

Constraint-Based Envelopes over Multiple Alternatives

Tatiana Kichkaylo*

USC/Information Sciences Institute
Marina del Rey, CA 90292, USA
tatiana@isi.edu

Abstract

Many real-world planning problems require sophisticated reasoning about numeric resources, including sharing of resources by different actions. Constraint-based planners offer machinery to represent complex constraints and dependencies between actions. However, such planners usually consider a single partial plan at a time, and limit themselves with finding a feasible solution. Alternatively, planning graph based planners use an aggregate data structure over multiple partial plans to guide search, which allows such planners to guarantee optimality of the solution. Typically, such planners restrict the form of supported resource expressions. In this paper we describe a planner for construction of computational grid workflows that combines these approaches. Our planner uses an envelope over multiple execution sequences represented by a constraint network to drive the search towards a good solution, while supporting expressive models for resource sharing and multi-resource reservations.

Introduction

Traditionally, in planning, resources are allocated to actions using an all-or-none policy; they cannot be shared. For example, at most one job can be executed on a machine at any given time.

In real-world planning problems, the amount of available resources and action requirements are often stated using real numbers. This more general model of resources creates additional degree of flexibility in the problem. For example, the number of jobs that can be executed on a machine in parallel depends on the resource requirements of the jobs and the amount of resources available on the machine. The situation becomes even more complicated when actions have durations and the resource availability changes over time.

In planning, the sequence of actions to be scheduled first needs to be chosen. Often, the same result may be achieved in different ways. The resource requirements of different options for achieving intermediate goals can interact in complex ways, affecting feasibility and optimality of the solution. For example, a job may require

a host with at least 16 CPUs. The execution time of such a job depends on the number of CPUs allocated for the job and their speed. Allocation of a larger number of CPUs to one job may decrease its running time, but make parallel execution of other jobs impossible.

Constraint-based planners (Smith, Frank, & Jónsson 2000; Rabideau, Engelhardt, & Chien 2000; Ghallab & Laruelle 1994) support very expressive resource models. However, such planners usually refine a single partial plan at any given moment and focus on finding a feasible solution. To produce solutions of good quality (minimum makespan, smaller resource consumption), such planners rely on domain-specific control knowledge, which needs to be provided by a human expert.

On the other hand, GraphPlan-based planners (Blum & Furst 1997; Koehler 1998; Gerevini & Serina 2002) build a data structure, called a planning graph, that aggregates information over multiple partial plans, essentially providing a logical envelope over all possibly reachable world states. Planning graphs help to find an optimal solution. Unfortunately, GraphPlan-based planners offer limited support for complex resource interactions and time-varying resource availability.

In this paper we describe GPRS (Grid Planner with Reservations and Sharing), a planner for constructing executable workflows in computational grid environments. Our planner combines ideas from planning graph and constraint-based planners. GPRS uses an envelope over multiple alternative partial plans for search guidance. To represent sharing and reservation of resources, numeric variables belonging to nodes of the envelope graph are organized in a constraint network. This allows GPRS to guide search towards an optimal solution while supporting expressive models for resource sharing and multi-resource reservations.

The rest of this paper is structured as follows. First, we present the problem of constructing grid workflows. Then, we describe the GPRS algorithm, and evaluate (i) its ability to handle expressive resource models, (ii) scalability of the planner, and (iii) quality of the solutions. We describe related work, and conclude with a discussion of limitations of our algorithm and future research directions.

*The work was done when the author was a student at New York University.

Workflow construction problem

The workflow construction problem (WCP) is the problem of designing executable workflows in computational grids. The objective of computational grids is to pool together distributed computational and storage equipment to efficiently solve computationally and data intensive tasks.

A typical grid environment consists of a network of hosts and links, which have resources, such as CPU, memory, and bandwidth. We model resource availability as a piecewise constant function of time.

A typical grid workflow consists of jobs that process files. A file has a globally unique name and a size. Each job is specified by a set of required files, a set of produced files, and resource requirements. A job can be executed on a host that has sufficient resources. The resources are occupied for the duration of the job and released upon its completion.¹ In general, the duration of the job is a function of the amount of resource available for the job.

Files can be transferred over network links, consuming link resources for the duration of the transfer. The latter depends on the size of the file and on the available network bandwidth. Note that it is possible to transfer files over multi-link paths. In this case, the duration is computed using the minimum link bandwidth along the path (path bandwidth). During the transfer, path bandwidth is reserved on all participating links, which allows high-bandwidth links to be shared between several parallel low-bandwidth transfers.

The workflow construction problem is, given a network topology and resource availability, a set of job descriptions, and an initial allocation of files on hosts, construct an executable workflow (execution plan) consisting of job executions on hosts and file transfers over paths such that a given file is obtained on a given network host as soon as possible. Constructing an optimal executable workflow may require resource sharing and trading off computation and communication.

The WCP can be viewed as a planning problem, where job executions and file transfers correspond to actions, and resource and file availability describe the state of the world.

In the example shown in Figure 1, the goal is to obtain a small file **Res** on host 3. **Res** can be computed in 1 time unit given two files **pA** and **pB** (for the purpose of this example we assume that the job duration is the same for all hosts). Each of these files can be computed in 2 time units from files **rA** and **rB** respectively. All four files, **pA**, **pB**, **rA**, and **rB**, are present in the network. Each of these files has size 100 units. The size of the **Res** file is 10 units.

Given the available link bandwidth, the optimal strategy (Figure 2) is to transfer **pA** from the remote host over path 1-2-3-4 (action **trpA**), recompute **pB** us-

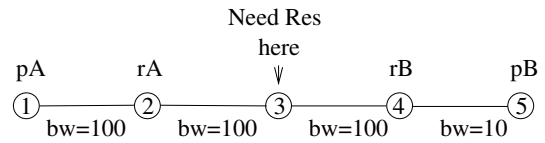


Figure 1: Bandwidth (bw) and file availability.



Figure 2: Solution to the problem shown on Figure 1 that trades off computation and communication.

ing the **rB** file (action **jobB**), compute **Res** on host 4 (**jobC**), and transfer the final result to the destination host (**trRes**). Note that, if the execution of the component producing the final result is co-located with the recomputation of **pB**, the time used to transfer **pA** overlaps with the computation of **pB**.

Finding this optimal solution requires reasoning about resource reservations and sharing, as well as an ability to compare different possible solutions.

GPRS

The Grid Planner with Reservations and Sharing (GPRS) combines the ideas of planning graph based algorithms and constraint propagation. GPRS builds an envelope over possible execution sequences to obtain completion time estimates for search guidance. Nodes of the envelope graph, which correspond to actions and propositions, may have multiple variables associated with them. The values of these variables are propagated using a constraint network. Currently, the plan extraction phase of GPRS is based on critical path scheduling.

Envelope graph

The purpose of the envelope graph is to obtain lower bound estimates on the completion time of various parts of the computation to guide the search towards the best (fastest) ways of achieving goals.

The graph has two types of nodes: **AND nodes** correspond to actions (job executions, file transfers), and **OR nodes** to propositions. In planning for computational grids, propositions describe the availability of a file on a network host.

A typical goal of a grid application is to produce a particular file on a particular host. While the size of such a goal is usually small (e.g., one file-host pair), the total amount of information about the state of the network and resource availability may be large. Therefore, GPRS constructs the envelope graph using regres-

¹Some resources, such as quotas for computing time in supercomputer centers, are consumed, rather than occupied and released. Currently, GPRS does not deal with these.

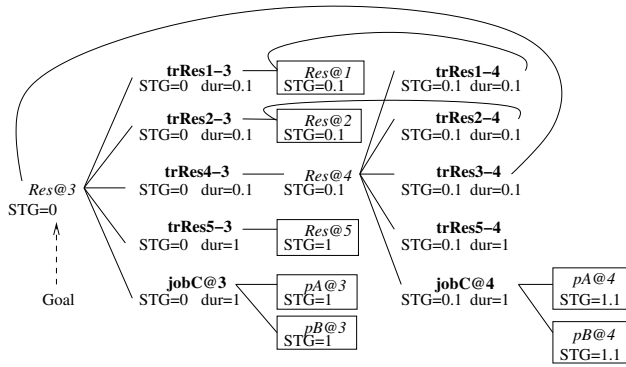


Figure 3: Part of the logical envelope graph. Bold font shows actions, italics shows propositions. Boxes correspond to leaf nodes. Numbers under nodes show shortest time to goal (STG) and action duration.

sion from the goals, which allows the planner to identify and bring in necessary information during the search. Figure 3 shows the initial phase of construction of the envelope graph for the problem shown in Figure 1.

For each OR node n_o , the **support** of n_o is a set of AND nodes corresponding to actions that can achieve the proposition of the node n_o . For example, availability of file *Res* on host 3 *Res@3* is supported by four actions for transferring the file from different network hosts and one job execution action. For each AND node n_a , the support of n_a is a set of OR nodes corresponding to preconditions of the operator of node n_a . Thus, action *jobC@3* is supported by availability of files *pA* and *pB* on host 3 (*pA@3* and *pB@3*).

Both AND and OR nodes can support multiple sinks. For example, *Res@2* supports *trRes2-3* and *trRes2-4*. Such reuse of nodes makes the envelope a general graph (with cycles) and reduces memory requirements for envelope construction.

Each of the nodes of the envelope graph may have several variables associated with it, including the *Earliest Completion Time* (ECT) and the *Shortest Time to Goal* (STG) variables. ECT corresponds to the lower bound on time for completing the current node after the start of the workflow. STG is the lower bound on time to reach the goal after completion of the current node. The planner optimizes the makespan of the computation and uses ECT variables to guide the search. The STG variables are used to order constraints for propagation as described below.

A **support tree** is defined recursively as a single support node for each proposition (OR) node and all support nodes for an action (AND) node.² The **best tree** of the envelope graph is a support tree in which the cheapest support node is chosen for every proposition. The envelope graph is expanded until the best tree is completely rooted in the initial state.

²Strictly speaking, a support tree is not necessarily a tree, but a DAG.

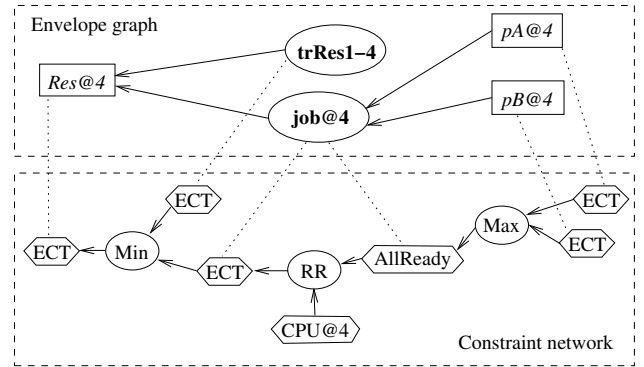


Figure 4: Envelope and constraint network. Dotted lines connect variables (hexagons) to the nodes they belong to.

Only leaf nodes belonging to the best tree are expanded. After expansion of a leaf node in the regression way, constraint propagation is performed, which can improve the lower bound estimates of the ECTs of nodes and change the set of nodes forming the best tree. Because of this non-uniform expansion of the envelope graph, GPRS essentially performs best-first search in the space of support trees (as opposed to GraphPlan (Blum & Furst 1997), which extends all branches of the envelope simultaneously).

Note that the envelope and the corresponding constraint network are optimistic in that they can miss dependencies between different branches. For example, logically independent jobs may compete for the same resources and, therefore might have to be scheduled sequentially rather than in parallel. Because of this optimism, the best tree of the envelope graph does not necessarily correspond to a valid solution, and additional expansion may be initiated during solution extraction.

Constraint propagation

Nodes of the constraint network, underlying the envelope graph of GPRS, may correspond to variables of the envelope nodes, such as earliest availability time of files on hosts (ECT for files), resource variables, and artificial variables.

Variables and constraints. For the ECT variables, the value of a variable is a number describing the current lower bound estimate of the completion time. Values of the resource variables may be represented as piecewise constant functions describing the maximum available levels of the resource as a function of time. Ultimately, the type of values variables can have is limited only by the constraint-propagation algorithm used.

Actions of a plan can be executed sequentially or in parallel. In both cases, the dependencies between the actions are represented using constraints.

For example, an execution of a job on a host can only start when all required files are available on that host.

This dependency is represented as follows (Figure 4). An artificial variable *AllReady* is created, which is connected to the ECT variables of all nodes corresponding to the required files using a Max constraint. The ECT variable of the job node is connected to the *AllReady* variable using a Resource Reservation (RR) constraint, which also takes into account resource availability. The reservation constraint directly computes the earliest completion time of the job given resource availability profiles, file availability time (*AllReady*), and an expression describing running time of the job as a function of available resources. The default implementation of the RR constraint searches for the earliest moment after *AllReady* when all required resources simultaneously satisfy job requirements for the duration of the job. Alternative implementations and types of constraints can be provided by the user to encode resource reservation policies specific for job types and capabilities of host operating systems/schedulers.

Actions that can be executed in parallel may affect each other via shared resources. The envelope of GPRS is built over multiple possible execution sequences, and not all actions that are included in the envelope, or even in its current best tree, would necessarily be incorporated in the final plan. Therefore, GPRS takes an optimistic approach. Only reservations of nodes chosen during solution extraction to be a part of the solution (committed nodes) affect resource availability visible to other actions. In other words, a resource can be promised to several actions at the same time. This can cause suboptimality of a solution.

Scheduling of constraint propagation. Constraint propagation is performed after every change to the envelope graph: expansion of a leaf node or commitment of a node during solution extraction. Since constraint propagation is the most frequent and expansive operation, it is desirable to make it as fast as possible.

GPRS requires that every variable belongs to at most one envelope node (resource variables are independent), and every constraint changes values of variables of a single node. The latter restriction can be enforced by creating artificial variables. The nodes of the envelope graph, in addition to ECT variables, also own Shortest Time to Goal (STG) variables (Figure 3), which are lower bounds on the completion time of the plan after completion of execution of the node. The values of STG variables are obtained by simple summation of lower bounds of action durations during the expansion of the envelope graph.

During the constraint propagation, constraints are scheduled in the order of decreasing STG of the nodes whose variables they affect. This technique permits almost loop free constraint propagation. (Some constraints may be executed more than once during the same propagation episode because of the loops in the envelope graph).

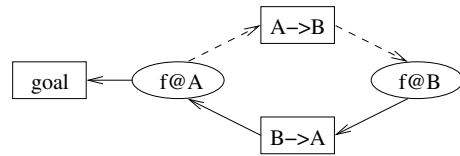


Figure 5: Envelope graph with an infinite loop.

Loop breaking. The ECT variables are lower bound estimates of the actual completion time. During construction of the envelope graph, the values of the ECT variables of leaf nodes are considered zero.

The envelope graph can contain loops, for example, when the same file *f* may be transferred back and forth between two network hosts A and B, and no job can produce this file (Figure 5). Expansion of a leaf node, e.g. *f@B* in our example, which increases the ECT of that node, may cause infinite loops in constraint propagation along graph cycles.

To avoid this, before expansion, the ECTs of all nodes reachable from the leaf node being expanded are set to infinity. Subsequent constraint propagation can only decrease the bounds, when possible. In case of an infinite loop, such as the one shown in Figure 5, the lower bounds of ECTs of unreachable nodes will remain infinity, and the constraint propagation will terminate after checking each node only once.

Solution extraction

The envelope graph is expanded until the best tree has all its leaves true in the initial state. The envelope graph is optimistic, because it can miss some interactions between different branches of the application DAG. Therefore, the best tree of the envelope graph does not necessarily represent a valid plan.

To construct a valid plan, we use critical path scheduling for the final solution extraction phase of the GPRS algorithm. A **critical path** is a path in the best tree leading from the root (the goal node) to a leaf, which chooses the most expensive support node for each AND node.

The planner **commits** nodes of the critical path starting from the leaf. Committing a node involves fixing the start time of the action and making all resource reservations belonging to the node permanent. Such commitment of reservations is possible as long as the constraint network is quiescent.

As a result of committing resource reservations, the availability of involved resources may change, which may affect other reservation constraints that involve the same resources, and, via constraint propagation, completion times of various nodes of the envelope graph. The change of ECT values of the graph nodes may change the portion of the graph considered to be the best tree and therefore result in further expansion of the envelope graph.

The current implementation of GPRS does not support backtracking. Once a node is committed, the com-

mitment cannot be revoked. In addition, when a leaf node is committed, all nodes of the critical path leading to this leaf node get frozen. The latter means that, although the values of variables of those nodes, including ECT, may change, the frozen nodes are guaranteed to be a part of the solution. Because the envelope graph is optimistic, such a non-backtracking nature of the solution extraction phase may result in suboptimal solutions. Moreover, the non-backtracking algorithm is incomplete and may fail to find a solution in the presence of budget restrictions (quotas on resource usage and/or deadlines).

On the other hand, the non-backtracking solution extraction phase is fast. Moreover, the use of the envelope graph during the solution extraction phase allows the planner to make continuous adjustments to the best tree including rescheduling actions and replacing whole subtrees. This flexibility usually leads to good quality of solutions, and appears to compensate in practice for the theoretical incompleteness of the algorithm.

Evaluation

In this section we evaluate the ability of GPRS to handle the expressiveness of the model of grid applications with explicit resource reservations and sharing, the scalability of the planner with respect to the network and application size, and the quality of solutions produced by the planner.

Handling expressiveness

A planner for computational grids needs to reason about sharing of numeric resources between jobs and data transfers running in parallel. The planner also needs to reason about action durations and start and completion times in the presence of time-varying resource availability. Finally, the planner needs to be able to select different options, such as file replicas or job types capable of producing a given data product, and trade off computation and communication so as to minimize the total duration of computation. To check if GPRS can correctly handle these tradeoffs, we ran the following experiment, which exercises the features listed above.

The abstract structure of the application is shown in Figure 6. The application consists of three jobs, organized in two levels. Each of the two jobs of the first level requires three input files, produces two output files, and takes 100 time units to complete. The third job requires four files and produces one file in 40 time units. The size of files *a*, *b*, *c*, *d*, and *e* is 500 units. Files *f*, *g*, *k*, and *q* are 100 units each. The final result file *r* has size 1000. Note that file *c* is required by two jobs, and file *k* can be produced by two different jobs. In the latter case the semantics is that the file *k* will contain exactly the same data regardless of how it was produced.

The network structure for our problem is shown in Figure 7. Replicas of several files are available on different network hosts as shown in the figure. The band-

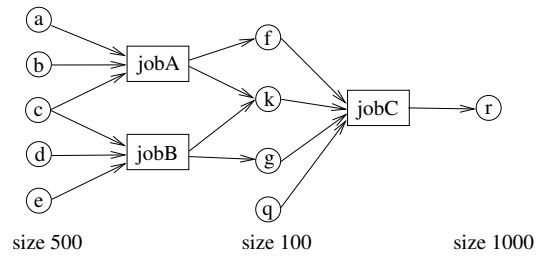


Figure 6: Abstract structure of a grid application. Circles represent files, and rectangles jobs.

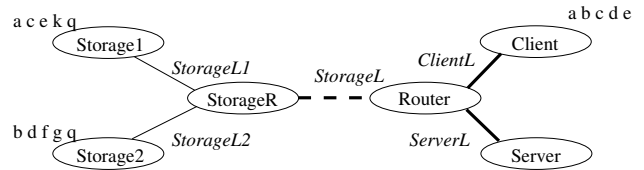


Figure 7: Network structure. Link names are shown in italics. Availability of file replicas is shown in normal font next to the hosts. Width of the lines corresponds to the link bandwidth. The dashed line shows the link, whose availability varies over time.

width of the links connecting storage nodes to the storage router is at most 5; the bandwidth of all other links is at most 10 units. We assume that only host *Server* can perform computation.

We further assume that the availability of the *Server* host and the availability of the link *StorageL* between the main *Router* and the storage router *StorageR* vary with time due to reservations from other ongoing computations. The availability windows for network resources are shown in Figure 8.

The goal of this problem is to obtain the *r* file on host *Client* as quickly as possible.

The plan found by GPRS is shown in Figure 9. This plan has the optimum duration given the resource availability. Note that because of the limited availability of both computational and link resources, the planner decided to recompute two of the intermediate data products and fetch the other two from where they are stored. The replica of file *q* is chosen so that the transfers of

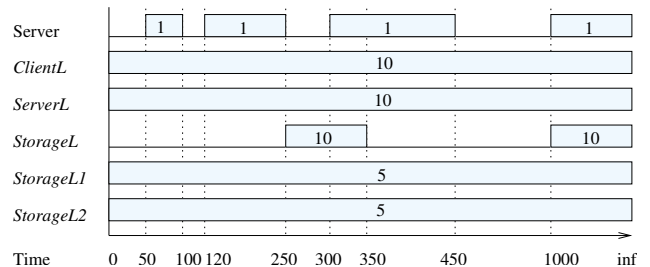


Figure 8: Resource availability. Numbers in the bars show the amount of the resources.

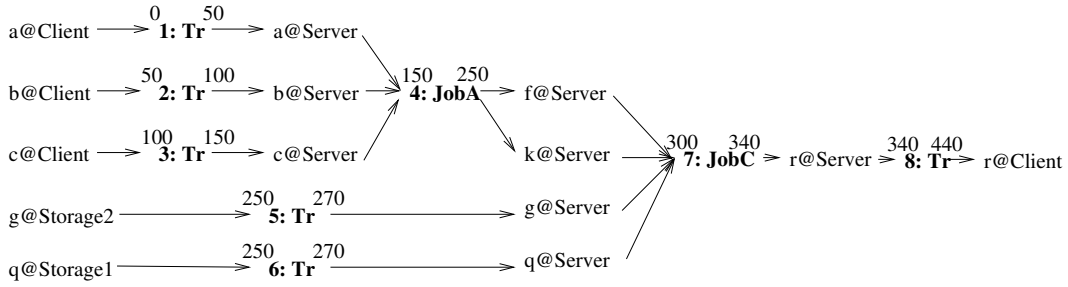


Figure 9: The final plan. File nodes describing availability of a file replica on a host are shown in normal font. Actions are shown in bold and preceded by a sequence number. The numbers above each action node are the start and end time of that action. The start time of an action depends on the earliest availability of the required files and on the resource availability.

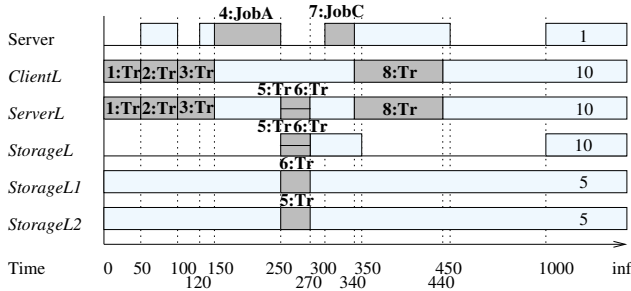


Figure 10: Resource reservations by actions. Multiple resources may be simultaneously used by a single action, and several actions can concurrently use the same resources (e.g. link *StorageL*).

files *g* and *q* can be done in parallel.

Before any commitments were made, GPRS considered execution of both jobs A and B to be the fastest way to obtain intermediate files. However, committing the decision to execute job A changed availability profile of *Server*. As a result of subsequent constraint propagation, the structure of the best tree changed to include file transfers from the storage hosts.

The planner also correctly handles multiple reservations and sharing of resources (Figure 10). For example, transfer of file *g* from *Storage2* to *Server* requires simultaneous reservation of bandwidth of three links. Two of these links (*StorageL* and *ServerL*) are used for transfer of file *q* from *Storage1* at the same time.

Performance

To evaluate the scalability of the planner we used parameterized synthetic applications and networks, whose structure matches that of typical grid workflows and environments.

Figure 11 shows the network used in our experiments. This network consists of C clusters each containing N computational hosts and one router, which connects the cluster to the central master router. Hosts in a cluster can be thought of as supercomputers in a supercomputer center. Hosts within a cluster are fully connected.

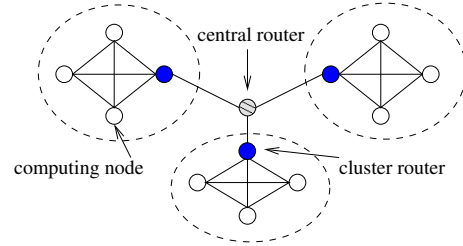


Figure 11: Synthetic network used in performance evaluation.

In total, the network contains $(N + 1) \times C + 1$ hosts, of which $N \times C$ can perform computation.

Figure 12 shows the application kernel used in the experiments. The structure of this kernel is modeled after existing grid applications such as Montage (Berriman *et al.* 2003). This application kernel is parameterized, which allows us to analyze scalability of the planner with respect to different properties of the application.

The application consists of S segments limited by the splitting and merging components. The portion of the segment between these components contains W parallel execution sequences, each consisting of H processing jobs. An instance of this application contains $(H \times W + 2) \times S - 1$ jobs and $((H + 1) \times W + 1) \times S$ files. The splitting and merging components serve as synchronization and data aggregation points. Jobs of a typical workflow can take tens of minutes to execute on large supercomputers, so it is feasible to invest up to several minutes of time on a workstation to construct and optimize a workflow.

In our experiments, we varied values of C , N , S , H , and W . In all cases, the goal is to achieve availability of the merged file of the last segment on the first computational host of the first cluster. Initially, all intermediate files of the first level of the first segment are available in the network. These initial files are distributed sequentially to all computing nodes of the network.

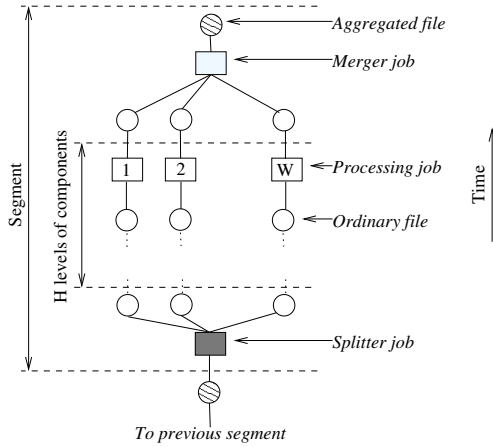


Figure 12: Application kernel.

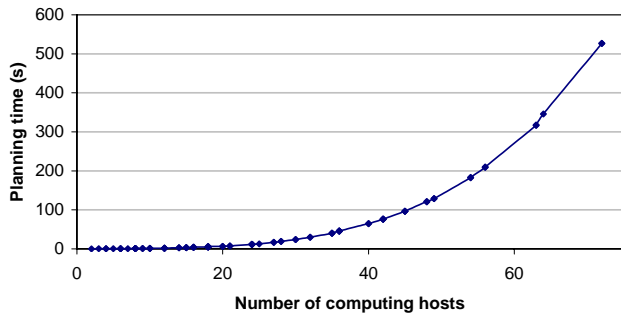


Figure 13: Scalability of GPRS wrt the network size.

Scalability with the network size. To evaluate scalability of GPRS with respect to the size of the network, we run the planner for a set of networks with parameters $C \in \{1..9\}$, $N \in \{2..9\}$ using the kernel application with $S = 1$, $H = 3$, $W = 5$. Figure 13 shows planning time as a function of the number of computing hosts in the network. As can be seen from the figure, the planning time grows fast. This can be explained by the fact that, to support simultaneous reservations of groups of network links, the planner considers all hosts of the network as targets for file transfers. The number of hosts contributes to the branching factor of the search space. The fact that the algorithm still scales to networks of considerable size can be explained by the pruning power of the envelope graph.

Scalability with the width of workflow. Next we evaluated scalability of GPRS with respect to each of the three parameters affecting the size of the workflow. Figure 14 shows planning time as a function of the total number of files in the generated workflow for the following experiments: $\{C = 2, N = 2, S = 1..36, H = 1, W = 5\}$, $\{C = 2, N = 2, S = 1, H = 2..100, W = 5\}$, $\{C = 2, N = 2, S = 1, H = 3, W = 5..125\}$.

As the results demonstrate, the planning time grows more than quadratically with the depth of the work-

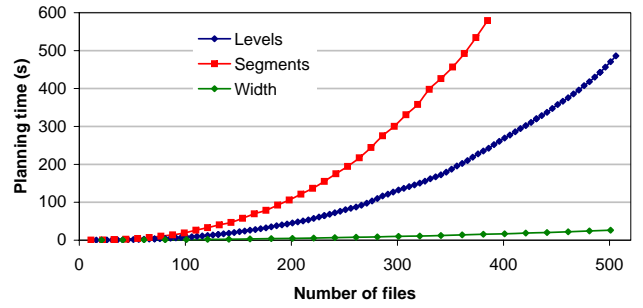


Figure 14: Scalability of GPRS wrt the size of the workflow.

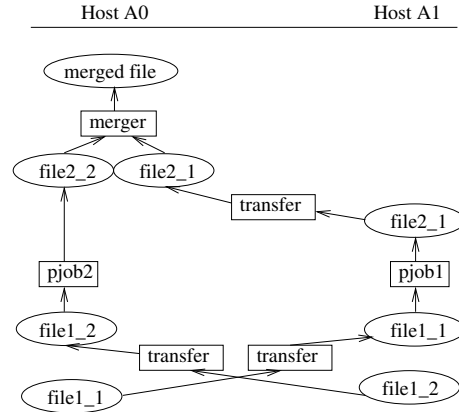


Figure 15: Suboptimal plan generated by GPRS.

flow. This can be explained by the fact that in the current implementation a complete constraint propagation is performed after every action choice during the solution extraction phase. The complexity of this propagation is close to linear with respect to the width of the workflow, but grows faster with respect to the depth of the workflow. We expect that a different (lazy) implementation of constraint propagation would lead to significant speedup of the algorithm. Note, however, that grid workflows tend to have few stages, and therefore scalability of the planner with respect to the depth of the workflow is less important than that with the number of parallel execution sequences.

Solution quality

Due to the use of the critical path scheduler with commitments for plan extraction, the plans found by GPRS may be suboptimal. For example, Figure 15 illustrates the plan produced by GPRS for the problem with $\{C = 1, N = 2, S = 1, W = 2, H = 1\}$.

The solution shown in the figure contains two file transfers more than the optimal solution. In this example, the scheduler made a greedy decision about scheduling the longest path of the workflow first, and then had to schedule the rest of the workflow using the remaining resources.

Despite possible suboptimality, GPRS still performs

good load balancing. It is problematic to find an exact optimum for problems with reasonable size. However, it is easy to check optimality of the solution in some special cases.

To assess the quality of solutions, we asked the planner to find a configuration of the kernel application described above with one segment with one job level of width 300 for a network with 2 clusters with 4 computing hosts each. We set the link bandwidths to a very high value, so that delays introduced by data transfers are negligible.

This application contains the total of 301 jobs, which can be executed on any of the 8 computing hosts. GPRS assigned 37 jobs to each of the hosts of the B cluster, 38 jobs to three hosts of A cluster (A1,A2, and A3), and 39 jobs to host A0. Since A0 is the host where the final answer was requested, the job assignment is indeed optimal.

The load-balancing effect can be explained by the fact that all decisions made by the scheduler are immediately taken into account by the envelope graph. The envelope is built over all possible execution sequences, and at any moment chooses the best way to achieve every subgoal given the current set of resource reservations.

Related work

The workflow construction problem searches for an optimum solution in the presence of complex (numeric) dependencies between actions. In designing an algorithm for solving such a problem, two decisions need to be made. First, action selection and resource allocation may be done simultaneously or separately. Second, during the search, the planner can consider each possible refinement of a plan separately, or perform aggregation using envelope-like structures.

Planners that reason about numeric resources during action selection typically support only limited form of functions (Koehler 1998; Refanidis & Vlahavas 2000). Separating these concerns allows Pegasus planner for the WCP (Blythe *et al.* 2003) to achieve good scalability. However, planners that separate action selection from resource reasoning may perform poorly in resource-driven domains (Srivastava 2000). This may be important, for example, when data transfers and job executions have comparable durations.

The second choice, single partial plan vs. envelope over multiple options, affects quality of the solution. Although single-plan algorithms may be faster, they rely on domain-specific control knowledge to guide the search (Blythe *et al.* 2003). Considering several options simultaneously helps to drive the search towards an optimum solution (Kichkaylo, Ivan, & Karamcheti 2004).

This paper presents an algorithm that uses constraint networks to construct envelopes over multiple alternative plans. This approach allows GPRS to combine the benefits of the harder options for both of the above choices with a reasonably small overhead.

Discussion and future work

Even when using a greedy critical path scheduler, GPRS still produces high quality plans. We believe that the reason for this is the use of the envelope over multiple sequences for search guidance. Every time a decision is committed to be a part of the plan, corresponding changes of the resource availability are propagated through the envelope. This may cause the change of the best tree towards the optimal solution given all committed decisions.

We believe that the planner's performance and the quality of the solution can be further improved. One possibility is to add explicit constraints between parallel execution sequences relying on the same resources. Adding such constraints will incur computational overhead. Whether or not this overhead would be justified by the improvements in the performance and solution quality is a topic for future research.

Another idea is to propagate the total resource requirements of each subtree (in addition to time) as it is done in (Kichkaylo, Ivan, & Karamcheti 2004).

Finally, it would be interesting to see how support for non-replenishable resources, e.g. quotas, which can be implemented using backtracking, affects performance of the planner.

References

- Berriman, G. B. *et al.* 2003. Montage: A grid enabled image mosaic service for the national virtual observatory. In *ADASS*.
- Blum, A., and Furst, M. 1997. Fast planning through planning graph analysis. *Artificial Intelligence* 90(1-2):281-300.
- Blythe, J.; Deelman, E.; Gil, Y.; Kesselman, C.; Agarwal, A.; Mehta, G.; and Vahi, K. 2003. The role of planning in grid computing. In *ICAPS*.
- Gerevini, A., and Serina, I. 2002. LPG: a planner based on local search for planning graphs. In *AIPS*.
- Ghallab, M., and Laruelle, H. 1994. Representation and control in IxTeT, a temporal planner. In *AIPS*.
- Kichkaylo, T.; Ivan, A.; and Karamcheti, V. 2004. Sekitei: An AI planner for constrained component deployment in wide-area networks. Technical Report TR2004-851, New York University.
- Koehler, J. 1998. Planning under resource constraints. In *ECAI*.
- Rabideau, G.; Engelhardt, B.; and Chien, S. 2000. Using generic preferences to incrementally improve plan quality. In *AIPS*.
- Refanidis, I., and Vlahavas, I. 2000. Heuristic planning with resources. In *ECAI*.
- Smith, D.; Frank, J.; and Jónsson, A. 2000. Bridging the gap between planning and scheduling. *Knowledge Engineering Review* 15(1).
- Srivastava, B. 2000. Realplan: Decoupling causal and resource reasoning in planning. In *AAAI*.