Scheduling in Multi-Organization Grids: Measuring the Inefficiency of Decentralization

Krzysztof Rzadca
krz@pjwstk.edu.pl

LIG, Grenoble Universities, France
Polish-Japanese Institute of Information Technology, Warsaw, Poland

.

Abstract. We present a novel, generic model of the grid that emphasises the roles of individual organizations that form the system. The model allows us to study the global behaviour of the system without introducing external forms of recompense. Using game-theory and equitable multicriteria optimization, we study three diverse types of computational grids: an off-line system with dedicated uniprocessors, an on-line system with divisible load and an off-line system with parallel jobs. Results show that, unless strong assumptions are made, the complete decentralization leads to a significant loss of performance.
Keywords: game theory, fairness, scheduling, grid, multi-objective optimization

1 Introduction

Grids are systems that allow users to access resources belonging to different administrative entities [1]. Such an administrative decentralization imposes new requirements on resource management systems, which must not only optimize the efficiency of the whole system, but also ensure that all the parties are treated fairly. Otherwise, resulting conflicts may break the grid agreements.

The goal of this paper is to present the generic multi-organizational model of the computational grid. The model allows us to study the problem of fair scheduling without the need to introduce any external forms of recompense, such as money. The analysis of the global behaviour of the system is thus fairly straightforward and does not require many out-of-model assumptions (such as e.g. supply-demand curves). At the same time, the model is general enough to be applied in a variety of systems. We use equitable optimization to study grids with strong central control and game-theory, when central control is weak.

The problem of grid scheduling was addressed in a number of papers. Grid economy approaches [2] introduce free market economy. [3] uses multicriteria optimization in context of divisible load scheduling. In [4], each job is scheduled independently by a broker. The global behaviour of the system cannot be, however, easily studied with those approaches.

This research was partly supported by the FP6 Network of Excellence CoreGRID funded by the European Commission (Contract IST-2002-004265)

This paper is organized as follows. In Section 2 we briefly present game theory and equitable optimization, tools used in analysis of our models. Section 3 presents the generic multi-organizational grid model and three approaches used for theoretical analysis. The following sections present example applications of the model: Section 4 in grids composed of dedicated uniprocessors; Section 5 in divisible load scheduling; Section 6 in parallel job scheduling.

2 Tools: Game-Theory and Equitable Optimization

Game theory [5] studies situations in which independent parties (*players*) make decisions (use *strategies*). For each player P_k, the *outcome* u_k of the game is a function of his/her strategy σ_k, but also of strategies of other players $\boldsymbol{\sigma} = [\sigma_1, \ldots, \sigma_N]$. The players are assumed to be *selfish* and *rational*, i.e. concerned only with maximizing their own outcomes. Furthermore, in strategic games (considered in this paper) players chose their strategies at the same time.

Nash Equilibrium (NE) is a profile of players' strategies such that no player has an incentive to unilaterally change his/her strategy. Given the NE strategies of other players, each player optimizes his/her outcome by playing a NE strategy. It is expected that a game will end in a NE. However, a NE does not necessarily results in the global optimum (usually defined as the optimal sum of players' outcomes). *Price of Anarchy (PoA)* measures the inefficiency of a NE by computing the ratio between the worst NE and the global maximum.

Game theory has been previously applied to the problem of scheduling. [6] considered selfish jobs competing for common infrastructure. [7] analyses a problem of electing one of selfish resources to execute a job.

Equitable optimization [8] incorporates the notion of distributive fairness to multi-criteria optimization. In multi-criteria optimization, a solution is considered optimal, if no outcome can be improved without worsening other outcome (Pareto optimality). Equitable optimization puts a further restriction. A transfer of any small amount from an outcome to any other relatively worse-off outcome results in a more equitable solution. We say that the latter solution *equitably dominates* the former (e.g. solution $[3, 2, 1]$ equitably dominates $[4, 2, 0]$). An equitably optimal solution is a solution that is not equitably dominated by any other solution. The notion of equitable optimality is broader than $\min\max$ fairness: in equitable optimization $[4, 2, 0]$ is as fair as $[3, 2, 2]$, yet $\min\max$ chooses the latter solution. An outcome is *equitably-optimal* iff it is not equitably dominated by any other outcome.

3 The Generic, Multi-Organizational Computational Grid Model

In our model, a *grid* is an agreement between selfish, independent organizations to share their resources. Thus, the central notion of our model is that of an *organization*, an entity that groups a resource donated to the grid and local users willing to employ the whole system.

3.1 The Core of the Model

An organization (denoted as O_k) is an administrative entity such as a laboratory or a faculty. Each organization contributes its resource (denoted as M_k) to the grid. By contributing, an organization expects that its users will have access to other resources in a fair manner. $\mathcal{O}$ denotes the set of all organizations $\mathcal{O} = \{O_1, \ldots, O_N\}$. Organizations are *independent* from each other. Thus, an organization is concerned only with the performance of the jobs produced by its members. Our notion of an organization differs from Virtual Organization, because we assume that an organization must own, and grant access to, a resource.

As there are no external users, each job (denoted as J_k^i) is local to some organization. J_k^i is ith job produced (and owned) by organization O_k. p_k^i denotes job's computation time. For M_k, jobs $\mathcal{J}_k$ produced by the resource's organization O_k are called *local jobs*. Remaining jobs $\mathcal{J}_{-k}$ assigned for execution on M_k are called *foreign jobs*.

We assume that there are no external means of recompense for accessing resources. An organization cannot explicitly "pay" other organization neither in some kind of money, nor in barter trade.

3.2 Additional Characteristics of the Model

In order to derive results, we make additional assumptions, commonly present in the theory of scheduling [9]. We assume that the exact size p_k^i of every job submitted to the system is known. Preemption is not allowed. A job that has been started must be completed. We do not consider communication times. We also assume that the system is perfectly reliable.

In order to assess the performance of the system, we measure the completion time of jobs [9]. C_k^i denotes the completion (finish) time of job J_k^i. To measure the performance experienced by organization O_k, we compute two aggregated measures. The *sum of completion times* is the sum $C_k = \sum_i C_k^i$ of completion times of jobs $\mathcal{J}_k$ owned by O_k. The *makespan* (maximum completion time) is the time when the last job of O_k finishes $C_{\max}(O_k) = \max_i C_k^i$. On the system level, the *global sum of completion times* ΣC is defined as the sum of completion times of all the jobs in the system $\Sigma C = \sum_k \sum_i C_k^i$. The *global makespan* $C_{\max}$ is the time when last job in the system finishes $C_{\max} = \max_{i,k} C_k^i$.

If a job J_k^i cannot be started before certain date (called release date r_k^i), it is usual to measure the flow time F_k^i defined as the time job J_k^i spends in the system, $F_k^i = C_k^i - r_k^i$. The aggregated measures are defined similarly.

3.3 Approaches for Optimization

We introduce a centralized, grid-level scheduler which proposes a schedule to each resource. However, the power of the centralized scheduler and, consequently, the kind of solutions it can impose on individual processors, depends heavily on the level of control the individual organizations have over their resources. We will study the problem from three perspectives, leading to three different approaches for optimization: multi-criteria optimization, game theory and constrained multi-criteria optimization.

Firstly, in the most restricted case, we assume that an organization is neither able to impose any schedule on its local resource, nor to quit the grid. The goal of the grid scheduler is to share the pool of available resources fairly among organizations. Consequently, the problem transforms into *equitable multi-criteria optimization* of performance measures of organizations.

Secondly, each organization may have complete control over the schedule of the local resource. Each organization is tempted to locally modify the solution proposed by the grid scheduler, if the organization's gain is increased. Consequently, such a problem must be analyzed with a *game-theoretic* approach. In the resulting game, the set of players is equal to the set of organizations $\mathcal{O}$. Strategy σ_k of player O_k is a schedule of jobs on player's local resource M_k. Finally, payoff function u_k for player O_k is the performance of player's local jobs $\mathcal{J}_k$.

Thirdly, we assume that each organization independently decides whether to join or to leave the grid. Once inside, the organization grants complete control over its resources to the grid scheduler. Yet, an organization will leave the system, if perceived performance is lower than the performance the organization could achieve being outside (called *self-reliant performance*). This problem is *equitable constrained multi-criteria optimization* of performance measures of respective organizations, with the constraints of the self-reliant performance.

The three approaches defined above are far from being exhaustive to the problem of grid scheduling. However, we claim that they are sufficiently varied to cover a number of grid application scenarios. The multi-criteria optimization approach is suitable for classic systems where a number of resources must be shared between organizations. An example scenario is a supercomputer bought by a state agency, that is later shared between a number of public laboratories and universities. The game theoretic perspective concerns systems with almost no central control, in which independent parties try to maximize their own gain, with no motivation to optimize the performance of the whole system. We expect that highly distributed, peer-to-peer systems will behave in that manner. Finally, the constrained multi-criteria optimization concerns systems where individual goals are noticed, but not necessarily selfishly maximized. Some level of trust and social control can be maintained. This perspective models a number of grids where there are a few participating organizations and the participation is somehow limited (like in e.g. academic grids).

4 Resource Management in Dedicated Grids

In this section, we consider the grid as a tool for accessing specialized resources. Each job $J^i_{k,l}$ in the system must be computed on specific resource M_l, not necessarily the one belonging to owner O_k of the job. The scheduling proposed by classic approaches is to execute jobs on each processor in order of their increasing computation times, regardless of owners of jobs (denoted as $\mathbf{SPT}$). However, this solution may be unfair for organizations with very popular equipment.

The following additional assumptions are made. The model is *off-line*. There are no release dates. Each resource has only one processor, therefore jobs are sequential.

Each organization O_k computes the *sum of completion times* C_k of locally produced jobs $\mathcal{J}_{k,\cdot} = \bigcup \mathcal{J}_{k,l}$.

As resources are uniprocessors, a schedule for resource M_l is a permutation of jobs $\mathcal{J}_{\cdot,l} = \bigcup \mathcal{J}_{k,l}$. However, we may restrict our attention to schedules that order jobs of each organization in non-decreasing processing time order (called *Shortest Processing Time, SPT*). In a SPT schedule, for each organization O_k, if $p^i_{k,l} < p^j_{k,l}$, $J^i_{k,l}$ is executed before $J^j_{k,l}$. Any non-SPT schedule is Pareto-dominated by a SPT schedule (the proof is by exchange argument on jobs in non-SPT order).

4.1 Optimization Approach

The scheduling problem remains hard, even if it is restricted to two organizations and one resource $P1||(\Sigma C^A_i, \Sigma C^B_i)$ [10]. There can be exponential number of Pareto-efficient schedules. The decision version of the problem is NP-Complete.

Equitable Walk (EW) [11] is a heuristics which produces a number of grid schedules by iterative modifications of the initial $\mathbf{SPT}$ schedule in order to improve the outcome of the disfavored organizations. The algorithm modifies the schedules by *switching* the order of two jobs executed one after another on the same resource. Although there is no guarantee about the optimality of the produced results, during experiments [11] EW delivered results close to optimal, running a few orders of magnitude faster than a reference exact algorithm

4.2 Game-Theoretic Approach

Assertive organization O_k, which is able to impose a schedule for $\mathcal{J}_{\cdot,k}$, would use a greedy *My Jobs First* (MJF) strategy, which schedules all the local jobs $\mathcal{J}_{k,k}$ before any foreign job. Given any strategies of the rest of organizations, MJF strategy will reduce the total finish time C_k. By $\mathbf{MJF}$ we denote the profile of strategies in which every organization uses MJF. We define the payoff $u_k(\boldsymbol{\sigma})$ for each player O_k as the *gain over* $\mathbf{MJF}$ for that player, $u_k(\boldsymbol{\sigma}) = C_k(\mathbf{MJF}) - C_k(\boldsymbol{\sigma})$.

Proposition 1. $\mathbf{MJF}$ *is the only Nash equilibrium of the one round, non-cooperative grid scheduling game.*

Proof. Assume that, for a particular instance $\mathcal{J} = \{J_{k,l}\}$, the grid scheduler is able to produce schedule $\boldsymbol{\sigma}^* = [\sigma^*_1, \ldots, \sigma^*_n]$, which results in non-negative payoff $u_k(\boldsymbol{\sigma}^*) \geq 0$ for all players and a positive payoff for at least one player. Consequently, there must be at least one player O_k, for whom the proposed strategy σ^*_k is different than MJF. Thus, in M_k's schedule, there is at least one foreign job $J^i_{l,k}$ scheduled before a local job $J^j_{k,k}$. If O_k decides to switch the order of execution of those two jobs, local job $J^j_{k,k}$ will be finished faster and, thus, player's payoff u_k will increase. At the same time payoff u_l will decrease. It follows that the strategy maximizing u_k is MJF, given that the others play any profile of strategies $\boldsymbol{\sigma_{-k}}$. Additionally, if all the other players play MJF, the only strategy which guarantees non-negative u_k for O_k is to play MJF as well.

In [11] we have shown an example instance, in which $\mathbf{MJF}$ strategies resulted in a $O(n)$ increase of makespan of every organization. Thus, the price of anarchy is at least linear with the number of jobs, and, consequently, the price the grid pays for the lack of control is considerable.

4.3 Constrained Optimization Approach

In dedicated grids, *self-reliant performance* corresponds to **MJF** strategies. Therefore, we can equitably optimize the gains $[u_k]$ defined in the previous section with a constraint $u_k \geq 0$ for each organization O_k. To produce such solutions, we use Adjusted EW (AEW) [11] algorithm, that maximizes u_k, instead of minimizing C_k. AEW starts with a **SPT** schedule. However, **SPT** may not be feasible, and, generally, AEW may be unable to produce a feasible result. Nevertheless, during our experiments [11], AEW always returned at least one feasible schedule, if such existed. Moreover, the constraint did not caused large loss of performance.

5 Load Balancing of Divisible Load

In this section, we consider the grid as a tool to compute divisible load jobs. Divisible load [12] models jobs that can be divided into a large number of fragments, that can be computed independently in parallel. In our model, jobs may be computed in parallel on many resources with the consent of the owners of the non-local resources.

We assume that the system must guarantee the latest finish time of every submitted job, which is announced to the user in the moment of job submission. We claim that such guarantee gives better quality of service than best-effort execution commonly used in distributed divisible load computing (e.g. in BOINC [13]).

We consider an *on-line* model with release dates. Jobs are unknown before they are released. Jobs are processed in FIFO order. O_k measures the flow time F_k of locally-produced jobs. As long as there are no local jobs, computing a foreign job is free. However, if a new local job is produced, such a foreign job is blocking the needed resources and thus delaying the newly-produced local job. Note that once foreign load is accepted, in order to guarantee its finish time, it cannot be interrupted. Consequently, the performance of the owner of the resource is degraded. For the theoretical analysis, we also assume that the jobs are produced by a Poisson process. With this assumption, we analyze the expected result (and not the worst-case) of the proposed algorithms and strategies.

5.1 Optimization Approach

In classic, centralized systems, a straightforward approach is to use the averaging load balancing algorithm (ALB). ALB sends parts of loads from overloaded to underloaded resources so that all resources finish at the same time. ALB appends such parts at the end of the schedule of an underloaded resource. With one organization and no communication costs, this strategy is optimal. It follows that, in our model, the expected result of ALB remains optimal[1].

[1] As in the worst case the on-line adversary produces a next local job every time some foreign load is accepted, the gain of the sender's completion time is equal to the loss of the receiver's completion time. Consequently, no algorithm that sends load can achieve better results than the local computation.

Proposition 2. *ALB is an equitably-optimal strategy for two-organizational grid when, on each resource, the load is composed of only one job.*
Proof. We will start with computing the delay $g(r, L_k, \Phi_k)$ in the start time of the "next" local job caused by the foreign load. $g(r, L_k, \Phi_k)$ is a function of the next job's unknown release time r, the length of the known local load L_k and the length of the incoming foreign load Φ_k (assuming that foreign load comes at time $t = 0$). If $r \leq L_k$ (the job is released before the local load is completed), $g(r, L_k, \Phi_k) = \Phi_k$ (the job is delayed by the size of the foreign load). If $r > L_k + \Phi_k$, $g(r, L_k, \Phi_k) = 0$ (the job is not delayed). Finally, if $L_k < r \leq L_k + \Phi_k$, $g(r, L_k, \Phi_k) = L_k + \Phi_k - r$.

Assuming that local jobs $\mathcal{J}_k$ are produced by a Poisson process with known mean time between arrivals λ_k, we can compute the expected value of the delay as: $EG(L_k, \Phi_k) = \Phi_k + \frac{e^{-\lambda_k L_k}}{\lambda_k}\Big(e^{-\lambda_k \Phi_k} - 1\Big)$. Note that $EG\,(L_k, \Phi_k) < \Phi_k$ for all positive values of Φ_k and L_k. ALB, assuming two resources, one job on each resource, $p_1^1 > p_2^1$, sends half of the difference in loads to the less loaded resource. The gain in the completion time C_1 of the sender $\Phi_2 = \frac{1}{2}(p_1^1 - p_2^1)$ is thus higher than the loss $EG(p_2^1, \Phi_2)$ of the receiver. The resulting load distribution optimizes thus both the worst utility (C_1) and the sum of utilities.

5.2 Game-Theoretic Approach

Even if ALB is optimal, a selfish organization holding full control over its resource will never accept incoming foreign load, as it may delay future local jobs. The expected delay $EG\,(L_k, \Phi_k)$ is positive for all positive values of Φ_k and L_k. Consequently, the only non-dominated action for the less-loaded resource is not to receive anything, which leads to significant loss of performance of the grid perceived as a whole.

5.3 Constrained Optimization Approach

As EG is positive, a receiver looses when cooperating. Thus, it is not possible to apply the constrained optimization to the original problem. However, we can modify the rules of the game, so that participating in the load-balancing algorithm becomes profitable even for less loaded organizations. Note that, through following mechanisms, we cannot *guarantee* that each organization will gain from LB. We only *increase the probability* that, on average, organizations gain.

Firstly, if the system forces the organizations to commit to their decisions for a *longer period* of time, the probability of being the receiver is similar to that of being the sender (assuming that the resources are similarly loaded). In our experiments [14], this mechanism was sufficient to make ALB the dominating strategy in grids composed of similarly loaded resources.

Secondly, if the load of resources differ, we introduce two mechanism in the load balancing algorithm, *iterativeness* and *bounds*, to distribute the gains from cooperation fairly among the participants. In iterative LB, firstly the least-loaded resources are balanced, then the resources are iteratively added in the order of their local load. In bounded LB, each organization O_k declares its participation level l_k. The algorithm ensures that, if a resource receives some load, its local queue will not be extended beyond the declared participation level. To motivate organizations to declare $l_k > 0$, an

overloaded resource cannot send more load than its l_k. In experiments [14], bounded, iterative LB in grids with one overloaded resource managed to improve F_i of underloaded resources, and thus to make load balancing the dominating strategy.

6 Parallel Job Scheduling

Here, we extend to multiple organizations the classic model of scheduling parallel, rigid jobs on a multiprocessor resource in order to minimize the makespan. Each organization O_k owns resource M_k with m processors and minimizes the maximum completion time (makespan) $C_{\max}(O_k)$ of the locally-produced jobs. A job J_k^i must be executed on q_k^i processors of exactly one resource. The model is *off-line*. There are no release dates. Consequently, we may assume that a foreign job executed after all local jobs does not cause any cost.

6.1 Optimization Approach

Multi-organizational scheduling is an extension of scheduling sequential jobs on two processors, which is NP-hard [15]. For parallel jobs scheduled on one resource, list scheduling (LS) algorithm is a $(2-\frac{1}{m})$-approximation of $C^*_{\max}$[16]. LS works in two phases. In the first phase, jobs are ordered into a list. In the second phase, the schedule is constructed by assigning jobs to processors in a greedy manner.

Two lower bounds on the global makespan can be defined. Let us denote as $W = \sum p_k^i q_k^i$ the total *surface* of the jobs, and as $p_{\max} = \max p_k^i$ the length of the longest job. Firstly, all the jobs must fit into available processors, so $C^*_{\max} \geq \bar{W} = \frac{W}{Nm}$. Secondly, the longest job must be executed, so $C^*_{\max} \geq p_{\max}$.

We have assumed that a job cannot be executed in parallel on two resources. Consequently, we cannot treat N resources as one resource having Nm processors.

Proposition 3. *LS is a* 3*-approximation of* $C^*_{\max}$.

Proof. The proof is by contradiction. Let us assume that the last job finishes after $3C^*_{\max}$. It is thus started after $2C^*_{\max}$. At the moment the last job is started, all the other resources are busy (otherwise, the job would have been started earlier). Let us denote as $LB = \max(\bar{W}, p_{max})$. Consequently, on each resource M_k we have $C_{\max}(M_k) \geq 2C^*_{\max} \geq 2LB$. On each resource M_k, we have [16] (as $LB \geq p_{max}$): $u_k(t) + u_k(t + LB) \geq m$ for $0 \leq t \leq \bar{W}$. After integrating this inequality, we get: $\int_0^{\bar{W}} u_k(t)dt + \int_0^{\bar{W}} u_k(t + LB)dt \geq m\int_0^{\bar{W}} 1dt$, i.e. $\int_0^{\bar{W}} u_k(t)dt + \int_{LB}^{LB+\bar{W}} u_k(t)dt \geq m\bar{W}$. After adding inequalities for every resource M_k, we get: $\sum_{1\leq k\leq N}(\int_0^{\bar{W}} u_k(t)dt + \int_{LB}^{LB+\bar{W}} u_k(t)dt) \geq Nm\bar{W} \geq W$. Left-hand side of the inequality is the surface of the jobs computed on all resources in periods $[0, \bar{W}]$ and $[LB, LB + \bar{W}]$. Those periods do not overlap. The surface computed is thus greater than the surface of all the tasks available, which leads to a contradiction.

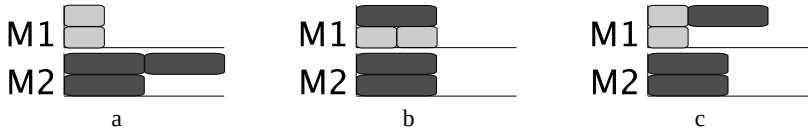

Fig. 1. Globally-optimal solution (b) extends $C_{max}(O_1)$ in comparison with the local solution (a). The best solution not extending O_1's makespan is (c).

6.2 Game-Theoretic Approach

Let us assume that each organization can control the schedule on the local resource, but not the allocation of jobs to resources, which is given. A strategy similar to MJF (Section 4.2) is as follows. An organization firstly schedules its local jobs. Then, foreign jobs are scheduled so that they do not delay any local job: either at the end of the schedule, or in gaps.

Proposition 4. **MJF** *is a Nash equilibrium of the scheduling game.*
Proof. (Sketch) Similarly to Proposition 1, given any profile of strategies of other players, MJF minimizes the $C_{\max}(O_k)$.

Proposition 5. *The Price of Anarchy is at least* $\frac{3}{2}$.
Proof. Consider an instance in Figure 1. **MJF** solution, depicted in (c), has $C_{\max} = 3$, whereas the global optimum has $C_{\max} = 2$.

6.3 Constrained Optimization Approach

We can use load-balancing techniques similar to those presented in the previous section to optimize the system-wide makespan, at the same time not worsening makespans of individual organizations. Multi-Organizational Load Balancing Algorithm (MOLBA) [17] starts with scheduling jobs $\mathcal{J}_k$ on local resource M_k with LS in Highest-First order (i.e. non-increasing number of required processors). Resulting $C_{\max}(O_k)$ form the constraint for the rest of the algorithm. Then, for organizations with $C_{\max}(O_k) \geq 3\bar{W} + p_{\max}$, all the jobs are mixed and rescheduled in HF order by LS algorithm on all resources (however, LS is adjusted so that on resources of organizations that only receive jobs, no local job is delayed). In [17], we proved that this algorithm is a 4-approximation of $C_{\max}$, at the same time not increasing any $C_{\max}(O_k)$.

7 Conclusions

The paper presented a new model of computational grid that emphasises its organizational heterogeneity. The notion of organization, that donates resources, but also consumes other resources, allows us to avoid using external forms of recompense that must be present in the classic, provider-consumer schemes. Through a series of applications we have demonstrated that the model is useful for theoretical analysis of various grids: working off-line or on-line, processing divisible load, sequential or parallel jobs. The perspectives of analysis ranged from systems that are almost like classic supercomputers (optimization), through partly distributed ones (constrained optimization), to systems that are highly distributed (game-theory). Such a spectrum allowed us to compare

the performance and thus to measure the cost of the decreased control. We have shown that in highly distributed systems the loss of performance of the grid is significant (with an exception of the last model, that required a strong off-line assumption). However, we were able to guarantee good results for all organizations, if partial control was granted to a centralized grid scheduler. We conclude that grids cannot be fully distributed to achieve acceptable performance. A strong control of community must be present.

In our future work we plan to, firstly, further investigate the scheduling of rigid, parallel jobs, and secondly, to apply the multi-organizational model in other contexts, such as data-intensive computation. In parallel, we plan to implement some of presented ideas in real-world grid resource managers.

Acknowledgements: The author would like to thank Fanny Pascual, Denis Trystram, and Adam Wierzbicki.

References

1. Foster, I.: What is the grid. `http://www-fp.mcs.anl.gov/~foster/Articles/WhatIsTheGrid.pdf` (2002)
2. Buyya, R., Abramson, D., Venugopal, S.: The grid economy. In: Special Issue on Grid Computing. Volume 93., IEEE Press (2005) 698–714
3. Marchal, L., Yang, Y., Casanova, H., Robert, Y.: A realistic network/application model for scheduling divisible loads on large-scale platforms. Proceedings of the International Parallel & Distributed Processing Symposium (IPDPS) **01** (2005) 48b
4. Kurowski, K., Nabrzyski, J., Oleksiak, A., Weglarz, J.: Multicriteria aspects of grid resource management. In Nabrzyski, J., Schopf, J.M., Weglarz, J., eds.: Grid resource management: state of the art and future trends. Kluwer (2004) 271–293
5. Osborne, M.J.: An Introduction to Game Theory. Oxford (2004)
6. Liu, J., Jin, X., Wang, Y.: Agent-based load balancing on homogeneous minigrids: Macroscopic modeling and characterization. IEEE TPDS **16**(7) (2005) 586–598
7. Kwok, Y.K., Song, S., Hwang, K.: Selfish grid computing: Game-theoretic modeling and nash performance results. In: Proceedings of CCGrid. (2005)
8. Kostreva, M.M., Ogryczak, W., Wierzbicki, A.: Equitable aggregations and multiple criteria analysis. EJOR **158** (2004) 362–377
9. Brucker, P.: Scheduling Algorithms. Springer (2004)
10. Agnetis, A., Mirchandani, P., Pacciarelli, D., Pacifici, A.: Scheduling Problems with Two Competing Agents. Operations Research **52**(2) (2004) 229–242
11. Rzadca, K., Trystram, D., Wierzbicki, A.: Fair game-theoretic resource management in dedicated grids. In: Proceedings of the IEEE International Symposium on Cluster Computing and the Grid (CCGRID07). (2007)
12. Robertazzi, T.: Ten reasons to use divisible load theory. Computer **36**(5) (2003) 63–68
13. Anderson, D.: BOINC: a system for public-resource computing and storage. In: Grid Computing, 2004. Proceedings. Fifth IEEE/ACM International Workshop on. (2004) 4–10
14. Rzadca, K., Trystram, D.: Promoting cooperation in selfish computational grids. European Journal of Operational Research (to appear) (2007)
15. Garey, M., Johnson, D.: Computers and Intractability: A Guide to the Theory of NP-Completeness. WH Freeman & Co. New York, NY, USA (1979)
16. Eyraud-Dubois, L., Mounie, G., Trystram, D.: Analysis of scheduling algorithms with reservations. In: Proceedings of IPDPS, IEEE Computer Society (2007) 1–8
17. Pascual, F., Rzadca, K., Trystram, D.: Cooperation in multi-organization scheduling. In: Euro-Par Proceedings. Volume 4641 of LNCS., Springer (2007)