Asynchronous Congestion Games

(Short Paper)

Michal Penn
Technion IIT
Haifa 32000, Israel
mpenn@ie.technion.ac.il

Maria Polukarov[*]
University of Southampton
SO17 1BJ, Southampton, UK
mp3@ecs.soton.ac.uk

Moshe Tennenholtz
Technion IIT
Haifa 32000, Israel
moshet@ie.technion.ac.il

ABSTRACT

We introduce a new class of games, *asynchronous congestion games* (ACGs). In an ACG, each player has a task that can be carried out by any element of a set of resources, and each resource executes its assigned tasks in a random order. Each player's aim is to minimize his expected cost which is the sum of two terms – the sum of the fixed costs over the set of his utilized resources and the expected cost of his task execution. The cost of a player's task execution is determined by the earliest time his task is completed, and thus it might be beneficial for him to assign his task to several resources. We prove the existence of pure strategy Nash equilibria in ACGs. Moreover, we present a polynomial time algorithm for finding such an equilibrium in a given ACG.

Categories and Subject Descriptors

C.2.4 [**Computer Systems Organization**]: Distributed Systems; I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence —*multiagent systems*

General Terms

Algorithms, Theory, Economics

Keywords

Congestion games, Asynchronous distributed systems, Pure-strategy Nash equilibrium, Algorithms

1. INTRODUCTION

Congestion games received a lot of attention in the recent game theory and computer science literature [4, 5, 8, 9, 10]. In a classic congestion game [14], each player chooses a subset of a set of available resources in order to perform his task. The cost of using a particular resource is determined by its congestion. The important property of congestion games is that they possess pure strategy Nash equilibria. Monderer and Shapley [10] introduced the notions of *potential function* and *potential game* and proved that the existence of a potential function implies the existence of a pure strategy Nash equilibrium. They also showed that the classes of finite potential games and congestion games coincide.

Classic congestion games can be viewed as synchronous: the cost suffered by a player when selecting a particular resource is determined only by the number of users who have chosen that resource, and does not take into account the actual *order* in which the assigned tasks are executed. In this paper we present a new class of games – *asynchronous congestion games* (ACGs) – that models noncooperative congestion settings in which resources execute their assigned tasks in a randomly chosen order. The random order of task execution reflects, for instance, a situation where players and resources are the elements of an asynchronous distributed system, in which each process has its own independent clock[1].

In ACGs, we consider a finite set of players, each having a unit length task that can be carried out by any element of a finite set of independent resources (machines). Each resource executes its assigned tasks in a randomly chosen order. As a result, a player may selfishly assign his task to several resources, hoping that his task will be completed in a short time by at least one of the resources. It is assumed that resource usage is costly; that is, every player has to pay for utilizing each of his chosen resources. More specifically, a player's aim is to minimize his expected total cost which is composed by the sum of the fixed costs over the set of his chosen resources and the cost of his task execution which is determined by the *minimum* completion time of his task by any of his chosen resources.

By considering the order of task execution, the study of ACGs is related to the literature on selfish scheduling. There are two types of selfish scheduling: scheduling involving *selfish machines* [2, 6, 12] in which resources attempt to optimize their own objectives, and scheduling involving *selfish tasks* [1, 3, 7] in which each participant's objective is to minimize the completion time of his task. The latter type is closely related to congestion games. In particular, a model in which a player is interested in minimizing the sum of the completion times of his tasks over the set of his resources, can be viewed as a variant of weighted congestion games.

Introducing a new class of games raises the important question of the existence of pure strategy equilibria as well as the computation of such equilibria. There are only few known classes of games which possess pure strategy equilibria, and there seems to be relatively little work providing efficient and exact algorithms for computing such equilibria. In this paper we introduce the class of ACGs and prove that these games possess a Nash equilibrium in pure strategies,

[*]The work was done when the author was a Ph.D. student at the Technion IIT, Haifa 32000, Israel.

Cite as: Asynchronous Congestion Games (Short Paper), Michal Penn, Maria Polukarov and Moshe Tennenholtz, *Proc. of 7th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2008)*, Padgham, Parkes, Müller and Parsons (eds.), May, 12-16., 2008, Estoril, Portugal, pp. XXX-XXX.

[1]The idea of using random ordering in order to reflect the asynchronous nature of processes in distributed systems is discussed, for example, in Monderer and Tennenholtz [11].

despite the non-existence of a potential function. In addition, we present a polynomial time algorithm for finding such an equilibrium in a given ACG.

The rest of the paper is organized as follows. In Section 2 we define our model. In Sections 3, 4 and 5 we present our results. In 3 we show that ACGs are not potential games. In Section 4 we prove that every ACG possesses a pure strategy Nash equilibrium, despite the non-existence of a potential function. In Section 5 we present an $O(nm^2)$ algorithm for computing such an equilibrium. We conclude in Section 6. Due to lack of space the proofs are omitted.

2. THE MODEL

Let $N = \{1, \ldots, n\}$ be a set of n players and let $M = \{e_1, \ldots, e_m\}$ be a set of m resources. Player $i \in N$ chooses a strategy $\sigma_i \in \Sigma_i$ which is a *nonempty* subset of the resources: $\Sigma_i = P(M) \smallsetminus \{\varnothing\}$. Given a subset $S \subseteq N$ of the players, the set of strategy combinations of the members of S is denoted by $\Sigma_S = \times_{i \in S} \Sigma_i$, and the set of strategy combinations of the complement subset of players is denoted by Σ_{-S} ($\Sigma_{-S} = \Sigma_{N \smallsetminus S} = \times_{i \in N \smallsetminus S} \Sigma_i$). The set of pure strategy profiles of all the players is denoted by Σ ($\Sigma = \Sigma_N$). Let $\sigma = (\sigma_1, \ldots, \sigma_n) \in \Sigma$ be a strategy profile. The (m-dimensional) *congestion vector* that corresponds to σ is $h(\sigma) = (h_e(\sigma))_{e \in M}$, where $h_e(\sigma) = \left|\{i \in N : e \in \sigma_i\}\right|$.

The *outcome* for player $i \in N$ from σ is the vector $x^i(\sigma) = \left(x^i_e(\sigma)\right)_{e \in M} \in \{1, \ldots, n, \infty\}^m$ of the ordering numbers of player i's task on all the resources, where $x^i_e(\sigma) \in \{1, \ldots, n\}$ for $e \in \sigma_i$ and $x^i_e(\sigma) = \infty$ for $e \notin \sigma_i$. The player's objective is to minimize his total cost that consists of the sum of the fixed costs over the set of resources he uses and the cost of the player's task execution. The fixed cost for utilizing each of the resources equals t units of money. The cost of task execution is a nonnegative, nondecreasing function of its completion time; thus, the longer it takes to complete the task execution, the greater is the cost incurred by the player. We assume that each player pays a fixed price, say c, for a unit of time his task is in the system before completed by at least one of the resources and, w.l.o.g., that this cost is one unit of money per unit of time. That is, the cost of a player's task execution is determined by the *minimum* among the completion times of his task by his chosen resources. Hence, the *cost* to player i from a strategy profile σ and his outcome $x^i(\sigma)$, is defined as $c_i\left(\sigma, x^i(\sigma)\right) = \min_{e \in \sigma_i} x^i_e(\sigma) + |\sigma_i| t$. Given a strategy profile σ, for any player $i \in N$ and resource $e \in \sigma_i$, let $X^i_e(\sigma)$ denote a random variable representing the ordering number of player i's task on resource e. Since it is assumed that each task requires a unit of time to be processed and each unit of time costs one unit of money, $X^i_e(\sigma)$ represents the cost to player i for his task execution by resource e. We assume that $X^i_e(\sigma)$ is uniformly distributed over $\{1, \ldots, h_e(\sigma)\}$. The *expected cost* of player i from strategy profile σ, $C_i(\sigma)$, is therefore:

$$C_i(\sigma) = E\left(\min_{e \in \sigma_i} X^i_e(\sigma)\right) + |\sigma_i| t = \sum_{q=1}^{\min_{e \in \sigma_i} h_e(\sigma)} \prod_{e \in \sigma_i} \frac{h_e(\sigma) - q + 1}{h_e(\sigma)} + |\sigma_i| t.^2$$

The aim of a player is to minimize his own expected cost.

3. POTENTIAL FUNCTION

Here we discuss the (non-)existence of a potential function in ACGs. In particular, in Theorem 1 below we show that any ACG with more than two players or resources is not a potential game. Hence, ACGs are not congestion games.

THEOREM 1. *Any 2×2 ACG is a potential game but **no** ACG with $n > 2$ players or $m > 2$ resources possesses a potential function*

4. THE EXISTENCE OF A PURE STRATEGY NASH EQUILIBRIUM

Despite the fact that ACGs, in general, are not potential games, in this section we prove that every ACG possesses a Nash equilibrium in pure strategies. If the number of resources is greater than or equal to the number of players ($m \geq n$) then the profile $\sigma = (e_i)_{i \in N}$ is a Nash equilibrium as well as an optimal strategy (one that minimizes the sum of the players' expected costs). If $m < n$ then proving the existence of such an equilibrium is quite complicated, as is demonstrated below.

Our proof uses the notion of stability under single moves, previously presented in [13], and proceeds as follows. We define three types of single moves (A-, D- and S-moves) and prove that a profile which is stable under all these moves is a Nash equilibrium (see Lemma 2). We observe that the DS-stable[3] profile is easy to find, but the existence of a profile which is stable under all three types of single moves is not obvious (see Lemma 3 and the discussion following it). We look for such a profile using two special types of A-moves ("one- and two-step additions", to be defined in the sequel). Lemma 4 furthermore describes how these additions affect DS-stable profiles. Based on this lemma, we prove that for some DS-stable profiles the above additions do not ruin the DS-stability (see Lemma 5). We complete our proof by showing that applying a finite series of addition operations to such a profile results in an equilibrium (see Lemma 6 and Corollary 7). We turn now to our results.

In a congestion setting, we are mainly interested in three types of *single moves*, where each type is a deviation involving a single resource, as follows.

DEFINITION 1. *For any strategy profile $\sigma \in \Sigma$ and for any player $i \in N$, the operation of adding precisely one resource to his strategy, σ_i, is called an **A-move** of i from σ. Similarly, the operation of dropping a single resource is called a **D-move**, and the operation of switching one resource with another is called an **S-move**.*

Lemma 2 below implies that any strategy profile in which no player wishes unilaterally to apply a single move, is a Nash equilibrium. This property is called the *single profitable move property* and it allows us to consider only single moves rather than considering all possible deviations.

LEMMA 2. ***(The single profitable move property)*** *Given an ACG, let $\sigma \in \Sigma$ be a strategy profile which is not in equilibrium, and let $i \in N$ be a player for which a profitable deviation from σ is available. Then, i has a profitable A-, D- or S-move from σ.*

By Lemma 2, in order to prove the existence of a pure strategy Nash equilibrium in games possessing the single profitable move property, it suffices to present a strategy profile for which no player wishes to unilaterally apply an A-, D- or S-move. This observation motivates the following definition.

[2]Note that if $t = 0$ then the dominant strategy of each player is to assign his task to all of the resources.

[3]A strategy profile which is stable under D- and S-moves (see Definition 2).

Definition 2. *A strategy profile σ is said to be* ***A-stable*** *(resp.,* ***D-stable, S-stable****) if there are no players with a profitable A- (resp., D-, S-) move from σ.*

In order to investigate stability under single moves in ACGs we use the notions of light and heavy resources as well as of even and nearly-even strategy profiles.

Definition 3. *Given a strategy profile σ, resource e' is called σ-**light** if $e' \in \arg\min_{e\in M} h_e(\sigma)$ and σ-**heavy** otherwise. A strategy profile σ with no heavy resources will be termed* ***even****. An even strategy profile with a common congestion of k on the resources will be termed* ***k-even****. A strategy profile σ satisfying $|h_e(\sigma) - h_{e'}(\sigma)| \leq 1$ for all $e, e' \in M$ will be termed* ***nearly-even****.*

Obviously, in a nearly-even strategy profile all heavy resources (if such exist) have the same congestion. Moreover, the notions of nearly-eveness and S-stability are strongly connected (see Lemma 3).

Lemma 3. *In an ACG, a strategy profile is S-stable if and only if it is nearly-even.*

Note that the pairwise intersections of the set of S-stable strategy profiles with the set of A-stable profiles or the set of D-stable profiles are not empty. In particular, the strategy profile $\sigma^M = (M, \dots, M)$ is A- and S-stable, while the profile $\sigma^0 = (e_{i \mod m})_{i\in N}$ is D- and S-stable (henceforth, "DS-stable"). However, at first glance, it is not clear whether there exists a profile which is stable under all three types of single moves. Intuitively, one can try to achieve a Nash equilibrium by selecting a profile which is stable under two types of single moves and applying on it a series of single moves of the third type. For instance, one can pick a DS-stable strategy profile and try to transform it into a Nash equilibrium by applying on it a series of profitable A-moves. However, simple examples show that such moves may destroy the D- or the S-stability of the selected profile; moreover, an A-move from the selected profile may initiate a long chain of D- and S-moves. Therefore, the chosen actions have to be picked out in a careful and subtle way. In this context, we first restrict the set of available A-moves to the subset of one- and two-step addition operations, as follows.

Let $\sigma \in \Sigma$ be a strategy profile, and for each player $i \in N$, let $e^i \in \arg\min_{e\in M\setminus\sigma_i} h_e$. That is, e^i is a lightest resource not previously chosen by i. Then, if there exists a profitable A-move for player i, then an A-move with e^i is profitable for i as well. If no player wishes to change his strategy in this manner, i.e. $C_i(\sigma) \leq C_i(\sigma_{-i}, \sigma_i + e^i)$ for all $i \in N$, then $C_i(\sigma) \leq C_i(\sigma_{-i}, \sigma_i + a)$ for all $i \in N$ and $a \in M \setminus \sigma_i$. Hence, σ is A-stable. Otherwise, let $N(\sigma)$ denote the subset of all players for which there exists e^i such that a unilateral addition of e^i is profitable. Let $a \in \arg\min_{e^i : i\in N(\sigma)} h_{e^i}$. Let also $i \in N(\sigma)$ be the player for which $e^i = a$. If a is σ-light, then let $\sigma' = (\sigma_{-i}, \sigma_i + a)$. In this case we say that σ' is obtained from σ by a *one-step addition* of resource a, and a is called an *added* resource. If a is σ-heavy then there exists a σ-light resource b and a player j such that $a \in \sigma_j$ and $b \notin \sigma_j$. Then let $\sigma' = \big(\sigma_{-\{i,j\}}, \sigma_i + a, \sigma_j - a + b\big)$. In this case we say that σ' is obtained from σ by a *two-step addition* of resource b, and b is called an *added* resource.

We notice that, in both cases, the congestion of each resource in σ' is the same as in σ, except for the added resource, with the congestion in σ' increased by 1. Thus, if σ is nearly-even then σ' is also nearly-even (since the added resource is σ-light). Then, Lemma 3 implies the S-stability of σ'. Lemma 4 below shows that if, in addition, σ is D-stable then the only potential cause for the non-D-stability of σ' is the existence of player $i \in N$ with $\sigma'_i = \sigma_i$ who wishes to drop the added resource a.

Lemma 4. *Let σ be a nearly-even and D-stable strategy profile of a given ACG, and let σ' be obtained from σ by a one- or two-step addition of resource a. Then, there are no profitable D-moves for any player $i \in N$ with $\sigma'_i \neq \sigma_i$. For $i \in N$ with $\sigma'_i = \sigma_i$, the only possible profitable D-move (if such exists) is to drop the added resource a.*

Note that although we did not succeed in keeping the D-stability, we have significantly reduced the set of possible post-addition D-moves. This motivates us to present the term of post-addition D-stability which plays a central role in our method, as follows.

Definition 4. *Let $\sigma \in \Sigma$ be a strategy profile and let σ' be obtained from σ by applying a one- or two-step addition operation. Then, based on Lemma 4, σ is said to be post-addition D-stable if σ' does not admit profitable D-moves with the added resource.*

Let $\Sigma^0 \subseteq \Sigma$ denote the subset of all D-stable strategy profiles, and let $\Sigma^1 \subseteq \Sigma^0$ be the subset of all even and D-stable strategy profiles. By Lemma 3, every profile in Σ^1 (if such exists) is S-stable. For any $\sigma \in \Sigma^1$, let h^σ denote the common congestion on the resources, and let $\Sigma^2 \subseteq \Sigma^1$ be the subset of Σ^1 consisting of all those profiles with maximum congestion on the resources: $\Sigma^2 = \arg\max_{\sigma\in\Sigma^1} h^\sigma$. Then,

Lemma 5. *Given an ACG, there exists a strategy profile $\sigma \in \Sigma^2$ that is either a pure strategy Nash equilibrium or post-addition D-stable.*

It is not clear, by first look, if the existence of a post-addition D-stable strategy profile implies the existence of a pure strategy Nash equilibrium. To show such an implication, post-addition D-stability should be preserved while applying a series of addition operations. In addition, such a series of addition operations should converge to a pure strategy Nash equilibrium in a finite number of steps. In this context, Lemma 6 and Observation 7 below provide the necessary steps for completing the proof of existence of a pure strategy equilibrium.

Lemma 6. *Given an ACG, let σ be a nearly-even and post-addition D-stable strategy profile, and let σ' be obtained from σ by applying on it a one- or two-step addition operation. If $\min_{e\in M} h_e(\sigma') = \min_{e\in M} h_e(\sigma)$ then σ' is also nearly-even and post-addition D-stable.*

Corollary 7. *By Lemma 6, if we can find a post-addition D-stable strategy profile σ' that lies in Σ^2, then a pure strategy Nash equilibrium can be achieved by applying on σ', in a sequential manner, less than m one-/two-step addition operations. This is because if we perform m addition operations then an even D-stable strategy profile σ'' with $h^{\sigma''} > h^{\sigma'}$ is obtained, contradicting $\sigma' \in \Sigma^2$.*

Theorem 8 below follows directly from Lemmas 5 and 6, and Corollary 7.

Theorem 8. *Every ACG possesses a Nash equilibrium in pure strategies.*

5. COMPUTATION OF A PURE STRATEGY NASH EQUILIBRIUM

We are now ready to present our Asynchronous Nash equilibrium (ANE)-algorithm that constructs a pure strategy

Nash equilibrium in any given ACG. Let us start with a brief description of the algorithm:

• Based on Lemma 6, the goal of the algorithm is to find a strategy profile in Σ^2 which is either a pure strategy Nash equilibrium or post-addition D-stable. In the latter case, the strategy profile can be turned into a Nash equilibrium by applying on it at most $m-1$ one-/two-step addition operations. For that, the algorithm has to determine a value $k^* = \max_{\sigma\in\Sigma^1} h^\sigma$ that represents the common congestion on the resources for any strategy profile in Σ^2.

• To find k^* as above, the algorithm uses a variable k initiated with the value $k = n$ and gradually decreases until k^* is found (Steps [0] – [1]).

• For $k = n$, the only even strategy profile with n being its common congestion is $\sigma = (M, \dots, M)$, which is obviously A- and S-stable. If σ is also D-stable then $k^* = n$, and the algorithm outputs σ and halts (Step [0]). Otherwise, $k^* < n$ and the algorithm proceeds with $k = n - 1$ (Step [1]).

• Given $\lfloor \frac{n}{m} \rfloor < k < n$, the algorithm checks whether a k-even D-stable strategy profile exists. If there is no such profile then $k^* < k$ and the algorithm proceeds with the next value of k (repeating Step [1]). Otherwise, $k^* = k$.

• If $k^* = \lfloor \frac{n}{m} \rfloor$ then the algorithm constructs a strategy profile $\sigma = (e_{i \bmod m})_{i\in N}$ (Step [2]). As we prove, σ is a Nash equilibrium.

• Otherwise, $k^* > \lfloor \frac{n}{m} \rfloor$. In this case, the algorithm constructs a k^*-even strategy profile σ with $n^* = n\left(\lfloor \frac{k^*m}{n} \rfloor + 1\right) - k^*m$ players using $\lfloor \frac{k^*m}{n} \rfloor$ resources and $n - n^* = k^*m - n\lfloor \frac{k^*m}{n} \rfloor$ players using $\lfloor \frac{k^*m}{n} \rfloor + 1$ resources (Step [3]). As we prove, the obtained σ is D- and S-stable. If σ is also A-stable then the algorithm outputs σ and halts (Step [4]). Otherwise, we show that $\sigma \in \Sigma^2$ and is post-addition D-stable. Then, based on Lemma 6 and Corollary 7, a pure strategy Nash equilibrium is achieved by applying at most $m-1$ one- or two-step additions on σ (Steps [5] – [9]).

The ANE-algorithm is presented below.

ANE-algorithm

```
Step [0]  If $t \le \sum_{q=1}^{n} \left(\frac{n-q+1}{n}\right)^{m-1} \frac{q-1}{n}$ then set
          $\sigma := (M, \dots, M)$ and QUIT;
          Otherwise, set $k := n-1$ and go to Step [1];
Step [1]  If $t > \sum_{q=1}^{k} \left(\frac{k-q+1}{k}\right)^{\lceil \frac{km}{n} \rceil - 1} \frac{q-1}{k}$ then set
          $k := k-1$; Otherwise go to Step [3];
Step [2]  If $k = \lfloor \frac{n}{m} \rfloor$ then set $\sigma := (e_{i \bmod m})_{i\in N}$
          and QUIT; Otherwise go to Step [1];
Step [3]  Set $n^* := n\left(\lfloor \frac{km}{n} \rfloor + 1\right) - km$;
          For $i = 1$ to $n^*$:
                Set $\sigma_i = \{e_r \in M : 1 \le r \le \lfloor \frac{km}{n} \rfloor\}$
                and reorder the resources:
                for all $e_r \in M$ set $e_r := e_{\left(r + \lfloor \frac{km}{n} \rfloor\right) \bmod m}$ ;
          If $n^* = n$ then go to Step [4];
          Otherwise, for $i = n^* + 1$ to $n$:
                Set $\sigma_i = \{e_r \in M : 1 \le r \le \lfloor \frac{km}{n} \rfloor + 1\}$
                and reorder the resources:
                for all $e_r \in M$ set $e_r := e_{\left(r + \lfloor \frac{km}{n} \rfloor + 1\right) \bmod m}$ ;
Step [4]  If $t \ge \sum_{q=1}^{k} \left(\frac{k-q+1}{k}\right)^{\lfloor \frac{km}{n} \rfloor} \frac{q-1}{k+1}$ then QUIT;
Step [5]  For all $i \in N$, select $e^i \in \arg\min_{e\in M \setminus \sigma_i} h_e(\sigma)$;
Step [6]  Set $N(\sigma) := \{i \in N : C_i(\sigma_{-i}, \sigma_i + e^i) < C_i(\sigma)\}$;
          If $N(\sigma) = \varnothing$ then QUIT;
Step [7]  Set $M(\sigma) := \{e \in M : \exists i \in N(\sigma), e = e^i\}$;
Step [8]  Select $a^* \in \arg\min_{e\in M(\sigma)} h_e(\sigma)$
          and $i^* \in \{i \in N(\sigma) : e^i = a^*\}$;
Step [9]  If $a^*$ is $\sigma$-light set $\sigma_{i^*} := \sigma_{i^*} + a^*$
          and go to Step [5];
          Otherwise select a $\sigma$-light resource $b^*$
          and $j^* \in \{i \in N : a^* \in \sigma_i, b^* \notin \sigma_i\}$,
          set $\sigma_{i^*} := \sigma_{i^*} + a^*$, $\sigma_{j^*} := \sigma_{j^*} - a^* + b^*$,
          and go to Step [5].
```

THEOREM 9. *The ANE-algorithm finds a pure strategy Nash equilibrium in any given ACG, and its time complexity is $O(nm^2)$.*

6. SUMMARY AND FUTURE WORK

In this paper, we introduced and investigated the class of asynchronous congestion games – ACGs – which extends the models of congestion games to allow for a random ordering of task execution. In an ACG, each player aims to minimize his own cost which is determined by the sum of two terms: the execution cost of his task which is assumed to be proportional to its completion time, and the sum of the fixed costs over the resources he uses. The completion time of the player's task is determined by the minimum among its completion times by all of his chosen resources.

We studied the existence of a pure strategy Nash equilibrium and a potential function in ACGs. We showed that only ACGs with 2 players and 2 resources are potential games, and any other ACG is not a potential game. Nevertheless, we proved that any ACG possesses a pure strategy Nash equilibrium. We presented a non-trivial polynomial time algorithm for constructing a pure strategy Nash equilibrium in a given ACG.

The model of ACGs can be extended in various ways. One can consider other probability distributions over the set of permutations (orders) of the tasks assigned to a particular resource. In addition, it will be a challenge to consider different processing times rather than these of single units. We believe that such extension will be significantly more difficult to analyze. It is also of interest to study the stability under deviations of coalitions and the social (in)efficiency of equilibria in ACGs.

7. REFERENCES

[1] E. Angel, E. Bampis, and F. Pascual. Truthful algorithms for scheduling selfish tasks on parallel machines. In *WINE-05*, pages 698–707, 2005.

[2] V. Auletta, R. Prisco, P. Penna, and G. Persiano. Determinisitc truthful approximation mechanisms for scheduling related machines. In *STACS-04*, pages 608–619, 2004.

[3] T. Carroll and D. Grosu. Selfish multi-user task scheduling. In *ISPDC-06*, pages 99–106, 2006.

[4] G. Christodoulou and E. Koutsoupias. The price of anarchy of finite congestion games. In *STOC-05*, pages 67–73, 2005.

[5] A. Fabrikant, C. Papadimitriou, and K. Talwar. The complexity of pure Nash equilibria. In *STOC-04*, pages 604–612, 2004.

[6] D. Grosu and T. Carroll. A strategyproof mechanism for scheduling divisible loads in distributed systems. In *ISPDC-05*, pages 83–90, 2005.

[7] E. Koutsoupias. Selfish task allocation. *Bulletin of EATCS*, 81:79–88, 2003.

[8] I. Milchtaich. Congestion games with player-specific payoff functions. *Games and Economic Behavior*, 13:111–124, 1996.

[9] D. Monderer. Solution-based congestion games. *Advances in Mathematical Economics*, 8:397–407, 2006.

[10] D. Monderer and L. Shapley. Potential games. *Games and Economic Behavior*, 14:124–143, 1996.

[11] D. Monderer and M. Tennenholtz. Distributed games. *Games and Economic Behavior*, 28:181–188, 1999.

[12] N. Nisan and A. Ronen. Algorithmic mechanism design. *Games and Economic Behavior*, 35(1/2):166–196, 2001.

[13] M. Penn, M. Polukarov, and M. Tennenholtz. Congestion games with load-dependent failures: identical resources. In *EC-07*, pages 210–217, 2007.

[14] R. Rosenthal. A class of games possessing pure-strategy Nash equilibria. *International Journal of Game Theory*, 2:65–67, 1973.