Fair Game-Theoretic Resource Management in Dedicated Grids

Krzysztof Rzadca
rzadca@imag.fr
LIG and PJIIT

Denis Trystram
LIG Grenoble University
51, avenue Jean Kuntzmann
38330 Montbonnot Saint Martin, France

Adam Wierzbicki
Polish-Japanese Institute of
Information Technology
Koszykowa 86
02-008 Warsaw, Poland

Abstract

We study two problems directly resulting from organizational decentralization of the Grid. Firstly, the problem of fair scheduling in systems in which the grid scheduler has complete control of processors' schedules. Secondly, the problem of fair and feasible scheduling in decentralized case, in which the grid scheduler can only suggest a schedule, which can be later modified by a processor's owner. Using game theory, we show that scheduling in decentralized case is analogous to the Prisoner's Dilemma game. Moreover, the Nash equilibrium results in significant performance drop. Therefore, a strong community control is required to achieve acceptable performance.

1 Introduction

Computational Grids [10] are distributed supercomputers of a very large scale. With the growing level of complexity of models used in many areas of modern science on one hand, and the growing volume of experimental data to analyze on the other hand, the access to computational power becomes a key apparatus [4] in areas as diverse as molecular biology, particle physics, physical chemistry, or civil engineering. Computational grid may become a convenient tool that provides enormous computing power required by such projects. Computational grids have been extensively studied since the end of the nineties and fairly complete middleware (such as the Globus Project [9]) is available. One of the most important remaining challenges is rather of economical, or even psychological nature. The Grid, by its definition [8], is inherently distributed, as it combines resources under different administrative domains. Consequently, certain rules of collaboration must exist, which specify how users from one domain access resources belonging to others. Providers must have some motivation to share their resources, expressed either by earning money for each CPU hour donated, by barter-trading the access, or by formal bilateral agreements between institutions.

1.1 Motivation and Objectives

It is difficult to use "pure" classic Combinatorial Optimization for optimizing the resource management on computing grids. An "instance" of grid cannot be easily represented, as there are too many parameters, not only the description of processors, applications and communication links, but also their interactions. Furthermore, it is hard to choose the "right" criterion, reflecting the diversity of needs or wishes of users, systems administrators and resource owners. Finally, the behavior of the grid seems to be unpredictable. No reasonable model (even probabilistic one) is available today.

The objective of this work is, firstly, to investigate the use of game theory [17] as an alternative for describing the problem of grid resource management. Secondly, we study the problem of multi-criteria scheduling with equitable optimization approach [13] and provide algorithms that produce solutions which are fair to all the participants.

1.2 Contribution

The most important contribution of this paper (Section 4.1) is a demonstration that scheduling in a computational grid composed of processors which can alter locally their schedule is analogous to the Prisoner's Dilemma game. Although global cooperation is profitable, each processor is tempted to "cheat" by favoring its local jobs. Moreover, we show that if everyone "cheats", the waste of performance experienced by every processor is significant (resulting performance is arbitrary far from the optimum). This result

The research was supported by the Polish Ministry of Science and Information Society Technologies under grant 3T11C 005 27 "Models and Algorithms for Efficient and Fair Resource Allocation in Complex Systems" and partly supported by the FP6 Network of Excellence CoreGRID funded by the European Commission (Contract IST-2002-004265). Krzysztof Rzadca is partly supported by the French Government Grant number 20045874.

proves that, although grids are decentralized, a strong society's control is essential to achieve acceptable performance.

To our best knowledge this paper presents also the first application of the axiomatic theory of equity to the problem of fair scheduling. We consider fairness both in game-theoretic and in pure optimization setting.

1.3 Organization

The paper is organized as follows. In Section 2 we introduce the model of the computational grid composed of dedicated processors and provide some general results on the size of the search space. Then, we introduce a centralized, grid-level scheduler which proposes fair ordering to each processor. The power of the centralized scheduler (and, consequently, the solutions it can impose on individual processors) depends heavily on the level of control the individual organizations have on their processors. Firstly, we can assume that the organization is not able to impose any ordering on its processor. The problem transforms then into multi-criteria optimization performed by the grid scheduler. We analyze this problem and present an algorithm which produce such solutions in Section 3. Secondly, each organization may have complete control over the schedule of the local processor. In such situation, the grid scheduler acts only as an advisor. The proposed solution must be profitable for each organization. However, each organization is tempted to modify it locally, if the organization's gain is increased. Consequently, such a problem must be analyzed with a game-theoretic approach (Section 4). Section 5 presents basic validation of the heuristics proposed. A short description of related work is provided in Section 6 before concluding in Section 7.

2 Grid Model

Informally, a computational grid logically interconnects several processing units such as clusters, supercomputers, but also pieces of specialized equipment like sophisticated displays, microscopes, or DNA sequencers (Figure 1). The actual physical network, consisting of high-performance network links, is out of the scope of this paper. Users are grouped in organizations, such as laboratories or faculties. Each organization owns one of the processors, which is the organization's contribution to the grid. By contributing, the organization expects that its users will be granted access to other processors in a fair manner. Each organization is concerned only with the performance (measured as the sum of completion times) of the jobs produced by its members. Processors have their local schedulers, which order jobs to be computed according to some criteria. A centralized grid scheduler helps to coordinate local schedulers. However, the organizations are independent, i.e. a local scheduler is not forced to follow the advice given by the grid scheduler.

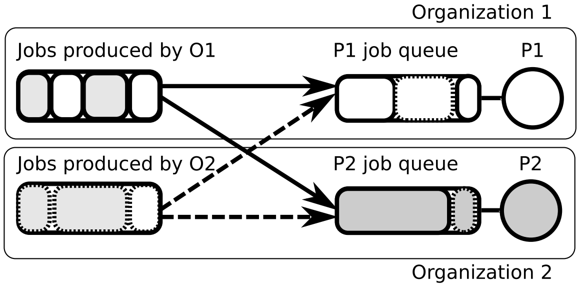

Figure 1. A grid composed of two organizations and two dedicated processors. P_1 processes white jobs, P_2 processes gray jobs. O_1 produced 7 jobs (plotted in continuous lines), 4 of them are not yet sent, two are waiting in P_1's queue and one is waiting in P_2's queue. O_2 produced 5 jobs (plotted in dotted lines).

In this paper we address two issues concerned with such an architecture. Firstly, what should be the properties of the schedules proposed by the grid scheduler to be acceptable by local schedulers. Secondly, how can the grid scheduler produce such schedules.

2.1 Notation and Preliminary Definitions

By $\mathcal{O} = \{O_1, \ldots, O_m\}$ we denote the set of independent organizations forming the grid. Each organization O_i owns a processor P_i. Processors are *dedicated*. Therefore, each job in the system must be executed on a specific processor.

$J^k_{i,q}$ is kth job which must be executed on processor P_q and which is produced (and owned) by organization O_i. Index k is used only to distinguish between jobs on a processor and does not imply the arrival or execution order. By $\mathcal{J}_{i,q}$ we denote the set of all jobs produced by O_i which must be executed on P_q. $n_{i,q} = |\mathcal{J}_{i,q}|$ is the number of such jobs.

We will use the standard game-theoretic notation of $-i$ to denote the set containing everything but the element i, so $\mathcal{J}_{-i,q} = \left(\bigcup_j \mathcal{J}_{j,q}\right) - \mathcal{J}_{i,q}$. Similarly, by $\cdot$ we denote the set containing all possibilities, e.g. $\mathcal{J}_{i,\cdot}$ is the set of jobs that are produced by organization O_i: $\mathcal{J}_{i,\cdot} = \bigcup_j J_{j,q}$.

From processor's P_q point of view, jobs produced by the processor's organization $\mathcal{J}_{q,q}$ are called the *local jobs*. Remaining jobs $\mathcal{J}_{-q,q}$ to be executed on this processor are called the *foreign jobs*. For organization O_i, *remote jobs* $\mathcal{J}_{i,-i}$ are the jobs produced by this organization which are to be executed on non-local processors.

A scheduler is an application which assigns start times to jobs. A scheduling problem is the problem how to assign such start times (formal definitions follow). A scheduling problem is considered *off-line* if all the jobs are known before the scheduling starts. Here, we also consider that all the jobs are ready to be executed (there are no *release dates*). A *clairvoyant* scheduler knows the size $p^k_{i,q}$ of each job $J^k_{i,q}$. There is *no preemption* if the job must be executed completely and cannot be interrupted after a processor has started to execute it. A processor can be shared by the jobs assigned to it in many ways. In *time sharing*, at any moment, a processor executes only one job.

By $C^k_{i,q}$ we denote the completion (finish) time of a job $J^k_{i,q}$. For an organization O_i, we may compute the sum of completion times as $C_i = \sum_q \sum_k C^k_{i,q}$ and the maximum completion time (makespan) as $C_{\max i} = \max_{k,q} C^k_{i,q}$. In the classic multiprocessor scheduling problem, the optimization of ΣC_i results in schedules better for users, whereas the optimization of the makespan $C_{\max}$ produces schedules which utilize the machine better [6].

For processor P_q, a *(list) schedule* is a permutation $\pi : \mathcal{J}_{\cdot,q} \rightarrow (J^{\pi(1)}_{\cdot,q}, J^{\pi(2)}_{\cdot,q}, \ldots, J^{\pi(n_{\cdot,q})}_{\cdot,q})$ of jobs $\mathcal{J}_{\cdot,q}$. A *scheduler* is an application which produces schedules, given the sets of jobs assigned to each processor. A non-preempting, time-sharing processor executing a schedule π, firstly executes the first job $J^{\pi(1)}_{\cdot,q}$, then, when the jobs finishes, executes the second one $J^{\pi(2)}_{\cdot,q}$ and so on. A Shortest Processing Time (SPT) schedule π_{SPT} is a schedule which orders the jobs $\mathcal{J}_{\cdot,q}$ according to non-decreasing sizes of jobs, i.e. $p^{\pi_{SPT}(k)}_{i,q} \leq p^{\pi_{SPT}(k+1)}_{i,q}$. If there is one processor and one organization, SPT schedule is optimal regarding the sum of completion times of jobs [3].

2.2 Problem Statement

We consider off-line, clairvoyant scheduling with no preemption on time-sharing processors. Each organization O_i is concerned with the sum of completion times C_i of the jobs $\mathcal{J}_{i,\cdot}$ which have been produced by its members. Organization O_i does not care about the performance of other organizations. However, as the processors are dedicated, organizations must submit jobs also to non-local processors. As a result, C_i depends heavily on the performance of jobs executed on processors other than the organization's local processor, which in turn are controlled by other organizations.

The scheduling problem considered in this paper is the *fair* ordering of jobs $\mathcal{J}_{\cdot,q}$ on each processor P_q which minimizes the sum of completion times C_i of all the organizations. Generally, the notion of fairness is hard to define precisely. In this work, we follow the axiomatic theory of equity, which basically states that the scheduling algorithm does not prefer *a priori* any organization (we present the axiomatic theory of equitable optimization in Section 3.1).

2.3 General Remarks

In a system having multiple objective functions, multicriteria approaches to optimization must be used. Those approaches usually produce a set of *Pareto-optimal* [17] solutions. In each Pareto-optimal solution, the result of one objective function cannot be improved without worsening one of the other objectives. We propose to further restrict the Pareto set to contain only *equitable solutions*. In such a set, the objective which is worse off cannot be improved without worsening much more one of the other objectives. Informally, equity guarantees that the resulting sum of completion times would be fair to every organization. However, if each organization is able to improve its objective by introducing some local modifications in the schedule, the resulting system must be analyzed by *game theoretic* approaches.

The following general property applies in both equitable optimization and in game-theoretic approach:

Proposition 1 *For a processor P_q, a schedule which orders jobs $\mathcal{J}_{i,q}$ originating from a organization O_i not in shortest processing time (SPT) order is Pareto-dominated by a SPT schedule for those jobs.*

Proof. The proof of this proposition is based on an exchange argument and it is analogous to the proof of the optimum of the SPT schedule for a single processor and a single organization. Let us assume that there are two jobs $J^k_{i,q}$ and $J^l_{i,q}$ executed in non-SPT order. Job k is longer ($p^k_{i,q} > p^l_{i,q}$), but is executed before job l ($C^k_{i,q} < C^l_{i,q}$). If those two jobs are swapped (resulting in completion times $C^{l*}_{i,q} < C^{k*}_{i,q}$) the sum of completion times of those two jobs will be decreased. $J^l_{i,q}$ completes earlier (i.e. $C^{l*}_{i,q} < C^k_{i,q}$, since the first job starts at the same moment, but it is shorter). $J^k_{i,q}$ completes at the same moment as $J^l_{i,q}$ did before the swap ($C^{k*}_{i,q} = C^l_{i,q}$). All the jobs scheduled between those two jobs will complete earlier, as they will start earlier. The rest of the jobs is not affected by the swap. Consequently, this swap reduces at least the sum of completion times C_i for the owner O_i of the jobs being swapped. □

Note that this proposition does not apply for jobs belonging to different organizations. Given two jobs with different owners $J^k_{i,q}$ and $J^l_{j,q}$, $J^k_{i,q}$ executed before $J^l_{j,q}$, swapping them would increase C_i, at the same time decreasing C_j. Consequently, neither solution would Pareto-dominate.

The consequence of the proposition presented above is that the number of reasonable (Pareto-efficient) schedules is reduced considerably. However, their number is still very large. Let us assume that there are only two organizations O_1 and O_2. Consider processor P_1 with $n_{1,1}$ local jobs and $n_{2,1}$ foreign jobs. A multiset (called also a bag) is a set which may contain multiple copies of an element. The schedule can be fully described by a multiset of size $n_{2,1}$, with elements $\{1, \ldots, n_{1,1}+1\}$. The idea is that each member of the multiset specifies a placement for one foreign job. In order to construct a schedule from the multiset, the local jobs are ordered according to SPT. Then, the shortest foreign job is assigned the smallest element from the multiset, which specifies its placement (1 – the job is placed before the shortest local job, 2 – before the second shortest local job, ..., $n_{1,1}+1$ – after the longest local job). This element (actually, one of the copies of the element) is removed from the multiset. The algorithm proceeds with the rest of foreign jobs (considering them in SPT order). Consequently, the number of SPT schedules for one processor is equal to the number of multisets of size $n_{2,1}$ with elements $\{1, \ldots, n_{1,1}+1\}$, which in turn is equal to the number of combinations with repetition of $(n_{1,1}+1)$ objects from which $n_{2,1}$ objects are chosen:

$$\binom{n_{1,1}+n_{2,1}}{n_{2,1}} = \frac{(n_{1,1}+n_{2,1})!}{n_{1,1}!n_{2,1}!} \in O(n_{1,1}^{n_{2,1}})$$

Note that although not necessarily all SPT schedules are Pareto-optimal, in the worst case the number of non-dominated schedules for a single processor is exponential with the number of foreign jobs [1]. A complete schedule on the grid level specifies schedules for each processor. In two processor case, each of $O(n_{1,1}^{n_{2,1}})$ schedules of the first processor is matched with $O(n_{2,2}^{n_{1,2}})$ schedules of the second processor. Most of the resulting combinations are Pareto-dominated. However, at least $O(n_{1,1}^{n_{2,1}})$ Pareto-optimal schedules remain. Thus, the number of Pareto-optimal grid schedules is also exponential.

3 Equitable Optimization Approach

In this section we assume that organizations cannot control the schedule on their processors. The problem considered is how to construct a grid schedule (consisting of schedules for each processor) which would treat all the participating organizations fairly, at the same time being efficient. This problem is, in fact, multicriteria minimization of the sum of completion times C_i of different organizations. In this section we will firstly characterize some general properties of *equitable multicriteria optimization*, and then propose an heuristic approach which produce equitable solutions of the multicriteria minimization – a local search method called Equitable Walk.

3.1 Fairness and Equitable Optimization

In this work, the concept of fairness is identified with *distributive fairness*, a sense narrower than social justice [18]. Distributive fairness is usually related to the question of distribution of some goods, resources or costs, be it kidneys for transplantation, parliament mandates, or the costs of water and electricity. Although extensively studied [19], fairness is a complex concept that depends much on cultural values, precedents, and the context of the problem. Therefore, precise definition is needed to use it in research.

Consider a *Multicriteria Optimization Problem (MOP)* with m objective functions $f_i(\mathbf{x})$ on a uniform scale. We assume, without loss of generality, that the objective functions are to be minimized. $\mathbf{f}(\mathbf{x})$ is a vector-function that maps the decision space $X = R^l$ into the criterion space of *outcome vectors* $Y = R^m$; $Q \subset X$ denotes the feasible set, $\mathbf{x} \in X$ denotes the vector of decision variables, $\mathbf{y} = [f_1(\mathbf{x}), \ldots, f_m(\mathbf{x})]$ denotes the outcome vector.

Distributive fairness can be precisely described by axioms, which formulate a *relation of preference* on the outcome vectors. A *equitable rational preference relation* is any symmetric and transitive relation satisfying the following axioms (see [13] for formal definitions):

symmetry The ordering of the outcome values is ignored (e.g. a solution $\mathbf{y}' = [4, 2, 0]$ is equally good as a solution $\mathbf{y}'' = [0, 2, 4]$).

monotony A outcome improving the value of one of the objectives is preferred, the values of other objectives are not deteriorated (e.g. $\mathbf{y}' = [3, 2, 0]$ is preferred to $\mathbf{y}'' = [4, 2, 0]$).

principle of transfers A transfer of any small amount from an outcome to any other relatively worse-off outcome results in a more preferred outcome vector (e.g. $\mathbf{y}' = [3, 2, 1]$ is preferred to $\mathbf{y}'' = [4, 2, 0]$).

A feasible solution $\mathbf{x} \in Q$ is an *equitably efficient* solution of MOP, iff there does not exist any $\mathbf{x}' \in Q$ such that $\mathbf{f}(\mathbf{x}')$ is preferred to $\mathbf{f}(\mathbf{x})$ by any equitable rational preference relation. Note that equitably efficient solutions are a subset of Pareto-optimal solutions to a MOP.

An Equitable Multicriteria Optimization Problem (EMOP) can be formulated on the basis of a MOP, such that the Pareto-optimal solutions to the EMOP will be equitably efficient solutions to the MOP. The original vector of outcomes of MOP is transformed into a vector of *cumulative ordered outcomes*. First, introduce the ordering map $\Theta : R^m \rightarrow R^m$ such that $\Theta(\mathbf{y}) = (\theta_1(\mathbf{y}), \theta_2(\mathbf{y}), \ldots, \theta_m(\mathbf{y}))$, which sorts an outcome vector $\mathbf{y}$ according to non-increasing values of individual outcomes. Next, apply a linear cumulative map to $\Theta(\mathbf{y})$, resulting in the *cumulative ordering map* $\bar{\Theta}(\mathbf{y}) = (\bar{\theta}_1(\mathbf{y}), \bar{\theta}_2(\mathbf{y}), \ldots, \bar{\theta}_m(\mathbf{y}))$ defined as $\bar{\theta}_i(\mathbf{y}) = \sum_{j=1}^{i} \theta_j(\mathbf{y})$ for $i = 1, \ldots, m$. The coordinates of vector $\bar{\Theta}(\mathbf{y})$ express, respectively: the largest outcome, the total of the two largest outcomes, the total of the three largest outcomes, etc.

The following theorem is a consequence of results from the theory of majorization:

Theorem 1 *[13] Outcome vector $\mathbf{y}' \in Y$ equitably dominates $\mathbf{y}'' \in Y$, iff $\bar{\theta}_i(\mathbf{y}') \leq \bar{\theta}_i(\mathbf{y}'')$ for all $i \in I$ where at least one strict inequality holds.*

Note that Theorem 1 permits to express equitable efficiency for MOP in terms of the Pareto-optimality for the EMOP:

Corollary 1 *A feasible solution $\mathbf{x} \in Q$ is an equitably efficient solution of the MOP, iff it is a Pareto-optimal solution of the EMOP, the problem with objectives $\bar{\Theta}(\mathbf{f}(\mathbf{x}))$:* $\min \{(\bar{\theta}_1(\mathbf{f}(\mathbf{x})), \ldots, \bar{\theta}_m(\mathbf{f}(\mathbf{x}))) \;:\; \mathbf{x} \in Q\}$

3.2 Equitable Walk (EW)

EW is a heuristics which produces a number of grid schedules by iterative modifications of the initial SPT schedule in order to improve the outcome of the organization disfavored by the SPT schedule. The resulting schedules are possibly equitable: the algorithm may produce non-equitable schedules and not all possible equitable schedules may be produced.

The algorithm modifies the schedules by *switching* the order of two jobs executed one after another on the same processor. Let us assume that job $J_{i,q}^k$ is executed before $J_{j,q}^l$. The *deterioration* from a particular switch can be defined as the difference between the decrease of C_j and the increase of C_i. For instance, if $p_{i,q}^k = 3$ and $p_{j,q}^l = 4$, the switch of order of those two jobs will reduce C_j by 3 and increase C_i by 4. The deterioration of that particular switch is thus 1.

Corollary 2 *A schedule which orders jobs on all processors according to SPT (regardless of their owners) is equitable.*

This is a direct consequence of Theorem 1. Such schedule (denoted $\mathbf{SPT}$) is optimal with regard to the sum of completion times of all jobs on all processors ($\sum_i \sum_q \sum_k C_{i,q}^k$). Therefore, it is also the minimal sum of sum of completion times of respective organizations ($\sum_i C_i$). Hence, $\mathbf{SPT}$ is Pareto-optimal solution of the aggregated equitable optimization problem, as it minimizes the sum of all criteria.

The algorithm starts with ordering jobs $\mathcal{J}_{\cdot,q}$ on every processor P_q according to SPT. Then, the organization O_i with the largest C_i is selected. For each O_i's remote job $J_{i,q}^k$, the algorithm tentatively advances the job by switching $J_{i,q}^k$ with the job executed immediately before. Similarly, each foreign job $J_{\cdot,i}^k$ executed on O_i's local processor P_i which is followed by a local job $J_{i,i}^l$ is tentatively delayed by switching $J_{\cdot,i}^k$ with $J_{i,i}^l$. From all the tentative moves performed, the one which results in the smallest deterioration is actually performed. Then, the grid schedule is appended to the list of results.

The algorithm iterates such moves until either C_i is no longer the largest completion time, or no further improvement is possible. In the first case, the algorithm can proceed with improving the payoff of the other organization, in the purpose of producing more solutions. However, in order not to introduce infinite loops, a taboo list of previously visited schedules must be kept (such schedules cannot be revisited).

The output of the algorithm is a list of grid schedules. The first ($\mathbf{SPT}$) and the second schedule produced are definitely equitable. However, the equitableness of the rest of the schedules is not guaranteed. Therefore, after the above algorithm stops, the list of solutions is cleaned. All the solutions which are not equitably-optimal wrt. other solutions on the list are removed. Moreover, the algorithm may not produce all possible equitable solutions (the equitable part of the Pareto-front), because it may be trapped in a local optimum. We provide an experimental analysis of the algorithm in Section 5.

4 Game-Theoretic Approach

In this section we assume that each organization O_i controls its local processor P_i and therefore is able to impose the scheduling of jobs $\mathcal{J}_{\cdot,i}$. Each organization wants to minimize the sum of completion times C_i of its jobs $\mathcal{J}_{i,\cdot}$. However, C_i depends also on the completion time of O_i's remote jobs $\mathcal{J}_{-i,\cdot}$, which in turn depends on the schedules imposed by other organizations. Game theory [17] studies the problems in which players maximize their returns which partially depend on actions of other players. Consequently, it seems to be an adequate tool to study this model.

More formally, a *grid scheduling game* can be defined as a game in the normal form:

- the set of players is equal to the set of organizations $\mathcal{O}$;
- a strategy σ_i of a player O_i is an ordering of $\mathcal{J}_{\cdot,l}$ (following the local SPT rule from Proposition 1, as all the other strategies are Pareto-dominated);
- a payoff function for a player O_i resulting from a profile of strategies $\boldsymbol{\sigma} = [\sigma_1, \ldots, \sigma_N]$ is the reduction of the player's sum of completion times with comparison to the selfish outcome of the game: $u_i(\boldsymbol{\sigma}) = C_i(\mathbf{MJF}) - C_i(\boldsymbol{\sigma})$.

In the first place, O_i can use a greedy *My Jobs First* (MJF) strategy, which schedules all the local jobs $\mathcal{J}_{i,i}$ before any foreign job. Given any strategies of the rest of organizations, MJF strategy will reduce the total finish time C_i, by advancing local jobs $\mathcal{J}_{i,i}$. Therefore, MJF is the *prudential strategy* for each player. However, a solution in which every organization schedules jobs according to MJF (denoted as $\mathbf{MJF}$) very often is not optimal and it is Pareto-dominated by other solutions. The goal of the grid scheduler is to propose a schedule at least Pareto-dominating $\mathbf{MJF}$. That is why we define the payoff $u_i(\boldsymbol{\sigma})$ for each player O_i as the *gain over* **MJF** for that player. The payoffs defined in such a way must be maximized. Any profile of strategies $\boldsymbol{\sigma}$ resulting, for a player O_i, in $u_i(\boldsymbol{\sigma}) < 0$ is not feasible. The proposed total execution time for O_i's jobs is greater than the longest possible total execution time when O_i decides to order jobs on its processor according to MJF. Therefore, O_i would play MJF, which would also reduce other payoffs.

We will analyze this game from two different perspectives. Firstly, in Section 4.1, we assume that there is no cooperation between the players. The grid scheduler proposes a schedule (a strategy) for each player. However, players are not obliged to follow the strategies proposed. We will show that such a game is analogous to the well-known Prisoner's Dilemma (PD) game. Secondly, in Section 4.2, we increase the power of the grid scheduler. If the scheduler proposes a schedule resulting in positive payoff for each player, the players must follow such schedule. We show how to chose a fair schedule in such setting.

4.1 Non-cooperative Game

Let us assume that, for a particular job configuration $\mathcal{J} = \{J_{i,q}\}$, a grid scheduler is able to produce schedule $\boldsymbol{\sigma^*} = [\sigma_1^*, \ldots, \sigma_n^*]$, which results in positive payoff $u_i(\boldsymbol{\sigma^*}) > 0$ for each player. Consequently, there must be at least one player O_i, for whom the proposed strategy σ_i^* is different than MJF. Thus, in P_i's schedule, there is at least one foreign job $J_{j,i}^k$ scheduled before a local job $J_{i,i}^l$. If O_i decides to switch the order of execution of those two jobs, the local job $J_{i,i}^l$ will be finished faster and, consequently, the player's payoff u_i will increase, assuming that the rest of the players orders their jobs according to $\boldsymbol{\sigma^*}$. At the same time the payoff of the owner of the delayed job u_j will decrease. It follows that the strategy maximizing u_i is MJF, given that the others play any profile of strategies $\boldsymbol{\sigma}$. Additionally, if all the other players play MJF, the only strategy which guarantees non-negative u_i for O_i is to play MJF as well. Accordingly, **MJF** is the only Nash equilibrium of the one shot, non-cooperative grid scheduling game.

Consequently, the game is analogous to the multi-player Prisoner's Dilemma (PD). The equivalent of the mutual collaboration in PD is when organizations follow scheduler's suggestion $\boldsymbol{\sigma^*}$. When one of the organizations O_i play $\sigma_i = \text{MJF}$, whereas the others play σ^* (a single betrayal in PD), O_i's payoff increases $u_i(\text{MJF}, \sigma_{-i}^*) > u_i(\sigma^*)$, and the others loose. If everyone plays MJF (multiple betrayal in PD), resulting payoff for each organization is 0.

When each player plays MJF, the resulting payoffs are 0, therefore everyone feels a performance drop. An instance (Figure 2) shows that this drop can be quite high. In the depicted instance, two processors P_1 and P_2 have identical loads: one local job of size $p_{1,1}^1 = p_{2,2}^1 = p$ and $n_{2,1} = n_{1,2}$ foreign jobs, each of size $p_{2,1}^k = p_{1,2}^k = \epsilon$. A **MJF** schedule results in sum of completion times of:

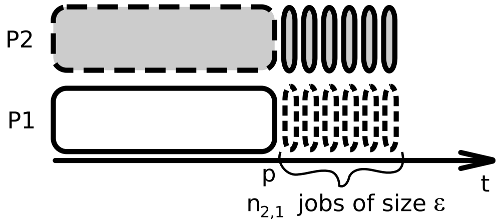

Figure 2. An instance with one large local job and n small foreign jobs on each processor which leads to a price of anarchy in $O(n)$.

$$C_1(\mathbf{MJF}) = p + (p+\epsilon) + (p+2\epsilon) + \cdots + (p + n_{1,2}\epsilon) =$$
$$= (n_{1,2}+1)p + \frac{1}{2}\epsilon n_{1,2}(n_{1,2}+1),$$

whereas an optimal schedule, in which the small foreign jobs are executed before the local job on both processors, results in total completion time of:

$$C_1^* = (\epsilon + 2\epsilon + \ldots + n_{1,2}\epsilon) + (n_{1,2}\epsilon + p) =$$
$$= \frac{1}{2}\epsilon n_{1,2}(n_{1,2}+1) + (n_{1,2}\epsilon + p).$$

The *price of anarchy* PoA is the ratio between the result in the worst Nash equilibrium and the socially-optimal result. Note that the social optimum are SPT schedules. Note also that in this instance, the social optimum is equal to the best feasible result, but it is not true in the general case. The price of anarchy is equal to:

$$PoA = \frac{2C_1(\mathbf{MJF})}{2C_1^*} \underset{\epsilon \to 0}{\longrightarrow} 2n + 2.$$

This result proves that the price the grid pays for the lack of control is very high.

Note that in real-world girds scheduling will be repeated, in function of new jobs arriving (such systems cannot be offline). This would lead to multiple iterations of the game presented above. In Infinitely Repeated PD, cooperation (following $\boldsymbol{\sigma^*}$) becomes profitable, as a single betrayal (playing MJF) can be punished during the next rounds by other players (by refusing to cooperate with the free-rider). Consequently, finding a good $\boldsymbol{\sigma^*}$ becomes an important problem, analyzed in the next section. However, the scheduling game will be not repeated infinitely, as collaboration in the Grid (e.g. in one Virtual Organization) is rather short-term, therefore the "shadow of the future" is reduced. In PD repeated N times (N known to players), defection is again the only Nash equilibrium.

4.2 Cooperative Game

In this section, we assume that the community, represented by the centralized grid scheduler, is able to impose a scheduling on the players, if the resulting payoffs $\boldsymbol{u}^* = [u_1, \dots, u_n]$ are *feasible*, i.e. $u_i \geq 0$ for each organization O_i. This guarantees that C_i does not rise in comparison with $C_i(\mathbf{MJF})$. Moreover, $\boldsymbol{u}^*$ should be equitably-optimal, as no organization should be preferred *a priori*.

To produce such solutions, we may use EW, the same method as in Section 3. EW must be adjusted to maximize u_i, instead of minimizing C_i. We will call the adjusted algorithm Game-theoretic Equitable Walk (GEW). GEW starts with a $\mathbf{SPT}$ schedule. Note that such schedule is still equitable (Corollary 2 holds), as $\sum_i u_i = \sum_i C_i(\mathbf{MJF}) - \sum_i C_i$; $\sum_i C_i(\mathbf{MJF})$ is constant and $\mathbf{SPT}$ minimizes $\sum_i C_i$, therefore maximizing the sum of all the variables. However, $\mathbf{SPT}$ may not be feasible, and, generally, GEW may be unable to produce a feasible result.

However, if players are able to form coalitions which exclude some organizations, a solution guaranteeing, for all O_i, $u_i^* \geq 0$, may not be feasible . For instance, two organizations O_i and O_j may decide to cooperate only bilaterally, executing jobs owned by other organizations after their jobs (i.e. after $\mathcal{J}_{i,i} \cup \mathcal{J}_{i,j} \cup \mathcal{J}_{j,i} \cup \mathcal{J}_{j,j}$) if the resulting payoffs u_i' and u_j' are greater than the respective payoffs when everyone cooperates ($u_i' > u_i^*$ and $u_j' > u_j^*$). In game-theoretic terms, there is no guarantee that $\boldsymbol{u}^*$ belongs to the *core* of the game. Such situations are analyzed by the cooperative n-person game theory. However, this theory usually assumes that that there are side payments, by which the coalition containing all the players may compensate O_i and O_j for cooperating with everyone. In our system, side payments may be difficult to realize without introducing external forms of payments (money). We are currently studying how to simulate such payments by choosing another grid schedule $\boldsymbol{\sigma}^{**}$, which brings the payoffs u_i^{**} and u_j^{**} closer to, respectively, u_i' and u_j'.

5 Experiments

In this section we will present preliminary experiments demonstrating that EW and GEW deliver satisfactory results. We compared the solutions produced by both algorithms to the set of equitably optimal solutions extracted from the results of the exhaustive search (matching all Pareto-optimal solutions for each processor). Consequently, the instances considered are quite small, as exhaustive search has an exponential complexity.

The number of jobs $n_{i,q}$ and jobs' sizes $p_{i,q}^k$ are uniformly distributed over, respectively, $\{1, \dots, \max n_{i,q}\}$ and $\{1, \dots, \max p_{i,q}^k\}$ ($\max n_{i,q}$ and $\max p_{i,q}^k$ are parameters of the experiment). For each combination of parameters, 100 instances were generated. There were 2 organizations. Table 1 presents aggregated results: number of instances in which at least one solution produced by the algorithm was equitably-dominated by one of the exhaustive search solutions (separately for EW and GEW); and a number of instances in which GEW produced at least one feasible solution (either a solution Pareto-dominating $\mathbf{MJF}$, or $\mathbf{MJF}$ in instances in which there were no exact solutions Pareto-dominating $\mathbf{MJF}$). The first two factors measure the equity achieved by EW and GEW, the last one the "fail-safe" mechanism of GEW (it is better to have a feasible solution, even though it is not equitable, than no solution at all).

Table 1. Experimental validation of EW and GEW. Each row is an aggregation over 100 randomized instances.

parameters		# inst. equitably dom		# inst. w/feasible
max $n_{i,q}$	max $p_{i,q}^k$	EQ	GEW	GEW solution
3	5	2	2	99
3	10	3	4	100
3	20	4	5	100
3	50	10	7	100
4	5	4	3	100
4	10	16	11	100
4	20	12	16	100
4	50	16	12	100
5	5	1	4	99
5	10	14	16	99
5	20	33	28	100
5	50	26	24	100

We can see that EW produces acceptable results. Firstly, in slightly less than 90% of instances both algorithms produced only equitable results (although a certain degradation of results can be seen with the increase in $\max n_{i,q}$ and, more surprisingly, $\max p_{i,q}^k$). Secondly, in all but 3 instances in which it was possible to dominate $\mathbf{MJF}$, GEW produced at least one feasible solution.

6 Related Work

The mainstream of the current research on scheduling and resource management [7] concerns systems in which the performance of all jobs is optimized. Usually, a common metric, such as the makespan C_{max}, or the sum of completion times ΣC_i is optimized and thus all the jobs are treated in a more or less equal manner.

In Grid computing, *multi-criteria* approaches may be used. Different criteria usually express performance of different jobs [16]. A scheduling algorithm is expected to deliver Pareto-optimal solutions. In Section 3 we argue that further restrictions on Pareto-optimal solutions should be imposed in order to achieve more equitable solutions.

Grid economic approaches [5][12] analyze the problem of resource management by market economy. Each resource has a (monetary) cost for its usage. Each user has a budget to spend on executing his/her jobs. The problem is that in highly heterogeneous settings the perfect competition assumption, stating that no single participant is able to influence the market price, is hardly fulfilled. Real-world grids are expected to be heterogeneous [11]. Our solutions solve the problem of scheduling in heterogeneous systems directly, without relying on free-market assumptions.

There were also some applications of game theory to the problem of grid resource management. [14] proposes a model where individual clusters (placed in e.g. different departments of an university) are visible as a one site in the grid. The model assumes that a job has been already accepted for the execution by the site. [14] studies which cluster from that site should eventually execute the job. Others [2][15] considered the infrastructure as a common property and the selfishness between individual jobs. We claim that our system models better academic grids, in which a job is viewed through the organization that has submitted it.

7 Conclusions and Future Work

In this work we have addressed the problem of resource management in highly heterogeneous computational grids. Our model emphasizes the miscellaneous nature of both the resources (by considering dedicated processors) and the users (by introducing the notion of an organization). We analyzed the model using two approaches: equitable optimization, when the grid scheduler has a complete control over all the local schedulers; and game theory, when local schedulers can alter their schedule. We proposed a simple heuristics to find solutions in both cases. Then we demonstrated that the complete decentralization may lead to a significant performance loss of the system. Therefore, a strong control performed by the grid community is indispensable.

We analyzed an off-line, clairvoyant problem, assumptions which might be considered unrealistic (yet they are quite normal in the scheduling theory). However, the goal of this research was rather to demonstrate certain phenomena than to propose a full-featured grid scheduler. On-line systems may use off-line algorithms to schedule jobs in batches. Partial clairvoyance can be acheived by combining users' run-time estimates with a prediction mechanism.

Our future work includes firstly, addressing the issue of cooperative games with players forming arbitrary coalitions, and secondly, providing efficient distributed algorithms for producing equitable solutions. Eventually, we plan to extend the theoretical results to computational grids composed of related multiprocessors (such as cluster computers of different architectures) and then to implement the solutions in real-world grid schedulers.

Acknowledgements Krzysztof Rzadca would like to thank prof. W. Ogryczak for helpful discussions on equity.

References

[1] A. Agnetis, P. Mirchandani, D. Pacciarelli, and A. Pacifici. Scheduling Problems with Two Competing Agents. *Operations Research*, 52(2):229–242, 2004.

[2] E. Angel, E. Bampis, and F. Pascual. The price of approximate stability for a scheduling game problem. In *Proceedings of Euro-Par*, volume 4128 of *LNCS*. Springer, 2006.

[3] J. Blazewicz. *Scheduling in Computer and Manufacturing Systems*. Springer, 1996.

[4] K. H. Buetow. Cyberinfrastructure: Empowering a third way in biomedical research. *Science*, 308:821–824, 2005.

[5] R. Buyya, D. Abramson, and S. Venugopal. The grid economy. In *Special Issue on Grid Computing*, volume 93, pages 698–714. IEEE Press, 2005.

[6] P. Dutot, L. Eyraud, G. Mounié, and D. Trystram. Scheduling on large scale distributed platforms: from models to implementations. *Int. J. Found. Comput. Sci.*, 16(2):217–237, 2005.

[7] D. G. Feitelson, L. Rudolph, and U. Schwiegelshohn. Parallel job scheduling a status report. In *Proceedings of JSSPP 2004*, volume 3277 of *LNCS*, pages 1–16. Springer, 2005.

[8] I. Foster. What is the grid. `http://www-fp.mcs.anl.gov/˜foster/Articles/WhatIsTheGrid.pdf`.

[9] I. Foster. Globus toolkit version 4: Software for service-oriented systems. In *IFIP International Conference on Network and Parallel Computing*, volume 3779 of *LNCS*, pages 2–13. Springer, 2005.

[10] I. Foster and C. Kesselman, editors. *The Grid 2. Blueprint for a New Computing Infrastructure*. Elsevier, 2004.

[11] R. Gruber, V. Keller, P. Kuonen, M.-C. Sawley, B. Schaeli, A. Tolou, M. Torruella, and T.-M. Tran. Towards an intelligent grid scheduling system. In *Proceedings of PPAM 2005*, volume 3911 of *LNCS*, pages 751–757, 2006.

[12] C. Kenyon and G. Cheliotis. Grid resource commercialization: economic engineering and delivery scenarios. In J. Nabrzyski, J. M. Schopf, and J. Weglarz, editors, *Grid resource management: state of the art and future trends*, pages 465–478. Kluwer Academic Publishers, Norwell, MA, USA, 2004.

[13] M. M. Kostreva, W. Ogryczak, and A. Wierzbicki. Equitable aggregations and multiple criteria analysis. *European Journal of Operational Research*, 158:362–377, 2004.

[14] Y.-K. Kwok, S. Song, and K. Hwang. Selfish grid computing: Game-theoretic modeling and nas performance results. In *Proceedings of CCGrid*, 2005.

[15] J. Liu, X. Jin, and Y. Wang. Agent-based load balancing on homogeneous minigrids: Macroscopic modeling and characterization. *IEEE Trans. on Parallel and Distributed Systems*, 16(7):586–598, 2005.

[16] L. Marchal, Y. Yang, H. Casanova, and Y. Robert. A realistic network/application model for scheduling divisible loads on large-scale platforms. *Proceedings of IPDPS*, 01:48b, 2005.

[17] M. J. Osborne. *An Introduction to Game Theory*. Oxford, 2004.

[18] J. Rawls. *The Theory of Justice*. Harvard Univ. Press, 1971.

[19] H. P. Young. *Equity: In Theory and Practice*. Princeton University Press, 1994.