Scalable Grid Resource Allocation for Scientific Workflows Using Hybrid Metaheuristics

Georg Buss, Kevin Lee, and Daniel Veit

Dieter Schwarz Chair of Business Administration, E-Business and E-Government
University of Mannheim, Schloss
Mannheim, Germany
{buss,lee,veit}@bwl.uni-mannheim.de

Abstract. Grid infrastructure is a valuable tool for scientific users, but it is characterized by a high level of complexity which makes it difficult for them to quantify their requirements and allocate resources. In this paper, we show that resource trading is a viable and scalable approach for scientific users to consume resources. We propose the use of Grid resource bundles to specify supply and demand combined with a hybrid metaheuristic method to determine the allocation of resources in a market-based approach. We evaluate this through the application domain of scientific workflow execution on the Grid.

1 Introduction

Many applications in science involve complex, large-scale and computationally intensive tasks which require a vast amount of computational resources provided in a globally distributed way. Computational Grids [1] allow resources from geographically distributed providers to be utilized by a single user to perform a single computational task. There is however a disconnect between the highly technical skills often necessary to utilize Grid resources and a scientists focus on their research domain. This is potentially limiting the use of Grids by scientific users. The two main challenges to solve this problem are i) improvements to high-level tools and task description languages [2], and ii) market structures to enable providers and users the ability to dynamically trade resources based on the principle of utility maximization [3].

A substantial amount of research has been undertaken in the area of high-level support for scientific usage of Grid resources, but there is of yet little practical support for market based resource allocation for end users. Current approaches to exchange Grid resources generally involve complex contractual relationships between institutions with grid resources; because of this there is no open market for trading grid resources between providers and users. This also makes it difficult for users to seek out and get, based on their valuation, access to specific resources for short term usage. More recently Cloud Computing has emerged [4], allowing rapid access to resources, but with very static pricing structures.

A promising solution to this problem of dynamic resource markets for Grids is the use of bundles to describe the resources offered or needed. These resources (such as processors, RAM or storage) are characterized by attributes (such as speed or size) and

complex scheduling requirements (such as uptime and required time spans) as well as a valuation or reservation price. Viewing Grid resource as complex 'bundles' allows their availability and requirement to be described and traded in an open market. Taking such an approach, the matching of providers and consumers of resources can take place in a open market using combinatorial exchange mechanisms [5].

The trading of Grid resources via an exchange mechanism requires a centralized institution. Users submit bundles of required resources which are matched with provider bundles of available resources. Previous approaches formulated this problem as a generalization of the $\mathcal{NP}$-complete combinatorial allocation problem [6] which is found to be inapproximable within given bounds[7]. In this paper we propose the use of hybrid-metaheuristics as a scalable approach to solve the winner determination problem in Grid resource trading to allow scientists to access resources though a marketplace.

The remainder of this paper is structured as follows. Section 2 introduces scientific workflow execution in the Grid to motivation our approach. In Section 3 we introduce a formal description of the problem of trading grid-based resources. Section 4 describes a solution to the defined problem using a hybrid-metaheuristics approach. Section 5 evaluates the proposed solution. Finally, Section 6 concludes and discusses future work.

2 Workflow Execution on the Grid

Complex scientific applications are typically modeled as a workflow [2] which describes the tasks to be performed, their dependencies as well as the data and computations required to be completed. To enable as wide a range of scientists as possible to utilize the powerful resources of global Grids, application workflows can be described in a high level logical or abstract way, without references to concrete resources. This allows scientists to concentrate on efficiently describing the application rather than how it is executed. Workflow engines such as Taverna[8], Triana[9] and Pegasus[2] convert abstract workflows to an executable workflow which can then be executed on the Grid.

To convert an abstract workflow to one that is executable, the workflow has to be refined by mapping it to actual grid-based computational and storage resources. The workflow engine performs a number of steps to produce a concrete workflow including finding the actual input files, optimizing the workflow structure, finding resources and generating grid submission configurations. For example, the Pegasus workflow engine [2] queries the Globus Monitoring and Discovery Service (MDS) [10] to determine the number, availability, characteristics of Grid resources.

As yet, workflow engines do not commonly make decisions to schedule resources based on the economic impact of those decisions, even though the resources in question are of great value and in great demand. In this paper we introduce a scalable trading mechanism which can be used to enhance the mapping process of workflow engines, such as the Pegasus Site-Selection [2] stage. Rather than attempting to minimize execution time [11], we propose a market-based site-selection mechanism that determines an allocation based on the valuation for resources.

In our approach, we assume that idle resources are announced to a grid monitoring service. These resources can then be offered to resource consumers as reserved instances through a trading mechanism. Resources are offered or requested as bundles

consisting of a number of parameters [5]. A resource offer or request is structured by the amount of required RAM, disk space and grid worker nodes. Requested hard disc space can be aggregated from various resources sellers, however, RAM and worker nodes have to be provided by a single seller. To allow the trading of resources, we assume that the Grid monitoring service runs the exchange mechanism collecting resource supply and demand requests. Requests timeout after a given time frame for a predefined amount of periods allowing reserved resources instance to be cleared.

3 Grid Resource Trading

To describe the problem of Grid resource trading we take the multi-attribute combinatorial exchange mechanism described in [5]. The set $G = \{g_1, \ldots, g_{|G|}\}$ specifies the computational resources available in the exchange mechanisms where G denotes all the goods to be traded in a trading market and a g_k is a specific resource. A bundle S_i denotes a subset of all the resources in G. Therefore the set $S = \{S_1, \ldots, S_{|S|}\}$ of bundles covers all the possible subsets of G. A computational resource g_k itself is defined by a set of cardinal quality attributes $A_k = \{a_1, \ldots, a_{|A_k|}\}$.

The exchange mechanism which is for the workflow scenario run by the grid monitoring service allows both the sellers and buyers of resources to place blind orders. Buyer orders specify what resources are required, and seller orders specify what resources are available. Potential buyers n out of the set $N = \{n_1, \ldots, n_{|N|}\}$ of buyers are allowed to submit an order of multiple bundle bids $B_n = \{B_{n,1}(S_1) \oplus \ldots \oplus B_{n,u}(S_i)\}$. The respective bundle bids are XOR concatenated.

$$\begin{aligned} B_{n,f}(S_i) = \{ & \langle v_n(S_i), s_n(S_i), e_n(S_i), l_n(S_i), \\ & q_n(S_i, g_1, a_{g_1,1}), \ldots, q_n(S_i, g_G, a_{g_G,A_j}), \\ & \gamma_n(S_i, g_1), \ldots, \gamma_n(S_i, g_G), \\ & \varphi_n(S_i, g_1, g_2), \ldots, \varphi_n(S_i, g_G, g_{G-1}) \rangle \} \end{aligned}$$

The valuation $v_n(S_i)$ is the amount the buyer is willing to pay for the bundle S_i per time slot. The number of slots the resources are required for is given by $s_n(S_i)$. A buyer bid defines a period of time slots within which the required slots have to be allocated. The period is given by $e_n(S_i)$ for the earliest possible time slot and $l_n(S_i)$ for the latest possible time slot. The minimum quality of the resources g_k contained in a bundle bid S_i is specified for each resource attribute a_{g_k,A_j} by $q_n(S_i, g_k, a_{g_k,A_j})$. In addition bundle bids may contain two types of fulfillment constraints. A coupling constraint $\gamma_n(S_i, g_1)$ specifies the maximum number of sellers allowed to allocate a required resource g_k. The co-allocation $\varphi_n(S_i, g_k, g_j)$ constraint requires a pair of resources g_k, g_j to be allocated from the same single seller.

Following the requirements for workflows (Section 2) the set of resources for the application domain is given as $G = \{$RAM, disk space, worker nodes$\}$. Each of these resources is described by a single attribute: $A_{RAM} = \{$Megabyte$\}$,$A_{HD-Space} = \{$Megabyte$\}$,$A_{Nodes} = \{$Number of nodes$\}$. For the scenario envisioned buyers request a certain quality of bundle S_1. The bundle S_1 is defined as $S_1 = \{$ RAM, HD-Space, Nodes$\}$; it contains all the types of resources specified. There is no splitting

constraint present as the HD-Space can be aggregated from the whole set of seller bids. The assumption that RAM and the number of nodes have to be allocated from the same seller is modeled buy the coupling constraint $\varphi_n(S_1, g_{RAM}, g_{Nodes})$. A buyer submits a request for a single bundle S_1 which matches her needs exactly.

Potential sellers m out of the set of $M = \{m_1, \ldots, m_{|M|}\}$ may submit an order of multiple bundle bids $B_m = \{B_{m,1}(S_i) \vee \ldots \vee B_{m,u}(S_i)\}$. The bundle bids are OR concatenated. Any number of seller orders may be part of the final allocation. A single seller bundle bid is of the form:

$$B_{m,f}(S_i) = \{\langle r_m(S_i), e_m(S_i), l_m(S_i), \\ q_m(S_i, g_1, a_{g_1,1}), \ldots, q_m(S_i, g_G, a_{g_G,A_j}), \rangle\}$$

The reservation price $r_m(S_i)$ specifies the minimum price a seller is willing to sell the specified bundle of resources per time slot. It is assumed that a seller bid is valid for the range of time slots given by $e_m(S_i)$ and $l_m(S_i)$. The quality of the resource services g_k is given by $q_m(S_i, g_k, a_{g_k,A_j})$. For the purpose of modeling the domain of workflows, sellers are able to submit offers in terms of two types of bundles. This is the bundle S_1 which was introduced above. The second bundle S_2 models an offer of HD-Space only ($S_2 = \{\text{HD-Space}\}$). A seller is free to submit an offer for both or one of the bundles either. This description includes free disposal (buyers do not care about taking extra units, sellers do not care about keeping units of winning bids) except when resources are coupled.

Given a collection of buyer and seller bundle orders the multi-attribute winner determination problem is to identify a set of winning bids out of the total set of bids. An optimal set of winning buyer and seller bids determines an allocation that maximizes the overall surplus while meeting time, capacity, coupling and co-allocation constraints. An allocation is described by the variables $x_{n,t}(S_i) \in \{0, 1\}$ and $y_{n,m,t}(S_i) \in [0, 1]$. The binary variable $x_{n,t} = 1$ if buyer n is allocated bundle S_i in time slot t. The real valued variable $y_{m,n,t}$ denotes the percentage of bundle S_i allocated from seller m to buyer n in time slot t. The surplus of an allocation is given by:

$$(x, y) \in arg\ max \left(\sum_{n \in N} \sum_{S_i \in S} \sum_{t \in T} v_n(S_i) x_{n,t} - \sum_{m \in M} \sum_{n \in N} \sum_{S_i \in S} \sum_{t \in T} r_m(S_i) y_{m,n,t} \right. \\ \left. | (x, y)\ is\ a\ feasible\ allocation \right)$$

4 Heuristic Solutions to the Grid Resource Trading Problem

Solving the problem with a standard solver optimally becomes computationally intractable for small problems of realistic size [5]. To tackle the problem we propose the use of local search based [12] hybrid metaheuristics. We focus on integrative hybrid metaheuristics that incorporate an exact algorithm into a metaheuristic to solve subproblems to optimality [13]. Section 4.1 details the problem representation as well as the operators which define the neighborhood structure and Section 4.2 presents solutions based on hybrid metaheuristics.

4.1 Problem Representation

A problem instance is represented as depicted in figure 1. The buyer orders are split up into the single bundle bids $B_{n,f}(S_i)$. The single buyer bundle bids are stored in an ordered set B_b. For each buyer bid $B_{n,f}(S_i)$ a list of possible time slots t is kept. For each of these time slots the available seller bundle bids $B_{m,f}(S_i)$ are stored in the set $B_{s,t}(S_i)$. The overall idea is to reduce the $|n| : |m|$ allocation problem to be scheduled into a given number of time slots to several $1 : |m|$ allocation problems to be solved for a single time slot t. The $1 : |m|$ allocation problem for a given buyer bundle bid $B_{n,f}(S_i)$ and a given time slot t can be formalized as follows:

$$y \in arg\ max \left(v_n(S_i) - \sum_{m \in M} \sum_{S_i \in S} r_m(S_i) y_{m,n} \ \ | y\ is\ a\ feasible\ allocation \right)$$

In case of no coupling or collocation constraints the problem becomes a linear, continuous, optimization problem. This type of problem can be solved efficiently by a linear programming solver. If coupling or co-allocation constraints are present the problem is of combinatorial nature with complexity reduced significantly compared to the original $|n| : |m|$ allocation problem.

The problem representation is evaluated by passing through the sequence of buyer bundle bids starting with the first bundle bid. The time slots the buyer bid is valid for $(e_n(S_i), l_n(S_i))$ are checked in the given order. A check of a time slot requires solving the, possibly constrained, $1 : |m|$ allocation problem. As soon as the amount of computational resources requested is available for a sufficient number of time slots the buyer bid is included into the allocation and the construction process is continued with the next buyer bid. A check of a time slot is valid only if there is a valid solution to the $1 : |m|$ allocation problem. Therefore no infeasible solution can be encoded by the problem representation. In case the buyer is already part of the allocation with another bid (XOR constraint) the evaluation of the specific bid is skipped and the process is continued with the next bid in the sequence. The evaluation process is summarized in algorithm 1. To solve the $1 : |m|$ problem we use the optimization engine lpsolve. Solving the problem to optimality covers the search of a large number of solutions.

Local search based metaheuristics start from an initial point in the search space and aim to iteratively improve on the current solution [12]. Solutions of higher quality are identified by the evaluation of the solutions within the neighborhood of the current solution. We introduce two operators, $\mathcal{N}_1(B_b)$ and $\mathcal{N}_2(T(B_{n,f}(S_i)))$, spanning the neighborhood structures. Both functions are applied to an ordered set and are defined as a transposition. A transposition is a function that swaps two elements of an ordered set. The neighborhood structure is given by any solution that can be reached by a single transposition on the elements of the respective sets.

The function $\mathcal{N}_1(B_b)$ operates on the set of buyer bids B_b. Swapping two bids at the positions (i, j) coincides to a repositioning in the evaluation procedure. In case of scarce resources an allocation for a bid in a leading position becomes more likely, and to sellers with lower reservation prices. The function $\mathcal{N}_2(T(B_{n,f}(S_i)))$ operates on the set of slots available for a given buyer bid. It swaps for each buyer bid two of the slots which changes the order these are passed in the evaluation procedure and therefore the allocation. Both operators are illustrated in figure 1.

Algorithm 1. Evaluate

$s \leftarrow 0$ \\allocation surplus
B_b \\ordered set of buyer bids
$T(B_{n,f}(S_i))$ \\ordered set of slots for a given buyer bid
$B_{s,t}(S_i)$ \\seller bids available in time slot t for the bundle S_i
$B_{alloc} \leftarrow \emptyset$ \\buyers that are part of the allocation (XOR - Constraint)
\\iterate over all buyer bids
while hasNext(B_b) **do**
  $B_{n,f}(S_i) \leftarrow$next(B_b)
  \\test if the buyer n is part of the allocation already
  if $n \in B_{alloc}$ **then**
    $continue$ with the next iteration
  end if
  $s^{'} \leftarrow 0$
  $slots \leftarrow s_n(S_i)$
  \\iterate over the slots
  while hasNext($T(B_{n,f}(S_i))$) **do**
    $B_{s,t}(S_i) \leftarrow next(T(B_{n,f}(S_i)))$
    $s^{''} \leftarrow$solve($B_{n,f}(S_i), B_{s,t}(S_i)$) \\solve the $1 : |m|$ allocation problem for time slot t
    if $isFeasilbe(s^{''})$ **then**
      $s^{'} \leftarrow s^{'} + s^{''}$ \\add the surplus to surplus of the allocation
      $slots \leftarrow slots - 1$ \\save that the slot was successfully allocated
    end if
    \\test if the allocation for $B_{n,f}(S_i)$ is completed
    if $slots == 0$ **then**
      $s \leftarrow s + s^{'}$ \\add surplus to overall surplus
      $B_{alloc} \cup n$ \\save that buyer n was successful with a bid
      $continue$ with the next buyer bid
    end if
  end while
end while

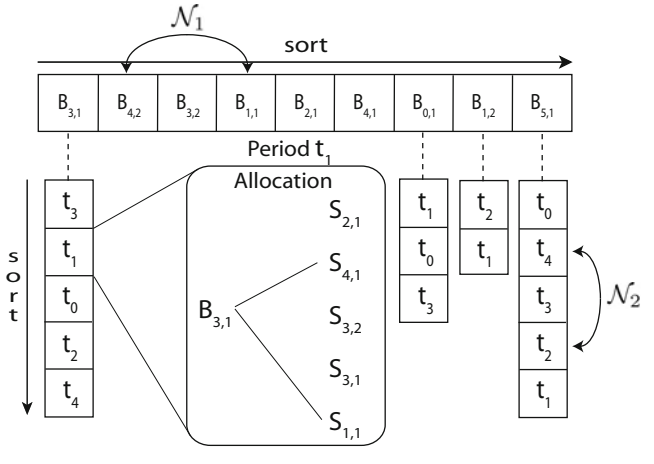

Fig. 1. Problem Representation

4.2 Metaheuristic Based Solution

Local search based methods require an initial solution to improve on, a suitable representation of the problem to be solved and a neighborhood structure to traverse the solution space. The initial solution is provided by a greedy type algorithm we proposed in [14]. The basic steps are illustrated in algorithm 2. The procedure starts with an empty allocation. In the initialization phase the buyer bundle bids and the respective time slots are ordered. The buyer bids are sorted in descending order according to the attractiveness for the inclusion into the allocation (cf. figure 1). The attractiveness of a bundle bid is calculated from the valuation of a bid adjusted by the weighted average consumption of resources (wac) and the flexibility ($flex$) in a time scheduling sense: $\frac{v_n(S_i)s_n(S_i)}{flex*wac}$.

$$flex = \frac{s_n(S_i)}{l_n(S_i) - e_n(S_i)} \tag{1}$$

$$wac(S_i) = \frac{\sum_{S_i \ni g_k} \sum_{e_n(S_i)}^{l_n(S_i)} \max_{a_{g_k,j} \in A_j} \frac{q_n(S_i, g_k, a_{g_k,j})}{\sum_{B_{s,t}(S_i) \ni S_j} q_m(S_j, g_k, a_{g_k,j})}}{l_n(S_i) - e_n(S_i)} \tag{2}$$

Algorithm 2. Greedy Implementation

```
s ← 0                          \\surplus
B_b                            \\set of buyer bids
T(B_{n,f}(S_i))                \\set of slots for a given buyer bid
p_1 ← v_n(S_i)s_n(S_i) / flex*wac   \\sorting criterion buyer bid
p_2 ← surplus                  \\sorting criterion seller bid

sort(B_b, p_1)
for all B_{n,f}(S_i) ∈ B do
   sortSlots(T(B_{n,f}(S_i)), p_2)
end for
s ← eval(B)
```

The parameter $flex$ given in equation 1 measures the flexibility of the buyer bid as a ratio of the number of slots requested and the number of slots available to schedule the resources request. The higher the flexibility, the more valuable a bid is considered and the closer flex is to zero which influences the overall rating positively. The parameter $wac(S_i)$ given in equation 2 describes the weighted average consumption of resources of a buyer bid per time slot. For each resource requested, in each time slot the supply and demand ratio for each of the attributes is calculated. The scarcity of a requested good for a given time slot is determined by the maximum of these ratios. The overall aggregation of the scarcity measures for all goods is divided by the number of requested slots to obtain the weighted average resource consumption on a slot base. In consequence the demand for more scarce resources is more significant in reducing the attractiveness of a buyer bundle bid. The slots for a buyer bid are sorted according to the optimal solution (calculated using lpsolve [15]) of the $1 : |m|$ allocation problem.

A simulated annealing (SA) version based on the original proposal [16] with an adapted cooling schedule was chosen for guiding the improvement on a starting solution. SA is an intensely studied metaheuristic which has provided good results for a various number of combinatorial problems [17]. A standard SA implementation takes four parameters. This is the initial temperature value of the system T_s, a stopping criterion given by a final temperature value T_e, a parameter p specifying the number of steps at each temperature level and a temperature reduction function [18].

Algorithm 3. Simulated Annealing Implementation

$B_b \leftarrow GenerateInitialSolution()$
$T_s \leftarrow surplus(getFirstElement(B_b))$ \\initial Temperature
$T_e \leftarrow 0.1T_s$
$t_{max} = 180\ seconds$
$k \leftarrow 3$ \\number of iterations at N_2 level
$p \leftarrow 1$ \\number of steps at each temperature level
while time limit not met **do**
  $B_b^{'} \leftarrow randomNeighbor(\mathcal{N}_1(B_b))$
  if $(eval(B_b) < eval(B_b^{'}))$ **then**
    $B_b \leftarrow B_b^{'}$
  else
    if $random[0,1) < P(B_b^{'}, T)$ **then**
      $B_b \leftarrow B_b^{'}$
    else
      $T \leftarrow UpdateTemperature(T,p)$
      continue with the next iteration of the main loop
    end if
  end if
  for k iterations **do**
    $B_b^{'} \leftarrow randomNeighbor(\mathcal{N}_2(B_b))$
    if $(eval(B_b) < eval(B_b^{'}))$ **then**
      $B_b \leftarrow B_b^{'}$
      continue with next iteration of the main loop
    end if
  end for
  $T \leftarrow UpdateTemperature(T,p)$
end while

The temperature reduction function is generally given by $T_{t+1} = \alpha T_t$ where $T_0 = T_s$. However, we are interested in a comparison of solution methods based on wall clock time: t_{max}. Therefore temperature is reduced in dependence of time instead of a static parameter: $T(t) = T_s(\frac{T_e}{T_s})^{\frac{t}{t_{max}}}$ [19]. The function ensures that $T(0) = T_s$ and $T(t_{max}) = T_e$. Between these points the temperature is set according to $T(t)$. SA always accepts movements to superior solutions. However its key feature is its mechanism to escape local optima by accepting with a certain probability a solution worse compared to the current solution. The probability for accepting a solution that causes a decrease of $|\Delta|$ in the objective function is given by the acceptance function $P(\Delta) = \exp(\frac{-|\Delta|}{T})$ [20]. This acceptance function implies first that a small decrease

of the objective function is more likely to be accepted than a large decrease. Second in the beginning of the search when T is high most down-hill moves are accepted.

Algorithm 3 shows how SA is applied to the problem. The initial solution is generated based on the greedy approach presented. T_s is set such that a decrease in the objective function value by the surplus of the first accepted bid is accepted with a probability of 80 percent [21]. The surplus of the first accepted bid is used to approximate the maximum change in the objective function by the exchange of two bids. This coincides to omitting the bid from the evaluation. T_e is set to ten percent of the value of T_s. While the time limit is not met a random neighbor $B_b^{'}$ according to $\mathcal{N}_1(B_b^{'})$ is chosen. $B_b^{'}$ is accepted either if it improves the solution quality or if it is accepted according to $P(\Delta)$. The search proceeds with the next neighbor $\mathcal{N}_1(B_b)$ otherwise. For each iteration the temperature is reduced. In case $B_b^{'}$ is accepted a refinement phase based on local search is started. The factor k determines how often an improvement according to the neighborhood $\mathcal{N}_2$ is tested. The first time the solution is improved the process is continued with the next iteration of the main loop.

5 Experimental Evaluation

This section evaluates the metaheuristic approach to grid resource trading using the approach introduced in Section 4. Two sets of Experiments are performed (setup as described in Section 5.1). We first asses the quality of the SA approach based on a comparison to the allocation computed by the commonly available standard solver lpsolve [15]. We then provide the results of the initial greedy heuristic and a first improvement local search algorithm equal to the SA approach but without the probabilistic element to escape local optima. Results are presented in Section 5.2.

5.1 Experiment Description

To study the heuristic methods under simulation we produce random variants from a real world data set. This allows us to keep the macro-structure of characteristic bids for the workflow domain fixed while varying the numbers of bids. The data is obtained from the execution of a 0.2 degree montage workflow [2] of Messier M17 on Grid resources. We assume that a buyer wants to allocate the resources required for a complete execution of such a workflow. This translates to specifying a bid for the bundle S_1 which satisfies the requirements at each point in time. Based on this the attributes of a buyer bid are set as follows: $A_{RAM} = \{86.4\}$, $A_{HD-Space} = \{171\}$, $A_{Nodes} = \{9\}$; a single execution of a workflow is assumed to take 5 whole time slots while a range of 10 time slots is open for trade. As valuations are not available from the data sets these are generated as the sum of a fixed amount of 1000 units and a variable, from uniform distribution drawn between 0 and 100 units. For seller bids on the bundle S_1 (S_2) the attributes are defined as: $A_{RAM} = \{168\}$, $A_{HD-Space} = \{300\}$, $A_{Nodes} = \{20\}$ ($A_{HD-Space} = \{100\}$). The reservation prices are determined by a fixed part 1500 (100) and a variable, from a uniform distribution drawn, value of up to 150 (10).

Two sets of experiments are performed (A and B) that differ in the distribution of buyer bids. For experiment A the requests are normally distributed with a mean value

of three and variance of one. This models a peak in demand for resources between time slot three and time slot eight. For experiment B the requests for resources are uniformly distributed over the time slots. For each type of experiments ten bidding scenarios differing in the number of buyer and seller bids were created. The scenarios comprise 20,40,60,80,100,120,140,160,180 and 200 bundle bids. For each scenario the number of bids is equally distributed between seller and buyer bids. Half of the sellers offer bundle S_1 and half of the sellers offer bundle S_2. For each of the ten bidding scenarios ten instances were generated. The time to solve the allocation problem was set to the reasonable time of three minutes.

5.2 Experiment Results and Analysis

To asses the quality of the approach described in Section 4.2 we first performed a comparison between the lpsolve solver and our approach, over different time requirements. We tested the scalability of lpsolve for very small scale bidding scenarios of type B experiments. Table 1 summarizes the results indicating whether the optimal solution was found (Opt), an intermediary solution was provided (Inter.) or the solver did not return any solution (KO). The results show that for very simple problems lpsolve returns an optimal solution for both 3 and 60 minutes. For slightly more complicated problems lpsolve returns intermeadiate results only, and no optimal solution. For complex problems lpsolve fails to produce any results. As lpsolve fails to produce results with relatively complex experiments, the following experiments evaluate the SA approach with the greedy approach detailed in Section 4.2 and a basic local search.

Table 2 summarizes the results of experiments A and B. The comparison of the heuristic methods is based on the percentage deviation to the best surplus found [22]. The deviation column shows the average percentage deviation of the ten simulation runs performed for the bidding scenario. The corresponding Best/10 column indicates the number of times the heuristic reported the best result. The last row of the table shows the average error for all bidding scenarios for a single heuristic method.

Table 1. Results using lpsolve for small scale scenarios

Time	Scenario			
	2/2	4/4	6/6	10/10
3 min. (Opt/Inter./KO)	3/0/0	0/3/0	0/1/2	0/0/3
60 min. (Opt/Inter./KO)	3/0/0	0/3/0	0/1/2	0/0/3

For both experiments SA shows on average the best performance. SA does not always find the allocation with the maximum surplus but is always closer to the best solution than local search. Both local search and SA improve significantly on the greedy starting solution. With increasing number of bids the factor of improvement decreases; this is because the time for the evaluation is kept constant while the size of the scenario and neighborhood of an allocation to be searched is increased significantly.

The variation between the results of experiment A and B can be explained by the structural differences of the underlying scenarios. The uniform distribution of time slot requirements for experiment B results in less competition for the offered resources

Table 2. Experimental Results and Analysis

Scenario	Experiment A					Experiment B				
	Sim. Annealing		Local Search		Greedy	Sim. Annealing		Local Search		Greedy
	Dev.	Best/10	Dev.	Best/10	Dev.	Dev.	Best/10	Dev.	Best/10	Dev.
10/10	0.00	10/10	4.83	0/10	35.14	0.00	10/10	5.11	0/10	26.02
20/20	0.17	9/10	2.64	1/10	32.13	1.37	5/10	1.83	5/10	27.06
30/30	0.02	8/10	2.25	2/10	27.26	0.30	8/10	1.88	2/10	25.30
40/40	0.55	6/10	1.14	4/10	28.48	0.81	7/10	1.85	3/10	23.32
50/50	0.51	7/10	2.45	3/10	22.74	0.87	8/10	3.05	2/10	19.37
60/60	0.24	7/10	2.97	3/10	21.29	1.63	5/10	2.40	5/10	16.87
70/70	0.00	10/10	7.29	0/10	21.69	2.17	5/10	1.44	5/10	15.51
80/80	0.35	8/10	7.10	2/10	17.02	0.53	8/10	2.62	2/10	14.21
90/90	0.00	10/10	5.80	0/10	12.16	0.13	7/10	1.68	3/10	10.17
100/100	0.17	7/10	4.37	3/10	8.05	0.61	8/10	5.81	2/10	9.14
Average	0.20	82/100	4.08	18/100	22.60	0.84	71/100	2.77	29/100	18.70

compared to the peak in demand modeled for the scenarios in experiment A. From a scheduling perspective there are few options to check in experiment B. The greedy approach in both experiments shows that the deviation to the best solution is bigger for the scenarios of experiment A. The same holds for the comparison of the local search procedures. The scenarios in experiment B require a less extensive search for a promising basic solution to improve on but fine tuning of the initial solution which is the domain of local search. This becomes evident by the smaller average deviation of the results of the local search procedure in comparison to the SA approach. Furthermore the best solution is identified for 71 percent of the simulation runs of experiment B by the SA approach in comparison to 82 percent of the simulation runs of experiment A. In summary, standard solvers like lpsolve provide allocations for very small scale scenarios only and heuristic solutions provide a scalable alternative producing good results for the problem of resource allocation. The simulated annealing-based approach presented here can be used to improve significantly on greedy results.

6 Conclusions

This paper has presented a hybrid metaheurstic approach to the trading of grid resources for the execution of scientific workflows. It has described a trading approach for grid resources and detailed metaheuristics to efficiently trade resources. These metaheuristics have been evaluated through the execution of scientific workflows. We showed that for the scientific workflow domain heuristic solutions provide a scalable alternative compared to standard solvers like lpsolve. Future work will involve the scaling up of the approach to more resource providers and consumers, and increasing the number and types of scientific workflows. Expanded work will involve the evaluation of different and more complex workflow types, as well as other types of applications.

References

1. Foster, I.: The grid: A new infrastructure for 21st century science. Physics Today 55(2), 42–47 (2002)
2. Deelman, E., et al.: Pegasus: A framework for mapping complex scientific workflows onto distributed systems. Scientific Programming 13(3), 219–237 (2005)

3. Broberg, J., Venugopal, S., Buyya, R.: Market-oriented grids and utility computing: The state-of-the-art and future directions. Journal of Grid Computing 6(3), 255–276 (2008)
4. Buyya, R., Yeo, C.S., Venugopal, S., Broberg, J., Brandic, I.: Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility. Future Gener. Comput. Syst. 25(6), 599–616 (2009)
5. Schnizler, B., Neumann, D., Veit, D., Weinhardt, C.: Trading grid services - a multi-attribute combinatorial approach. European Journal of Operational Research 187(3), 943–961 (2008)
6. Rothkopf, M.H., Pekeč, A., Harstad, R.M.: Computationally manageable combinational auctions. Management Science 44(8), 1131–1147 (1998)
7. Sandholm, T.: Algorithm for optimal winner determination in combinatorial auctions. Artificial Intelligence 135(1-2), 1–54 (2002)
8. Oinn, T., et al.: Taverna: lessons in creating a workflow environment for the life sciences: Research articles. Concurr. Comput.: Pract. Exper. 18(10), 1067–1100 (2006)
9. Taylor, I., Shields, M., Wang, I., Harrison, A.: Visual grid workflow in triana. Journal of Grid Computing 3(3), 153–169 (2005)
10. Fitzgerald, S.: Grid information services for distributed resource sharing. In: Proc. 10th IEEE Intl. Symposium on High Performance Distributed Computing (2001)
11. Lee, K., Paton, N.W., Sakellariou, R., Fernandes, A.A.A.: Utility driven adaptive workflow execution. In: CCGrid (2009)
12. Blum, C., Roli, A.: Metaheuristics in combinatorial optimization: Overview and conceptual comparison. ACM Computing Surveys 35(3), 268–308 (2003)
13. Puchinger, J., Raidl, G.R.: Combining metaheuristics and exact algorithms in combinatorial optimization: A survey and classification. In: Artificial Intelligence and Knowledge Engineering Applications: A Bioinspired Approach, pp. 41–53. Springer, Berlin (2005)
14. Buss, G., Lee, K., Veit, D.: Scalable grid resource trading with greedy heuristics. To appear in Fourth International Workshop on P2P, Parallel, Grid and Internet Computing (3PGIC-2010) (February 2010)
15. Lpsolve 5.5.0.14, a mixed integer linear programming (milp) solver, `http://lpsolve.sourceforge.net/`
16. Kirkpatrick, S., Gelatt Jr., C.D., Vecchi, M.P.: Optimization by simulated annealing. Science 220(4598), 671–680 (1983)
17. Suman, B., Kumar, P.: A survey of simulated annealing as a tool for single and multiobjective optimization. Journal of the Operational Research Society 57(10), 1143–1160 (2006)
18. Aarts, E.H.L., Korst, J.H.M., van Laarhoven, P.J.M.: Simulated annealing. In: Aarts, E.H.L., Lenstra, J.K. (eds.) Local Search in Combinatorial Optimization, pp. 91–120. John Wiley & Sons, Chichester (1997)
19. Petersen, H.L., Madsen, O.B.G.: The double travelling salesman problem with multiple stacks - formulation and heuristic solution approaches. European Journal of Operational Research 198(1), 139–147 (2009)
20. Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A., Teller, E.: Equation of state calculations by fast computing machines. Journal of Chemical Physics 21, 1087–1092 (1953)
21. Kouvelis, P., Kim, M.W.: Unidirectional loop network layout problem in automated manufacturing systems. Oper. Res. 40(3), 533–550 (1992)
22. Rardin, R.L., Uzsoy, R.: Experimental evaluation of heuristic optimization algorithms: A tutorial. Journal of Heuristics 7(3), 261–304 (2001)