No. 2007–40

PARTITIONING SEQUENCING SITUATIONS AND GAMES

By Marloes Gerichhausen, Herbert Hamers

June 2007

ISSN 0924-7815

Partitioning sequencing situations and games

Marloes Gerichhausen

Tilburg University, Faculty of Economics and Business Administration, Department of Econometrics and OR, P.O.Box 90153, 5000 LE, Tilburg, The Netherlands, e-mail: M.Gerichhausen@uvt.nl

Herbert Hamers

Tilburg University, Faculty of Economics and Business Administration, Department of Econometrics and OR, P.O.Box 90153, 5000 LE, Tilburg, The Netherlands, e-mail: H.J.M.Hamers@uvt.nl

Abstract:

The research that studies the interaction between sequencing situations and cooperative games, that started with the paper of Curiel et al. (1989), has become an established line of research.

This paper introduces a new model in this field: partitioning sequencing situations and games. The characteristic of partitioning sequencing situations is that the jobs arrive in batches, and those jobs that arrive in earlier batches have some privileges over jobs in later arrived batches. For partitioning sequencing situations we introduce and characterise the partitioning equal gain splitting rule. Next, we define cooperative games that arise from these partitioning sequencing situations. It is shown that these games are convex. Moreover, we present a game independent expression for the Shapley value. Finally, it is shown that the partitioning equal gain splitting rule can be used to generate a core allocation and can be viewed as the average of two specific marginal vectors.

Keywords: Sequencing situations, sequencing games.
JEL classification: C71

1 Introduction

In one-machine partitioning sequencing situations a number of jobs need to be processed on a single machine. It is assumed that each job is assigned to a different agent (player), who has a specific cost function depending on the completion time of his job. An initial order of the jobs is assumed before the processing of the machine starts. Moreover, the jobs in the initial order are partitioned into connected sets. The jobs from each set have initial rights, which are represented by the maximum number of positions they can shift backwards, a so called disruption level. The objective is to find a processing order of the jobs that minimises the aggregated costs function of all agents, taking into account the disruption levels.

In practice partitioning sequencing situations are faced constantly by manufacturing companies, especially those producing custom products to order rather than standard products to stock. Customers call in over time to place orders. It is for these companies necessary to develop a production schedule a priory, to provide coordination for necessary activities, i.e., shipping. However it often happens that another customer calls, and additional orders arrive. These additional orders must then be integrated into the schedule, such that, for example, the costs are optimised and the disruption of the old schedule is also controlled.

The paper of Curiel et al. (1989) started a new line of research that investigates the interaction between sequencing situations and cooperative games. In their paper one-machine sequencing situations and associated games were investigated. The main focus was on allocation rules (e.g. equal gain splitting rule, Shapley value (Shapley (1953)) and compromise value (Tijs (1981)), and game theoretical properties (e.g. balancedness, convexity). In this paper we take a similar approach as in Curiel et al. (1989) on the class of partitioning sequencing situations. First, we focus on the allocation of the cost savings in partitioning sequencing situations that can be obtained by rearranging the jobs from its initial order to an optimal order. We introduce the partitioning equal gain splitting rule ($\mathcal{PEGS}$-rule). This rule is inspired on the algorithm of Hall and Potts (2004), that finds an optimal order for partitioning sequencing situations. Moreover, the $\mathcal{PEGS}$-rule is characterised using efficiency, symmetry and consistency. Especially the consistency property is attractive, which has been applied in several OR-situations, like assignment situations (cf. Owen (1992)), flow-situations (cf. Reijnierse et al. (1996)) and minimum cost spanning tree situations (cf. Feltkamp et al. (1994)). For a survey on consistency we refer to Thomson (1990). Second, we define a cooperative game that corresponds to a partitioning sequencing situation. It is shown that partitioning sequencing games are convex by decomposing them into a non-negative linear combination of unanimity games. The latter makes it possible to provide a game independent expression for the Shapley value. Finally, it is shown that the $\mathcal{PEGS}$-rule can be viewed as the average of two specific marginal vectors.

Indeed the research in the field of sequencing situations and related games is quite extensive. Hamers et al. (1995), Borm et al. (2002), Hamers et al. (2005) and Van Velzen (2006) considered sequencing situations in which the jobs have ready times, due dates, precedence relations and controllable processing times, respectively. All these papers focus on convexity of the corresponding games. Others investigated games that arise from sequencing situations with multiple machines. Hamers et al. (1999), Calleja et al. (2006) and Slikker (2005) focus on balancedness for these games. Finally, Van Velzen and Hamers (2003) and Slikker (2006) considered a relaxation of sequencing games and proved that these games have a non-empty core.

This paper is organised as follows. In Section 2 we describe the partitioning sequencing situations with two connected sets and introduce and characterise the $\mathcal{PEGS}$-rule. In Section 3 partitioning sequencing games are introduced. It is shown that these games are convex. Moreover, an explicit expression for the Shapley value is provided and it is shown that the $\mathcal{PEGS}$-rule provides a core element. Section 4 shows that the results with respect to the allocation rule and the partitioning sequencing games can be extended to partitioning sequencing situations in which

there are more than two connected sets. Finally, Section 5 concludes.

2 Partitioning sequencing situations

In this section we introduce the class of partitioning sequencing situations with two connected sets. Moreover, we introduce and characterise the partitioning equal gain splitting ($\mathcal{PEGS}$-) rule, an allocation rule that divides the profit that can be obtained by rearranging the jobs from its initial order into an optimal order.

In an one-machine partitioning sequencing situation there is a queue of agents, each with one job, before a machine. Each agent has to process his job on this machine. The finite set of agents is denoted by $N = \{1, ..., n\}$. By a bijection $\sigma : N \to \{1, ..., n\}$ we can describe the position of the agents (jobs) in the queue. Specifically, $\sigma(i) = j$ means that agent i is in position j. We assume that there is an initial order $\sigma_0 : N \to \{1, ..., n\}$ on the agents before the processing of the machine starts. Without loss of generality we assume in the remaining of this paper that $\sigma_0(i) = i$ for all $i \in N$. Moreover, we assume that N is partitioned into two sets J_1 and J_2, in such a way that $J_1 = \{1, \cdots, n_1\}$ and $J_2 = \{n_1 + 1, ..., n\}$. One can view the initial order as being determined by two separate arrivals of a set of jobs (corresponding to the agents). In the remaining of the paper we will only refer to the job (or, agent) and not referring to its corresponding agent (or, job).

Further, it is assumed that the jobs of set J_1 have disruption level k, with k being a positive integer. The disruption level indicates the number of jobs of J_2 that can take a position in $\{1, \cdots, n_1\}$ if the jobs are reordered from its initial order. In other words, an order σ is feasible if and only if $\sigma(i) \leq n_1 + k$, for all $i \in J_1$. The set of all feasible orders is denoted by $\mathcal{F}$. The processing time p_i of the job of agent i is the time the machine takes to handle this job. The completion time of agent i, with respect to an order σ is defined by $C(i, \sigma) = \sum_{\sigma(j) \leq \sigma(i)} p_j$. For each agent $i \in N$ the costs of the time spent in the system is described by the linear cost function $c_i : [0, \infty) \to I\!R$ defined by $c_i(t) = t$. Hence, the costs of an agent only depend on its completion time.

A *partitioning sequencing situation* is denoted as $\Gamma(N) = (N, J_1, J_2, \sigma_0, p, k)$, where N is the set of agents, J_1, J_2 is a connected partition of N, σ_0 is the initial order, $p = (p_i)_{i \in N} \in I\!R^N_+$ the processing times and k the disruption level of the jobs in J_1. In the remainder we denote with $PSEQ$ the class of all partitioning sequencing situations.

The objective is to minimise the total completion time of the jobs, i.e., $\min_{\sigma \in \mathcal{F}} \sum_{j=1}^{n} C(j, \sigma)$. Observe that if $k \geq n_2$, where $n_2 = |J_2|$, then a partitioning sequencing situation is equal to the classical one-machine sequencing situation as discussed in Curiel et al. (1989). Hence in the remainder of the paper we assume that $k < n_2$.

The following algorithm, introduced by Hall and Potts (2004), establishes an optimal order of a partitioning sequencing situation. The algorithm is based on the Shortest Processing Time (SPT) rule. The SPT rule schedules the jobs in non-decreasing order of their processing times.

Algorithm Hall and Potts
Let $\Gamma(N) = (N, J_1, J_2, \sigma_0, p, k)$ be a partitioning sequencing situation. In the first step of the algorithm, the SPT rule is applied to both partitioned sets J_1 and J_2, separately, resulting in a new order σ_1. In the second step the order σ_2 is obtained by applying the SPT rule on the jobs that are in position $1, \cdots, n_1 + k$ of the order σ_1.

Hall and Potts prove that the order σ_2 is optimal. The following example illustrates the algorithm.

Example 1 Let $N = \{1, 2, \cdots, 8\}$, $J_1 = \{1, 2, 3, 4\}$, $J_2 = \{5, 6, 7, 8\}$, $\sigma_0 = (1, 2, \cdots, 8)$, $p = (2, 9, 8, 4, 3, 7, 10, 1)$ and $k = 2$. The initial order of jobs has a total completion time of 201. We reduce these costs using the described algorithm. First we rearrange the jobs of J_1 and J_2, respec-

tively, by applying the SPT rule. This results in the orders: $(1, 4, 3, 2)$ and $(8, 5, 6, 7)$, respectively. Hence, the new order after the first step is $\sigma_1 = (1, 4, 3, 2, 8, 5, 6, 7)$. Because $n_1 + k = 6$ we apply in the second step the SPT-rule on the jobs in position $1, \cdots, 6$, which are $(1, 4, 3, 2, 8, 5)$. We get $\sigma_2 = (8, 1, 5, 4, 3, 2, 6, 7)$ as an optimal order with a total completion time of 143. Hence, the total cost savings, obtained by rearranging the jobs from its initial order into the optimal order, are $201 - 143 = 58$. The steps of the algorithm are displayed in Figure 1.

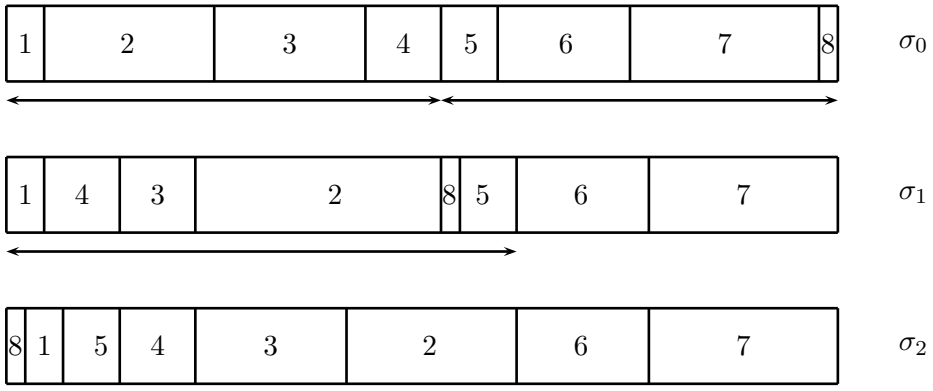

Figure 1: The consecutive steps in the Hall and Potts algorithm.

◁

Example 1 illustrates that agents standing in front of a machine, before the machine starts processing, can save money by rearranging their positions. In the remaining part of this section we introduce and characterise the $\mathcal{PEGS}$-rule, an allocation rule for partitioning sequencing situations. The $\mathcal{PEGS}$-rule arises from a non-aggregated solution concept, which is inspired by the algorithm of Hall and Potts (2004). A non-aggregated solution concept provides a specification of the reward each agent can obtain by cooperating with any other agent. Suijs et al. (1997) gave a non-aggregated solution for sequencing situations as studied in Curiel et al. (1989). Formally, a non-aggregated solution f is a map assigning to each partitioning sequencing situation $\Gamma(N) \in PSEQ$ a matrix $W \in I\!R_+^{N \times N}$, where an element w_{ij} of W represents the non-negative gain assigned to agent i for cooperation with agent j. The aggregated allocation corresponding to a solution W can be found by multiplying W with the unit vector $e^N = (1 \cdots 1)^T \in I\!R^N$.

Before we define the non-aggregated solution we need some notation. We define $g_{ij} = (p_i - p_j)_+$ as the possible gains which can be obtained by a neighbour switch of i and j, if i is in front of j. The set of i and his predecessors with respect to σ is denoted by $P(\sigma, i) = \{t | \sigma(t) \leq \sigma(i)\}$. Next, define recursively the set B_j for $j = n_1 + k, \cdots, n$ in the following way.

$$B_j = \begin{cases} P(\sigma_0, j) \cap J_2 & \text{if } j = n_1 + k \\ (B_{j-1} \cup \{j\}) \backslash \{c_j\} & \text{if } j = n_1 + k + 1, \cdots, n \end{cases} \tag{1}$$

where

$$c_j = \text{argmin}\{t | t \in B_{j-1} \cup \{j\}, p_t \geq p_m \text{ for all } m \in B_{j-1} \cup \{j\}\} \tag{2}$$

is the first agent with the largest processing time of the set $B_{j-1} \cup \{j\}$. Observe that the set B_{n_1+k} is the set with the first k jobs of J_2. In every step, one job is added to the set and the job with the highest processing time is removed from the set. It may happen that the job that is added and removed in the same step. Note that B_j is the set of k jobs of $\{n_1+1, \cdots, j\}$ with smallest processing times. Hence, B_n is the set of k jobs of J_2 with the smallest processing times.

In the following we describe a procedure that determines the $\mathcal{PEGS}$-rule. In the first step the gains between the agents in J_1 (J_2) are divided in the following way. In the algorithm all agents of J_1 (J_2) are ordered in SPT order. This order can be obtained by neighbour switches that all have non-negative gains. The gain obtained in such a neighbour switch is divided equally among both agents involved. Hence, if $i, j \in J_1$ or $i, j \in J_2$ then $\frac{1}{2}g_{ij}$ is assigned to both players i and j.

Next, we divide the possible gains between the agents of J_1 and the first k agents of J_2. Because the disruption level is k, any pair (i, j) with $i \in J_1$ and $j \in B_{n_1+k}$ can switch and can obtain a gain of g_{ij}. To both agents half of this possible gain is assigned, i.e., each agent receives $\frac{1}{2}g_{ij}$.

In the following consecutive steps we take $j = n_1 + k + 1, ..., n$, in which j is increased with one unit after each step. We distinguish two cases. If $c_j = j$ then agent j will remain in position and receives no gains in this step. If $c_j \neq j$, then $p_j \leq p_{c_j}$, and agent c_j and j will switch position. This means that agent c_j cannot switch anymore with jobs of J_1 due to the disruption level, but their gains are already allocated. The gain agent j and agent $i \in J_1$ can obtain is g_{ij}, but the gain g_{ic_j} is already divided. Hence, the net profit agent j can obtain with agent i is $g_{ij} - g_{ic_j}$. This net profit is divided equally between agent i and j, which is $\frac{1}{2}(g_{ij} - g_{ic_j})$.

Next, the (non-)aggregated $\mathcal{PEGS}$-rule is formally introduced. Let $\Gamma(N)$ be a partitioning sequencing situation. Define the symmetric matrix W, where $i, j \in \{1, ..., n\}$, $i \leq j$ as follows:

$$w_{ij} = \begin{cases} \frac{1}{2}g_{ij} & \text{for } (i, j \in J_1) \text{ or } (i, j \in J_2) \text{ or } (i \in J_1 \text{ and } j \in B_{n_1+k}) \\ \frac{1}{2}(g_{ij} - g_{ic_j}) & \text{for } i \in J_1 \text{ and } j \in J_2 \backslash B_{n_1+k} \end{cases} \tag{3}$$

where B_{n_1+k} and c_j are defined as in (1) and (2). Then the non-aggregated $\mathcal{PEGS}$-rule is defined by $\mathcal{PEGS}(\Gamma(N)) = W$. The aggregated $\mathcal{PEGS}$-rule is defined as We^N. Example 2 will illustrate the (non-)aggregated $\mathcal{PEGS}$-rule.

Example 2 Consider the partitioning sequencing situation of Example 1. Recall that the total cost savings are equal to 58 . The non-aggregated $\mathcal{PEGS}$-rule is equal to the matrix W, as defined in (3), which is

$$W = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.5 \\ 0 & 0 & 0.5 & 2.5 & 3 & 1 & 0 & 3 \\ 0 & 0.5 & 0 & 2 & 2.5 & 0.5 & 0 & 3 \\ 0 & 2.5 & 2 & 0 & 0.5 & 0 & 0 & 1.5 \\ 0 & 3 & 2.5 & 0.5 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0.5 & 0 & 0 & 0 & 0 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4.5 \\ 0.5 & 3 & 3 & 0.5 & 1 & 3 & 4.5 & 0 \end{pmatrix}$$

Observe that in the (1-4)x(1-4) block and (5-8)x(5-8) of the matrix W represents the equal divisions of gains of neighbour switches within the set J_1 and J_2, respectively. The gains within those blocks are equally divided. For example, player 2 and 3 gain $g_{23} = 1$ in their neighbour switch. Hence, each player receives according to this switch $w_{23} = w_{32} = \frac{1}{2}g_{23} = 0.5$.
The other entries of the matrix are obtained by using the steps described in (1) and (2) recursively. First, we obtain $B_6 = \{1, 2, 3, 4, 5, 6\} \cap \{5, 6, 7, 8\} = \{5, 6\}$. Next, because $c_7 = 7$ we get that $B_7 = \{5, 6\} \cup \{7\} \backslash \{7\} = \{5, 6\}$. Finally, because $c_8 = 6$ we get $B_8 = \{5, 8\}$.

Now, we can divide the gains between the jobs of J_1 and the jobs of J_2, using (3).
For job $i \in J_1$ and job $j \in \{5, 6\}$ the gains are equally divided. For example, $w_{25} = \frac{1}{2}g_{25} = 3$.

For job $i \in J_1$ and job 7 we get $w_{i7} = \frac{1}{2}(g_{i7} - g_{i7}) = 0$ for all $i \in J_1$.
At last, the gains job $i \in J_1$ and job 8 can obtain is equal to $w_{i8} = \frac{1}{2}(g_{i8} - g_{i6})$. Which gives for example $w_{18} = \frac{1}{2}(g_{18} - g_{16}) = 0.5$.
The aggregated $\mathcal{PEGS}$-rule assigns to this partitioning sequencing situation the vector $We^N = (0.5, 10, 8.5, 6.5, 7, 4.5, 4.5, 16.5)$, which reflects the total profit for each player. $\triangleleft$

In the final part of this section, we characterise the non-aggregated $\mathcal{PEGS}$-rule. Therefore we need the notions of connected coalitions and reduced partitioning sequencing situations. A set of agents is called *connected with respect to* σ if for all $i, j \in S$ and all $k \in N$ with $\sigma_0(i) < \sigma_0(k) < \sigma_0(j)$ it holds that $k \in S$. The set of all non-empty connected coalitions with respect to the initial order σ_0 is denoted with $con(\sigma_0)$. A partitioning sequencing situation reduced to a connected coalition S, is the sequencing situation remaining when all agents outside of coalition S are left out of consideration. A *reduced partitioning sequencing situation with respect to* S is described by $\Gamma(N|_S) = (S, J_1 \cap S, J_2 \cap S, \sigma_0^S, p^S, k)$, with $p^S = (p_i)_{i \in S}$ and for all $i, j \in S$ it holds that $\sigma_0^S(i) < \sigma_0^S(j)$ if and only if $\sigma_0(i) < \sigma_0(j)$.

Let f be a non-aggregated solution concept, that assigns to each partitioning sequencing situation $\Gamma(N)$ a matrix $f(\Gamma(N)) \in I\!R^{N \times N}$ and let σ^* denote an optimal order for $\Gamma(N)$. We introduce the following three properties.

- **Efficiency**: f is efficient if for all $\Gamma(N)$ it holds that:

$$\sum_{i,j \in N} f(\Gamma(N))_{ij} = \sum_{i \in N} C(\sigma_0, i) - \sum_{i \in N} C(\sigma^*, i)$$

- **Symmetry**: f is called symmetric if for all $\Gamma(N)$ it holds that $f(\Gamma(N))_{ij} = f(\Gamma(N))_{ji}$ for all $i, j \in N$.

- **Consistency**: f is called consistent if for all $S \in con(\sigma_0)$ different from N it holds that $f(\Gamma(N))|_S = f(\Gamma(N|_S))$, where $f(\Gamma(N))|_S$ is the matrix with all columns and rows of members outside S deleted.

Efficiency means that exactly the total cost savings are allocated to the agents. Symmetry tells us that the (extra) gain two agents can obtain is equally divided by the two agents. Consistency means that connected sub-coalitions obtain the same division if they renegotiate the (sub)solution on basis of the same solution concept to an intuitively appealing reduced situation.

Lemma 1 shows that the $\mathcal{PEGS}$-rule is efficient. Before we formulate the Lemma, we define the vector e^S as

$$e_i^S = \begin{cases} 1 & \text{if } i \in S \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

Moreover, in spite of slight abuse of notation, we denote the transpose of e^S also by e^S.

Lemma 1 *Let $\Gamma(N) = (N, J_1, J_2, \sigma_0, p, k)$ be a partitioning sequencing situation, let S be a connected set and let σ_S^* present the optimal order of S, then*

$$e^S W e^S = \sum_{i \in S} C(\sigma_0, i) - \sum_{i \in S} C(\sigma_S^*, i)$$

Proof. First we calculate $e^S W e^S$. Observe that for S connected $(We^S)_j = \sum_{i\in S} w_{ji} = \sum_{i\in S} w_{ij} = \sum_{i\in S, i<j} w_{ij} + \sum_{i\in S, i>j} w_{ji}$, for all $j \in S$. Using (3) we obtain

$$(We^S)_j = \begin{cases} \sum_{i\in S,i<j} \frac{1}{2} g_{ij} + \sum_{i\in J_1\cap S, i>j} \frac{1}{2} g_{ji} + \sum_{i\in B_{n_1+k}\cap S} \frac{1}{2} g_{ji} + \sum_{i\in (J_2\setminus B_{n_1+k})\cap S} \frac{1}{2}(g_{ji} - g_{jc_i}) \\ \qquad\qquad \text{if } j \in J_1 \cap S \\ \sum_{i\in J_1\cap S} \frac{1}{2} g_{ij} + \sum_{i\in J_2\cap S, i<j} \frac{1}{2} g_{ij} + \sum_{i\in S, i>j} \frac{1}{2} g_{ji} \\ \qquad\qquad \text{if } j \in B_{n_1+k} \cap S \\ \sum_{i\in J_2\cap S, i<j} \frac{1}{2} g_{ij} + \sum_{i\in S, i>j} \frac{1}{2} g_{ji} + \sum_{i\in J_1\cap S} \frac{1}{2}(g_{ij} - g_{ic_j}) \\ \qquad\qquad \text{if } j \in (J_2\setminus B_{n_1+k}) \cap S \end{cases} \tag{5}$$

Let $S = \{t, t+1, ..., r-1, r\}$, and define $B_r = \emptyset$ if $r \leq n_1$ and $B_r = \{n_1+1, ...r\}$ if $n_1+1 \leq r < n_1+k$. Recall, that B_r is already defined by (1) if $r \geq n_1 + k$. Then

$$\begin{aligned} e^S W e^S &= \sum_{j\in S} (We^S)_j \\ &= \sum_{i,j\in J_1\cap S} g_{ij} + \sum_{i,j\in J_2\cap S} g_{ij} + \sum_{i\in J_1\cap S, j\in B_{n_1+k}\cap S} g_{ij} + \sum_{i\in J_1\cap S, j\in (J_2\setminus B_{n_1+k})\cap S} (g_{ij} - g_{ic_j}) \\ &= \sum_{i,j\in J_1\cap S} g_{ij} + \sum_{i,j\in J_2\cap S} g_{ij} + \sum_{i\in J_1\cap S, j\in J_2\cap S} g_{ij} - \sum_{i\in J_1\cap S, j\in (J_2\setminus B_{n_1+k})\cap S} g_{ic_j} \\ &= \sum_{i,j\in J_1\cap S} g_{ij} + \sum_{i,j\in J_2\cap S} g_{ij} + \sum_{i\in J_1\cap S, j\in B_r} g_{ij} \\ &= \sum_{i\in S} C(\sigma_0, i) - \sum_{i\in S} C(\sigma_S^*, i) \end{aligned}$$

whereby the second equation is obtained by reorganizing the terms of (5). The third equation is obtained by rewriting the previous expression. The fourth equation is obtained by the definition of B_r. The last equation follows from the algorithm of Hall and Potts (2004) applied to S. □

Theorem 1 *The non-aggregated $\mathcal{PEGS}$-rule is the unique non-aggregated rule satisfying efficiency, symmetry and consistency.*

Proof. First it is shown that the $\mathcal{PEGS}$-rule satisfies efficiency, symmetry and consistency. Efficiency follows by Lemma 1 by taking $S = N$. Symmetry is a consequence of the definition of W and consistency follows from the definition of W and $\Gamma(N|_S)$.

Let f be a non-empty solution concept satisfying symmetry, efficiency and consistency. With induction on the number of agents we show that $f(\Gamma(N)) = \mathcal{PEGS}(\Gamma(N))$. If $n = 1$ efficiency yields $f(\Gamma(N)) = [0] = \mathcal{PEGS}(\Gamma(N))$ for all $\Gamma(N)$. Now, assume that $f(\Gamma(N)) = \mathcal{PEGS}(\Gamma(N))$ for all $n < m$. Take $n = m$ and let $\Gamma(N, J_1, J_2, \sigma_0, p, k)$. Now, reduce $\Gamma(N)$ to $\Gamma(N_{|S})$ with $S = N\setminus\{1\}$ and $S = N\setminus\{n\}$, respectively. Applying consistency and using the induction hypothesis yields

$$f(\Gamma(N))_{ij} = f(\Gamma(N|_S))_{ij} = \mathcal{PEGS}(\Gamma(N|_S))_{ij} = \mathcal{PEGS}(\Gamma(N))_{ij}$$

for all pairs $(i, j) \neq (1, n)$ and $(i, j) \neq (n, 1)$. Efficiency and symmetry gives then

$$f(\Gamma(N))_{1n} = f(\Gamma(N))_{n1} = \mathcal{PEGS}(\Gamma(N))_{n1} = \mathcal{PEGS}(\Gamma(N))_{1n}.$$

Hence $f(\Gamma(N)) = \mathcal{PEGS}(\Gamma(N))$. □

3 Partitioning sequencing games

In this section we define partitioning sequencing games, a class of cooperative games that arises from partitioning sequencing situations. We show that these games are convex. Further, a game independent expression for the Shapley value is given. Moreover, we show that the aggregated $\mathcal{PEGS}$-rule provides a core element.

Let $\Gamma(N) = (N, J_1, J_2, \sigma_0, p, k)$ and let σ^* be an optimal order of N. Then the maximal cost savings of coalition N is equal to $\sum_{i\in N}[C(\sigma_0, i) - C(\sigma^*, i)]$. Now, we want to determine the maximum cost savings of a coalition $S \subset N$. For this, we introduce the admissible set of rearrangements. A feasible order $\sigma : N \to \{1, \ldots, n\}$ is called *admissible for S with respect to σ_0* if $P(\sigma_0, i) \cap (N\backslash S) = P(\sigma, i) \cap (N\backslash S)$ for all $i \in N$. Hence players (agents) of S are not allowed to jump over players outside coalition S. The set of admissible rearrangements of a coalition S with respect to σ_0 is denoted by $\mathcal{A}(S)$. Observe, since an admissible order is feasible, the disruption level is not violated.

The value of a coalition is defined as the maximal cost savings of the coalition which can be achieved by admissible rearrangements. Formally, the *partitioning sequencing game* (N, v) corresponding to a partitioning sequencing situation $\Gamma(N)$ is defined by:

$$v(S) = max_{\sigma\in\mathcal{A}(S)}\{\sum_{i\in S}[C(\sigma_0, i) - C(\sigma, i)]\} \tag{6}$$

for all $S \in 2^N$, and $v(\emptyset) = 0$.

Observe that the admissibility definition implies that the value of each disconnected coalition can be written as the sum of the value of its maximally connected components, i.e.,

$$v(T) = \sum_{S\in T\backslash\sigma_0} v(S) \tag{7}$$

where S is maximally connected in T if S is connected and $S \cup \{i\}$ is not connected for every $i \in T\backslash S$ and $T\backslash\sigma_0$ denotes the set of maximally connected components of T.

The following proposition shows that a partitioning sequencing game can be written as a non-negative linear combination of unanimity games. The *unanimity game* (N, u_S) for $S \subset N$ is defined by $u_S(T) = 1$ if $S \subset T$ and $u_S(T) = 0$ otherwise.

Proposition 1 *Let $\Gamma(N) = (N, J_1, J_2, \sigma_0, p, k)$ be a partitioning sequencing situation and let (N, v) be the corresponding partitioning sequencing game. Then*

$$v = \sum_{i,j\in J_1} g_{ij}u_{[i,j]} + \sum_{i,j\in J_2} g_{ij}u_{[i,j]} + \sum_{\substack{i\in J_1,\\ j\in B_{n_1+k}}} g_{ij}u_{[i,j]} + \sum_{\substack{i\in J_1,\\ j\in J_2\backslash B_{n_1+k}}} (g_{ij} - g_{ic_j})u_{[i,j]} \tag{8}$$

where $[i, j] = \{i, i + 1, ..., j - 1, j\}$ and B_{n_1+k} and c_j as defined in (1) and (2) respectively.

Proof. Because of (7), it is sufficient to prove the statement for connected coalitions. Let S be connected set and let σ^*_S be an optimal order of S. Then

$$\begin{aligned}
v(S) &= \sum_{i\in S} C(\sigma_0, i) - \sum_{i\in S} C(\sigma^*_S, i) \\
&= e^S W e^S \\
&= \sum_{i,j\in J_1\cap S} g_{ij} + \sum_{i,j\in J_2\cap S} g_{ij} + \sum_{\substack{i\in J_1\cap S\\ j\in B_{n_1+k}\cap S}} g_{ij} + \sum_{\substack{i\in J_1\cap S\\ j\in (J_2\backslash B_{n_1+k})\cap S}} (g_{ij} - g_{ic_j}) \\
&= \sum_{i,j\in J_1} g_{ij}u_{[i,j]}(S) + \sum_{i,j\in J_2} g_{ij}u_{[i,j]}(S) + \sum_{\substack{i\in J_1,\\ j\in B_{n_1+k}}} g_{ij}u_{[i,j]}(S) + \sum_{\substack{i\in J_1,\\ j\in J_2\backslash B_{n_1+k}}} (g_{ij} - g_{ic_j})u_{[i,j]}(S)
\end{aligned}$$

whereby the first equation is by (6) and the second equation by Lemma 1. The third equation follows from (5) and the last equation is obtained using the definition of unanimity games. □

Because unanimity games are convex games and partitioning sequencing games are a non-negative combination of convex games, we obtain the following theorem.

Theorem 2 *Partitioning sequencing games are convex games.*

In fact Proposition 1 shows that partitioning sequencing games are σ_0-pairing games. Curiel et al. (1994) call a game a σ_0-pairing game if v is an element of the non-negative cone generated by the games $\{u_{[ij]_{\sigma_0}} | \sigma_0(i) < \sigma_0(j)\}$, and they showed that the class of σ_0-pairing games contains also the class of sequencing games introduced in Curiel et al. (1989). Shapley (1953) introduced the Shapley value, defined as

$$\Phi_i(v) = \sum_{S:i\notin S} \frac{|S|!(n-1-|S|)!}{n!}(v(S\cup\{i\}) - v(S)).$$

The next theorem provides a game independent expression for the Shapley value.

Theorem 3 *Let $\Gamma(N) = (N, J_1, J_2, \sigma_0, p, k)$ be a partitioning sequencing situation and let (N, v) be the corresponding partitioning sequencing game. Then for all $m \in N$ it holds*

$$\begin{aligned}\Phi_m(v) &= \sum_{\substack{i,j\in J_1,\\ i\le m\le j}} \frac{g_{ij}}{j-i+1} + \sum_{\substack{i,j\in J_2,\\ i\le m\le j}} \frac{g_{ij}}{j-i+1} \\ &+ \sum_{\substack{i\in J_1, j\in B_{n_1+k},\\ i\le m\le j}} \frac{g_{ij}}{j-i+1} + \sum_{\substack{i\in J_1, j\in J_2\setminus B_{n_1+k},\\ i\le m\le j}} \frac{(g_{ij}-g_{ic_j})}{j-i+1}\end{aligned}$$

with B_{n_1+k} and c_j as defined in (1) and (2), respectively.

Proof. Let h_{ij}, with $i, j \in \{1, \cdots, n\}$ and $i < j$, be defined as

$$h_{ij} = \begin{cases} g_{ij} & \text{if } (i,j\in J_1) \text{ or } (i,j\in J_2) \text{ or } (i\in J_1, j\in B_{n_1+k}) \\ g_{ij} - g_{ic_j} & \text{if } i\in J_1, j\in J_2\setminus B_{n_1+k}\end{cases} \tag{9}$$

Then according to Proposition 1 we have $v = \sum_{i<j} h_{ij} u_{[ij]}$. Curiel et al. (1994) showed that the Shapley value of such a game is equal to

$$\Phi_m(v) = \sum_{\substack{i\le m\le j\\ i\neq j}} \frac{h_{ij}}{j-i+1},$$

which completes the proof. □

Note that the Shapley value for partitioning sequencing games divides the (extra) gain of two players i and j equally between all players involved, $(i, i+1, \cdots, j)$.

The next theorem provides an expression for the β-value. The β-value of a game is defined by Curiel et al. (1994), as the average of two marginal vectors. i.e. $\beta(v) = \frac{1}{2}m^{\sigma_0}(v) + \frac{1}{2}m^{\sigma_0^{-1}}(v)$, where σ_0^{-1} is the reverse order of σ_0, hence $\sigma_0^{-1}(i) = n - i + 1$.

Theorem 4 *Let $\Gamma(N) = (N, J_1, J_2, \sigma_0, p, k)$ be a partitioning sequencing situation and let (N, v) be the corresponding partitioning sequencing game. Then*

$$\beta(v) = We^N$$

Proof. Let $v = \sum_{i<j} h_{ij} u_{[ij]}$ with h_{ij} defined as in (9). Then

$$\begin{aligned}
\beta_j(v) &= \frac{1}{2}(\sum_{i<j} h_{ij} + \sum_{i>j} h_{ji}) \\
&= \frac{1}{2}(\sum_{\substack{i<j, i,j\in J_1 \\ i\in J_1, j\in B_{n_1+k} \\ i<j, i,j\in J_2}} g_{ij} + \sum_{i\in J_1, j\in J_2\setminus B_{n_1+k}} (g_{ij} - g_{ic_j}) + \sum_{\substack{j<i, j,i\in J_1 \\ j\in J_1, i\in B_{n_1+k} \\ j<i, j,i\in J_2}} g_{ji} + \sum_{\substack{j\in J_1, \\ i\in J_2\setminus B_{n_1+k}}} (g_{ji} - g_{jc_i}) \\
&= \begin{cases}
\frac{1}{2}\sum_{i<j} g_{ij} + \frac{1}{2}\sum_{j<i, i\in J_1} g_{ji} + \frac{1}{2}\sum_{i\in B_{n_1+k}} g_{ji} + \frac{1}{2}\sum_{i\in J_2\setminus B_{n_1+k}} (g_{ji} - g_{jc_i}) & \text{if } j \in J_1 \\
\frac{1}{2}\sum_{i<j} g_{ij} + \frac{1}{2}\sum_{i>j} g_{ji} & \text{if } j \in B_{n_1+k} \\
\frac{1}{2}\sum_{i<j, i\in J_2} g_{ij} + \frac{1}{2}\sum_{i\in J_1} (g_{ij} - g_{ic_j}) + \frac{1}{2}\sum_{i>j} g_{ji} & \text{if } j \in J_2\setminus B_{n_1+k}
\end{cases} \\
&= (We^N)_j,
\end{aligned}$$

where the first equality is a result from Curiel et al. (1994). □

Corollary 1 *Let $\Gamma(N) = (N, J_1, J_2, \sigma_0, p, k)$ be a partitioning sequencing situation and let (N, v) be the corresponding partitioning sequencing game. Then the aggregated $\mathcal{PEGS}$-rule of $\Gamma(N)$ coincides with the $\beta(v)$.*

Because partitioning sequencing games are convex games we get the following corollary.

Corollary 2 *Let $\Gamma(N) = (N, J_1, J_2, \sigma_0, p, k)$ be a partitioning sequencing situation and let (N, v) be the corresponding partitioning sequencing game. Then the aggregated $\mathcal{PEGS}$-rule provides a core element.*

4 Partitioning sequencing situations with more than two sets

In this section we discuss partitioning sequencing situations in which the jobs are partitioned in more than two connected sets. It is shown that the results obtained in the previous sections still hold in this more general setting.

Let $\Gamma(N, m) = (N, J_1, J_2, \cdots, J_m, \sigma_0, p, k_1, k_2, \cdots, k_{m-1})$ be an m-partitioning sequencing situation, i.e. a partitioning sequencing situation with m connected sets of jobs. Furthermore, let for each set J_s the last position be denoted by $N_s = \sum_{i\leq s} n_i$, where $n_i = |J_i|$. Like we had before, a certain disruption level is given for each set, i.e., the disruption level of the jobs from J_s is k_s, which means that these jobs have to be scheduled within the first $N_s + k_s$ positions. Observe that it is allowed to have a different disruption level for each set. Furthermore we assume that $k_s < n_{s+1}$ for all $s = \{1, \cdots, m-1\}$, which means that the jobs of set J_s cannot be disrupted by all jobs of the set J_{s+1}. Now we extend the algorithm of Hall and Potts (2004) straightforward to more sets.

Extended Hall and Potts Algorithm
$\Gamma(N, m) = (N, J_1, J_2, \cdots, J_m, \sigma_0, p, k_1, k_2, \cdots, k_{m-1})$ be an m-partitioning sequencing situation. In the first step of the algorithm, the SPT rule is applied to the partitioned sets $J_1, \cdots, J_m$ separately, which results in σ_1. For $r \in \{2, ..., m\}$, the order σ_r is obtained by applying the SPT rule on the jobs that are in position $N_{m-r} + 1, ..., N_{m-r+1} + k_{m-r+1}$ of the order σ_{r-1}.

Theorem 5 *Let $\Gamma(N) = (N, J_1, \cdots, J_m, \sigma_0, p, k_1, \cdots, k_{m-1})$ be a m-partitioning sequencing situation. Then the order σ_m obtained by the extended Hall and Potts Algorithm is an optimal order of N.*

Proof. The proof is by induction on the number of partitioning sets. For $m = 2$ the extended algorithm is equal to the algorithm of Hall and Potts (2004), which provides an optimal order of N. Now, assume for $m = q$ that σ_q is optimal. Hence, we have to prove that the extended algorithm provides an optimal order for $m = q + 1$. We will prove that σ_{q+1} is an optimal order for N.

The first step of the algorithm results in σ_1. Let $J'_{q+1} = \{\sigma_1^{-1}(N_q + k_q + 1), \cdots, \sigma_1^{-1}(n)\}$. Observe that the jobs in J'_{q+1} are the jobs with the largest processing time in J_{q+1}. Hence, there exists an optimal order σ^* of N such that $\sigma^{*-1}(i) = \sigma_1^{-1}(i)$ for all $i \in \{N_q + k_q + 1, \cdots, n\}$. Now, consider $\Gamma(N \backslash J'_{q+1}) = (N \backslash J'_{q+1}, J'_1, \cdots, J'_q, \sigma_1, p_{|N \backslash J'_{q+1}}, k_1, \cdots, k_{q-1})$ where J'_i is the SPT order of J_i, for $i \in \{1, \cdots, q-1\}$, and J'_q is the SPT order of $J_q \cup \{\sigma_1^{-1}(N_q + 1), \cdots, \sigma_1^{-1}(N_q + k_q)\}$. Since $\Gamma(N \backslash J'_{q+1})$ is partitioned in q sets, we have that the order τ_q, obtained after applying the extended Hall and Potts algorithm, is optimal for $\Gamma(N \backslash J'_{q+1})$. Now, define the order τ^* on N by $\tau^*(i) = \tau_q(i)$ if $i \in N \backslash J'_{k+1}$ and $\tau^*(i) = \sigma_{q+1}(i)$ if $i \in J'_{k+1}$. Obviously, τ^* is optimal for N and since $\tau_q(i) = \sigma_{q+1}(i)$ for all $i \in N \backslash J'_{k+1}$, the proof is complete. □

The following example illustrates the algorithm.

Example 3 Let $N = \{1, 2, \cdots, 12\}$, $J_1 = \{1, 2\}$, $J_2 = \{3, 4, 5\}$, $J_3 = \{6, 7, 8\}$, $J_4 = \{9, 10, 11, 12\}$, $\sigma_0 = (1, 2, \cdots, 12)$, $p = (10, 12, 11, 9, 5, 6, 8, 7, 4, 3, 2, 1)$ and $k_1 = k_2 = 2$ and $k_3 = 3$. By applying the first step of the algorithm, rearranging each set in SPT order, we get $\sigma_1 = \{1, 2, 5, 4, 3, 6, 8, 7, 12, 11, 10, 9\}$. Next, we rearrange the jobs on position $N_2+1, \cdots, N_3+k_3$, which are $\{\sigma_1(6), \cdots, \sigma_1(11)\} = \{6, 8, 7, 12, 11, 10\}$, resulting in the new order $\sigma_2 = \{1, 2, 5, 4, 3, 12, 11, 10, 6, 8, 7, 9\}$. Then, we apply the SPT rule to $\{5, 4, 3, 12, 11\}$ which results in $\sigma_3 = \{1, 2, 12, 11, 5, 4, 3, 10, 6, 8, 7, 9\}$. And last we apply SPT to $\{1, 2, 12, 11\}$ and get $\sigma_4 = \{12, 11, 1, 2, 5, 4, 3, 10, 6, 8, 7, 9\}$. Total cost savings: $638 - 491 = 147$. Figure 2 displays the steps of the extended algorithm of Hall and Pots. ◁

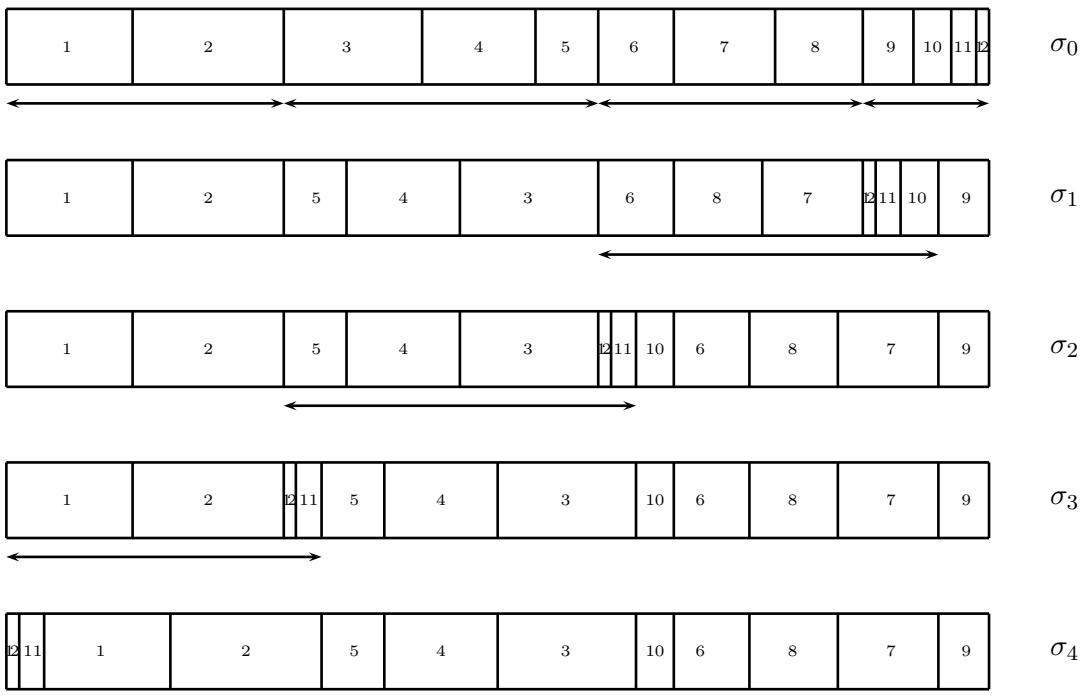

Figure 2: The steps in the extended algorithm of Hall and Potts.

Similar to the partitioning sequencing situation with two connected sets, we describe a non-aggregated solution in which the gains are divided. For notational convenience, we assume $k_1 \leq k_2 \leq \cdots \leq k_{m-1}$. Indeed, if k_i is not non-decreasing, then a similar approach can be followed, but the expressions of the $\mathcal{PEGS}$-rule become more complicated. Again we start with the recursive construction of the sets B_j^r for each $r = 1, \cdots m-1$ and $j = N_r + k_r, \cdots, n$

$$B_j^r = \begin{cases} P(\sigma_0, j) \cap J_{r+1} & \text{if } j = N_r + k_r \\ (B_{j-1}^r \cup \{j\}) \backslash \{c_j^r\} & \text{if } j = N_r + k_r + 1, \cdots, n \end{cases} \tag{10}$$

where

$$c_j^r = \text{argmin}\{t | t \in B_{j-1}^r \cup \{j\}, p_r \geq p_m \text{ for all } m \in B_{j-1}^r \cup \{j\}\} \tag{11}$$

is the agent with the largest processing time of the set $B_{j-1}^r \cup \{j\}$. Observe if $m = 2$, we are in the situation of two partitioning sets.

The division of gains between two jobs is done similarly as before with equal division for jobs from the same set and for jobs from set J_s and the first k_s jobs of the next set $B_{N_s+k_s}^s$. Furthermore, the net gains $(g_{ij} - g_{ic_j^s})$ from J_s and jobs behind $N_s + k_s$ are equally divided.

Now we can denote the (non-)aggregated $\mathcal{PEGS}$-rule for partitioning sequencing situations with more than two connected sets formally. Let $\Gamma(N)$ be a partitioning sequencing situation. Define the symmetric matrix W, where $i, j \in \{1, ..., n\}$ and $i \leq j$, as follows:

$$w_{ij} = \begin{cases} \frac{1}{2} g_{ij} & \text{for } (i, j \in J_r) \text{ or } (i \in J_r \text{ and } j \in B_{N_r+k_r}^r) \text{ and } r \in \{1, \cdots, m-1\} \\ \frac{1}{2}(g_{ij} - g_{ic_j^r}) & \text{for } i \in J_r \text{ and } j \in (\bigcup_{i=r+1}^m J_i) \backslash B_{N_r+k_r}^r \text{ and } r \in \{1, \cdots, m-1\} \end{cases} \tag{12}$$

where $B_{N_r+k_r}^r$ and c_j^r are calculated using (10) and (11) respectively. Then the non-aggregated $\mathcal{PEGS}$-rule is defined by $\mathcal{PEGS}(\Gamma(N)) = W$. The aggregated $\mathcal{PEGS}$-rule is defined as We^N. Example 4 illustrates the algorithm and the (non-)aggregated $\mathcal{PEGS}$-rule.

Example 4 Consider the partitioning sequencing situation of Example 3. Recall that the total cost savings are equal to 147 . We use (10) and (11) to calculate all B_j^r and c_j^r. We show the construction of the recursive sets B_j^r and c_j^r for $r = 1, 2$ and 3. Observe that for $r = 1$ we have $N_1 + k_1 = 4$ and consequently, $B_4^1 = \{3, 4\}$. Similarly, for $r = 2$ we have $N_2 + k_2 = 7$ and for $r = 3$ we have $N_3 + k_3 = 11$ which result in $B_7^2 = \{6, 7\}$ and $B_{11}^3 = \{9, 10, 11\}$, respectively.

	$r=1$		$r=2$		$r=3$	
$j=4,$		$B_4^1 = \{3,4\}$				
$j=5,$	$c_5^1 = 3$	$B_5^1 = \{4,5\}$				
$j=6,$	$c_6^1 = 4$	$B_6^1 = \{5,6\}$				
$j=7,$	$c_7^1 = 7$	$B_7^1 = \{5,6\}$		$B_7^2 = \{6,7\}$		
$j=8,$	$c_8^1 = 8$	$B_8^1 = \{5,6\}$	$c_8^2 = 7$	$B_8^2 = \{6,8\}$		
$j=9,$	$c_9^1 = 6$	$B_9^1 = \{5,9\}$	$c_9^2 = 8$	$B_9^2 = \{6,9\}$		
$j=10,$	$c_{10}^1 = 5$	$B_{10}^1 = \{9,10\}$	$c_{10}^2 = 6$	$B_{10}^2 = \{9,10\}$		
$j=11,$	$c_{11}^1 = 9$	$B_{11}^1 = \{10,11\}$	$c_{11}^2 = 9$	$B_{11}^2 = \{10,11\}$		$B_{11}^3 = \{9,10,11\}$
$j=12,$	$c_{12}^1 = 10$	$B_{12}^1 = \{11,12\}$	$c_{12}^2 = 10$	$B_{12}^2 = \{11,12\}$	$c_{12}^3 = 9$	$B_{12}^3 = \{10,11,12\}$

Note that the steps in B_j^r are straightforward. Using the calculations from the algorithm and applying (12) gives us the following result for the non-aggregated $\mathcal{PEGS}$-rule.

$$W = \begin{pmatrix} 0 & 0 & 0 & 0.5 & 2.5 & 1.5 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0.5 & 1.5 & 3 & 1.5 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0.5 & 0 & 1 & 3 & 2.5 & 1.5 & 0.5 & 1.5 & 1.5 & 1 & 1 \\ 0.5 & 1.5 & 1 & 0 & 2 & 1.5 & 0.5 & 0.5 & 1.5 & 1.5 & 1 & 1 \\ 2.5 & 3 & 3 & 2 & 0 & 0 & 0 & 0 & 0.5 & 1 & 1 & 1 \\ 1.5 & 1.5 & 2.5 & 1.5 & 0 & 0 & 0 & 0 & 1 & 1.5 & 2 & 1.5 \\ 0 & 0 & 1.5 & 0.5 & 0 & 0 & 0 & 0.5 & 2 & 2.5 & 3 & 1.5 \\ 0 & 0 & 0.5 & 0.5 & 0 & 0 & 0.5 & 0 & 1.5 & 2 & 2.5 & 1.5 \\ 1 & 1 & 1.5 & 1.5 & 0.5 & 1 & 2 & 1.5 & 0 & 0.5 & 1 & 1.5 \\ 1 & 1 & 1.5 & 1.5 & 1 & 1.5 & 2.5 & 2 & 0.5 & 0 & 0.5 & 1 \\ 1 & 1 & 1 & 1 & 1 & 2 & 3 & 2.5 & 1 & 0.5 & 0 & 0.5 \\ 1 & 1 & 1 & 1 & 1 & 1.5 & 1.5 & 1.5 & 1.5 & 1 & 0.5 & 0 \end{pmatrix}$$

Finally, the aggregated $\mathcal{PEGS}$-rule assigns to this partitioning sequencing situation the vector $We^N = (8.5, 10.5, 14, 12.5, 14, 13, 11.5 , 9, 13, 14, 14.5, 12.5)$. ◁

It turns out that the results we obtained, for the partitioning sequencing situation with two connected sets, in the previous sections, can easily be extended to a partitioning sequencing situation with more than two connected sets, because the set B_n^r represents exactly the jobs which can interchange with jobs from J_r without violating the disruption levels (like in the case with two connected sets). Therefore the proofs are a straight forward generalisation of the previous proofs, and therefore they are omitted.

Theorem 6 *Let $\Gamma(N,m) = (N, J_1, J_2, \cdots, J_m, \sigma_0, p, k_1, k_2, \cdots, k_{m-1})$ with $k_1 \leq \cdots \leq k_{m-1}$ be a m-partitioning sequencing situation with m partitioned sets, and let (N,v) be the corresponding partitioning sequencing game. Then the following statements hold:*

(a) The generalised $\mathcal{PEGS}$ is the unique non-aggregated rule satisfying efficiency, symmetry and consistency;

(b) The partitioning sequencing game (N,v) is convex;

(c) There is a game independent expression for the Shapley value of the game:

$$\Phi_h(v) = \sum_{\substack{i,j\in J_s,\\ i\leq h\leq j,\\ s\in\{1,\cdots,m\}}} \frac{g_{ij}}{j-i+1} + \sum_{\substack{i\in J_s, j\in B^s_{N_s+k_s},\\ i\leq h\leq j,\\ s\in\{1,\cdots,m\}}} \frac{g_{ij}}{j-i+1} + \sum_{\substack{i\in J_s, j\geq N_s+k_s,\\ i\leq h\leq j,\\ s\in\{1,\cdots,m\}}} \frac{(g_{ij} - g_{ic_j^s})}{j-i+1}$$

(d) $\beta(v) = We^N$;

(e) The generalised aggregated $\mathcal{PEGS}$-rule generates a core element.

5 Final remarks

In the former sections we studied partitioning sequencing situations in which all jobs have an equal weight and the processing times are chosen arbitrary. In this section we look at the general class of partitioning sequencing problems in which the weights and processing times of the jobs are arbitrary chosen.

First we consider the class of partitioning sequencing situations in which the processing times are all equal ($p = 1$) and the cost functions are linear, i.e., $c_i(t) = \alpha_i t$ with the weight $\alpha_i > 0$ for all $i \in N$. The objective is to minimise the total weighted completion time. In this class the Largest Urgency (LU)-rule, i.e., the jobs are ordered in a non-decreasing order of their weight factor, takes the role of the SPT-rule in the Hall and Potts (2004). Using the same arguments as before, we can show that we can obtain the same results as stated in Theorem 6, using $g_{ij} = (\alpha_j - \alpha_i)_+$.

The following Example shows that for partitioning sequencing games in which the processing times and weights are arbitrary, the corresponding partitioning sequencing games need not be convex.

Example 5 Let $(N, J_1, J_2, \sigma_0, p, \alpha, k)$ be a general partitioning sequencing situation with $N = \{1, 2, 3, 4\}$, $J_1 = \{1, 2\}$, $J_2 = \{3, 4\}$, σ_0 =1-2-3-4, $p = (1, 50, 10, 1)$,$\alpha = (5, 4, 11, 10)$ and $k = 1$. The game corresponding to this problem:
$v(\{i\}) = 0$ for all $i \in N$, $v(\{1, 2\}) = v(\{1, 3\}) = v(\{1, 4\}) = v(\{2, 4\}) = 0$, $v(\{2, 3\}) = v(\{1, 2, 3\}) = 510$, $v(\{3, 4\}) = v(\{1, 3, 4\}) = 89$, $v(\{1, 2, 4\}) = 0$, $v(\{2, 3, 4\}) = 585$ and $v(\{1, 2, 3, 4\}) = 590$.
Take $S = \{1, 3, 4\}$, $T = \{3\}$ and $i = 2$. Then it is easily to see that the game is not convex: $v(S \cup \{i\}) - v(S) \leq v(T \cup \{i\}) - v(T)$. ◁

Nevertheless, it follows from (7) and LeBreton et al. (1991) that partitioning sequencing games are balanced.

Acknowledgements: We wish to thank Marco Slikker for his useful comments on a preliminary version.

References

Borm, P., G. Fiestras-Janeiro, H. Hamers, E. Sanchez, and M. Voorneveld (2002). On the convexity of games corresponding to sequencing situations with due dates. *European Journal of Operational Research*, **136**, 616–634.

Calleja, P., A. Estevez-Fernandez, P. Borm, and H. Hamers (2006). Job scheduling, cooperation, and control. *OR Letters*, **34**.

Curiel, I., G. Pederzoli, and S. Tijs (1989). Sequencing games. *European Journal of Operational Research*, **40**, 344–351.

Curiel, I., J. Potters, V. Rajendra Prasad, S. Tijs, and B. Veltman (1994). Sequencing and cooperation. *Operations Research*, **42**, 566–568.

Curiel, I., J. Potters, V. Rajendra Prasad, S. Tijs, and B. Veltman (1994). Cooperation in one machine scheduling. *Zeitschrift für Operations Research (Now: Mathematical Methods of Operations Research)*, **38**, 113–129.

Feltkamp, V., S. Tijs, and S. Muto (1994). Minimum cost spanning extension problems: the proportional rule and the decentralized rule. CentER Discussion Paper 9496, Tilburg University, Tilburg, The Netherlands.

Hall, N. and C. Potts (2004). Rescheduling for new orders. *Operations Research*, **52**, 440–453.

Hamers, H., P. Borm, and S. Tijs (1995). On games corresponding to sequencing situations with ready times. *Mathematical Programming*, **70**, 1–13.

Hamers, H., F. Klijn, and J. Suijs (1999). On the balancedness of multimachine sequencing games. *European Journal of Operational Research*, **119**, 678–691.

Hamers, H., F. Klijn, and B. van Velzen (2005). On the convexity of precedence sequencing games. *Annals of Operations Research*, **137**, 161–175.

LeBreton, M., G. Owen, and S. Weber (1991). Strongly balanced cooperative games. *International Journal of Game Theory*, **20**, 419–427.

Owen, G. (1992). The assignment game: the reduced game. *Annals of Economics and Statistics*, **25**, 71–79.

Reijnierse, J., M. Maschler, J. Potters, and S. Tijs (1996). Simple flow games. *Games and Economic Behavior*, **16**, 238–260.

Shapley, L. (1953). A value for n-person games. In: H. Kuhn and A. Tucker (Eds.), *Contributions to the theory of games II*, Volume 28 of *Annals of Mathematics Studies*, pp. 307–317. Princeton: Princeton University Press.

Slikker, M. (2005). Balancedness of sequencing games with multiple parallel machines. *Annals of Operations Research*, **137**.

Slikker, M. (2006). Relaxed sequencing games have a nonempty core. *Naval Research Logistics*, **53**.

Suijs, J., H. Hamers, and S. Tijs (1997). On consistency of reward allocation rules in sequencing situations. In: W. Klein Haneveld, O. Vrieze, and L. Kallenberg (Eds.), *Ten years LNMB*, pp. 223–232. Amsterdam: CWI Tract.

Thomson, W. (1990). The consistency principle. pp. 187–215. San Diego, California: Academic Press.

Tijs, S. (1981). Bounds for the core and the τ-value. In: O. Moeschlin and D. Pallaschke (Eds.), *Game theory and mathematical economics*, pp. 123–132. Amsterdam: North Holland Publishing Company.

Velzen, B. van (2006). Sequencing games with controllable processing times. *European Journal of Operational Research*, **172**, 64–85.

Velzen, B. van and H. Hamers (2003). On the balancedness of relaxed sequencing games. *Mathematical Methods of Operations Research*, **57**, 287–297.