Partition Equilibrium Always Exists in Resource Selection Games*

Elliot Anshelevich, Bugra Caskurlu, and Ameya Hate

Department of Computer Science, Rensselaer Polytechnic Institute, Troy, NY.

Abstract. We consider the existence of Partition Equilibrium in Resource Selection Games. Super-strong equilibrium, where no subset of players has an incentive to change their strategies collectively, does not always exist in such games. We show, however, that partition equilibrium (introduced in [4] to model coalitions arising in a social context) always exists in general resource selection games, as well as how to compute it efficiently. In a partition equilibrium, the set of players has a fixed partition into coalitions, and the only deviations considered are by coalitions that are sets in this partition. Our algorithm to compute a partition equilibrium in any resource selection game (i.e., load balancing game) settles the open question from [4] about existence of partition equilibrium in general resource selection games. Moreover, we show how to always find a partition equilibrium which is also a Nash equilibrium. This implies that in resource selection games, we do not need to sacrifice the stability of individual players when forming solutions stable against coalitional deviations. In addition, while super-strong equilibrium may not exist in resource selection games, we show that its existence can be decided efficiently, and how to find one if it exists.

1 Introduction

In multi-agent systems, it is common to assume that the agents will change their existing behavior if they can reduce their cost by doing so. This assumption is at the heart of the study of Nash equilibrium in various settings. The concept of Nash equilibrium, however, becomes relevant only in scenarios where agents cannot form coalitions, and change their behavior as a group. The *Strong Equilibrium* [1] solution concept, where any subset of agents can form a coalition and deviate together if it is beneficial to all of them, addresses the weaknesses of the Nash equilibrium solution concept for the settings where players can form coalitions. A strong equilibrium represents the scenario where any group of players could form a coalition, and everyone has to strictly benefit from a deviation. In this paper, we relax these assumptions, and consider the cases where only some of the subsets of players could group themselves together into a deviating coalition, and where not everyone in a coalition has to strictly improve their utility in order to deviate.

* This work supported in part by NSF CCF-0914782.

We study these solution concepts in the context of Resource Selection Games (RSGs). RSGs model a wide range of scenarios, where a set of players are selecting exactly one of various resources, with the cost of using a resource depending on the type of the resource, as well as the number of players selecting this particular resource. They present a framework that can be used to model the problems of various communities like operations research, economics, computing systems, transportation, and communication networks. The atomic selfish routing game [2, 6, 13, 14] on parallel link networks and selfish machine assignment [3, 5, 10, 14] for identical jobs are among various problems modeled as RSGs in the algorithmic game theory community. RSGs fall into the class of potential games [11, 12] for which existence of a pure Nash equilibrium is guaranteed. Holzman and Law-Yone [8, 9] proved the existence of strong equilibrium in RSGs as well. However, *Super-Strong Equilibrium* (see below) is not guaranteed to exist in RSGs, which led Feldman and Tennenholtz [4] to define a concept of *Partition Equilibrium* and study its existence in the context of Resource Selection Games. In this paper, we greatly extend their results by showing the existence of partition equilibrium in *every* Resource Selection Game, as well as how to compute it.

1.1 Related Solution Concepts

The *Strong Equilibrium* (SE) solution concept assumes a coalition will deviate only if the deviation is strictly profitable to all members of the coalition. In a strong equilibrium, no subset of players is able to deviate with every player in the group strictly improving their utility.

Super-Strong Equilibrium (SSE) considers *weakly-profitable deviations*, where a coalition will deviate provided that no member of the coalition becomes worse off, and at least one member of the coalition strictly benefits. A super-strong equilibrium is a solution where no subset of players has such a deviation. This solution concept makes more sense in many settings, especially if agents will somewhat care about the utility of other agents (which perfectly make sense if the agents are friends, colleagues, family members). While strong equilibrium is guaranteed to exist in RSGs, there are RSG instances where super-strong equilibrium may not exist, even with 2 identical machines and 3 players [4]. Additionally, if we consider the formation of player coalitions as arising from a social context (i.e., a group of friends decide to form a coalition together), then the assumption that *any* subset of players can form a coalition is quite strong.

Partition Equilibrium was first defined in [4] as an attempt to model coalitions that arise from a social context. In this setting, the specification of the game contains a fixed partition T over the set of players. This partition divides the players into non-overlapping coalitions. In this solution concept, the only permissible deviations are the ones where a coalition is one of the sets in the fixed partition T. A solution is a stable solution if no coalition has an *weakly-profitable deviation*, i.e., a deviation where at least one member of the coalition strictly benefits and no member of the coalition becomes worse off. [4] called such a stable solution a *T-SSE*, since a partition equilibrium is a super-strong equilibrium, but with the only coalitions that are allowed to deviate being the

sets of partition T. Also observe that unlike strong equilibrium or super-strong equilibrium, partition equilibrium solutions are not a subset of Nash equilibrium solutions.

Feldman and Tennenholtz [4] studied the existence of partition equilibrium in the context of resource selection games and proved that partition equilibrium exists in the following special cases:

- All the resources are identical, i.e., they share the same latency function, or
- There are only 2 resources in the system, or
- Each coalition is composed of 1 or 2 players.

Note that partition equilibrium is a solution concept in a non-transferable utility game, i.e., money transfers among the players are not allowed. The *Collusion Equilibrium* solution concept [7] is the analogue of partition equilibrium in transferable utility games. In this solution concept, there is also a fixed partition over the players which forms the non-overlapping coalitions. The only difference is that money transfers among the players are permitted, and therefore a deviation is an improving deviation if it reduces the total cost of the players in the coalition. Observe that collusion equilibrium is a stronger solution concept in the sense that an allocation of players to resources that constitutes a collusion equilibrium (no coalition can reduce its total cost by deviating) is also a partition equilibrium allocation but not vice versa. Hayrapetyan, Tardos and Wexler [7] studied the existence and computation of collusion equilibrium in the context of resource selection games. They proved the existence of collusion equilibrium (and therefore, partition equilibrium) in the special case where the latency functions of the resources are convex. Their proof is constructive, i.e., they give an algorithm that produces a collusion equilibrium solution which may *not* be a Nash equilibrium solution.

1.2 Our Results

Our main result is the proof of existence (and efficient computation) of an allocation A of players to resources such that A is *both* a partition equilibrium and a Nash equilibrium allocation. This result holds for general resource allocation games, with no assumptions about the latency functions of different resources (except them being increasing), on the size of the coalitions, or on the number of machines. This resolves an open question from [4] about the existence of partition equilibrium for general RSGs. Moreover, our results provide the interesting insight that for every partition T there exists a solution where no coalition of T would gain by deviating (i.e., it is a T-SSE), *and* no single player would gain by deviating (i.e., it is a Nash equilibrium). This implies that we do not need to sacrifice the stability of individual players when forming solutions stable against coalitional deviations.

In Section 2, we present a formal definition of resource selection games and give a complete characterization of Nash equilibrium solutions for these games. In Section 3, we give a set of sufficient conditions for coalitions such that if a

coalition satisfies the given conditions on a Nash equilibrium allocation, then that coalition does not have an improving deviation. In Section 4, we give an algorithm that produces a Nash equilibrium allocation of players to resources such that all coalitions satisfy the sufficient conditions given in Section 3. In Section 5, we show that for any resource selection game instance, the existence of super-strong equilibrium is efficiently decidable, and if super-strong equilibrium exists, then we can compute one efficiently.

In summary, this paper shows that we can always find a SSE if one exists, but even for games which do not admit a SSE, we can find a solution that is stable for any set in a given partition T, as well as for any individual player.

2 Model and Preliminaries

We now formally define the resource selection game. We have n players (jobs) and m resources (machines). The strategy of each player is to select exactly one of the m machines. Each machine i has a strictly increasing latency function $f_i(n_i)$ which only depends on the number of players n_i that select machine i. The cost of each player that selected machine i is $f_i(n_i)$.

In this paper we will consider partition equilibrium and super-strong equilibrium (SSE), both of which are solution concepts involving stability against coalitional deviations. Specifically, by an *improving deviation* by a coalition of players C, we will mean a *weakly-profitable deviation*, i.e., a deviation where no player in C increases their cost, and at least one player of C strictly decreases their cost.

A SSE is an allocation of jobs to machines, so that no subset of jobs has an improving deviation. As shown in [4], a SSE does not always exist, although a strong equilibrium (where a deviation will only occur if every member of a coalition strictly profits) always exists in resource selection games [8, 9].

Now suppose that we have a fixed partition $T = T_1, \ldots, T_k$ over the set of players such that $T_i \cap T_j = \emptyset$, i.e., the sets are not overlapping. Each set T_i represents a coalition of players that are willing to deviate as a group. Then, a *partition equilibrium* or T-SSE is an allocation of jobs to machines such that no set of jobs in partition T has an improving deviation. Then, it is clear that a SSE is also a T-SSE for every partition T, as well as a Nash equilibrium. A T-SSE, on the other hand, is not necessarily a Nash equilibrium.

2.1 Nash Equilibrium

Since we are going to show that the existence of an allocation that is a T-SSE and a Nash equilibrium, we first give a complete characterization of Nash equilibrium solutions.

Let u be the minimum makespan of our system, i.e., the minimum value of $\max_i f_i(n_i)$ that can be achieved for any allocation of jobs to machines. Notice that since the latency of a machine depends only on the number of jobs assigned to this machine, then u is easily computable using a greedy algorithm. We classify

the machines into two groups. A resource i is called a 'type 1' resource if there exists a positive integer z such that $f_i(z) = u$. In other words, a resource is a 'type 1' resource if it can attain a latency of u. We say that a resource i is a 'type 2' resource if it cannot attain a latency of u, i.e., there is no positive integer z such that $f_i(z) = u$.

For each machine i, define m_i as the maximum number of jobs a machine can accept while i attains a latency at most u, i.e., $m_i = \max_z\{f_i(z) \leq u\}$.

Proposition 1. *An allocation A of jobs onto machines is a Nash equilibrium if and only if each type 2 machine i is allocated exactly m_i jobs and each type 1 machine i is allocated either m_i or $m_i - 1$ jobs, with at least one type 1 machine i allocated exactly m_i jobs.*

Proof. ***if:*** Note that when a job deviates it has to move to another machine, thereby increasing the number of jobs on that machine. If the number of jobs on any machine increases, then that machine will experience a latency of at least u. Since all jobs are currently experiencing a cost of at most u, the latency of any job after moving to a different machine will not decrease. This proves that if all the above conditions are satisfied then the allocation is a Nash equilibrium.

only if: If the makespan of a solution is more than u, say α, then this means that some machine i has more than m_i jobs on it. This implies that there exists a machine j that has less than m_j jobs on it. Then by transferring a job from machine i to j we can reduce the latency faced by that job from α to at most u. Hence a Nash equilibrium will always have a makespan of u. Also if any type 2 machine has less than m_i jobs, or a type 1 machine has less than $m_i - 1$ jobs on it, then by moving a job that faces a latency of u to this machine, we can reduce its latency. It is trivial to see that any type 2 machine will not have more than m_i jobs on it since that will increase the makespan to more than u. Hence any such allocation will not form a Nash equilibrium. This proves that in order for an allocation to be a Nash equilibrium, all the above conditions must be fulfilled. ■

By Proposition 1, some type 1 machines i are allocated m_i jobs and therefore the jobs on them are experiencing a cost of u, and some type 1 machines are (possibly) allocated $m_i - 1$ jobs and the jobs on those machines are experiencing a cost strictly less than u. Given a Nash equilibrium solution, we use the term *high machine* to refer to a type 1 machine i that has m_i jobs and use H to denote this set of machines. We use the term *low machine* to refer to all other type 1 machines and use L to denote this set of machines throughout the paper. We use R to denote the set of type 2 machines.

Given a game instance (i.e., the set of machines with their latency functions, the set of players, and the partition specified on it), the set of type 1 and type 2 machines can be readily decided, i.e., the same set of machines will be type 1 machines and the same set of machines will be type 2 machines in any Nash equilibrium allocation A. However, the splitting of type 1 machines into high machines H and low machines L depends on the Nash equilibrium solution

selected. Let A and A' be two different Nash equilibrium allocations and let H, H' and L, L' be the corresponding high and low machines for these Nash equilibrium allocations. Observe that $|H| = |H'|$ (and therefore $|L| = |L'|$) even though H and H' (and therefore L and L') may be different sets of machines. The number of high machines in any Nash equilibrium will be same.

3 Sufficient Conditions for Stability

In this paper, we want to construct a Nash equilibrium solution that is also a partition equilibrium for any given partition of the players. So, we want to construct a Nash equilibrium allocation such that none of the coalitions has an improving deviation. Given a Nash equilibrium allocation A, whether a coalition T_k has an improving deviation or not depends on the number of jobs of this coalition allocated to each machine. In this section, we will give a set of sufficient conditions for a coalition T_k not to have an improving deviation. Observe that if all the coalitions satisfy these sufficient conditions then none of the coalitions will have an improving deviation, which implies that the allocation A is also a partition equilibrium. For a type 1 machine i, we use l_i to denote $f_i(m_i - 1)$, i.e., the latency that it would experience if it were a low machine.

Following lemmas will help in finding the sufficient conditions for stability:

Lemma 1. *If the number of jobs on a machine k is the same before and after an improving deviation, then there exists an equivalent improving deviation (i.e., with the same number of jobs on each machine) where no jobs move to or from machine k.*

Proof. Consider a machine k that has the same number of jobs before and after an improving deviation D by a coalition. Let the latency of this machine be θ. Since the number of jobs on this machine did not change, for every job j that left this machine there exists another job i which entered the machine. Denote the latencies of these before deviation by α_j and α_i respectively. Similarly, let the latencies after deviation be β_j and β_i respectively. We can conclude that $\alpha_j = \beta_i = \theta$. Also, since this was an improving deviation, then $\alpha_i \geq \theta$ and $\beta_j \leq \theta$.

Now consider another deviation D' which is exactly the same as the original except for that fact that the final positions of the jobs i and j are interchanged. Let the new final latencies be β'_i and β'_j. Notice that $\beta'_j = \theta$ and $\beta'_i = \beta_j \leq \theta \leq \alpha_i$. This means that jobs i and j do not increase their latency after deviation D' as compared to their original latencies before deviation.

It is clear that the two deviations are equivalent in the sense that they result in the same number of jobs on each machine. We must still show that deviation D' is an improving deviation, i.e., that if one of these jobs reduced their latency in the original deviation D then this also holds in the new deviation D'. If job j strictly reduced its latency after deviation D, i.e., $\beta_j < \theta$, then after deviation D' latency of job i will strictly reduce. This is because now the following inequality holds true: $\beta'_i = \beta_j < \theta \leq \alpha_i$. If i strictly reduced its latency after deviation

D, i.e., $\alpha_i > \theta$, then for the new deviation D', the following inequality holds: $\alpha_i > \theta \geq \beta_j = \beta'_i$. Hence job i will also reduce its latency after deviation D'.

All pairs of such jobs can be interchanged to create a new deviation in which jobs of machine k do not take part. By above arguments, if the original deviation was improving then the new deviation will also be improving. ■

Lemma 2. *If a coalition T_k, that has 0 or 1 jobs on a high machine i in a Nash equilibrium allocation A, has an improving deviation D, then T_k has another improving deviation D', where no jobs move to or from i.*

Proof. Let A' be the allocation after T_k takes deviation D. Note that if coalition T_k does not have any job on a high machine i in allocation A, then it does not have any job on i in allocation A' as well, since otherwise the jobs that move to i will face a cost of more than u. This would imply that D cannot be an improving deviation, since all jobs have cost at most u in A.

Assume coalition T_k has exactly 1 job on a high machine i. T_k can have either 0 or 1 jobs on i in allocation A'. If T_k has 1 job on i in A' then there exists an improving deviation where no job moves to or from i by Lemma 1. Consider the case where T_k has 0 jobs on i in A'. Since the number of jobs on i has decreased then the number of jobs on some other machine j has increased. Since no machine can have a latency more than u in A', then j is a high machine in allocation A'. We can obtain another improving deviation D' by moving one of the jobs of T_k from j to i. In this deviation, the job that ends up on machine i still has cost of u (just as it did after deviation D), and all other jobs of T_k have cost after D' that is no more than their cost after D.

Therefore, if T_k has an improving deviation D and has 0 or 1 jobs on a high machine i, then it has an improving deviation D', where no jobs move to or from i by Lemma 1. ■

With the use of these lemmas we now state the sufficient conditions for stability in the following theorem:

Theorem 1. *Given a Nash equilibrium allocation A and a coalition T_k, let x_i denote the number of jobs of the coalition T_k allocated to machine i in A. Then coalition T_k does not have an improving deviation if for every high machine i such that $x_i \geq 2$ the following conditions are satisfied:*

- *for every low machine j such that $l_j > l_i$, we have that $x_j \geq x_i$ and*
- *for every low machine j such that $l_j \leq l_i$, we have that $x_j \geq x_i - 1$.*

Proof. For the purpose of contradiction, assume there exists a coalition T_k that satisfies all the conditions and yet has an improving deviation D. Let A' denote the allocation of jobs to machines if coalition T_k takes its improving deviation D, and x'_i be the number of jobs T_k has on machine i in allocation A'. Since the allocation of the jobs of all coalitions except T_k are the same in both A and A', the change in the number of jobs on any machine i is as much as the change in the number of jobs coalition T_k has on i. No machine i can have more than m_i

jobs allocated to it in allocation A' since otherwise, the jobs on i (at least one of which is a member of T_k) will experience a latency more than u, which will imply that the deviation is not an improving deviation.

If coalition T_k has 0 or 1 jobs on a high machine i in allocation A then there exists another improving deviation D', where no jobs move to or from i by Lemma 2. We will assume that D has this property. Notice that if $x_i < 2$ for a high machine i, then i is also a high machine in allocation A'. Let $x_h = \max_{i \in H}\{x_i\}$. We will first show that $x_h \geq 2$. Otherwise, all machines in H remain high after the deviation. If any other machine $j \notin H$ became high after deviation D, then jobs on j would experience a cost of u. However, all jobs with cost of u in A are on machines of H after the deviation, which means that the jobs on j have strictly increased their cost due to deviation D, and therefore D could not be an improving deviation. Thus, if $x_h < 2$, then the set of high machines is the same before and after D. In addition, if any machine $j \notin H$ has less jobs in A' than it did in A, then another machine must have more jobs, which would cause those jobs to experience a cost of at least u. By the argument above, this cannot happen, and so $x_h \geq 2$.

Lemma 3. *Let H' be the set of machines with latency of exactly u in allocation A'. Then, $|H'| \leq |H|$.*

Proof. In allocation A, coalition T_k has $\sum_{i \in H} x_i$ jobs experiencing a latency of u, whereas in allocation A', coalition T_k has $\sum_{i \in H'} x'_i$ jobs experiencing a latency of u. Let $x_h = \max_{i \in H}\{x_i\}$ and let $x_l = \min_{i \in L}\{x_i\}$. The sufficient conditions state that $x_l \geq x_h - 1$, since $x_h \geq 2$ as shown above. If $i \in H'$ was a low machine before the deviation, then $x'_i = x_i + 1$, and so it has at least as many jobs of T_k in A' as any high machine of allocation A. If $i \in H'$ was a high machine before the deviation, then $x'_i = x_i$. Thus $|H'| > |H|$ would imply that $\sum_{i \in H'} x'_i > \sum_{i \in H} x_i$, which means that coalition T_k has more jobs that are experiencing a latency of u in allocation A' than allocation A. However, that would contradict with D being an improving deviation, and so it has to be that $|H'| \leq |H|$. ■

Note that the total number of jobs in any Nash equilibrium allocation A can be expressed as $\sum_{i \in R} m_i + \sum_{i \in H} m_i + \sum_{i \in L} (m_i - 1)$. If a type 2 machine $i \in R$ has less than m_i jobs in A' then the number of machines that has latency of u would be strictly more than $|H|$, i.e., $|H'| > |H|$. Therefore, the number of jobs coalition T_k has on any type 2 machine in allocation A' is exactly x_i. Since deviation D does not change the number of jobs T_k has on any type 2 machine, there exists an equivalent improving deviation where the jobs of type 2 machines do not change by Lemma 1, and we will assume that D has this property.

Observe that if a type 1 machine i has less than $m_i - 1$ jobs in allocation A', then $|H'| > |H|$, thus violating Lemma 3. Therefore, every type 1 machine has either m_i or $m_i - 1$ jobs in allocation A'. In other words, by Proposition 1 A' is also a Nash equilibrium allocation.

Since A' is a Nash equilibrium, we can assume without loss of generality that deviation D made a certain number of high machines become low, and the same

number of low machines become high. Using Lemma 1 we can assume that the machines on which the number of jobs did not change also did not take part in deviation D. Let the set of machines the become low after the deviation be H^- and the set of machines that become high after the deviation be L^+. We know that $|H^-| = |L^+|$. Now consider the total latency faced by jobs on machines belonging to $H^- \cup L^+$ before deviation, say α, and after the deviation, say β.

$$\alpha = \sum_{i \in H^-} ux_i + \sum_{j \in L^+} l_j x_j$$
$$\beta = \sum_{i \in H^-} (x_i - 1)l_i + \sum_{j \in L^+} (x_j + 1)u$$

We now prove the following lemma:

Lemma 4. *For every perfect matching P between the machines of H^- and L^+ that pairs $i \in H^-$ with $j \in L^+$, it must be true that $l_j \leq l_i$ and $x_j = x_i - 1$.*

Proof. Let P be any perfect matching between the machines of H^- and L^+ (note that $|H^-| = |L^+|$). Consider a pair of machines $(i, j) \in P$ such that $i \in H^-$ and $j \in L^+$. If $l_j > l_i$ then we know that $x_j \geq x_i$. This means that the total number of jobs facing a latency u on machines i, j after deviation: $(x_j + 1)$ is strictly more than before: (x_i). If $l_j \leq l_i$ then we know that $x_j \geq x_i - 1$. This would mean that the total number of jobs facing a latency u on machines i, j after deviation: $(x_j + 1)$ is at least as much as before: (x_i).

This implies that if there exists even one pair of machines (i, j) such that $l_j > l_i$, then the total number of jobs facing a latency of u after deviation will strictly increase. On the other hand if $l_j \leq l_i$ but $x_j > x_i - 1$ then too it is easy to see that the number of jobs facing a latency of u after deviation strictly increases.

Hence D is a valid deviation only if for every $(i, j) \in P, l_j \leq l_i$ and $x_j = x_i - 1$.
■

Consider any perfect matching P between the machines of H^- and L^+. We can now compare the values of α and β:

$$\begin{aligned}
\alpha &= \sum_{i \in H^-} ux_i + \sum_{j \in L^+} l_j x_j \\
&= \sum_{j \in L^+} u(x_j + 1) + \sum_{j \in L^+} l_j x_j \quad \ldots \text{(Lemma 4)} \\
&= \sum_{j \in L^+} u(x_j + 1) + \sum_{(i,j) \in P} l_j (x_i - 1) \quad \ldots \text{(Lemma 4)} \\
&\leq \sum_{j \in L^+} u(x_j + 1) + \sum_{i \in H^-} l_i (x_i - 1) \quad \ldots \text{(For every } (i, j) \in P, l_j \leq l_i) \\
&= \beta
\end{aligned}$$

This means that the total cost faced by jobs of machines of $H^- \cup L^+$ will at best remain the same. If the latency of some job decreases then the latency of some other has to increase in order to keep the sum constant. This means that the latency faced by every job can at best remain the same. But an improving deviation requires that D must strictly improve the latency of some job. Hence an improving deviation D does not exist. ■

4 Partition Equilibrium

We now present an algorithm that constructs an allocation of jobs such that all the sufficient conditions of Theorem 1 are satisfied. Thus we will create a Nash equilibrium that is also a partition equilibrium. For this purpose we will use the properties of Nash equilibrium as described in Section 2. Particularly we will use the fact that, given the total number of jobs, every Nash equilibrium will have the same number of high machines, which we denote by q. The algorithm gives an allocation of jobs over all type-1 machines. Since the sufficient conditions do not have restrictions on jobs of the type-2 machines, remaining jobs can be arbitrarily allocated to them so that each machine has m_i jobs. We also define a set of *active machines* as all machines i that have less than m_i jobs on them. Let $q = |H|$ be the number of high machines in any Nash equilibrium allocation. The algorithm is as follows:

- Begin with an empty allocation. Note that all machines are active at this time.
- Obtain an ordering on the set of all active machines based on non-increasing values of their l_i.
- For every coalition, place jobs sequentially starting from the first active machine according to the above ordering. If the number of jobs in this coalition exceeds the number of active machines then rollover and continue placing jobs from the first active machine in the ordering.
- If at any step a machine i has m_i jobs placed on it, i.e. i becomes high, then remove it from the set of active machines.
- When q machines become high, place remaining jobs on the active machines arbitrarily such that they have $m_i - 1$ jobs on them.

Example Consider an example with 4 machines and 2 coalitions in Figure 1. Coalition 1 has 6 jobs and coalition 2 has 4 jobs. All the machines in this example are of type-1. Also they have been sorted in non-increasing order of their l_i-values. The blocks represent the jobs and height of the j'th block on machine i is given by $f_i(j) - f_i(j-1)$. Figure 1 illustrates various stages during the implementation of the algorithm. Observe that in any Nash equilibrium for this input exactly $q = 2$ machines will be high.

Notice that making sure that our algorithm places no more than m_i jobs on any machine is crucial not only to create a Nash equilibrium, but also to create a partition equilibrium. For example, in Figure 1, if we did not stop adding jobs

to machines once they have m_i jobs, then we would end up with $3 = m_2 + 1$ jobs on machine 2. If $f_2(3) > f_4(3)$, then this would not be a partition equilibrium, since Coalition 1 would have an improving deviation by moving two of its jobs to machine 4 from machine 2, and one job to machine 2 from machine 4.

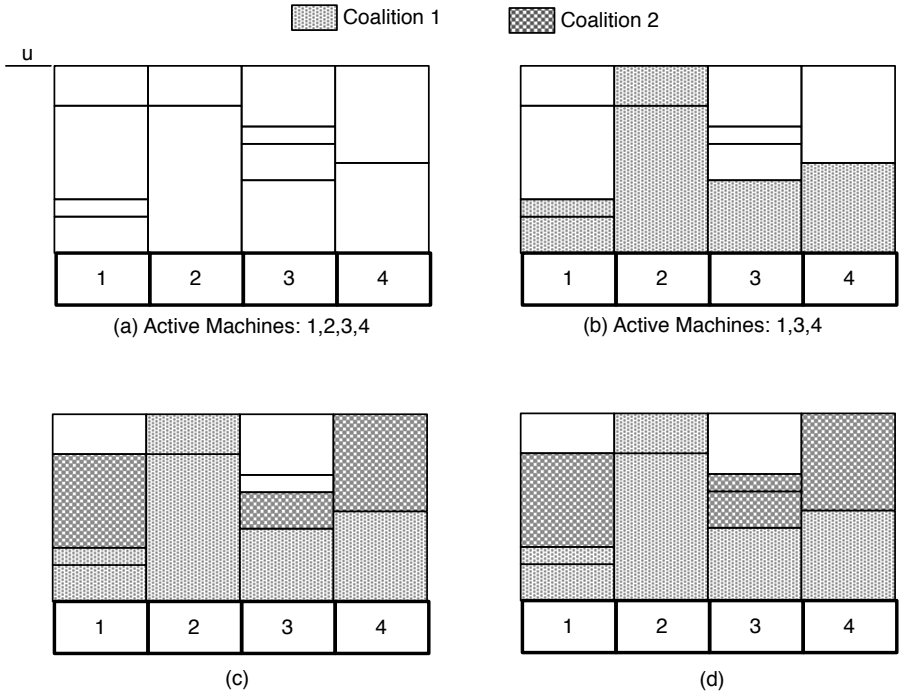

Fig. 1. (a) In the beginning all machines are active. (b) Jobs of coalition 1 are placed and machine 2 becomes inactive. (c) 3 out of 4 jobs of coalition 2 have been placed and $q = 2$ machines have become high. (d) The remaining job is placed on machine 3 making it low. Sufficiency conditions of Theorem 1 are now satisfied.

Theorem 2. *The above algorithm produces a partition equilibrium and a Nash equilibrium.*

Proof. The algorithm makes exactly q machines high hence due to the property of NE we know that there are sufficient jobs to make rest of the machines low, i.e., put $m_i - 1$ jobs on them. Consider a coalition C. If this coalition has only 0 or 1 jobs on every high machine then the conditions of Theorem 1 are fulfilled, and so C has no improving deviation. Consider a high machine i on which coalition C has more than one jobs. Let us look at the time-step when the algorithm has put α jobs of coalition C on i. Now before putting the $(\alpha + 1)$'th job on i

the algorithm puts one job on every low machine. This is true because the low machines are exactly the ones that do not run out of space for jobs until the high machines are completely filled. This implies that for every low machine j, $x_j \geq x_i - 1$.

Also if a low machine j is such that $l_j > l_i$ then the algorithm puts one job on every such machine j before i. This follows from the ordering obtained on the machines on the basis of the l_i-values. This means that if $l_j > l_i$ then $x_j \geq x_i$.

This proves that both sufficient conditions of Theorem 1 are fulfilled by the final allocation, which is also a Nash equilibrium by Proposition 1. Hence the allocation obtained by the algorithm is a partition equilibrium. ■

5 Existence and Computation of SSE

For a resource selection game, SSE may or may not exist (see [4] for an example where it does not exist). In this section, we show that for a given instance of a resource selection game, we can efficiently determine whether there exists a SSE or not. We also give an algorithm that finds a SSE if it exists.

Theorem 3. *Given a resource selection game G, there is a polynomial time algorithm that returns a SSE allocation if it exists, and returns "no" if G does not have a SSE.*

Proof. Since every SSE is also a Nash equilibrium, then each type 2 machine i has to have exactly m_i jobs in any SSE allocation.

Recall that the total number of jobs in the system is exactly as much as $\sum_{i \in R} m_i + \sum_{i \in H} (m_i - 1) + \sum_{i \in L} (m_i - 1) + q$ in any Nash equilibrium allocation, with q being the number of high machines in any Nash equilibrium. If all type 1 machines are high machines, i.e., $L = \emptyset$, then all the machines in the system will have exactly m_i jobs and no coalition can have an improving deviation. This is because any non-trivial deviation would require moving a job so that its resulting latency is strictly more than u, and so cannot be an improving deviation. Therefore, if $L = \emptyset$ then any Nash equilibrium allocation is also a SSE allocation. So, a SSE allocation can be obtained simply by assigning m_i jobs to all machines.

Consider the case, where $L \neq \emptyset$. Assume that a high machine i has 2 or more jobs. Consider a coalition composed of 2 jobs such that both are allocated to i. If one of the jobs of this coalition moves to a low machine, then the cost of the moving job will not change, while the other member of the coalition strictly benefits. Therefore, an allocation where a high machine i has 2 or more jobs is not a SSE if $L \neq \emptyset$. Thus, G does not have a SSE if $L \neq \emptyset$ and G does not have at least q type 1 machines for which $m_i = 1$.

If there are at least q type 1 machines for which $m_i = 1$ then any Nash equilibrium allocation A where q of the type 1 machines, for which $m_i = 1$ are high machines, is a SSE. This is because no subset of players has more than 1 job on any high machine in A, and therefore no coalition has an improving deviation by Theorem 1. SSE allocation then can simply be obtained by placing

1 job on q machines for which $m_i = 1$ and assigning the remaining jobs to all other machines in a way that every type 2 machine has exactly m_i jobs and every remaining type 1 machine has exactly $m_i - 1$ jobs allocated to it. ■

References

1. Aumann, R.: Acceptable Points in General Cooperative n-person Games. In: Contributions to Theory of Games IV, Princeton Univ. Press, Princeton, N.J. (1959)
2. Awerbuch, B., Azar, Y.,Epstein, L.: The Price of Routing Unsplittable Flow. In: Proceedings of the thirty-seventh annual ACM symposium on Theory of computing, pp. 57-66. ACM, New York (2005)
3. Durr, C., Kim Thang, N.: Non-clairvoyant Scheduling Games. In: SAGT 2009. LNCS, vol. 5814, pp. 135-146. Springer, Heidelberg (2009)
4. Feldman, M., Tenneholtz, M.: Partition Equilibrium. In: SAGT 2009. LNCS, vol. 5814, pp. 48-59. Springer, Heidelberg (2009)
5. Fiat, A., Kaplan, H., Levi, M., Olonetsky, S.: Strong Price of Anarchy for Machine Load Balancing. In: ICALP 2007. LNCS, vol. 4596, pp. 583-594. Springer, Heidelberg (2007)
6. Fotakis, D., Kontogiannis, S., Spirakis, P.: Atomic Congestion Games Among Coalitions. In: ICALP 2006. LNCS, vol. 4051, pp. 572-583. Springer, Heidelberg (2006)
7. Hayrapetyan, A., Tardos, É., Wexler, T.: The Effect of Collusion in Congestion Games. In: Proceedings of the thirty-eighth annual ACM symposium on Theory of computing, pp. 89-98. ACM, New York (2006)
8. Holzman, R., Law-Yone, N.: Strong Equilibrium in Congestion Games. :Games and Economic Behavior 21, 85-101 (1997)
9. Holzman, R., Law-Yone, N.: Network structure and strong equilibrium in route selection games. :Mathematical Social Sciences 46, 193-105 (2003)
10. Immorlica, N., Li, L., Mirrokni, V., Schulz, A.,: Coordination Mechanisms for Selfish Scheduling. In: WINE 2005. LNCS, vol. 3828, pp. 55-69. Springer, Heidelberg (2005)
11. Monderer, D., Shapley, L.: Potential Games. In: Games and Economic Behavior 14, 124-143 (1996)
12. Rosenthal, R.: A class of games possessing pure-strategy Nash equilibria. In: International Journal of Game Theory 2, 65-67 (1973)
13. Roughgarden, T.: Selfish Routing with Atomic Players. In: Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms, pp. 1184-1185. Society for Industrial and Applied Mathematics, Philadelphia (2005)
14. Suri, S., Toth, C., Zhou, Y.: Selfish Load Balancing and Atomic Congestion Games. In: Algorithmica, 47(1), 79-96 (2007).