Taxonomies of the Multi-criteria Grid Workflow Scheduling Problem

M. Wieczorek, R. Prodan
{marek,radu}@dps.uibk.ac.at
University of Innsbruck
Technikerstraße 21a, A-6020 Innsbruck, Austria

A. Hoheisel
{andreas.hoheisel}@first.fraunhofer.de
Fraunhofer FIRST
Kekuléstraße 7, D-12489 Berlin, Germany

CoreGRID Technical Report
Number TR-0106
August 30, 2007

Institute on Grid Information, Resource and Workflow Monitoring Services
Institute on Resource Management and Scheduling

CoreGRID - Network of Excellence
URL: http://www.coregrid.net

CoreGRID is a Network of Excellence funded by the European Commission under the Sixth Framework Programme

Project no. FP6-004265

Taxonomies of the Multi-criteria Grid Workflow Scheduling Problem

M. Wieczorek, R. Prodan
{marek,radu}@dps.uibk.ac.at
University of Innsbruck
Technikerstraße 21a, A-6020 Innsbruck, Austria

A. Hoheisel
{andreas.hoheisel}@first.fraunhofer.de
Fraunhofer FIRST
Kekuléstraße 7, D-12489 Berlin, Germany

CoreGRID TR-0106
August 30, 2007

Abstract

The workflow scheduling problem which is considered difficult on the Grid becomes even more challenging when multiple scheduling criteria are used for optimization. The existing approaches can address only certain variants of the multi-criteria workflow scheduling problem, usually considering up to two contradicting criteria being scheduled in some specific Grid environments. A comprehensive description of the problem can be an important step towards more general scheduling approaches. Based on the related work and on our own experience, we propose several novel taxonomies of the multi-criteria workflow scheduling problem, considering five facets which may have a major impact on the selection of an appropriate scheduling strategy: scheduling process, scheduling criteria, resource model, task model, and workflow model. We analyze different existing workflow scheduling approaches for the Grid, and classify them according to the proposed taxonomies, identifying the most common use cases and the areas which have not been sufficiently explored yet.

1 Introduction

Scheduling of computational tasks on the Grid is a complex optimization problem which may require different scheduling criteria to be considered. Usually, execution time is applied as the most important criterion. In some other cases, the global efficiency (job throughput) should be maximized by the Grid system. In market models (especially in business Grids), economic cost optimization is also considered. Other possible criteria include quality of results, reliability of service, etc. 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 workflow scheduling on the Grid, most of them addressing two specific criteria (usually execution time and economic cost), by applying some specific approaches invented for specific cases. Our goal is to analyze the general problem of Grid workflow scheduling, by discovering regularities and irregularities between different problem variants. We aim at providing a study which can be used as a basis to move towards a scheduling approach addressing different problem classes for multiple scheduling criteria. The rest of the paper is organized as follows. In Section 2, we formally describe the problem which we want to address. Section 3 provides our contribution to the state of the art.

This research work is carried out under the FP6 Network of Excellence CoreGRID funded by the European Commission (Contract IST-2002-004265).

We introduce several taxonomies of the workflow scheduling problem for different aspects, considering both different problem variants and different approaches used to solve the problem. At the end of the section, we summarize the performed case study, by classifying several existing workflow scheduling approaches according to the taxonomies introduced previously. Finally, Section 4 concludes the paper and provides a short roadmap for the future work.

2 Grid workflow scheduling problem

We define Grid workflow scheduling as the problem of assigning different Grid services to different workflow tasks. Every workflow is a *directed graph* (digraph) $w \in \mathcal{W}, w = (\mathcal{V}, \mathcal{E})$ consisting of a set of nodes $\mathcal{V}$ and a set of edges $\mathcal{E}$, where nodes and edges represent tasks $\tau \in \mathcal{T}$ and data transfers $dt \in \mathcal{D}$ (as we explain in Section 3.5, the mapping between the sets $\mathcal{V}$, $\mathcal{E}$, and the sets $\mathcal{T}$, $\mathcal{D}$ can differ, depending on the current workflow model). In some workflow representations applied in the related work cited by us, workflow elements may have special semantics that defines complex workflow constructs (loops, parallel loops, `if`/`switch` conditions). Workflows expressed in such formalisms (e.g., Petri Nets [26], BPEL [47], AGWL [23]) can be systematically reduced during the runtime to simple Directed Acyclic Graphs (DAG), for instance by means of loop unrolling and by predicting and evaluating the conditions [35]. In case of any full-ahead workflow scheduling approach, such conversion has to be performed globally for the whole workflow each time when the scheduling is triggered. The set $\mathcal{S}$ contains all the *services* that are available for scheduling in the Grid and that implement different workflow tasks. In order to run a workflow, every task of the workflow has to be mapped to a service that implements the task. For every task $\tau_i \in \mathcal{T}$, there is a set $\mathcal{S}_i = \{s_{i1}, ..., s_{ip_i}\} \subset \mathcal{S}$ of the services which implement the task τ_i, where p_i may differ for different i. A *schedule* is defined as a function $sched_w : \mathcal{T} \mapsto \mathcal{S}$, where $sched_w$ assigns to each task $\tau_i \in \mathcal{T}$ a service $s \in \mathcal{S}_i$, creating a complete *schedule* (mapping) of the workflow w. Set $\mathcal{SC}$ contains all possible schedules for all workflows $w \in \mathcal{W}$. The *cost model* for workflows is described by n multiple *scheduling criteria* $C_i, 1 \leq i \leq n, n \in \mathbb{N}^+$, for instance by execution time, economic cost, and quality of results. The *partial cost functions* $cost_i : \mathcal{S} \mapsto \mathbb{R}, 1 \leq i \leq n$, defined for each scheduling criterion C_i, assign to each service $s_j \in \mathcal{S}$ its *partial cost* c_i^j (e.g., “execution time of 5 minutes”, “economic cost of 5\$”, “quality of results 100%”). In the remainder of this paper, we will sometimes refer to the cost of a service $s \in \mathcal{S}$ which is mapped to a task $\tau \in \mathcal{T}$ (i.e., where $sched_w(\tau) = s$) as the *cost of the task* τ. Similarly to the partial cost functions, the *total cost functions* $cost_i^{tot} : \mathcal{W} \times \mathcal{SC} \mapsto \mathbb{R}, 1 \leq i \leq n$ assign to a workflow $w \in \mathcal{W}$ scheduled by $sched_w \in \mathcal{SC}$ its *total costs* c_i^{tot}, calculated based on the partial costs of the services mapped to the workflow tasks. The optimization goal is to find a schedule $sched_w$ with the *best possible* total costs $c_i^{tot}, 1 \leq i \leq n$. As we describe in Section 3.2, the total costs can be evaluated in different ways.

3 Taxonomies in workflow Grid scheduling

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 will analyze in detail 5 different facets of the problem:

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

For every facet, we propose a certain *taxonomy* which classifies different scheduling approaches into different possible *classes*. 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 notation *subject-predicate-object*, which we extend in some cases to distinguish between different *sub-classes* of the problem. The proposed taxonomies can by no means be considered to be exhaustive, as our attempt is to create a model only for a certain subset of the general workflow

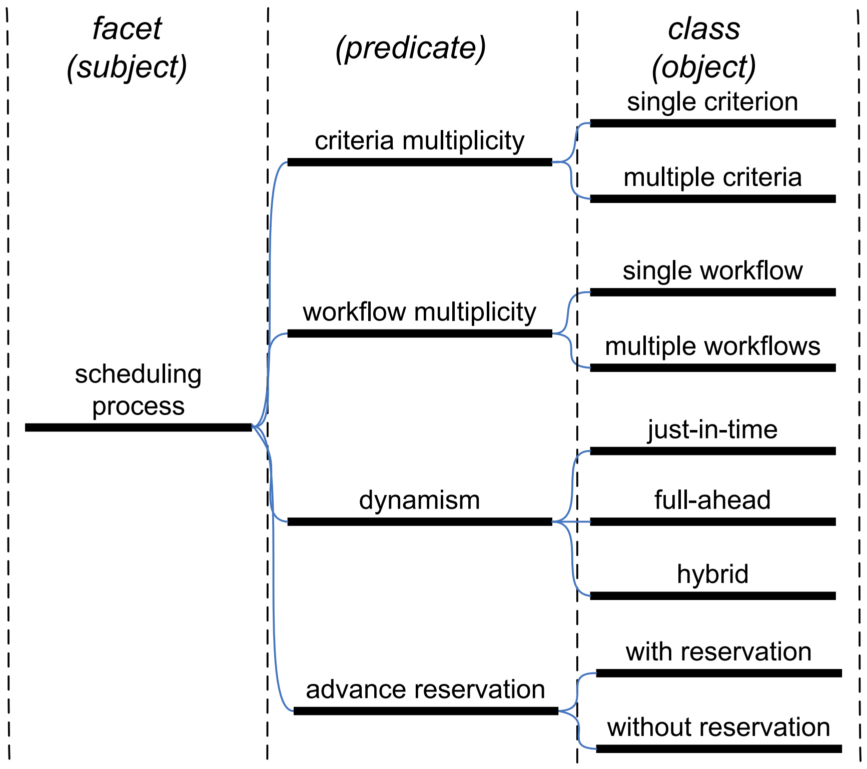

Figure 1: Taxonomy of workflow scheduling process

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, which partially come from the related work. Some of those examples are taken from [20], which provides a more complete analysis of the scheduling problem on the Grid.

3.1 Taxonomy of scheduling process

Different classes of Grid workflow scheduling can be distinguished with respect to different properties of the scheduling process (see Fig. 1). In this section, we will analyze both the information processed by the scheduler, and the way in which this information is being processed.

3.1.1 Criteria multiplicity

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

- *Single criterion*. The 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 which consider more than one criterion (e.g., [21, 59, 57, 58, 45, 8, 44]), and many of them consider the trade-off between execution time and economic cost. Vienna Grid Environment [8, 9] proposes a scheduling approach for multiple criteria (*Quality of Service parameters*), usually for execution time and economic cost. It applies a general multi-criteria scheduling approach, by using an optimization technique based on integer programming [43] to optimize a weighted goal function combining different QoS parameters.

Some other criteria are the main focus for the Grid-wide optimization (see Section 3.2) and for the pipelined workflows (see Section 3.5). In Instant Grid [27], a simple resource ranking model based on the number of CPUs and the last known load is created dynamically, in order to optimize the profit of the Grid. In [45], the scheduling of pipelined workflows is optimized with respect to the throughput and the latency of workflow execution.

3.1.2 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. 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 can optimize the execution of more than one workflow at a time. The work presented in [63] distinguishes three different approaches to the problem, the first one based on a sequential scheduling of multiple graphs (DAGs), the second one which 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 paper 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).

3.1.3 Dynamism

Workflow scheduling is a process which prepares workflows for an actual execution, therefore scheduling and execution should be considered together, and the time relation between them may differ for different scheduling approaches. In [18], three different types of workflow scheduling are distinguished: *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 going to be executed. The first approach combines the two former approaches by performing full-ahead planning every time a new scheduling decision needs to be made. Based on this classification, we distinguish the following three classes of scheduling processes:

- *Just-in-time scheduling* (in-time local scheduling). The scheduling decision for an individual task is postponed as long as possible, and performed before the task execution starts (fully dynamic approach).
- *Full-ahead planning* (full-plan-ahead). The whole workflow is scheduled before its execution starts (fully static approach).
- *Hybrid*. The scheduling approach combines the two aforementioned approaches.

Just-in-time scheduling is represented by many simple scheduling heuristics like Min-min, Max-min, Suffrage, and XSuffrage. These approaches are also applied to schedule parameter sweep workflows on the Grid [13]. Two typical example approaches which fall into the second class are presented in [42] and [57]. 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). The static planning can be applied only if the *meta data* for performance prediction is known in advance. The hybrid approach proposed in [19] 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. Another hybrid approach presented in [60] 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 needed, as well as a monitoring system which can control the fulfillment of these criteria. An example of such acceptance criteria are the *performance contracts* proposed in [52], which define the expectation concerning the execution time of the applications, and which are applied in the GrADS system [17, 6].

3.1.4 Advance reservation

When scheduling a workflow, we should take into consideration the environment in which the workflow will be executed. Most of the Grid environments are based on local resource managements with standard queuing systems which can give only a guarantee that a task submitted to the Grid will be executed at some time point. Many of the systems (e.g., Pegasus [19]) are based on DAGMan [16] which is a simple workflow processor which processes workflows and sends workflow tasks to local queuing systems. This simple model can be extended by applying *advance reservation*, which 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 his 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.

When considering queuing systems, the Grid scheduler should be aware if the queues on resources have finite or infinite length (capacity). In case of the finite-length queues, it is possible that queues become full and some jobs are lost, which may cause the need for their resubmission. Different advance reservation models for workflow Grid scheduling are proposed in [44, 54, 62]. In [44], different algorithms for resource provisioning are proposed, which reserve time slots on resources based on the economic cost and the execution time criteria. The approach presented in [54] proposes a workflow scheduling approach 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 putting 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 which does not impose any fairness policy. In [62], an advance reservation model is proposed 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*.

3.2 Taxonomy of scheduling criteria

The scheduling criteria may be characterized by various properties (e.g., workflow structure dependence, calculation method) which 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 differ with respect to the Grid actor (e.g., resource consumer, 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.

3.2.1 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 executed the workflow (e.g., execution time, economic cost).
- *Grid-wide*. The optimization criterion is defined for the Grid environment (e.g., resource usage, fairness of execution).

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 [3], Sun Grid Engine [46], LSF [1], Maui [15]), and is also applied for workflow scheduling, for instance in [63] where fairness of multiple workflow executions is considered as one of the optimization goals. Dynamic cost models based on Grid Economy and on other negotiation-based strategies, which are described more in detail later in this section, can be used to equilibrate between the requirements of the user and of the Grid. *Market equilibrium* which is the goal of any economy-based technique is a desirable state from the point of view of the Grid environment. Some study is conducted in [31, 33, 32] to compare the influence which different negotiation strategies have on resource utilization on the Grid.

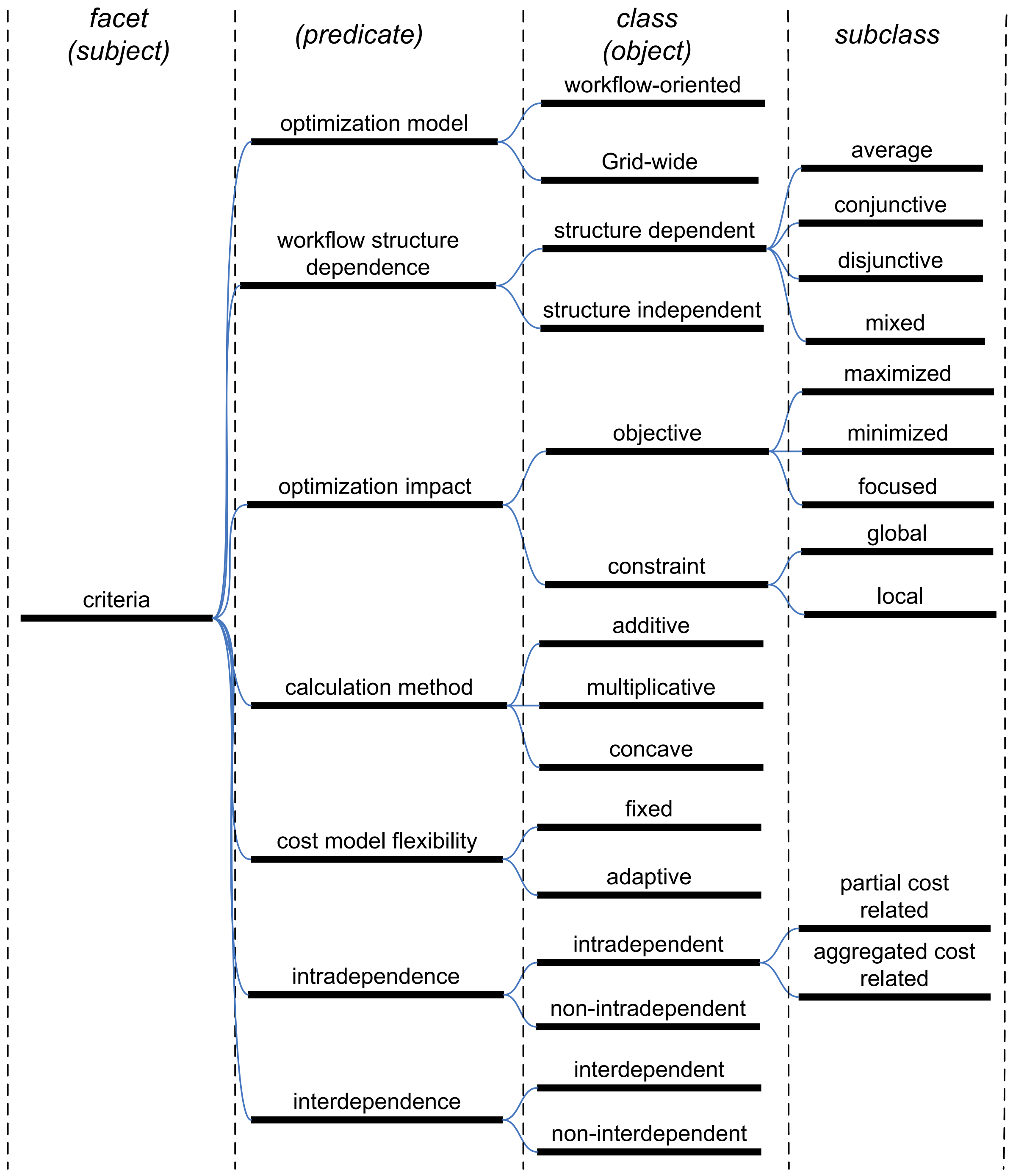

Figure 2: Taxonomy of workflow scheduling criteria

3.2.2 Workflow structure dependence

Whereas tasks in a task batch are independent, workflows contain dependencies between tasks which 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).

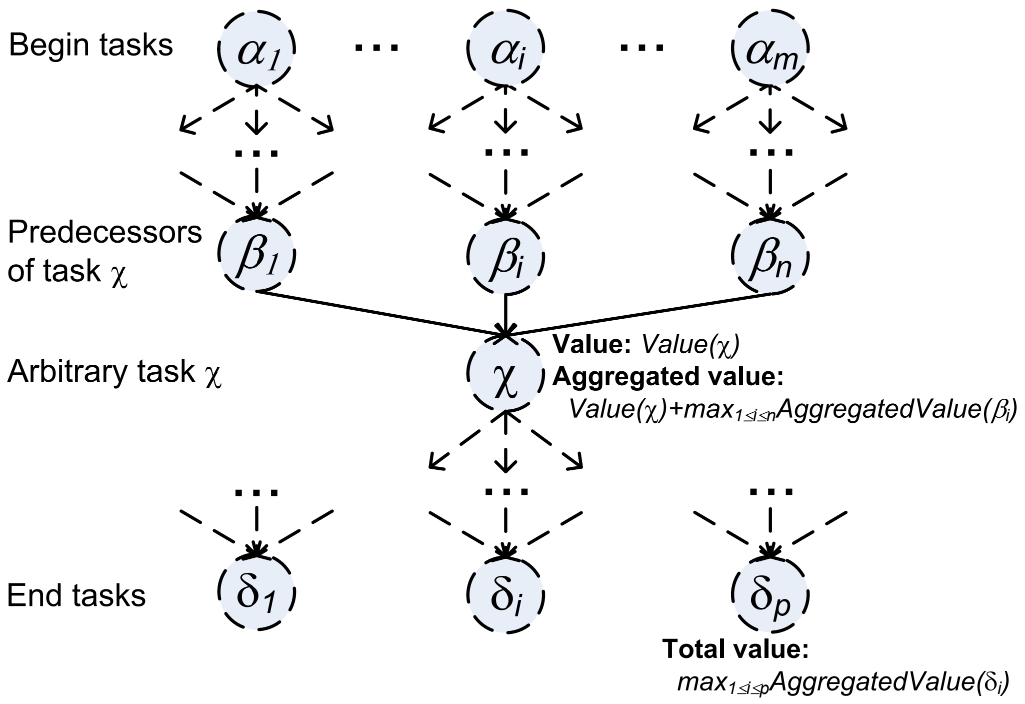
Begin tasks
α_1
α_i
α_m
Predecessors of task χ
β_1
β_i
β_n
Arbitrary task χ
χ
Value: $Value(\chi)$
Aggregated value: $Value(\chi)+max_{1\le i\le n}AggregatedValue(\beta_i)$
End tasks
δ_1
δ_i
δ_p
Total value: $max_{1\le i\le p}AggregatedValue(\delta_i)$

Figure 3: Recursive calculation of aggregated costs for a structure dependent criterion

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

Most of the existing workflow scheduling approaches only optimize execution time which is a structure dependent criterion. Some multi-criteria workflow scheduling approaches (e.g., [21, 59, 57, 58, 8]) also consider economic cost which is structure independent. Some other scheduling criteria can belong to either of the two classes, depending on the way the user defines them. Let us denote by *quality of results* any kind of qualitative description (for instance, expressed in percentage) of the results produced by alternative services (this quality will usually be higher for an expensive commercial application than for its open-source equivalent). To calculate the quality of the final results, the user can either simply multiply the quality of the results produced by individual workflow tasks, or can also consider the dependencies between different tasks and the order in which the partial results are produced, defining in this way a structure dependent function which calculates the quality of results.

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. Let us consider as an example the calculation of execution time. In order to calculate the total execution time, we calculate the *aggregated costs* (execution times) for all workflow tasks $\tau \in \mathcal{T}$ in a workflow $w \in \mathcal{W}$, and use the maximum aggregated cost as the total cost (execution time) of the workflow. A 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 $\beta_i, 1 \leq i \leq n$. The aggregated costs are calculated recursively, so the same scheme would also apply for the tasks $\beta_i, 1 \leq i \leq n$. The aggregated cost function will be denoted $acost : \mathcal{T} \times \mathcal{W} \times \mathcal{SC} \mapsto \mathbb{R}$. In case of execution time, the aggregated costs of the predecessors are aggregated by finding 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 quality of results), the aggregation function can calculate the mean (or weighted mean) over the arguments. Such function is referred to as *averaging function*. Many different averaging functions are proposed in the literature ([51, 36]). For our taxonomy, we chose four averaging functions which seem to be most relevant from the point of view of workflow scheduling:

- *Averaging*. *Averaging functions* give outputs which lie between the greatest and the smallest elements 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).
- *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 which is introduced in the later part of this section. However, an aggregation function can only be defined for the structure dependent criteria, and it applies only to a part of the cost calculation procedure (i.e., to the aggregation of the predecessor costs).

3.2.3 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 established for a certain criterion (e.g., by a budget limit or a time 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 which 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 (e.g., [42, 19, 34, 37]), there is an optimization objective defined for execution time (time minimization). A common way to deal with a multi-criteria scheduling [50] is to define an optimization objective for one criterion, and to establish constraints for all the other criteria. The scheduling techniques presented in [59, 57, 58, 21] 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 connected 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 the quality of results or the security and reliability of 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 which 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 variants 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 approaches (e.g., [8]) distinguish *global constraints* and *local constraints*:

- *Global constraint*. A constraint defined for the whole workflow.
- *Local constraint*. A constraint defined for a single workflow task.

3.2.4 Calculation method

Another classification can be done with respect to the operation used for the 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 a difference that the partial costs are added up taking into consideration also the structure of the workflow (see Fig. 3). There exist a large number of criteria for which it is convenient to express costs as real numbers from the range $[0, 1]$ (e.g., quality of results, probability of failure, availability rate, security). 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 in [56]. 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 should be distinguished:

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

3.2.5 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 have 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*, which 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 result from negotiations between different actors of the Grid.

Adaptive pricing have been extensively studied in the past (although usually not for workflow scheduling), and different models have been proposed. An important class of such models originates from human economy, so the common name to refer to them is *Grid Economy*. Many Grid Economy models have been enumerated and discussed in [11, 10], where a Grid architecture realizing them has also been proposed. In the *commodities market model*, prices are established centrally 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 the their offers. The *auction model* supports 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 in [11] 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 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 [55]. The scheduling approach proposed in [49] uses the commodities market model to determine the cost of resource usage in context of non-workflow streaming applications. The approches based on a single market and on multiple markets are compared in this work. The work presented in [55] compares the economic models based on the commodities market and on the second-price Vickrey auctions, showing the superiority of the former approach in terms of the economic factors like 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. A real workflow scheduling approach based on an economic model is introduced in [14], in which the first-price auction model is applied. Workflows are scheduled in a full-ahead manner, and the scheduling is performed together with bidding for resources. The distance of individual tasks from the end of the workflow determines how *urgent* each task is; the more urgent tasks are given higher prices during the auction in order to increase the possibility of meeting the deadline defined for the workflow.

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. A good introduction to the problem of automatic negotiation is presented in [28]. 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 in [31, 33, 32] proposes non-workflow scheduling techniques using heuristic-based negotiation strategies. The heuristics are implemented through special *utility functions* which 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.

3.2.6 Intradependence

The notion of intradependence of scheduling criteria has a major impact on the 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 can be 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 we can say that the scheduling decisions depend on one another within a scheduling criterion. Also for execution time, the scheduling decisions made for some tasks may influence the aggregated costs of some other tasks (because tasks consume resources whose amount is limited). On the other hand, the scheduling decisions made for criteria like reliability, quality of results, or the economic cost calculated in a simple price model does not seem to show any intradependence. From this point of view, we will distinguish two classes of criteria:

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

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* for some other tasks. For a change in execution time, a scheduling decision made for a workflow task does not always change the execution times of other tasks, however it usually 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).

3.2.7 Interdependence

When considering multiple scheduling criteria, we may observe that some of them strongly depend on others, whilst 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 its more reliable counterparts. On the other hand, the economic cost of a service usage 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 criteria, the multi-criteria optimization problem can often be reduced to the optimization of a goal function being a simple product. Therefore, when considering groups of criteria, we will distinguish the following two disjoint classes:

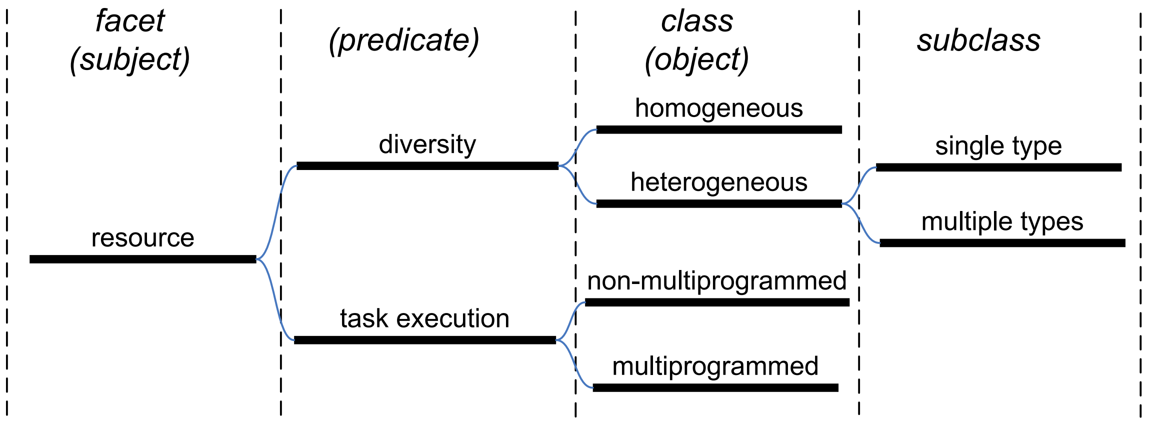

Figure 4: Taxonomy of Grid resources

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

A workflow scheduling approach based on the idea of interdependent criteria reduction is proposed in the Instant-Grid [27]. The two criteria (number of CPUs and the last known load) 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-wide optimization perspective applied).

3.3 Taxonomy of Grid resources

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 to 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 the way in which the resources handle the execution of tasks. The proposed taxonomy of Grid resources from the point of view of workflow scheduling is shown in Fig. 4.

3.3.1 Diversity

One of the main characteristics of the Grid resources is their *heterogeneity*. Therefore, most of the existing Grid environments belong to the second one 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 types, different performance, different load, etc.).

Heterogeneity can be understood as the existence of diverse characteristics (e.g., CPU speed, RAM 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 which have 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*. The resources of the same type (e.g., computational resources) differ with respect to their characteristics (e.g., CPU speed, RAM size).
- *Multiple types*. The resources differ with respect to their types (e.g., computational, storage, and network resources).

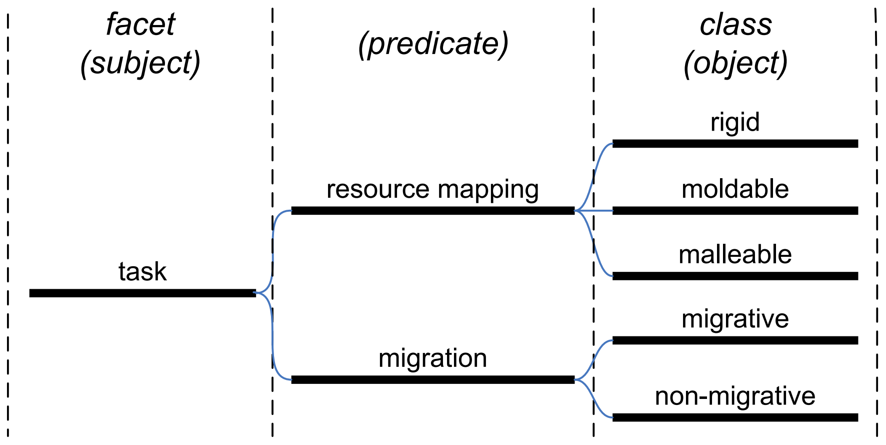

Figure 5: Taxonomy of workflow tasks

The existing workflow scheduling approaches we are aware of address the former variant of the problem, although the characteristics of some types of resources (e.g., network bandwidth, storage size) are sometimes included in the description of the computational resources (e.g., [25]).

Much effort has been put into addressing multiple types of resources on the Grid. Stork [30] aims at "making data placement a first class citizen in the Grid", by handling data transfers tasks in a similar way as execution tasks. The concept of Open Grid Service Architecture (OGSA) [24] 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) [53] developed within the VIOLA project aims at co-allocation of different types of resources (currently, compute resources and network resources) in multiple administrative domains.

3.3.2 Task execution

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 to be executed on a resource at the same time.
- *Multiprogrammed*. The scheduler can schedule multiple tasks to be executed on a resource at the same time.

The resources from these two classes are sometimes referred to also as *disjunctive* and *cumulative*, respectively [4]. Most of the existing Grid environments consist of parallel machines being managed by local resource managers which 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 which we are aware of address the non-multiprogrammed resource model. In [41], a scheduler called O-OSKAR is proposed, which schedules workflows of general (not necessarily computational) activities on multiprogrammed resources. The problem is approached as a Meta-CSP (Meta- Constraint Satisfaction Problem), and solved using an algorithm called ISES [2].

3.4 Taxonomy of workflow tasks

Workflow tasks may differ with respect to their requirements and characteristics which have to be taken into consideration when scheduling a workflow. The proposed taxonomy of tasks is depicted in Fig. 5.

3.4.1 Resource mapping

In a similar way as a single resource can be used by multiple tasks at a time (see Section 3.3), also a single task may require multiple resources to be used (e.g., parallel MPI and PVM programs). We can distinguish three classes of tasks, with respect to its resource mapping requirements:

- *Rigid*. A task requires a fixed number of resources to be used (usually, one resource).

- *Moldable*. A task requires multiple Grid resources to be used, and the number of resources required by the task is not known *a priori* but determined before the execution starts.
- *Malleable*. A task requires multiple Grid resources to be used which may be added or withdrawn from a job according to the current system state.

The processing speed of a task (referred to as the *processing speed function*) is usually a nonlinear function of the number of processors allocated to the task. Most of the existing workflow scheduling approaches assume that tasks belong to the first class. 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 [39]: the first step aims at finding an optimal *allocation* for each task, and the second step determines a *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., [40, 39],[12],[45, 25]).

A typical algorithm which deals with the problem of workflow scheduling of moldable tasks in homogeneous environments is the Critical Path and Area-based algorithm (CPA) [40]. This algorithm aims at finding the best compromise between the length of the critical path, and the *average area* T_A which measures the mean processor-time area required by the application. Formally, $T_A = \frac{1}{R}\sum_{i=1}^{N}(t(\tau_i, N_p(t_i)) \cdot N_p(t_i))$, where R denotes the total number of resources, N the total number of tasks, $\tau_i, 1 \leq i \leq N$ a task, $N_p(\tau_i)$ the number of resources allocated to the task τ_i, and $t(\tau_i, N_p(\tau_i))$ the execution time of the task τ_i executed on $N_p(\tau_i)$ resources. In [39], CPA is extended to 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, and (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 in [12]. 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 which extends the existing Heterogeneous Earliest Finish Time (HEFT) algorithm [61] 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 single 1-processor configuration of each site. Vienna Grid Environment [8] applies heuristics to determine the number of processors required to execute an MPI job within the user-specified time constraints.

The work presented in [25] addresses 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 3.5), and the third type, which is exploited 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*), which are originally described as *single-node plans* (where *node* refers to a computational node), and which are subsequently converted to *multi-node plans* (in which 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 *costlies* (i.e., most time consuming) parallelizable operators.

The problem of scheduling of malleable tasks in a parallel environment is addressed in [7]. 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 projects [6] 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.

The requirement of multiple resources for a task is connected with the concept of *co-allocation*, i.e., the simultaneous allocation of resources in multiple sites. In the KOALA Grid Scheduler [38], co-allocation is done by the Co-allocator (CO) which is responsible for finding the execution sites with enough idle processors for the tasks. In the

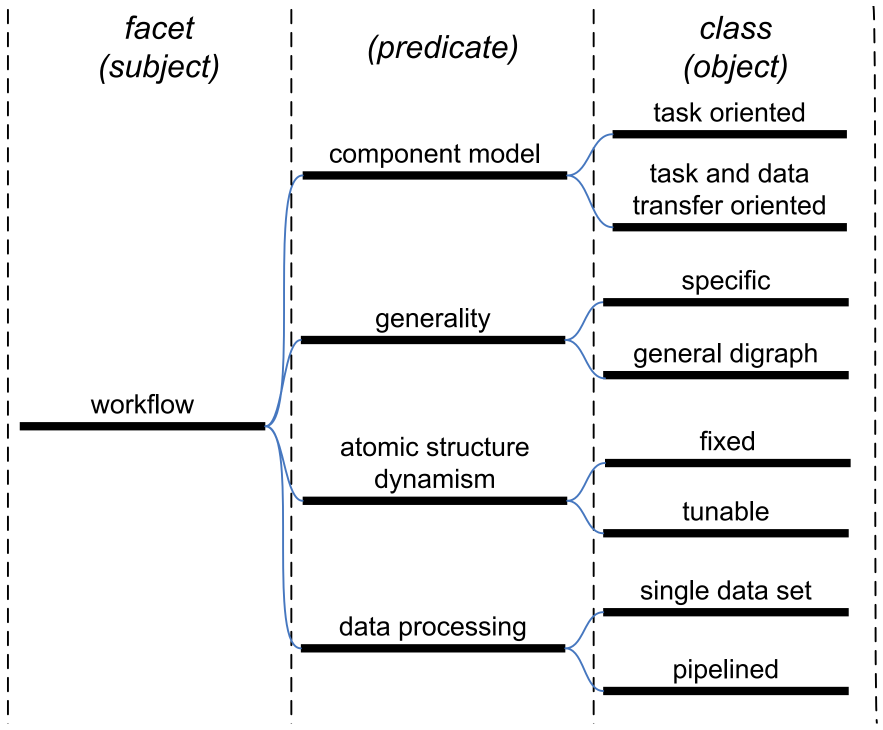

Figure 6: Taxonomy of workflow model

MetaScheduling-Service (MSS) [53], developed within the VIOLA project, heterogeneous resources are co-allocated across multiple administrative domains.

3.4.2 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 enabled.

Task migration is rarely applied in the real Grid, due to well-known problems with the implementation of reliable and effective task migration. All existing implementations are restricted only to specific platforms, and impose strict prerequisites on the tasks which can be migrated [5]. The only Grid workflow system we are aware of which supports task migration is GrADS [6].

3.5 Taxonomy of workflow model

The taxonomy depicted in Fig. 6 differentiates workflows with respect to their representation and behavior.

3.5.1 Component model

From the scheduling point of view, workflows may differ with respect to the way computational tasks and data transfers are represented in them. We can distinguish two classes of workflow models:

- *Task oriented*. Computational tasks are represented as graph nodes. Data transfers are represented as graph edges.
- *Task and data transfer oriented*. Both computational tasks and data transfers are represented as graph tasks.

The existing Grid workflow scheduling approaches are based predominantly on the former model. There are only few workflow representations which support the latter model (e.g., Karajan [29] and Stork [30]). 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 *VGE services* representing data transfers. The distributed query workflows used in [25] include also special workflow nodes called *exchange operators* which involve communication between other workflow nodes.

3.5.2 Generality

Although the workflow model specified by us in Section 2 is the directed graph (digraph), many existing workflows have a well-defined structure which can be described by a simpler model being a subset of the general digraph model (e.g., a master-worker workflow with well-defined parallel sections of identical tasks). For specific workflow models, there may exist some specialized algorithms which produce better results than any general-purpose digraph scheduling algorithm. Therefore, we will distinguish the following two workflow models:

- *Specific*. The workflow structure has certain regularities, so it can be described by a well-defined subset of the general digraph model (e.g., parameter sweep applications).
- *General digraph*. The workflow is a general digraph defined in Section 2.

Many existing approaches are based on a specific workflow model. The work presented in [45] considers a pipelined workflow model based on a sequence of data parallel tasks. The workflows used to model distributed database queries in [25] are based on a special tree-like structure constructed according to certain restricted composition rules. The regular structure of the workflows considered in [19] allowed to introduce the idea of workflow partioning which consists in converting the workflow to a sequence of subworkflows. The dynamic scheduling of the parameter sweep applications considered in [34] is approached by a special prioritization policy which gives higher priority to the tasks whose so-called *children's ancestors* have already been finished. In [13], several heuristics for dynamic scheduling of parameter sweep applications (Min-min, Max-min, Suffrage) are compared, and a new heuristic called XSuffrage is proposed. In the Abstract Grid Workflow Language (AGWL) [23] used in ASKALON [22], the 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 general digraph model [35]. The general digraph model of workflows is addressed for instance in [42, 63, 37, 21, 59].

3.5.3 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 to increase 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 may modify the workflow structure (for optimization purposes) within the scheduling process, in contrast to the approaches which 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 [48], workflows are created on demand and semantically tuned by the components called Workflow Composition Tool (WCT) and Automatic Application Builder (AAB) before the tasks are mapped to services. Also in PEGASUS [19], the workflows are first converted from an *abstract* to a *concrete* form. Three different restructuring techniques are involved in this process. Firstly, data sets which 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.

Secondly, 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 multi-processor 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., [45]), 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.

3.5.4 Data processing

This classification distinguishes two different types of workflow processing, which 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 data set.* The workflow is executed once, for a single set of input data.
- *Pipelined*. The workflow is executed many times, for multiple data sets which 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 in [45] 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 which can be mapped onto a parallel machine in a variety of ways, employing different combinations of task and data parallelism.

In [49], the authors 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 workflow scheduling, they provide only a solution for scheduling of single processing units.

3.6 Classification of the existing Grid systems

To summarize the material presented in this section, in Table 1 we show a survey of different existing scheduling approaches, classified according to the proposed taxonomies. In this survey, we concentrate only on the workflow scheduling approaches dedicated for the Grid, although the term “Grid” may not be explicitly mentioned in all of them. In order to make the comparison more concise, we do not show there the classifications introduced by us for scheduling criteria (except for optimization model and cost model flexibility). Instead, we just state explicitly whether the compared approach considers execution time, economic cost, or other kinds of criteria. Several times, groups of multiple approaches are described in a single table row. It was done when the approaches were proposed by the same authors and were logically related (e.g., were developed within the same project).

4 Conclusions

The presented study shows that multi-criteria scheduling on the Grid is a complex problem for which multiple variants can be distinguished based on different possible aspects. Obviously, it is not feasible in general to develop a single scheduling approach which works efficiently for all classes of the problem. For instance, it is rather unlikely that a scheduling approach which works well for workflows consisting of rigid tasks running on non-multiprogrammed resources will work equally good for a pipelined workflow processing a stream of video data, containing moldable tasks which can share the same resources. Therefore, when developing any general scheduling strategy, the first step should be to identify the set of problem classes which can be approached in a similar way.

There exist some multi-criteria workflow scheduling approaches, most of them considering execution time as the most important scheduling criterion. In most of the cases, the scheduling process performed for the criteria is workflow-oriented. The existing workflow scheduling approaches are usually based on full-ahead planning. Most of them are designed for task oriented general digraphs and on the data processing model based on a single data input set. The pipelined workflows, which are characteristic only for some specific areas (e.g., for multimedia systems) have considerably different behaviors and require different scheduling techniques.

There are almost no workflow scheduling approaches which are based on an adaptive cost model for criteria. Such cost models present a very promising research direction, as they can lead towards scheduling techniques applicable

Paper/System	multiple criteria	execution time	economic cost	other criteria	multiple workflows	just-in-time	full-ahead	hybrid	advance reservation	workflow-oriented	Grid-wide	fixed cost model	adaptive cost model	heterogeneous resources	multiprogrammed resources	moldable tasks	malleable tasks	migrative tasks	task oriented	task and data transfer oriented	specific workflow model	general digraph model	tunable workflows	single data set workflows	pipelined workflows
GrADS [6, 17]	-	+	-	-	-	-	+	+	-	+	-	+	-	+	-	+	+	+	+	-	-	+	-	+	-
Vienna Grid Environment [8, 9]	+	+	+	+	-	+	+	-	+	+	-	+	-	+	-	+	-	-	+	-	-	+	+	+	-
Casanova et al. [12]	-	+	-	-	-	-	+	-	-	+	-	+	-	+	-	+	-	-	+	-	-	+	-	+	-
Casanova et al. [13]	-	+	-	-	-	+	-	-	-	+	-	+	-	+	-	-	-	-	+	-	+	-	-	+	-
Chien et al. [14]	-	+	-	-	-	-	+	-	-	+	-	-	+	+	-	-	-	-	+	-	-	+	-	+	-
PEGASUS [18, 19]	-	+	-	-	+	-	+	+	-	+	-	+	-	+	-	+	-	-	+	-	+	+	+	+	-
Singh et al. [44]	+	+	+	-	-	-	+	-	+	+	-	+	-	+	-	+	-	-	+	-	-	+	-	+	-
Tsiakkouri et al. [21]	+	+	+	-	-	-	+	-	-	+	-	+	-	+	-	-	-	-	+	-	-	+	-	+	-
Sakellariou, Zhao [42, 62, 63]	-	+	-	-	+	-	+	-	+	+	-	+	-	+	-	-	-	-	+	-	-	+	-	+	-
ASKALON [22, 35, 54, 23]	-	+	-	-	+	-	+	-	+	+	+	+	-	+	-	-	-	-	+	-	+	+	-	+	-
Gounaris [25]	-	+	-	-	-	-	+	-	-	+	-	+	-	+	-	+	-	-	-	+	+	-	-	+	-
K-WfGrid [26, 48]	+	+	-	+	-	-	+	-	-	+	-	+	-	+	-	-	-	-	+	-	-	+	+	+	-
Instant-Grid [27]	+	-	-	+	+	+	-	-	-	-	+	+	-	+	-	-	-	-	+	-	-	+	-	+	-
Ma, Buyya [34]	-	+	-	-	-	+	+	-	-	+	-	+	-	+	-	-	-	-	+	-	+	-	-	+	-
Yu, Buyya, et al. [57, 58, 59]	+	+	+	-	-	-	+	-	+	+	-	+	-	+	-	-	-	-	+	-	-	+	-	+	-
Mika et al. [37]	-	+	-	-	-	-	+	-	-	+	-	+	-	+	-	-	-	-	+	-	-	+	-	+	-
N'Takpé, Suter [39]	-	+	-	-	-	-	+	-	-	+	-	+	-	+	-	+	-	-	+	-	-	+	-	+	-
Radulescu et al. [40]	-	+	-	-	-	-	+	-	-	+	-	+	-	-	-	+	-	-	+	-	-	+	-	+	-
Subhlok, Vondran [45]	+	-	-	+	-	-	+	-	-	+	-	+	-	-	-	+	-	-	+	-	+	-	+	-	+
Yu, Shi [60]	-	+	-	-	-	-	+	+	+	+	-	+	-	+	-	-	-	-	+	-	-	+	-	+	-

Table 1: Survey of Grid workflow scheduling approaches

in utility Grids with paid access to resources, and which can address the challenges like Service Level Agreements (SLAs). Advance reservation can be applied as a logical extension of such models. There is still a large research potential for scheduling of malleable tasks, and for the multiprogrammed resource model, although it is not certain whether the latter problem class has any significant practical meaning (we are not aware of any workflow scheduling research for the Grid which addresses this problem). Another interesting research area is related with the heterogeneity model based on multiple resource types. Also workflow tuning and task migration as optimization methods seem to be underrepresented among the existing scheduling approaches.

The current study shows that the Grid workflow scheduling problem is still not fully addressed by the existing work. We believe that the presented taxonomies will facilitate development of scheduling approaches capable of dealing with some of the distinguished problem classes. In the future, we are planning to invent a generic scheduling approach for two or more criteria, exploring different types of criteria. An economic model provided for multiple consumers and providers, incorporating price negotiation and advance reservation, seems to be most appropriate for our goals. Starting from simple cases (bi-criteria scheduling), we will try to move towards more complicated problem classes, considering different types of intradependence, and different characteristics of tasks and workflows.

References

[1] Platform Computing Inc. Platform LSF. http://www.platform.com/Products/Platform.LSF.Family/.

[2] A. O. A. Cesta and S. Smith. A Constrained-Based Method for Project Scheduling with Time Windows. *Journal of Heuristics*, 8:109–136, 2002.

[3] Altair Engineering, Inc. PBS Professional. http://www.altair.com/software/pbspro.htm.

[4] P. Baptiste, C. Le Pape, and W. Nuijten. *Constraint-based Scheduling: Applying Constraint Programming to Scheduling Problems*, volume 39 of *International Series in Operations Research and Management Science*. Kluwer Academic Publishers, Norwell, MA, 2001.

[5] J. Basney, M. Litzkow, T. Tannenbaum, and M. Livny. Checkpoint and migration of unix processes in the condor distributed processing system. Technical Report Technical Report 1346, April 1997.

[6] Berman, F. et al. New Grid Scheduling and Rescheduling Methods in the GrADS Project. *International Journal of Parallel Programming*, 33:209–229(21), June 2005.

[7] J. Blazewicz, M. Machowiak, J. Weglarz, M. Kovalyov, and D. Trystram. Scheduling Malleable Tasks on Parallel Processors to Minimize the Makespan: Models and Algorithms for Planning and Scheduling Problems. *Annals of Operations Research*, 129:65–80(16), July 2004.

[8] I. Brandic, S. Benkner, G. Engelbrecht, and 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*, pages 108–115, Washington, DC, USA, 2005. IEEE Computer Society.

[9] I. Brandic, S. Pllana, and S. Benkner. Amadeus: A Holistic Service-oriented Environment for Grid Workflows. *gccw*, 0:259–266, 2006.

[10] R. Buyya, D. Abramson, and S. Venugopal. The Grid Economy. In M. Parashar and C. Lee, editors, *Proceedings of the IEEE*, volume 93 of *Special Issue on Grid Computing*, pages 698–714. IEEE Press, New Jersey, USA, Mar 2005.

[11] R. Buyya, H. Stockinger, J. Giddy, and D. Abramson. Economic Models for Management of Resources in Peer-to-Peer and Grid Computing. Technical Report 0108001, Economics Working Paper Archive at WUSTL, 2001. available at http://ideas.repec.org/p/wpa/wuwpco/0108001.html.

[12] H. Casanova, F. Desprez, and F. Suter. From Heterogeneous Task Scheduling to Heterogeneous Mixed Parallel Scheduling. In M. Danelutto, D. Laforenza, and M. Vanneschi, editors, *Proceedings of the 10th International Euro-Par Conference (Euro-Par'04)*, volume 3149 of *Lecture Notes in Computer Science*, pages 230–237, Pisa, Italy, August/September 2004. Springer.

[13] H. Casanova, A. Legrand, D. Zagorodnov, and F. Berman. Heuristics for Scheduling Parameter Sweep Applications in Grid Environments. In *Proceedings of 9th Heterogeneous Computing Workshop (HCW)*, pages 349–363, Cancun, Mexico, May 2000.

[14] C.-H. Chien, P. H.-M. Chang, and 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*, pages 867–872, Washington, DC, USA, 2005. IEEE Computer Society.

[15] Cluster Resources, Inc. Maui Cluster Scheduler. http://www.clusterresources.com/pages/products/maui-cluster-scheduler.php.

[16] DAGMan (Directed Acyclic Graph Manager). http://www.cs.wisc.edu/condor/dagman/.

[17] H. Dail, O. Sievert, F. Berman, H. Casanova, A. YarKhan, S. Vadhiyar, J. Dongarra, C. Liu, L. Yang, D. Angulo, and I. Foster. Scheduling in the Grid Application Development Software Project, 2003.

[18] E. Deelman, J. Blythe, Y. Gil, and C. Kesselman. Workflow Management in GriPhyN. *Grid Resource Management, State of the Art and Future Trends*. pages 99–116, 2004.

[19] 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, and D. Katz. Pegasus: a Framework for Mapping Complex Scientific Workflows onto Distributed Systems. *Scientific Programming Journal*, 13(2), November 2005.

[20] F. Dong and S. G. Akl. Scheduling Algorithms for Grid Computing: State of the Art and Open Problems. Technical Report 2006-504, School of Computing, Queen's University, Kingston, Ontario, January 2006.

[21] H. Z. E. Tsiakkouri, R. Sakellariou and M. D. Dikaiakos. Scheduling Workflows with Budget Constraints. In S. Gorlatch and M. Danelutto, editors, *In Proceedings of the CoreGRID Workshop "Integrated research in Grid Computing"*, pages 347–357, Nov. 2005.

[22] T. Fahringer, R. Prodan, R. Duan, F. Nerieri, S. Podlipnig, J. Qin, M. Siddiqui, H.-L. Truong, A. Villazon, and M. Wieczorek. ASKALON: A Grid Application Development and Computing Environment. In *6th International Workshop on Grid Computing (Grid 2005)*, Seattle, USA, Nov. 2005. IEEE Computer Society Press.

[23] T. Fahringer, J. Qin, and 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)*, Cardiff, UK, May 9-12, 2005. IEEE Computer Society Press.

[24] I. Foster, C. Kesselman, and S. Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *International Journal of Supercomputing Applications*, 15(3), 2002.

[25] A. Gounaris, R. Sakellariou, N. Paton, and A. Fernandes. A novel approach to resource scheduling for parallel query processing on computational grids. *Distributed and Parallel Databases*, 19:87–106(20), May 2006.

[26] A. Hoheisel. User tools and languages for graph-based Grid workflows: Research Articles. *Concurr. Comput. : Pract. Exper.*, 18(10):1101–1113, 2006.

[27] A. Hoheisel and H. Rose. Konzept für das Scheduling von Workflow-Aktivitäten in Instant Grid. Technical report, Fraunhofer Institut für Rechnerarchitektur und Softwaretechnik, June 2006.

[28] N. R. Jennings, P. Faratin, A. R. Lomuscio, S. Parsons, C. Sierra, and M. Woodlidge. Automated Negotiation: Prospects, Methods and Challenges. *International Journal of Group Decision and Negotiation*, 10(2):199–215, 2001.

[29] Java CoG Kit Karajan Guide. http://www.cogkit.org/current/manual/workflow.pdf.

[30] T. Kosar and M. Livny. Stork: Making data placement a first class citizen in the grid, 2004.

[31] J. Li and R. Yahyapour. A Negotiation Model Supporting Co-Allocation for Grid Scheduling. In *Proceedings of 7th IEEE/ACM International Conference on Grid Computing (Grid'06)*, Barcelona, Spain, 2006. IEEE Computer Society Press.

[32] J. Li and R. Yahyapour. Learning-Based Negotiation Strategies for Grid Scheduling. In *IEEE Int'l Symposium on Cluster Computing and the Grid (CCGrid 2006), Singapore*, pages 567–583. IEEE Press, 2006.

[33] J. Li and R. Yahyapour. Negotiation Strategies for Grid Scheduling. In *The First International Conference on Grid and Pervasive Computing (GPC2006)*, volume 3947 of *Lecture Notes in Computer Science*, pages 42–52, Tunghai University, Taiwan, 2006. Springer-Verlag.

[34] T. Ma and 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)*, Rio de Janeiro, Brazil, Oct. 24-27 2005. IEEE Computer Society Press.

[35] M. Mair, J. Qin, M. Wieczorek, and T. Fahringer. Workflow conversion and processing in the askalon grid environment. *2nd Austrian Grid Symposium*, 2007.

[36] C. F. Mela and 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):165–181, 1995.

[37] M. Mika, G. Waligóra, and J. Weglarz. Workflow Management in GriPhyN. *Grid Resource Management, State of the Art and Future Trends*. pages 295–318, 2004.

[38] H. Mohamed and D. Epema. The Design and Implementation of the KOALA Co-Allocating Grid Scheduler. In *European Grid Conference*, volume 3470 of *Lecture Notes in Computer Science*, pages 640–650. Springer-Verlag, 2005.

[39] T. N'Takpé and F. Suter. Critical path and area based scheduling of parallel task graphs on heterogeneous platforms. In *Proceedings of the 12th International Conference onParallel and Distributed Systems, ICPADS 2006*, Minneapolis, Minnesota, USA, July 2006. IEEE Computer Society Press.

[40] A. Radulescu, C. Nicolescu, A. J. C. van Gemund, and 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)*, pages 39–41, San Francisco, USA, Apr. 2001. IEEE Computer Society Press.

[41] M. D. Rodriguez Moreno, D. Borrajo Millán, and D. Meziat Luna. *Representing and Planning tasks with time and resources*. PhD thesis, Universidad de Alcalá, Spain, Dec. 2003.

[42] R. Sakellariou and H. Zhao. A Hybrid Heuristic for DAG Scheduling on Heterogeneous Systems. In *IPDPS*, 2004.

[43] A. Schrijver. *Theory of Linear and Integer Programming*. Wiley, 1987. SchRI a 87:1 1.Ex.

[44] G. Singh, C. Kesselman, and E. Deelman. Application-Level Resource Provisioning on the Grid. *e-science*, 0:83, 2006.

[45] J. Subhlok and G. Vondran. Optimal Use of Mixed Task and Data Parallelism for Pipelined Computations. *Journal of Parallel and Distributed Computing*, 60:297–319(23), March 2000.

[46] Sun Microsystems, Inc. Grid Engine. http://gridengine.sunsource.net/.

[47] T. Andrews, et al. Business Process Execution Language for Web Services. Technical report, BEA Systems, et al., May 2003.

[48] The K-WfGrid project. http://www.kwfgrid.net.

[49] L. Tian and K. M. Chandy. Resource allocation in streaming environments. In *Proceedings of 7th IEEE/ACM International Conference on Grid Computing (Grid'06)*, Barcelona, Spain, 2006. IEEE Computer Society Press.

[50] V. T'kindt and J. Billaut. *Multicriteria Scheduling*. Springer Verlag, Berlin, 2002.

[51] A. Čaplinskas and J. Gasperovič. Techniques to Aggregate the Characteristics of Internal Quality of an IS Specification Language. *Informatica*, 16(4), 2005.

[52] F. Vraalsen, R. A. Aydt, C. L. Mendes, and D. A. Reed. Performance Contracts: Predicting and Monitoring Grid Application Behavior. *Lecture Notes in Computer Science*, 2242:154–166, 2001.

[53] O. Wäldrich, W. Ziegler, and P. Wieder. A Meta-Scheduling Service for Co-allocating Arbitrary Types of Resources. Technical Report TR-0010, Institute on Resource Management and Scheduling, CoreGRID - Network of Excellence, December 2005.

[54] M. Wieczorek, M. Siddiqui, A. Villazon, R. Prodan, and 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*, page 82, Washington, DC, USA, 2006. IEEE Computer Society.

[55] R. Wolski, J. S. Plank, J. Brevik, and T. Bryan. Analyzing Market-Based Resource Allocation Strategies for the Computational Grid. 15(3):258–281, Fall 2001.

[56] X. Xiao and L. M. Ni. Internet QoS: A Big Picture. *IEEE Network*, 13(2):8–18, Mar. 1999.

[57] J. Yu and 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)*, Paris, France, June 2006. IEEE, IEEE CS Press.

[58] J. Yu and R. Buyya. Scheduling Scientific Workflow Applications with Deadline and Budget Constraints using Genetic Algorithms. *Scientific Programming Journal*, 14(1), 2006.

[59] J. Yu, R. Buyya, and 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)*, Melbourne, Australia, Dec. 2005. IEEE, IEEE CS Press.

[60] Z. Yu and W. Shi. An Adaptive Rescheduling Strategy for Grid Workflow Applications. In *Proceedings of the 21st IPDPS 2007*, Long Beach, USA, Mar 26 -30 2007. IEEE Computer Society Press.

[61] H. Zhao and R. Sakellariou. An Experimental Investigation into the Rank Function of the Heterogeneous Earliest Finish Time Scheduling Algorithm. In *Euro-Par*, pages 189–194, 2003.

[62] H. Zhao and R. Sakellariou. Advance Reservation Policies for Workflows. In *Proceedings of the 12th International Workshop on Job Scheduling Strategies for Parallel Processing*, volume 4376 of *Lecture Notes in Computer Science*, pages 47–67, Saint-Malo, France, June 2006. Springer-Verlag.

[63] H. Zhao and R. Sakellariou. Scheduling Multiple DAGs onto Heterogeneous Systems. In *15th Heterogeneous Computing Workshop (HCW'06)*, Rhodes, Greece, April 2006. IEEE Computer Society Press.