Drop Out Monotonic Rules for Sequencing Situations

Fernández, C.; Borm, P.E.M.; Hendrickx, R.L.P.; Tijs, S.H.

Publication date:
2002

No. 2002-51

DROP OUT MONOTONIC RULES FOR SEQUENCING SITUATIONS

By Cristina Fernández, Peter Borm, Ruud Hendrickx, Stef Tijs

May 2002

ISSN 0924-7815

Discussion paper

Drop Out Monotonic Rules for Sequencing Situations

Cristina Fernández[1] Peter Borm[2] Ruud Hendrickx[2,3]
Stef Tijs[2]

Abstract

This paper focuses on a new monotonicity property for sequencing situations. A sequencing solution is called drop out monotonic if no player will be worse off whenever one of the players decides to drop out of the queue before processing starts. This intuitively appealing property turns out to be very strong: we show that there is at most one rule satisfying both stability and drop out monotonicity. The existence of this rule is established in special classes of sequencing situations.

1 Introduction

For various classes of economic situations in which agents can cooperate, allocation rules have been developed, which handle the problem of allocating the rewards or cost savings from cooperation among the agents involved. Often, for such an economic situation, one can define a corresponding cooperative game. In this setting, *stable* rules are interesting, which assign allocations that are core elements of the corresponding games.

In this paper, we consider stable rules, which behave well in case agents drop out, leaving a reduced economic situation. We say that a rule is *drop out monotonic* if applying the rule to the reduced situation yields an allocation, which, depending on the context, either makes all remaining players better off or all players worse off than in the original situation.

[1]Facultad de Ciencias Económicas y Empresariales, Departamento de Economia Aplicada I, Universidad de Sevilla, Av. Ramón y Cajal, n°1. This author acknowledges financial support through DGICYT grant BFM 2001-2378.

[2]CentER and Department of Econometrics and Operations Research, Tilburg University, P.O. Box 90153, 5000 LE Tilburg, The Netherlands.

[3]Corresponding author. E-mail: ruud@kub.nl.

If the cooperative games corresponding to reduced situations are subgames of the the original game, then a stable and drop out monotonic rule generates a population monotonic allocation scheme (pmas) for the original game (Sprumont (1990)). In the cases of linear production situations (Owen (1975)), airport situations (Littlechild and Owen (1973)) and holding situations (Tijs et al. (2000)), the game corresponding to a reduced situation after one player drops out is a subgame of the original game. So here, the existence of stable and monotonic rules boils down to the existence of a pmas. Such pmas-es do not always exist for linear production games. However, for airport situations, the Shapley value induces one of many stable and drop out monotonic rules. For holding games, the rule which gives all gains to the so-called holding house keeper is also a pmas.

The property of drop out monotonicity introduced in this paper is inspired by the so-called fairness condition introduced in Ambec and Sprumont (2000). They study the problem of water management from a game theoretical point of view: given a river of certain capacity flowing through a number of countries with certain demand for water, how should the water of the river be allocated?

The fairness condition states that whenever one of the countries ceases to demand water (drops out), all other countries should be better off. Contrary to the examples mentioned before, the reduced situation after a player drops out does not give rise to a subgame of the original game. Ambec and Sprumont show that there is a unique allocation rule which satisfies both stability (ie, generates a core element) and the fairness condition. This rule (the μ-rule) is the marginal vector corresponding to the ordering of the countries along the river (from upstream to downstream).

This paper studies the drop out monotonicity property in the context of sequencing situations, as introduced in Curiel et al. (1989), in which there is also a natural ordering of the players forming the queue. Indeed, in the most basic class of sequencing situations (with linear cost functions), a result similar to Ambec and Sprumont is established. Within a more general class of sequencing situations (with regular cost functions), it turns out that there is *at most* one stable and drop out monotonic rule, which must be the analogue of the μ-rule. Finally, we introduce the class of sequencing situations with linear cost functions, in which one of the players faces a due date. It turns out that in this class, the μ-rule is indeed stable and drop out monotonic if the processing times of the agents are equal.

This paper is organised as follows. In section 2, we introduce the basic sequencing

model and define the μ-rule. In section 3, we define drop out monotonicity for sequencing situations and show that if the cost functions are regular, there can be at most one stable and drop out monotonic rule. In section 4, we introduce the class of sequencing situations with a single due date and show that in this class, the μ-rule is stable and drop out monotonic.

2 Sequencing situations

In a one-machine sequencing situation, as introduced in Curiel et al. (1989), there is a queue of players, each with one job, in front of a machine. Each player must have his job processed on this machine. The finite set of players is denoted by $N = \{1, ..., n\}$. The positions of the players in the queue are described by a bijection σ, where $\sigma(i) = j$ means that player i is at position j in the queue. The set of all such bijections is denoted by Π_N. We assume that the initial order on the jobs before the processing of the machine starts is $\sigma_0 \in \Pi_N$, defined by $\sigma_0(i) = i$ for all $i \in N$. The processing time $p_i > 0$ of the job of player i is the time the machine takes to handle this job. For each player $i \in N$, the costs of spending time in the system can be described by a cost function $k_i : \mathbb{R}_+ \to \mathbb{R}$, where $k_i(t)$ represents the costs of player i if his job is completed in t time units. Costs are assumed to be additive: the total costs of a coalition $S \subset N$ equal the sum of the individual costs of the members of S. Furthermore, the costs functions are regular, ie, increasing in t. A *sequencing situation* can be described by a triple (N, p, k) with $p = (p_i)_{i \in N}$ and $k = (k_i)_{i \in N}$.

In this section, we pay special attention to the class of sequencing situations with linear cost functions, ie, $k_i(t) = \alpha_i t$ for all $t \in \mathbb{R}_+$ with $\alpha_i \geq 0$. A *sequencing situation with linear cost functions* is denoted by (N, p, α) with $\alpha = (\alpha_i)_{i \in N}$.

The completion time $C(\sigma, i)$ of the job of player i if the jobs are processed (in a semi-active way) according to the order $\sigma \in \Pi_N$ is given by

$$C(\sigma, i) = \sum_{\{j \in N | \sigma(j) \leq \sigma(i)\}} p_j.$$

A processing order is called semi-active if there does not exist a job which could be processed earlier without altering the processing order, ie, if there are no unnecessary delays. The total costs of all players if the jobs are processed according to the order σ equal $\sum_{i \in N} k_i(C(\sigma, i))$. Clearly, because Π_N is finite, there exists an order for which total costs are minimised.

In the linear case, a processing order that minimises total costs of N is an order in which the jobs are processed in decreasing order with respect to the urgency index u_i defined by $u_i = \frac{\alpha_i}{p_i}$ (cf. Smith (1956)).

Example 2.1 Consider a linear one-machine sequencing situation (N, p, α), where $N = \{1, 2, 3\}$, $p = (2, 2, 1)$ and $\alpha = (4, 6, 5)$. Then the urgencies for the players are $u_1 = 2$, $u_2 = 3$ and $u_3 = 5$, respectively. Hence, the optimal processing order is $(3, 2, 1)$ with total costs $5 \cdot 1 + 6 \cdot 3 + 4 \cdot 5 = 43$. $\triangleleft$

Note that an optimal order can be obtained from the initial order by consecutive switches of neighbours i, j with i directly in front of j and $u_i < u_j$. This process will be referred to as the Smith algorithm.

For a sequencing situation (N, p, k) the costs $C_S(\sigma)$ of coalition S with respect to a processing order σ equal $C_S(\sigma) = \sum_{i \in S} k_i(C(\sigma, i))$. We want to determine the minimal costs of a coalition S when its members decide to cooperate. For this, we have to define which rearrangements of the coalition S are admissible with respect to the initial order. A bijection $\sigma \in \Pi_N$ is called *admissible* for S if it satisfies the following condition:

$$P(\sigma, j) = P(\sigma_0, j)$$

for all $j \in N \backslash S$, where for any $\tau \in \Pi_N$ the set of predecessors of a player $j \in N$ with respect to τ is defined as $P(\tau, j) = \{k \in N \mid \tau(k) < \tau(j)\}$.

This condition implies, in particular, that the starting time of each player outside the coalition S is equal to his starting time in the initial order and the players of S are not allowed to "jump" over players outside S. The set of admissible orders for a coalition S is denoted by $\mathcal{A}(S)$.

We define the *sequencing game* (N, c) corresponding to the sequencing situation (N, p, k) by

$$c(S) = \min_{\sigma \in \mathcal{A}(S)} \sum_{i \in S} k_i(C(\sigma, i)) \tag{2.1}$$

for all $S \subset N$.

In case the cost functions are linear, expression (2.1) can be rewritten in terms of $g_{ij} = \max\{0, \alpha_j p_i - \alpha_i p_j\}$, which equals the cost savings attainable by player i and j when i is directly in front of j, regardless of the exact position in the order. For this we need the notion of a connected coalition. A coalition S is called *connected*

with respect to σ if for all $i, j \in S$ and $\ell \in N$ such that $\sigma(i) < \sigma(\ell) < \sigma(j)$ it holds that $\ell \in S$. The Smith algorithm and (2.1) imply the following proposition.

Proposition 2.1 *Let (N, p, α) be a linear sequencing situation and let (N, c) be the corresponding sequencing game. Then for any coalition S that is connected with respect to σ_0 we have*

$$c(S) = \sum_{i \in S} \alpha_i C(\sigma_0, i) - \sum_{i,j \in S : \sigma_0(i) < \sigma_0(j)} g_{ij}.$$

For a coalition T that is not connected with respect to σ_0 the definition of admissible orders implies that

$$c(T) = \sum_{S \in T \backslash \sigma_0} c(S),$$

where $T \backslash \sigma_0$ is the set of components of T, a component of T being a maximally connected subset of T.

A *sequencing rule* is a function f assigning to every sequencing situation (N, p, k) a vector $f(N, p, k) \in \mathbb{R}^N_+$ such that $\sum_{i \in N} f_i(N, p, k) = c(N)$. In this paper, we investigate one specific rule for the class of sequencing games with regular cost functions: the μ-rule, which is the marginal vector corresponding to the initial order σ_0, defined by

$$\mu_j(N, p, k) = c(P_j) - c(P_{j-1})$$

for all $j \in N$, where $P_j = P(\sigma_0, j) = \{1, \ldots, j\}$. In case the cost functions are linear, we can use Proposition 2.1 to rewrite this as

$$\mu_j(N, p, \alpha) = c(\{j\}) - \sum_{i \in N : i < j} g_{ij}.$$

According to this rule, the gain g_{ij} goes fully to player j, who is behind i in the queue.

Since every sequencing game is σ_0-component additive (cf. Curiel et al. (1995)), $\mu(N, p, k) \in C(c)$ for every (N, p, k).[1] So letting the players at the front of the queue pay the highest costs and attributing the gains to the players at the back of the queue results in a stable outcome.

[1]For sequencing games arising from linear cost functions, concavity of these games can be used to establish stability of the μ-rule (cf. Curiel et al. (1989)).

3 Drop out monotonicity

Suppose that one player in the queue decides to wait no longer and drops out. One natural question in this situation is how the costs of the other players will be affected by this. It seems natural that none of the players should be worse off if one of them drops out of the queue. Formally, a rule f is called *drop out monotonic* if for all sequencing situations (N, p, k) and all $q \in N$ we have

$$f_j(N, p, k) \geq f_j((N, p, k)^{-q})$$

for all $j \in N \backslash \{q\}$, where $(N, p, k)^{-q} = (N \backslash \{q\}, (p_1, \ldots, p_{q-1}, p_{q+1}, \ldots, p_n), (k_1, \ldots, k_{q-1}, k_{q+1}, \ldots, k_n))$ and the initial order in $(N, p, k)^{-q}$ is σ_0 restricted to $N \backslash \{q\}$.

Proposition 3.1 *μ is drop out monotonic on the class of sequencing situations with linear cost functions.*

Proof: Let (N, p, α) be a sequencing situation with linear cost functions, let $q \in N$ and let $j \in N \backslash \{q\}$. If $j < q$, then $\mu_j(N, p, \alpha) = c(\{j\}) - \sum_{i \in N: i<j} g_{ij} = \mu_j((N, p, \alpha)^{-q})$. If $j > q$, then $\mu_j((N, p, \alpha)^{-q}) = (\sum_{i=1}^{j} p_i - p_q)\alpha_j - \sum_{i \in N: i<j} g_{ij} + g_{qj} = ((\sum_{i=i}^{j} p_i)\alpha_j - \sum_{i \in N: i<j} g_{ij}) + (g_{qj} - p_q\alpha_j) = \mu_j(N, p, \alpha) - \min\{p_j a_q, p_q a_j\} \leq \mu_j(N, p, \alpha)$. □

Proposition 3.1 shows that the μ-rule is drop out monotonic in case the cost functions are linear. The question now arises whether this is the only rule satisfying this property. In the following theorem, we show that within the class of sequencing situations with regular cost functions (not necessarily linear), the μ-rule is the only possible stable and drop out monotonic rule.

Theorem 3.2 *Let f be a rule on the class of sequencing situations with regular cost functions. If f is stable and drop out monotonic, then f equals the μ-rule.*

Proof: Let (N, p, k) be a sequencing situation with regular cost functions and let f be a stable and drop out monotonic rule. Denote the corresponding game by (N, c) and denote $f_i^S = f_i(S, (p_j)_{j \in S}, (k_j)_{j \in S})$ and $\mu_i = \mu_i(N, p, k)$ for all $i \in N$ and $S \subset N, S \neq \emptyset$. We show that $f = \mu$ by an inductive argument.

First, from drop out monotonicity it follows that $f_1^N \geq f_1^{\{1\}}$. From stability we have $f_1^N \leq c(\{1\}) = f_1^{\{1\}}$. Hence, $f_1^N = f_1^{\{1\}} = c(\{1\}) = \mu_1$.
Next, let $j \in \{2, \ldots n\}$. Assume $f_i^N = f_i^{P_{j-1}} = \mu_i$ for all $i \in P_{j-1}$. From drop out monotonicity we have $f_i^N \geq f_i^{P_j}$ for all $i \in P_j$, so $\sum_{i \in P_j} f_i^N \geq \sum_{i \in P_j} f_i^{P_j} = c(P_j)$. By stability, $\sum_{i \in P_j} f_i^N \leq c(P_j)$. So $\sum_{i \in P_j} f_i^N = c(P_j)$ and, using the induction hypothesis, $f_j^N = c(P_j) - \sum_{i \in P_{j-1}} f_i^N = c(P_j) - c(P_{j-1}) = \mu_j$.
Hence, we may conclude that $f = f^N = \mu$. □

It follows from Proposition 3.1 and Theorem 3.2 that drop out monotonicity and stability together characterise the μ-rule on the class of sequencing situations with linear cost functions.

Another interesting type of cost function arises when the players face a due date. Suppose player $i \in N$ must have his job completed before a certain due date $d_i \geq 0$. If he manages to do this, waiting is costless and if he is tardy, he has to pay a penalty of one unit. So,

$$k_i(t) = \begin{cases} 1 & \text{if } t > d_i, \\ 0 & \text{if } t \leq d_i. \end{cases} \tag{3.1}$$

Finding an optimal order for these cost functions boils down to minimising the number of tardy jobs. An efficient algorithm for this is provided by Moore (1968). The following proposition follows immediately.

Proposition 3.3 *The μ-rule is drop out monotonic on the class of sequencing situations with cost functions as in (3.1).*

4 Sequencing with a single due date

In this section, we consider the class of sequencing situations in which one of the players, $i^* \in N$, faces a due date, and the players in $N \backslash \{i^*\}$ have linear cost functions as before. Player i^* also has a linear cost parameter α_{i^*}, but if his job is completed after a certain due date d, he also has to pay a fixed penalty $\gamma > 0$.

Note that the cost functions in this framework are regular, so we can apply Theorem 3.2 to conclude that there is at most one stable and drop out monotonic rule, which must be the μ-rule. Contrary to the situation with only linear cost functions, however, the μ-rule may not be drop out monotonic, as shown in the next example.

Example 4.1 Consider the sequencing situation with $N = \{1, 2, 3\}$, processing times $p_1 = 1$, $p_2 = 5$ and $p_3 = 1$ and cost functions $k_1(t) = 1000t$,

$$k_2(t) = \begin{cases} 100 & \text{if } t > 5, \\ 0 & \text{if } t \leq 5, \end{cases}$$

and $k_3(t) = 10t$. For the grand coalition, the optimal order is $(1, 3, 2)$ with costs 1000+100+20=1120, while for coalition $\{1, 2\}$, the optimal order is $(1, 2)$ with costs 1000+100=1100. So, according to the μ-rule, player 3 should pay 20.

Now, consider the situation in which player 1 has dropped out of the queue. The optimal order for coalition $\{2, 3\}$ is then $(2, 3)$ with costs 0+60=60 and the costs for coalition $\{2\}$ equal 0. So, in this reduced situation, player 3 has to pay 60 according to the μ-rule, which is more than in the original situation. Hence, the μ-rule is not drop out monotonic. $\triangleleft$

In the remainder of this section, we restrict ourselves to sequencing situations with a single due date and equal processing times (which we assume to be 1). It turns out that for this class of situations, drop out monotonicity of the μ-rule can be established.

A *sequencing situation with a single due date* is represented by a 5-tuple $(N, \alpha, i^*, d, \gamma)$ with player set $N = \{1, \ldots, n\}$, a vector of cost parameters $\alpha \in \mathbb{R}^N_+$, player $i^* \in N$ facing the due date $d \in \mathbb{N}$ and penalty $\gamma > 0$. Such a situation is a sequencing situation (N, p, k) with $p_i = 1$ for all $i \in N$, $k_i(t) = \alpha_i t$ for all $i \in N \backslash \{i^*\}$ and

$$k_{i^*}(t) = \begin{cases} \alpha_{i^*} t & \text{if } t \leq d, \\ \alpha_{i^*} t + \gamma & \text{if } t > d. \end{cases}$$

Let $(N, \alpha, i^*, d, \gamma)$ be a sequencing situation with a single due date. With this situation we associate two games, $\hat{c}$ and c. The game $\hat{c}$ is the formal sequencing game defined by applying (2.1) to the actual cost functions $(k_i)_{i \in N}$. The game c is the auxiliary sequencing game that arises if γ is set to 0, ie, the game corresponding to the linear sequencing situation (N, p, α) with $p_i = 1$ for all $i \in N$.

Games arising from linear sequencing situations are convex. The single due date game $\hat{c}$ need not be convex, as is shown in the following example.

Example 4.2 Consider $(N, \alpha, i^*, d, \gamma)$ with $N = \{1, 2, 3\}$, $\alpha = (10, 2, 5)$, $i^* = 2$, $d = 1$ and $\gamma = 10$. Then $\hat{c}(N) - \hat{c}(\{1, 2\}) = 37 - 22 > 26 - 14 = \hat{c}(\{2, 3\}) - \hat{c}(\{2\})$. $\triangleleft$

Let $j \in N$ and let $\sigma_j \in \Pi_{P_j}$ be the unique urgency order on P_j (recall $P_j = \{1, \ldots, j\}$), where ties are broken by some fixed order on the players, starting with i^*. Then the following proposition is immediately clear.

Proposition 4.1 *The optimal order on P_j is either σ_j or the order in which the completion time of job i^* is exactly the due date and all other jobs are ordered in decreasing urgency.*

In order to prove drop out monotonicity of the μ-rule, we need to introduce some auxiliary lemmas and notation. By β_j we denote the cost of making player i^* in time, starting from the urgency order σ_j, so

$$\beta_j = \begin{cases} \displaystyle\sum_{i \in N: d \leq \sigma_j(i) < \sigma_j(i^*)} \alpha_i - (\sigma_j(i^*) - d)\alpha_{i^*} & \text{if } d < \sigma_j(i^*) \text{ and } i^* \leq j, \\ 0 & \text{if } \sigma_j(i^*) \leq d \text{ or } i^* > j. \end{cases}$$

Note that $\beta_j \geq 0$. From Proposition 4.1 it follows that

$$\hat{c}(P_j) = c(P_j) + A_j,$$

where $A_j = \min\{\gamma, \beta_j\}$ represents the extra costs as a result of the due date.

For $q \in N$, we denote by β_j^{-q} and A_j^{-q} the corresponding costs in the situation where player q has dropped out of the queue. Next, we define $\mu_j = \mu_j(N, p, \alpha) = c(P_j) - c(P_{j-1})$ and $\hat{\mu}_j = \mu_j(N, p, k) = \hat{c}(P_j) - \hat{c}(P_{j-1})$ and μ_j^{-q} and $\hat{\mu}_j^{-q}$ accordingly.

By Proposition 3.1 we have $\mu_j^{-q} \leq \mu_j$. To show that the μ-rule is drop out monotonic on the class of sequencing situations with a single due date, we have to show that $\hat{\mu}_j^{-q} \leq \hat{\mu}_j$ for all $j \neq q$, or equivalently,

$$A_j - A_{j-1} - (A_j^{-q} - A_{j-1}^{-q}) \geq -\min\{\alpha_j, \alpha_q\}, \tag{4.1}$$

where the right hand side equals $\mu_j^{-q} - \mu_j$ by the proof of Proposition 3.1.

For $i^* \leq j, q \leq j$ we have the following expressions for β:

$$\beta_j = \begin{cases} \displaystyle\sum_{\substack{i \in N: \\ d \leq \sigma_j(i) < \sigma_j(i^*)}} \alpha_i - (\sigma_j(i^*) - d)\alpha_{i^*} & \text{if } d < \sigma_j(i^*), \\ 0 & \text{if } \sigma_j(i^*) \leq d. \end{cases}$$

$$\beta_j^{-q} = \begin{cases} \beta_j & \text{if } d < \sigma_j(i^*) < \sigma_j(q), \\ \displaystyle\sum_{\substack{i\in N:\\ d\le\sigma_j(i)<\sigma_j(i^*)}} \alpha_i - \alpha_q - (\sigma_j(i^*) - d - 1)\alpha_{i^*} & \text{if } d < \sigma_j(q) < \sigma_j(i^*), \\ \displaystyle\sum_{\substack{i\in N:\\ d<\sigma_j(i)<\sigma_j(i^*)}} \alpha_i - (\sigma_j(i^*) - d - 1)\alpha_{i^*} & \text{if } \sigma_j(q) \le d < \sigma_j(i^*), \\ 0 & \text{if } \sigma_j(i^*) \le d. \end{cases}$$

For $q = j$, the previous implies

$$\beta_{j-1} = \begin{cases} \beta_j & \text{if } d < \sigma_j(i^*) < \sigma_j(j), \\ \displaystyle\sum_{\substack{i\in N:\\ d\le\sigma_j(i)<\sigma_j(i^*)}} \alpha_i - \alpha_j - (\sigma_j(i^*) - d - 1)\alpha_{i^*} & \text{if } d < \sigma_j(j) < \sigma_j(i^*), \\ \displaystyle\sum_{\substack{i\in N:\\ d<\sigma_j(i)<\sigma_j(i^*)}} \alpha_i - (\sigma_j(i^*) - d - 1)\alpha_{i^*} & \text{if } \sigma_j(j) \le d < \sigma_j(i^*), \\ 0 & \text{if } \sigma_j(i^*) \le d. \end{cases}$$

Finally, for $q \le j - 1$ we have

$$\beta_{j-1}^{-q} = \begin{cases} \beta_{j-1} & \text{if } \ d < \sigma_j(i^*) < \sigma_j(q), \\ \beta_j^{-q} & \text{if } \ d < \sigma_j(i^*) < \sigma_j(j), \\ & \quad \sigma_j(q) < \sigma_j(i^*), \\ \displaystyle\sum_{\substack{i\in N:\\ d\le\sigma_j(i)<\sigma_j(i^*)}} \alpha_i - \alpha_j - \alpha_q - (\sigma_j(i^*) - d - 2)\alpha_{i^*} & \text{if } \ d < \sigma_j(q) < \sigma_j(i^*), \\ & \quad d < \sigma_j(j) < \sigma_j(i^*), \\ \displaystyle\sum_{\substack{i\in N:\\ d<\sigma_j(i)<\sigma_j(i^*)}} \alpha_i - \alpha_q - (\sigma_j(i^*) - d - 2)\alpha_{i^*} & \text{if } \ \sigma_j(j) \le d < \sigma_j(q) < \sigma_j(i^*), \\ \displaystyle\sum_{\substack{i\in N:\\ d<\sigma_j(i)<\sigma_j(i^*)}} \alpha_i - \alpha_j - (\sigma_j(i^*) - d - 2)\alpha_{i^*} & \text{if } \ \sigma_j(q) \le d < \sigma_j(j) < \sigma_j(i^*), \\ \displaystyle\sum_{\substack{i\in N:\\ d+2\le\sigma_j(i)<\sigma_j(i^*)}} \alpha_i - (\sigma_j(i^*) - d - 2)\alpha_{i^*} & \text{if } \ \sigma_j(q) \le d, \\ & \quad \sigma_j(j) \le d, \\ & \quad d + 2 \le \sigma_j(i^*), \\ 0 & \text{if } \ \sigma_j(i^*) \le d \text{ or} \\ & \quad \sigma_j(q) \le d, \\ & \quad \sigma_j(j) \le d, \\ & \quad \sigma_j(i^*) = d + 1. \end{cases}$$

Lemma 4.2 *If $i^* \le j, q \le j$, then $\beta_j \ge \beta_j^{-q}$.*

Proof: Assume $i^* \le j, q \le j$. If $q = i^*$, then $\beta_j^{-q} = 0$ and hence, $\beta_j \ge \beta_j^{-q}$. So assume $q \ne i^*$ and distinguish between the following four cases:

a) $\sigma_j(i^*) < \sigma_j(q)$. Then $\beta_j = \beta_j^{-q}$.

b) $d < \sigma_j(q) < \sigma_j(i^*)$. Then $\beta_j - \beta_j^{-q} = \alpha_q - \alpha_{i^*} \geq 0$.

c) $\sigma_j(q) \leq d < \sigma_j(i^*)$. Then $\beta_j - \beta_j^{-q} = \alpha_{\sigma_j^{-1}(d)} - \alpha_{i^*} \geq 0$.

d) $\sigma_j(q) < \sigma_j(i^*) \leq d$. Then $\beta_j = \beta_j^{-q} = 0$.

□

Corollary 4.3 *If $i^* \leq j, q \leq j$, then $\beta_j \geq \beta_{j-1}$, $A_j \geq A_j^{-q}$ and $A_j \geq A_{j-1}$.*

In the following lemma, we establish (4.1) for some easy cases.

Lemma 4.4 *If $q \geq j$, $i^* \geq j$ or $q = i^*$, then $A_j - A_{j-1} - (A_j^{-q} - A_{j-1}^{-q}) \geq 0$.*

Proof: Distinguish between the following five cases:

a) $q > j$. Then $A_j^{-q} = A_j$ and $A_{j-1}^{-q} = A_{j-1}$.

b) $q = j$. Then $A_j^{-q} = A_{j-1}^{-q}$ and $A_j \geq A_{j-1}$ by Corollary 4.3.

c) $i^* > j$. Then $A_j = A_{j-1} = A_j^{-q} = A_{j-1}^{-q} = 0$.

d) $i^* = j$. Then $A_{j-1} = A_{j-1}^{-q} = 0$ and $A_j \geq A_j^{-q}$ by Corollary 4.3.

e) $i^* = q$. Then $A_j^{-q} = A_{j-1}^{-q} = 0$ and $A_j \geq A_{j-1}$ by Corollary 4.3.

□

Lemma 4.5 *If $i^* < j, q < j$, $q \neq i^*$ and $\alpha_j \geq \alpha_q$, then $\beta_j^{-q} \geq \beta_{j-1}$.*

Proof: Assume $i^* < j, q < j$, $q \neq i^*$ and $\alpha_j > \alpha_q$, then $\sigma_j(j) < \sigma_j(q)$. (The proof for $\alpha_j = \alpha_q$ is similar.) Distinguish between the following seven cases:

a) $d < \sigma_j(i^*) < \sigma_j(j) < \sigma_j(q)$. Then $\beta_j^{-q} = \beta_{j-1} = 0$.

b) $d < \sigma_j(j) < \sigma_j(i^*) < \sigma_j(q)$. Then $\beta_j^{-q} - \beta_{j-1} = \alpha_j - \alpha_{i^*} \geq 0$.

c) $\sigma_j(j) \leq d < \sigma_j(i^*) < \sigma_j(q)$. Then $\beta_j^{-q} - \beta_{j-1} = \alpha_{\sigma_j^{-1}(d)} - \alpha_{i^*} \geq 0$.

d) $d < \sigma_j(j) < \sigma_j(q) < \sigma_j(i^*)$. Then $\beta_j^{-q} - \beta_{j-1} = \alpha_j - \alpha_q \geq 0$.

e) $\sigma_j(j) \leq d < \sigma_j(q) < \sigma_j(i^*)$. Then $\beta_j^{-q} - \beta_{j-1} = \alpha_{\sigma_j^{-1}(d)} - \alpha_{i^*} \geq 0$.

f) $\sigma_j(j) < \sigma_j(q) \leq d < \sigma_j(i^*)$. Then $\beta_j^{-q} - \beta_{j-1} = 0$.

g) $\sigma_j(i^*) \leq d$. Then $\beta_j^{-q} = \beta_{j-1} = 0$.

□

Similarly, one can prove the following lemma.

Lemma 4.6 *If $i^* < j, q < j$, $q \neq i^*$ and $\alpha_j \leq \alpha_q$, then $\beta_{j-1} \geq \beta_j^{-q}$.*

In Lemmas 4.7 and 4.8, we establish (4.1) for the cases not covered in Lemma 4.4.

Lemma 4.7 *If $i^* < j, q < j$, $q \neq i^*$ and $\alpha_j \geq \alpha_q$, then $A_j - A_{j-1} - (A_j^{-q} - A_{j-1}^{-q}) \geq -\alpha_q$.*

Proof: Assume $i^* < j, q < j$, $q \neq i^*$ and $\alpha_j \geq \alpha_q$. It follows from Lemmas 4.2 and 4.5 that $\beta_j \geq \beta_j^{-q} \geq \beta_{j-1} \geq \beta_{j-1}^{-q}$. Define $\ell = A_j - A_{j-1} - (A_j^{-q} - A_{j-1}^{-q})$. Distinguish between the following five cases:

a) $\gamma \leq \beta_{j-1}^{-q}$. Then $\ell = 0$.

b) $\beta_{j-1}^{-q} < \gamma \leq \beta_{j-1}$. Then $\ell = \beta_{j-1}^{-q} - \gamma$.

c) $\beta_{j-1} < \gamma \leq \beta_j^{-q}$. Then $\ell = -\beta_{j-1} + \beta_{j-1}^{-q}$.

d) $\beta_j^{-q} < \gamma \leq \beta_j$. Then $\ell = \gamma - \beta_{j-1} - (\beta_j^{-q} - \beta_{j-1}^{-q})$.

e) $\beta_j < \gamma$. Then $\ell = \beta_j - \beta_{j-1} - (\beta_j^{-q} - \beta_{j-1}^{-q})$.

From these five cases it follows that $\ell \geq -\beta_{j-1} + \beta_{j-1}^{-q}$, so it suffices to show $\beta_{j-1} - \beta_{j-1}^{-q} \leq \alpha_q$. Assume $\alpha_j > \alpha_q$, then $\sigma_j(j) < \sigma_j(q)$. (The proof for $\alpha_j = \alpha_q$ is similar.) Distinguish between the following six cases:

a) $\sigma_j(i^*) < \sigma_j(q)$. Then $\beta_{j-1} = \beta_{j-1}^{-q}$.

b) $d < \sigma_j(j) < \sigma_j(q) < \sigma_j(i^*)$. Then $\beta_{j-1} - \beta_{j-1}^{-q} = \alpha_q - \alpha_{i^*} \leq \alpha_q$.

c) $\sigma_j(j) \leq d < \sigma_j(q) < \sigma_j(i^*)$. Then $\beta_{j-1} - \beta_{j-1}^{-q} = \alpha_q - \alpha_{i^*} \leq \alpha_q$.

d) $\sigma_j(q) \leq d \leq \sigma_j(i^*) - 2$. Then $\beta_{j-1} - \beta_{j-1}^{-q} = \alpha_{\sigma_j^{-1}(d+1)} - \alpha_{i^*} \leq \alpha_q$.

e) $\sigma_j(q) \leq d = \sigma_j(i^*) - 1$. Then $\beta_{j-1} = \beta_{j-1}^{-q}$.

f) $\sigma_j(i^*) \leq d$. Then $\beta_{j-1} = \beta_{j-1}^{-q} = 0$.

□

Similarly, one can prove the following lemma, using Lemma 4.6.

Lemma 4.8 *If $i^* < j, q < j$, $q \neq i^*$ and $\alpha_q \geq \alpha_j$, then $A_j - A_{j-1} - (A_j^{-q} - A_{j-1}^{-q}) \geq -\alpha_k$.*

From Lemmas 4.4, 4.7 and 4.8 one readily concludes the following theorem.

Theorem 4.9 *μ is drop out monotonic on the class of sequencing situations with a single due date.*

Since the resulting sequencing games are σ_0-component additive, the μ-rule also satisfies stability (cf. Curiel et al. (1995)). As a result, the μ rule is the unique stable and drop out monotonic rule on this class of situations.

References

Ambec, S. and Y. Sprumont (2000). Sharing a river. Cahiers de recherche du C.R.D.E. 08–2000, Université de Montréal, Montréal, Canada.

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 (1995). Sequencing and cooperation. *Operations Research*, **42**, 566–568.

Littlechild, S. and G. Owen (1973). A simple expression for the Shapley value in a special case. *Management Science*, **20**, 370–372.

Moore, J. (1968). An n-job, one machine sequencing algorithm for minimising the number of late jobs. *Management Science*, **15**, 102–109.

Owen, G. (1975). On the core of linear production games. *Mathematical Programming*, **9**, 358–370.

Smith, W. (1956). Various optimizers for single-stage production. *Naval Research Logistics Quarterly*, **3**, 59–66.

Sprumont, Y. (1990). Population monotonic allocation schemes for cooperative games with transferable utility. *Games and Economic Behavior*, **2**, 378–394.

Tijs, S., A. Meca, and M. López (2000). Benefit sharing in holding situations. CIO Discussion Paper I-2000-01, Universidad Miguel Hernández, Elche, Spain.