scheduling of scientific workflows for the Grid, in: Proc. of the Eighth IEEE International Symposium on Cluster Computing and the Grid, CCGrid 2008, IEEE Computer Society, Lyon, France, 2008.
[11] F. Neubauer, A. Hoheisel, J. Geiler, Workflow-based Grid applications, Future Generation Comput. Syst. 22 (1–2) (2006) 6–15.
[12] Andreas Hoheisel, Martin Alt, Petri Nets, in: Ian J. Taylor, Dennis B. Gannon, Ewa Deelman, Matthew Shields (Eds.), Workflows for e-Science – Scientific Workflows for Grids, Springer, ISBN: 978-1-84628-519-6, 2006.
[13] A. Hoheisel, User tools and languages for graph-based Grid workflows: Research articles, Concurr. Comput. Pract. Exper. 18 (10) (2006) 1101–1113.
[14] The K-WfGrid project. http://www.kwfgrid.net.
[15] T. Fahringer, J. Qin, S. Hainzer, Specification of Grid workflow applications with AGWL: An Abstract Grid Workflow Language, in: Proceedings of IEEE International Symposium on Cluster Computing and the Grid 2005, CCGrid 2005, IEEE Computer Society Press, Cardiff, UK, 2005.
[16] T. Fahringer, R. Prodan, R. Duan, J. Hofer, F. Nadeem, F. Nerieri, S. Podlipnig, J. Qin, M. Siddiqui, H.-L. Truong, A. Villazón, M. Wieczorek, ASKALON: A development and grid computing environment for scientific workflows, in: Workflows for e-Science, Springer-Verlag, 2007, pp. 450–471.
[17] T. Andrews, et al., Business process execution language for web services, Tech. Rep., BEA Systems, et al., May 2003.
[18] K. Krauter, R. Buyya, M. Maheswaran, A taxonomy and survey of Grid resource management systems for distributed computing, Softw. Pract. Exper. 32 (2) (2002) 135–164.
[19] S. Venugopal, R. Buyya, K. Ramamohanarao, A Taxonomy of data Grids for distributed data sharing, management, and processing, ACM Comput. Surv. 38 (1) (2006) 1–53.
[20] S. Zanikolas, R. Sakellariou, A taxonomy of Grid monitoring systems, Future Generation Comput. Syst. 21 (1) (2005) 163–188.
[21] C.S. Yeo, R. Buyya, A taxonomy of market-based resource management systems for utility-driven cluster computing, Softw. Pract. Exper. 36 (13) (2006) 1381–1419.
[22] K. Plankensteiner, R. Prodan, T. Fahringer, A. Kertész, P. Kacsuk, Fault tolerant behaviour in state-of-the-art workflow management systems, in: S. Gorlatch, P. Fragopoulou, T. Priol (Eds.), CoreGRID Integration Workshop, Crete University Press, Hersonissos, Greece, 2008, pp. 455–466.
[23] H.L. Truong, S. Dustdar, T. Fahringer, Performance metrics and ontologies for Grid workflows, Future Generation Comput. Syst. 23 (6) (2007) 760–772. http://dblp.uni-trier.de/db/journals/fgcs/fgcs23.html#TruongDF07.
[24] W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, A.P. Barros, Workflow patterns, Distrib. Parallel Databases 14 (1) (2003) 5–51.
[25] W.M. van der Aalst, Workflow patterns. http://www.workflowpatterns.com/.
[26] J. Yu, R. Buyya, A taxonomy of workflow management systems for Grid computing, J. Grid Comput. 3 (3–4) (2005) 171–200.
[27] T.L. Casavant, J.G. Kuhl, A taxonomy of scheduling in general-purpose distributed computing systems, IEEE Trans. Softw. Eng. 14 (2) (1988) 141–154.
[28] F. Dong, S.G. Akl, Scheduling algorithms for Grid computing: State of the art and open problems, Tech. Rep. 2006-504, School of Computing, Queen's University, Kingston, Ontario, January 2006.
[29] Resource Description Framework. http://www.w3.org/RDF/.
[30] Jablonski, Workflow management: Modeling concepts, architecture and implementation, 1996.
[31] B. Kiepuszewski, A.H.M. ter Hofstede, W.M.P. van der Aalst, Fundamentals of control flow in workflows, Acta Inform. 39 (3) (2003) 143–209. http://dblp.uni-trier.de/db/journals/acta/acta39.html#KiepuszewskiHA03.
[32] Java CoG Kit Karajan Guide. http://www.cogkit.org/current/manual/workflow.pdf.
[33] T. Kosar, M. Livny, Stork: Making data placement a first class citizen in the Grid, 2004.
[34] A. Gounaris, R. Sakellariou, N. Paton, A. Fernandes, A novel approach to resource scheduling for parallel query processing on computational Grids, Distrib. Parallel Databases 19 (May) (2006) 87–106 (20).
[35] M. Mair, J. Qin, M. Wieczorek, T. Fahringer, Workflow conversion and processing in the ASKALON Grid environment, in: 2nd Austrian Grid Symposium.
[36] A. Kertesz, G. Sipos, P. Kacsuk, Multi-Grid brokering with the P-GRADE portal, in: J. Volkert, T. Fahringer, D. Kranzlmuller, W. Schreiner (Eds.), 2nd Austrian Grid Symposium, OCG Verlag, Austria, 2006, pp. 166–178.
[37] R. Sakellariou, H. Zhao, A hybrid heuristic for DAG scheduling on heterogeneous systems, in: IPDPS, 2004.
[38] H. Zhao, R. Sakellariou, Scheduling multiple DAGs onto heterogeneous systems, in: 15th Heterogeneous Computing Workshop, HCW'06, IEEE Computer Society Press, Rhodes, Greece, 2006.
[39] M. Mika, G. Waligóra, J. Weglarz, A metaheuristic approach to scheduling workflow jobs on a Grid, Grid Res. Manag. State Art Future Trends (2004) 295–318.
[40] T. Ma, R. Buyya, Critical-path and priority based algorithms for scheduling workflows with parameter sweep tasks on global grids, in: Proceedings of the 17th International Symposium on Computer Architecture and High Performance Computing, SBAC-PAD 2005, IEEE Computer Society Press, Rio de Janeiro, Brazil, 2005.
[41] D. Abramson, R. Buyya, J. Giddy, A computational economy for Grid computing and its implementation in the nimrod-g resource broker, Future Generation Comput. Syst. 18 (8) (2002) 1061–1074.
[42] H. Casanova, A. Legrand, D. Zagorodnov, F. Berman, Heuristics for scheduling parameter sweep applications in Grid environments, in: Proceedings of 9th Heterogeneous Computing Workshop, HCW, Cancun, Mexico, 2000, pp. 349–363.
[43] J. Subhlok, G. Vondran, Optimal use of mixed task and data parallelism for pipelined computations, J. Parallel Distrib. Comput. 60 (March) (2000) 297–319 (23).
[44] E. Deelman, G. Singh, M.-H. Su, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, K. Vahi, G.B. Berriman, J. Good, A. Laity, J.C. Jacob, D. Katz, Pegasus: a Framework for mapping complex scientific workflows onto distributed systems, Sci. Programm. J. 13 (2) (2005) 219–237.
[45] P. Luo, K. Lü, Z. Shi, Q. He, Distributed data mining in Grid computing environments, Future Generation Comput. Syst. 23 (1) (2007) 84–91.
[46] P. Luo, K. Lü, R. Huang, Q. He, Z. Shi, A heterogeneous computing system for data mining workflows in multi-agent environments, Expert Syst. 23 (November) (2006) 258–272 (15).
[47] T. Gubała, D. Harężlak, M. Bubak, M. Malawski, Constructing abstract workflows of applications with workflow composition tool, in: M. Bubak, S. Unger (Eds.), Proceedings of Cracow'06 Grid Workshop, October 15–18, 2006, Krakow, Poland, in: The Knowledge-based Workflow System for Grid Applications, ACC Cyfronet AGH, 2007, pp. 25–30.
[48] L. Dutka, J. Kitowski, S. Natanek, Automatic application builder — Tool for automatic service selection, in: M. Bubak, S. Unger (Eds.), Proceedings of Cracow'06 Grid Workshop, October 15–18, 2006, Krakow, Poland, in: The Knowledge-based Workflow System for Grid Applications, ACC Cyfronet AGH, 2007, pp. 31–38.

[49] L. Tian, K.M. Chandy, Resource allocation in streaming environments, in: Proceedings of 7th IEEE/ACM International Conference on Grid Computing, Grid'06, IEEE Computer Society Press, Barcelona, Spain, 2006.
[50] S. Schmidt, Quality-of-service-aware data stream processing, Ph.D. Thesis, Dresden, 2007.
[51] Altair Engineering, Inc., PBS Professional. http://www.altair.com/software/pbspro.htm.
[52] Sun Microsystems, Inc., Grid Engine. http://gridengine.sunsource.net/.
[53] Platform Computing Inc., Platform LSF. http://www.platform.com/Products/Platform.LSF.Family/.
[54] Cluster Resources, Inc., Maui Cluster Scheduler.
[55] M. Wieczorek, M. Siddiqui, A. Villazon, R. Prodan, T. Fahringer, Applying advance reservation to increase predictability of workflow execution on the Grid, in: E-SCIENCE'06: Proceedings of the Second IEEE International Conference on e-Science and Grid Computing, IEEE Computer Society, Washington, DC, USA, 2006, p. 82.
[56] B. Schuller, B. Demuth, H. Mix, K. Rasch, M. Romberg, S. Sild, U. Maran, P. Bala, E. del Grosso, M. Casalegno, N. Piclin, M. Pintore, W. Sudholt, K. Baldrige, Chemomentum — UNICORE 6 based infrastructure for complex applications in science and technology, in: UNICORE Summit 2007, Springer-Verlag, Rennes, France, 2007.
[57] A. Hoheisel, H. Rose, Konzept für das Scheduling von Workflow-Aktivitäten in Instant Grid, Tech. Rep., Fraunhofer Institut für Rechnerarchitektur und Softwaretechnik, June 2006.
[58] J. Li, R. Yahyapour, A negotiation model supporting co-allocation for Grid scheduling, in: Proceedings of 7th IEEE/ACM International Conference on Grid Computing, Grid'06, IEEE Computer Society Press, Barcelona, Spain, 2006.
[59] J. Li, R. Yahyapour, Negotiation strategies for Grid scheduling, in: The First International Conference on Grid and Pervasive Computing, GPC2006, in: Lecture Notes in Computer Science, vol. 3947, Springer-Verlag, Tunghai University, Taiwan, 2006, pp. 42–52.
[60] J. Li, R. Yahyapour, Learning-based negotiation strategies for Grid scheduling, in: IEEE Int'l Symposium on Cluster Computing and the Grid, CCGrid 2006, IEEE Press, Singapore, 2006, pp. 567–583.
[61] L. Pipino, Y.W. Lee, R.Y. Wang, Data quality assessment, Communications of the ACM 45 (4) (2002) 211–218.
[62] P. Fleury, Dynamic scheme selection in image coding, Ph.D. Thesis, Lausanne, 1999.
[63] D.P. Ballou, H.L. Pazer, Modeling data and process quality in multi-input, multi-output information systems, Manag. Sci. 31 (2) (1985) 150–162.
[64] A. Čaplinskas, J. Gasperovič, Techniques to aggregate the characteristics of internal quality of an IS specification language, Informatica, Lith. Acad. Sci. 16 (4) (2005) 519–540. http://www.vtex.lt/Informatica/htm/INFO614.htm.
[65] C.F. Mela, D.R. Lehmann, Using fuzzy set theoretic techniques to identify preference rules from interactions in the linear model: An empirical study, Fuzzy Sets Syst. 71 (2) (1995) 165–181.
[66] X. Xiao, L.M. Ni, Internet QoS: A big picture, IEEE Netw. 13 (2) (1999) 8–18.
[67] R. Buyya, H. Stockinger, J. Giddy, D. Abramson, Economic models for management of resources in Peer-to-Peer and Grid computing, Tech. Rep. 0108001, Economics Working Paper Archive at WUSTL. Available at http://ideas.repec.org/p/wpa/wuwpco/0108001.html (2001).
[68] R. Buyya, D. Abramson, S. Venugopal, The Grid economy, in: M. Parashar, C. Lee (Eds.), Proceedings of the IEEE, in: Grid Computing, vol. 93, IEEE Press, NJ, USA, 2005, pp. 698–714 (special issue).
[69] R. Wolski, J.S. Plank, J. Brevik, T. Bryan, Analyzing Market-Based Resource Allocation Strategies for the Computational Grid 15 (3) (2001) 258–281.
[70] C.-H. Chien, P.H.-M. Chang, V.-W. Soo, Market-oriented multiple resource scheduling in Grid computing environments, in: AINA'05: Proceedings of the 19th International Conference on Advanced Information Networking and Applications, IEEE Computer Society, Washington, DC, USA, 2005, pp. 867–872.
[71] N.R. Jennings, P. Faratin, A.R. Lomuscio, S. Parsons, C. Sierra, M. Woodlidge, Automated negotiation: Prospects, methods and challenges, International Journal of Group Decision and Negotiation 10 (2) (2001) 199–215.
[72] G. Singh, C. Kesselman, E. Deelman, Application-level resource provisioning on the Grid, e-science 0 (2006) 83.
[73] G.C. Sih, E.A. Lee, A compile-time scheduling heuristic for interconnection-constrained heterogeneous processor architectures, IEEE Trans. Parallel Distrib. Syst. 4 (2) (1993) 175–187.
[74] H. Kellerer, U. Pferschy, D. Pisinger, Knapsack Problems, Springer-Verlag, Berlin, 2004.
[75] D.P. Bertsekas, Dynamic Programming and Optimal Control, Vol. I, Athena Scientific, Belmont, MA, 2005.
[76] A. Schrijver, Theory of Linear and Integer Programming, Wiley, 1987, schRI a 87:1 1.Ex.
[77] E. Deelman, J. Blythe, Y. Gil, C. Kesselman, Workflow management in GriPhyN, Grid Res. Manag. State Art Future Trends (2004) 99–116.
[78] H. Topcuouglu, S. Hariri, M. you Wu, Performance-effective and low-complexity task scheduling for heterogeneous computing, IEEE Trans. Parallel Distrib. Syst. 13 (3) (2002) 260–274.
[79] A. Jugravu, T. Fahringer, JavaSymphony, a programming model for the Grid, Future Gener. Comput. Syst. 21 (1) (2005) 239–246.
[80] Z. Yu, W. Shi, An adaptive rescheduling strategy for Grid workflow applications, in: Proceedings of the 21st IPDPS 2007, IEEE Computer Society Press, Long Beach, USA, 2007.
[81] F. Vraalsen, R.A. Aydt, C.L. Mendes, D.A. Reed, Performance Contracts: Predicting and Monitoring Grid Application Behavior, in: Lecture Notes in Computer Science, vol. 2242, 2001, pp. 154–166.
[82] H. Dail, O. Sievert, F. Berman, H. Casanova, A. YarKhan, S. Vadhiyar, J. Dongarra, C. Liu, L. Yang, D. Angulo, I. Foster, Scheduling in the Grid Application Development Software Project, 2003.
[83] F. Berman, et al., New Grid scheduling and rescheduling methods in the GrADS project, Internat. J. Parallel Programm. 33 (June) (2005) 209–229 (21).
[84] H. Zhao, R. Sakellariou, Advance reservation policies for workflows, in: Proceedings of the 12th International Workshop on Job Scheduling Strategies for Parallel Processing, in: Lecture Notes in Computer Science, vol. 4376, Springer-Verlag, Saint-Malo, France, 2006, pp. 47–67.
[85] I. Foster, C. Kesselman, S. Tuecke, The anatomy of the Grid: Enabling scalable virtual organizations, Internat. J. Supercomput. Appl. 15 (3) (2001) 2001.
[86] O. Wäldrich, W. Ziegler, P. Wieder, A meta-scheduling service for co-allocating arbitrary types of resources, Tech. Rep. TR-0010, Institute on Resource Management and Scheduling, CoreGRID - Network of Excellence, December 2005.
[87] P. Baptiste, C. Le Pape, W. Nuijten, Constraint-based Scheduling: Applying Constraint Programming to Scheduling Problems, in: International Series in Operations Research and Management Science, vol. 39, Kluwer Academic Publishers, Norwell, MA, 2001.
[88] M.D. Rodriguez Moreno, D. Borrajo Millán, D. Meziat Luna, Representing and planning tasks with time and resources, Ph.D. Thesis, Universidad de Alcalá, Spain, December 2003.
[89] A.O.A. Cesta, S. Smith, A constrained-based method for project scheduling with time windows, J. Heuristics 8 (2002) 109–136.
[90] A.C. Sodan, X. Huang, Adaptive time/space sharing with SCOJO, Internat. J. High Performance Comput. Netw. 4 (5–6) (2006) 256–269.
[91] D.G. Feitelson, L. Rudolph, Towards convergence in job schedulers for parallel supercomputers, in: IPPS'96: Proceedings of the Workshop on Job Scheduling Strategies for Parallel Processing, Springer-Verlag, London, UK, 1996, pp. 1–26.
[92] T. N'Takpé, F. Suter, Critical path and area based scheduling of parallel task graphs on heterogeneous platforms, in: Proceedings of the 12th International Conference on Parallel and Distributed Systems, ICPADS 2006, IEEE Computer Society Press, Minneapolis, MN, USA, 2006.
[93] A. Radulescu, C. Nicolescu, A.J.C. van Gemund, P.P. Jonker, CPR: Mixed task and data parallel scheduling for distributed systems, in: Proceedings of the 15th International Parallel and Distributed Processing Symposium, IPDPS, IEEE Computer Society Press, San Francisco, USA, 2001, pp. 39–41.
[94] H. Casanova, F. Desprez, F. Suter, From heterogeneous task scheduling to heterogeneous mixed parallel scheduling, in: M. Danelutto, D. Laforenza, M. Vanneschi (Eds.), Proceedings of the 10th International Euro-Par Conference, Euro-Par'04, in: Lecture Notes in Computer Science, vol. 3149, Springer, Pisa, Italy, 2004, pp. 230–237.
[95] K. Aida, H. Casanova, Scheduling mixed-parallel applications with advance reservations, in: Proceedings of the 17th IEEE International Symposium on High Performance Distributed Computing, HPDC-15, IEEE Press, Boston, MA, 2008.
[96] J. Błazewicz, M. Machowiak, J. Węglarz, M. Kovalyov, D. Trystram, Scheduling malleable tasks on parallel processors to minimize the makespan: Models and algorithms for planning and scheduling problems, Ann. Oper. Res. 129 (July) (2004) 65–80 (16).
[97] J. Basney, M. Litzkow, T. Tannenbaum, M. Livny, Checkpoint and migration of unix processes in the condor distributed processing system, Tech. Rep. Technical Report 1346, April 1997.
[98] D. Buntinas, C. Coti, T. Hérault, P. Lemarinier, L. Pilard, A. Rezmerita, E. Rodriguez, F. Cappello, Blocking vs. non-blocking coordinated checkpointing for large-scale fault tolerant MPI Protocols, Future Generation Comput. Syst. 24 (1) (2008) 73–84. http://dblp.uni-trier.de/db/journals/fgcs/fgcs24.html#BuntinasCHLPRRC08.
[99] S. Pellegrini, F. Giacomini, A. Ghiselli, A practical approach for a workflow management system, in: CoreGRID Workshop on Grid Middleware, Springer Verlag, Dresden, Germany, 2007.

Marek Wieczorek received his Master's degree in Computer Science from the AGH University of Science and Technology in Cracow, Poland, in 2003. From 2004 he joined the Institute of Computer Science, University of Innsbruck, Austria, where he works as Research Assistant in the Distributed and Parallel Systems group and is conducting his Ph.D. studies in Computer Science. He is interested in algorithmics, and his research is focused on scheduling of workflow applications on the Grid. He participated in several national and European projects, including the IST-2004-511385 K-WfGrid project. He is the author and co-author of over 10 scientific papers, including one journal article.

Andreas Hoheisel received his diploma in Geophysics with the subsidiary subjects of meteorology and geology from the University to Cologne. Since 2000 he is a researcher at the ISY department of Fraunhofer FIRST in Berlin/Germany. Main work areas include model coupling and integration, service-oriented architectures, Grid computing, and workflow management.

Radu Prodan received his Master's degree in Computer Science from the Technical University of Cluj-Napoca, Romania, in 1997. Between 1998 and 2001 he served as Research Assistant in Switzerland at ETH Zurich, University of Basel and the Swiss Centre for Scientific Computing. In 2001 he joined the Institute for Software Science, University of Vienna, where he earned his Ph.D. in 2004 from the Vienna University of Technology. Prodan is currently an assistant professor at the Institute of Computer Science, University of Innsbruck. He is interested in distributed software architectures, compiler technology, performance analysis, and scheduling for parallel and Grid computing. Prodan participated in several national and European projects and is currently workpackage leader in the IST-034601 edutain@grid project. He is the author of over 40 papers, including one book, seven journal articles, and one IEEE best paper award.