Finding and verifying the nucleolus of cooperative games

Márton Benedek* Jörg Fliege† Tri-Dung Nguyen‡

May 25, 2020

Abstract

The nucleolus offers a desirable payoff-sharing solution in cooperative games, thanks to its attractive properties —it always exists and lies in the core (if the core is non-empty), and it is unique. The nucleolus is considered as the most 'stable' solution in the sense that it lexicographically minimizes the dissatisfactions among all coalitions. Although computing the nucleolus is very challenging, the Kohlberg criterion offers a powerful method for verifying whether a solution is the nucleolus in relatively small games (i.e., with the number of players $n \leq 15$). This approach, however, becomes more challenging for larger games because of the need to form and check a criterion involving possibly exponentially large collections of coalitions, with each collection potentially of an exponentially large size. The aim of this work is twofold. First, we develop an improved version of the Kohlberg criterion that involves checking the 'balancedness' of at most $(n-1)$ sets of coalitions. Second, we exploit these results and introduce a novel descent-based constructive algorithm to find the nucleolus efficiently. We demonstrate the performance of the new algorithms by comparing them with existing methods over different types of games. Our contribution also includes the first open-source code for computing the nucleolus for games of moderately large sizes.

*Institute of Economics, Centre for Economic and Regional Studies, Hungarian Academy of Sciences, Tóth Kálmán u. 4., Budapest, 1097, Hungary, benedek.marton@krtk.mta.hu, ORCID: 0000-0002-7492-1174; Corvinus University of Budapest, Fővám tér 8., Budapest, 1093, Hungary; Budapest University of Technology and Economics, Egry József u. 1., Budapest, 1111, Hungary.

†Mathematical Sciences, University of Southampton, University Road, Southampton, SO17 1BJ, United Kingdom, J.Fliege@soton.ac.uk.

‡Mathematical Sciences, Business School and CORMSIS, University of Southampton, Southampton, SO17 1BJ, United Kingdom, T.D.Nguyen@soton.ac.uk.

1 Introduction

Cooperative games model situations where players can form coalitions to jointly achieve some objective. Assuming that it is more beneficial for the players to work together, a natural question is how to divide the reward of the collaboration among the players in such a way that ensures the stability of the grand coalition, i.e. avoiding any subgroup of players to break away in order to form their own coalition and increase their total payoff. Solution concepts in cooperative games provide the means to achieve this.

In a cooperative game (with transferable utilities), each coalition of players is associated with a value, a real number that represents what that coalition could achieve by working together, independently of other players. We are looking for a stable allocation of the value associated with the grand coalition, that includes every player in the game. A natural requirement from such an outcome is to allocate exactly the grand coalition value, and to do that *individually rationally*, i.e. each player should receive at least her stand-alone value. There are games where no such outcome exists, however, for our purposes in particular, we consider games where at least one individually rational outcome exists.

Applying the same concept to all groups of players, *coalitionally rational* outcomes form the core, guaranteeing to every coalition at least the amount that they could achieve by breaking away from the grand coalition. In this sense, core outcomes can be considered stable. However, it is possible that no payoff vector satisfies this condition, and a core outcome might not exist. Furthermore, in the appealing case of a non-empty core, one might find multiple core payoffs, offering possibly different levels of stability.

There are other solution concepts which provide outcomes that are, in a certain sense, as stable as possible. The first such solution concept is called the *least core*, which minimizes the worst level of dissatisfaction, i.e. the difference of what a coalition could achieve on their own and the amount allocated to the coalition, among all the coalitions. Note that least core payoffs always exist, but such a payoff vector might still not be unique.

Least core outcomes minimize the worst (largest) dissatisfaction level among all coalitions over the set of efficient payoff vectors, that allocate exactly the grand coalition value. Since there might be multiple of such outcomes, we might be interested in minimizing the second (third, etc.) largest dissatisfaction level of the remaining coalitions among these outcomes. By lexicographically minimizing the non-increasingly ordered dissatisfactions of all coalitions, we arrive at one of the most widely known solution concepts in cooperative game theory, the *nucleolus*, which is the 'most stable' individually rational outcome. In this paper we are focusing on the computation and the verification of the nucleolus.

The nucleolus was introduced in 1969 by Schmeidler [1] as a solution concept with attractive properties: it always exists (in a game with individually rational outcomes), it is unique, and it lies in the core, if the core is non-empty. Despite the desirable properties that the nucleolus has, its computation is, however, very challenging because the process involves the lexicographical minimization of 2^n excess values, where n denotes the number of players. While there is a few classes of games whose nucleoli can be computed in polynomial time (e.g. [2, 3, 4, 5, 6, 7]), it has been shown that finding the nucleolus is NP-hard for many classes of games, such as the utility games with non-unit capacities [6] and the weighted voting games [8].

While finding the nucleolus is very difficult, Kohlberg [9] provides a necessary and sufficient condition for a given imputation to be the nucleolus, which we will describe in the next section. This set of criteria is particularly useful for relatively small games (e. g. less than 10 players). The verification of it, however, becomes time consuming when the number of players exceeds 15, and becomes computationally extremely demanding when the number of players exceeds 20, even if we have an educated guess on the nucleolus based on the structure of a game. This is because the criterion involves the formation of collections of (tight) coalitions from all 2^n possible coalitions and iteratively verifying if unions of these collections are 'balanced' in a way to be described in details in Section 2.2. *The first aim of our work is to resolve these issues and propose a new improved set of criteria for verifying the nucleolus.*

Kopelowitz [10] suggested using nested linear programming (LP) to compute a closely related solution concept, the kernel of a game. This encouraged a number of researchers to focus on the computation of the nucleolus using LPs, rather than sharpening the Kohlberg criterion[1]. For example, Kohlberg [12] presents a single LP with $\mathcal{O}(2^n!)$ constraints which later on is improved by Owen [13] to $\mathcal{O}(4^n)$ constraints (at the cost of having larger coefficients). Puerto and Perea [14] recently introduced a different single-LP formulation with $\mathcal{O}(4^n)$ constraints and $\mathcal{O}(4^n)$ decision variables and with coefficients in $\{-1, 0, 1\}$. The nucleolus can also be found by solving a sequence of LPs. However, either the number of LPs involved is exponentially large ([15], [16]) or the sizes of the LPs are exponential ([17], [18], [19], [20]). *Our second aim is to directly solve the lexicographical minimization problem via introducing a new descent-based approach.* We compare our method with classical sequential LP methods (primal and dual sequences as described in [17]), the prolonged simplex method of [18], and the simplex implementation for finding the nucleolus from Derks and Kuipers [19].

The four key contributions of our work are:

[1]The only result we are aware of is the nonlinear approximation described in [11].

- We present a new set of necessary and sufficient conditions for a solution to be the nucleolus in Section 3.1. The number of collections of coalitions to be checked for balancedness is at most $(n-1)$ (instead of exponentially large as in the original Kohlberg criterion).
- We derive a new lexicographical descent algorithm for finding the nucleolus in Section 4. The new algorithm is distinguished from existing methods in that we directly solve the lexicographical minimization problem by iteratively finding improving directions through the balancedness checking procedure within the improved Kohlberg criterion.
- We demonstrate the performance of the proposed methods through numerical tests on various types of games in Section 5.
- We develop the first open-source code for computing the nucleolus of moderately large sizes in [21]. For completeness it also includes the implementation of algorithms from [17], [18] and [19].

In addition, we provide further contributions such as:

- The balancedness condition is essentially equivalent to solving a linear program with strict inequalities —a somewhat undesirable situation in mathematical programming. We provide an efficient tool for checking the balancedness condition in Section 3.3, requiring solving less number of LPs.
- While checking the Kohlberg criterion, we might end up having to store collections of exponentially large number of coalitions. We provide a method for reducing the storage size of these collections to at most $(n-1)$ coalitions in Section 3.2.

2 Notations and Preliminaries

2.1 Notations

Let n be the number of players and $\mathcal{N} = \{1, 2, \ldots, n\}$ be the set of all the players. A *coalition* $\mathcal{S}$ is a subset of players; i. e. $\mathcal{S} \subseteq \mathcal{N}$. The *characteristic function* $v : 2^{\mathcal{N}} \mapsto \mathbb{R}$ maps each coalition to a real number $v(\mathcal{S})$ (such that $v(\emptyset) = 0$). An outcome in a game is a payoff vector (payoffs, for short) $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ of real numbers, with x_i $(i \in \mathcal{N})$ being the share of player i. We focus on profit games and assume that it is more desirable to have higher shares. All our results can be extended to cost games through transforming the characteristic function to the corresponding profit game.

Let us denote $\mathbf{x}(\mathcal{S}) = \sum_{i \in \mathcal{S}} x_i$. Given the total payoff $v(\mathcal{N})$, *efficient* outcomes $\mathbf{x}$, also called *preimputations*, satisfy $\sum_{i \in \mathcal{N}} x_i = v(\mathcal{N})$. Let us denote by **PI** the set of these: $\mathbf{PI} = \{\mathbf{x} \in \mathbb{R}^n :$

$\mathbf{x}(\mathcal{N}) = v(\mathcal{N})\}$. The set of *imputations*, denoted by $\mathbf{I}$, contain efficient outcomes that satisfy *individual rationality*; that is, $x_i \geq v(\{i\}), \forall i \in \mathcal{N}$. The *core* of the game is the set of all efficient payoffs $\mathbf{x}$ such that no coalition has an incentive to break away, i.e. $\mathbf{x}(\mathcal{S}) \geq v(\mathcal{S})$ for all $\mathcal{S} \subsetneq \mathcal{N}$.

For each outcome $\mathbf{x}$, the *excess value* of a coalition $\mathcal{S}$ is defined as $d(\mathcal{S}, \mathbf{x}) := v(\mathcal{S}) - \mathbf{x}(\mathcal{S})$, which can be regarded as the level of dissatisfaction the players in coalition $\mathcal{S}$ have with respect to the proposed payoff vector $\mathbf{x}$. Then the *least core* is defined as follows: the set of preimputations $\{\mathbf{x} \in \mathbf{PI} : d(\mathbf{x}, \mathcal{S}) \leq \varepsilon^* \ \forall \mathcal{S} \subsetneq \mathcal{N}, \mathcal{S} \neq \emptyset\}$ form the least core, where ϵ^* is the smallest value such that the set is nonempty.

For any imputation $\mathbf{x}$, let $\Theta(\mathbf{x}) = (\Theta_1(\mathbf{x}), \Theta_2(\mathbf{x}), \ldots, \Theta_{2^n}(\mathbf{x}))$ be the vector of all the 2^n excess values at $\mathbf{x}$ sorted in a non-increasing order; i.e., $\Theta_i(\mathbf{x}) \geq \Theta_{i+1}(\mathbf{x})$ for all $1 \leq i < 2^n$. Let us denote $\Theta(\mathbf{x}) <_L \Theta(\mathbf{y})$ if there exists $r \leq 2^n$ such that $\Theta_i(\mathbf{x}) = \Theta_i(\mathbf{y}), \forall 1 \leq i < r$ and $\Theta_r(\mathbf{x}) < \Theta_r(\mathbf{y})$. Then $\nu(\mathcal{N}, v) \in \mathbf{I}$ is the *nucleolus* (ν for short) if $\Theta(\nu) <_L \Theta(\mathbf{x}),\ \forall \mathbf{x} \in \mathbf{I},\ \mathbf{x} \neq \nu$.

If we only require $\mathbf{x}$ and ν to be preimputations, we arrive at the definition of the prenucleolus, which can be seen as the most stable efficient outcome. In this paper every result is focusing on the nucleolus, hence throughout the paper we consider only games with non-empty imputation set. However, the aim is to develop algorithms applicable to a general class of games, thus we make no further assumptions on the characteristic function. Moreover, with suitable modifications, every result can be applied to the prenucleolus, making them applicable to every cooperative game (with transferable utilities).

For each collection $Q \subseteq 2^{\mathcal{N}}$, let us denote the size of Q by $|Q|$. We associate each collection Q with a weight vector in $\mathbb{R}^{|Q|}$ with each element denoting the weight of the corresponding coalition in Q. Throughout this paper, we use bold font for vectors and italic font for scalars. Whenever it is clear from context, we are going to omit the argument $\mathbf{x}$ from maximal dissatisfaction levels ϵ_k, tight sets T_0 and T_k, collection of tight sets H_k, and so on (the latter notions introduced in Section 2.2).

For $\mathcal{S} \subseteq \mathcal{N}$, let us denote by $\mathbf{e}(\mathcal{S})$ the *characteristic vector of* $\mathcal{S}$ in $\{0,1\}^n$ whose ith element is equal to one if and only if player i is in coalition $\mathcal{S}$. With this, for all $\mathbf{x} \in \mathbb{R}^n$, we have $\mathbf{x}(\mathcal{S}) = \sum_{i \in \mathcal{S}} x_i = \mathbf{x}^T \mathbf{e}(\mathcal{S})$. Furthermore we can consider (linear) spans and the rank of collections: coalition $\mathcal{S}$ is in the linear span of collection Q if its characteristic vector $\mathbf{e}(\mathcal{S})$ is in $\text{span}(\{\mathbf{e}(\mathcal{T}) : \mathcal{T} \in Q\})$ and $\text{rank}(Q) := \text{rank}(\{\mathbf{e}(\mathcal{T}) : \mathcal{T} \in Q\})$. Next, we formally define the concept of balancedness.

Definition 1. *A collection of coalitions $Q \subseteq 2^{\mathcal{N}}$ is* balanced *if there exists a weight vector $\omega \in \mathbb{R}^{|Q|}_{>0}$ such that $\mathbf{e}(\mathcal{N}) = \sum_{\mathcal{S} \in Q} \omega_{\mathcal{S}} \mathbf{e}(\mathcal{S})$. Given a collection $T_0 \subseteq 2^{\mathcal{N}}$, a collection $Q \subseteq 2^{\mathcal{N}}$ is called* T_0-balanced *if there exist weight vectors $\gamma \in \mathbb{R}^{|T_0|}_{\geq 0}$ and $\omega \in \mathbb{R}^{|Q|}_{>0}$ such that $\mathbf{e}(\mathcal{N}) = \sum_{\mathcal{S} \in T_0} \gamma_{\mathcal{S}} \mathbf{e}(\mathcal{S}) + \sum_{\mathcal{S} \in Q} \omega_{\mathcal{S}} \mathbf{e}(\mathcal{S})$.*

Remark 1. *We make the following observations about balancedness:*

a) Balancedness implies T_0-balancedness for any T_0, while for $T_0 = \emptyset$ the two concepts are equivalent.

b) All results in this paper are concerned with the nucleolus. These results and the corresponding algorithms to be described can be adapted for the prenucleolus by setting $T_0 = \emptyset$.

2.2 Algorithmic view of the Kohlberg criterion

We first formalize the concept of balancedness and summarize the main results of Kohlberg [9] from an algorithmic viewpoint. For any efficient payoff distribution $\mathbf{x} \in \mathbf{PI}$, Kohlberg [9] first defines the following sets of coalitions: $T_0(\mathbf{x}) = \{\{i\}, i = 1, \ldots, n \ : \ x_i = v(\{i\})\}$, $H_0(\mathbf{x}) = \{\mathcal{N}\}$ and $H_k(\mathbf{x}) = H_{k-1}(\mathbf{x}) \cup T_k(\mathbf{x}), k = 1, 2, \ldots$, where for each $k \geq 1$,

$$T_k(\mathbf{x}) = \underset{\mathcal{S} \notin H_{k-1}(\mathbf{x})}{\operatorname{argmax}} \{v(\mathcal{S}) - x(\mathcal{S})\}, \quad \epsilon_k(\mathbf{x}) = \max_{\mathcal{S} \notin H_{k-1}(\mathbf{x})} \{v(\mathcal{S}) - x(\mathcal{S})\}.$$

Here, $T_k(\mathbf{x})$ includes all coalitions that have the same excess value $\epsilon_k(\mathbf{x})$ and $\epsilon_1(\mathbf{x}) > \epsilon_2(\mathbf{x}) > \ldots$, while $T_0(\mathbf{x})$ contains the players for which $\mathbf{x}$ is on the boundary of violating individual rationality. We call $T_k(\mathbf{x})$ the set of 'tight' coalitions in the sense that coalition $\mathcal{S}$ belongs to $T_k(\mathbf{x})$ if and only if the constraint $v(\mathcal{S}) - x(\mathcal{S}) = \epsilon_k(\mathbf{x})$ is active/tight. In the followings, the terms 'collection of coalitions' (collection for short) and 'subset of the power set $2^{\mathcal{N}}$' are equivalent and are used interchangeably.

For any collection of coalitions Q, let us define

$$Y(Q) = \{\mathbf{y} \in \mathbb{R}^n \ : \ \mathbf{y}(\mathcal{S}) \geq 0 \ \forall \mathcal{S} \in Q, \ \mathbf{y}(\mathcal{N}) = 0\}.$$

We have $Y(Q) \neq \emptyset$ since $\mathbf{0} \in Y(Q)$. The first key result in Kohlberg [9] that will be exploited in this work is the following lemma:

Lemma 1 (Kohlberg [9])**.** *Given a collection $T_0 \subseteq 2^{\mathcal{N}}$, a collection $T \subseteq 2^{\mathcal{N}}$ is T_0-balanced if and only if $\mathbf{y} \in Y(T_0 \cup T)$ implies $\mathbf{y}(\mathcal{S}) = 0, \forall \mathcal{S} \in T$.*

This result allows the author to define two sets of equivalent properties regarding a sequence of collections $(Q_0, Q_1, \ldots)$:

Definition 2. *$(Q_0, Q_1, \ldots)$ has Property I if for all $k \geq 1$, the following claim holds: $\mathbf{y} \in Y(\cup_{j=0}^{k} Q_j)$ implies $\mathbf{y}(\mathcal{S}) = 0, \ \forall \mathcal{S} \in \cup_{j=1}^{k} Q_j$.*

Definition 3. *$(Q_0, Q_1, \ldots)$ has Property II if for all $k \geq 1$, $\cup_{j=1}^{k} Q_j$ is Q_0-balanced.*

The main result of [9] can be summarized in the following theorem:

Theorem 1 (Kohlberg [9]). *For games with a non-empty imputation set, the followings are equivalent: (a)* $\mathbf{x}$ *is the nucleolus; (b)* $(T_0(\mathbf{x}), T_1(\mathbf{x}), \ldots)$ *has Property I; (c)* $(T_0(\mathbf{x}), T_1(\mathbf{x}), \ldots)$ *has Property II.*

For the sake of completeness, in Appendix A of the e-companion [22] we provide a proof of Theorem 1 slightly different than the one in [9]. To appreciate the practicality of the Kohlberg criterion and for convenient development later, we present the algorithmic view of the criterion in Algorithm 1.

Algorithm 1: (Original) Kohlberg algorithm for verifying if a payoff vector is the nucleolus of a cooperative game.

Input: Game $(\mathcal{N}, v)$, imputation $\mathbf{x} \in \mathbf{I}$;

Output: Conclude if $\mathbf{x}$ is the nucleolus or not;

1. Initialization: Set $H_0 = \{\mathcal{N}\}$, $T_0 = \{\{i\} : x_i = v(\{i\}), i = 1, \ldots, n\}$ and $k = 1$;

while $H_{k-1} \neq 2^{\mathcal{N}} \setminus \{\emptyset\}$ **do**

2. Set $T_k = \underset{\mathcal{S} \notin H_{k-1}}{\operatorname{argmax}} \{v(\mathcal{S}) - \mathbf{x}(\mathcal{S})\}$;

if $(\cup_{j=1}^{k} T_j)$ *is* T_0*-balanced* **then**

3. Set $H_k = H_{k-1} \cup T_k$, $k = k + 1$ and continue

else

4. Stop the algorithm and conclude that $\mathbf{x}$ **is not** the nucleolus

end

end

5. Conclude that $\mathbf{x}$ **is** the nucleolus.

In this algorithm, we iteratively form the tight sets T_j $(j = 0, 1, \ldots)$ until either all the coalitions are included, and we conclude that the input payoff vector is the nucleolus (i.e. stopping at Step 5), or stop at a point where the union of the tight coalitions is not T_0-balanced (in Step 4), in which case we conclude that the payoff vector is not the nucleolus.

3 An improved Kohlberg criterion

The Kohlberg criterion, as described in Section 2.2, offers a powerful tool to assess whether a given payoff distribution is the nucleolus by providing necessary and sufficient conditions. These conditions can be used in relatively small or well-structured games, where a potential candidate for the nucleolus can be easily identified and where checking the balancedness of the corresponding tight sets can be done easily (possibly analytically). For larger games, it is inconvenient to apply the Kohlberg criterion as it

could involve forming and checking the balancedness of exponentially large number of subsets of tight coalitions (this is the case when the *while loop* in Algorithm 1 takes an exponentially large number of steps), each of which could be of exponentially large size. This section aims to resolve these issues.

3.1 Bounding the number of iterations to $(n-1)$

The key idea to check the Kohlberg criterion in a more efficient way is to note that, once we have obtained and verified the T_0-balancedness of $\cup_{j=1}^{k} T_j$, we do not have to be concerned about those coalitions that belong to $\text{span}(\cup_{j=1}^{k} T_j)$. In brief, this is because once a collection is T_0-balanced, its span is also T_0-balanced as formalized in the following lemma:

Lemma 2. *For any collection $T_0 \subseteq 2^{\mathcal{N}}$, the following results hold:*

(a) If a collection T is T_0-balanced, then $\text{span}(T)$ is also T_0-balanced[2]*.*

(b) If collections U, V are T_0-balanced then $U \cup V$ and $\text{span}(U) \cup \text{span}(V)$ are also T_0-balanced.

(c) If U is T_0-balanced and $U \subseteq V$, then $\text{span}(U) \cap V$ is also T_0-balanced.

We provide a proof of Lemma 2 in Appendix E of [22]. With these results, we can provide an improved Kohlberg algorithm as shown in Algorithm 2.

The differences between Algorithm 2 and Algorithm 1 are: (a) the stopping condition of the while loop has been changed from $H_{k-1} \neq 2^{\mathcal{N}} \setminus \{\emptyset\}$ to $\text{rank}(H_{k-1}) < n$, and (b) the search space at Step 2 has been changed from $\mathcal{S} \notin H_{k-1}$ to $\mathcal{S} \notin \text{span}(H_{k-1})$. As a result, we have the following desirable property:

Theorem 2. *The while-loop in Algorithm 2 terminates after at most $(n-1)$ iterations and it correctly decides whether a given imputation is the nucleolus.*

Proof. First, by the construction in Step 2 of the algorithm, $T_k \cap \text{span}(H_{k-1}) = \emptyset$ and hence, by Step 3, we have that $\text{rank}(H_k) = \text{rank}(H_{k-1} \cup T_k)$ keeps increasing. Therefore,

$$n \geq \text{rank}(H_k) = \text{rank}(H_{k-1} \cup T_k) \geq \text{rank}(H_{k-1}) + 1 \geq \text{rank}(H_0) + k = k + 1,$$

and hence the algorithm (i.e. the while loop) terminates in at most $(n-1)$ iterations. Here, we also note that the algorithm terminates at either Step 4 or Step 5 with complementary conclusions.

[2] Lemma 2.4 from [23].

Algorithm 2: Improved Kohlberg Algorithm for verifying if a payoff vector is the nucleolus.

Input: Game $(\mathcal{N}, v)$, imputation $\mathbf{x} \in \mathbf{I}$;

Output: Conclude if $\mathbf{x}$ is the nucleolus or not;

1. Initialization: Set $H_0 = \{\mathcal{N}\}$, $T_0 = \{\{i\} : x_i = v(\{i\}), i = 1, \ldots, n\}$ and $k = 1$;

while $\text{rank}(H_{k-1}) < n$ **do**

2. Find $T_k = \underset{\mathcal{S} \notin \text{span}(H_{k-1})}{\text{argmax}} \{v(\mathcal{S}) - \mathbf{x}(\mathcal{S})\}$;

if $(\cup_{j=1}^{k} T_j)$ *is* T_0*-balanced* **then**

3. Set $H_k = H_{k-1} \cup T_k$, $k = k + 1$ and continue;

else

4. Stop the algorithm and conclude that $\mathbf{x}$ **is not** the nucleolus.

end

end

5. Conclude that $\mathbf{x}$ **is** the nucleolus.

Proving that the algorithm correctly decides whether an impuation is the nucleolus is equivalent to showing that (a) if $\mathbf{x}$ is the nucleolus then the algorithm correctly terminates at Step 5, and (b) if the algorithm terminates at Step 5, then the input payoff vector must be the nucleolus.

Part (a): We first note that, although the sequences of T_k and H_k generated from Algorithm 2 are generally different from those in Algorithm 1, these are the same in the initialization and the first iteration; that is, T_0, T_1, H_0, H_1 are the same in both algorithms. Therefore, if $\mathbf{x}$ is the nucleolus, then T_1 must be T_0-balanced as a direct result from the Kohlberg criterion described in Theorem 1. Thus, the algorithm goes through to Step 3 at $k = 1$. Suppose, for the purpose of deriving a contradiction, that the algorithm goes through to Step 4 instead of Step 5, for some index $k > 1$; that is $(\cup_{j=1}^{k} T_j)$ is not T_0-balanced. By Lemma 1, there exists $\mathbf{y} \in \mathbb{R}^n$ such that

$$\mathbf{y}(\mathcal{S}) \geq 0, \forall \mathcal{S} \in \cup_{j=0}^{k} T_j;\ \mathbf{y}(\mathcal{N}) = 0;\ \mathbf{y}(\mathcal{S}') > 0,\ \text{for some } \mathcal{S}' \in \cup_{j=1}^{k} T_j. \tag{1}$$

Notice, however, that $\cup_{j=1}^{k-1} T_j$ is T_0-balanced by the construction in Step 3 of the previous iteration. Therefore, $\mathcal{S}' \notin H_{k-1}$ since otherwise Lemma 1 is violated. Thus, $\mathcal{S}' \in T_k$ and hence (1) leads to

$$(\mathbf{x} + \mathbf{y})(\mathcal{S}) \geq \mathbf{x}(\mathcal{S}), \forall \mathcal{S} \in T_k;\ (\mathbf{x} + \mathbf{y})(\mathcal{S}') > \mathbf{x}(\mathcal{S}'),\ \text{for some } \mathcal{S}' \in T_k.$$

As a result

$$d(\mathcal{S}, \mathbf{x} + \mathbf{y}) \leq d(\mathcal{S}, \mathbf{x}), \forall \mathcal{S} \in T_k;\ d(\mathcal{S}', \mathbf{x} + \mathbf{y}) < d(\mathcal{S}', \mathbf{x}),\ \text{for some } \mathcal{S}' \in T_k;$$

that is, for all coalitions in T_k, the corresponding excess values for $(\mathbf{x}+\mathbf{y})$ are not greater than that of $\mathbf{x}$ with at least one strict inequality for some coalition $\mathcal{S}'$. Thus,

$$\Phi^{T_k}(\mathbf{x}+\mathbf{y}) <_L \Phi^{T_k}(\mathbf{x}), \tag{2}$$

where, for each collection of coalitions Q, Φ^Q is the non-increasingly ordered excess values with respect to *only* those coalitions in Q. Since H_{k-1} is T_0-balanced by the construction in Step 3 of the previous iteration, $\text{span}(H_{k-1})$ is also T_0-balanced by Lemma 2. Thus, $\mathbf{y}(\mathcal{S}) = 0,\ \forall \mathcal{S} \in \text{span}(H_{k-1})$ and

$$\Phi^{\text{span}(H_{k-1})}(\mathbf{x}+\mathbf{y}) =_L \Phi^{\text{span}(H_{k-1})}(\mathbf{x}). \tag{3}$$

From (2) and (3) we have

$$\Phi^{\text{span}(H_{k-1})\cup T_k}(\mathbf{x}+\mathbf{y}) <_L \Phi^{\text{span}(H_{k-1})\cup T_k}(\mathbf{x}). \tag{4}$$

Note that (4) also holds if we scale $\mathbf{y}$ by any positive factor δ, i. e.

$$\Phi^{\text{span}(H_{k-1})\cup T_k}(\mathbf{x}+\delta\mathbf{y}) <_L \Phi^{\text{span}(H_{k-1})\cup T_k}(\mathbf{x}). \tag{5}$$

For all $\mathcal{S} \notin (\text{span}(H_{k-1}) \cup T_k)$ we have $v(\mathcal{S}) - \mathbf{x}(\mathcal{S}) < \epsilon_k$. Thus, there exists $\delta > 0$ small enough such that $\mathbf{x}+\delta\mathbf{y}$ is an imputation and that

$$v(\mathcal{S}) - (\mathbf{x}+\delta\mathbf{y})(\mathcal{S}) < \epsilon_k,\ \forall \mathcal{S} \notin (\text{span}(H_{k-1}) \cup T_k). \tag{6}$$

Results (5) and (6) imply that the $|\text{span}(H_{k-1}) \cup T_k|$ largest excess values at $\mathbf{x}$ are lexicographically larger than those at $(\mathbf{x}+\delta\mathbf{y})$. As a result, $\Phi(\mathbf{x})$ is lexicographically larger than $\Phi(\mathbf{x}+\delta\mathbf{y})$ considering all coalitions, which means $\mathbf{x}$ is not the nucleolus, i. e. we have arrived at a contradiction.

Part (b): If the algorithm bypassed Step 4 and went to Step 5, then $(\cup_{j=1}^{k} T_j)$ is T_0-balanced for all k until $\text{rank}(H_{k-1}) = n$. Let $\mathbf{z}$ be the nucleolus; then by its definition, its worst excess value should be no larger than the worst excess value of $\mathbf{x}$, which is equal to ϵ_1. Thus, the excess value of $\mathbf{z}$ over any coalition, including those in T_1, must be at most ϵ_1; i. e.

$$(\mathbf{z}-\mathbf{x})(\mathcal{S}) \geq 0, \forall \mathcal{S} \in T_1.$$

Notice that $(\mathbf{z}-\mathbf{x})(\mathcal{N}) = 0$ and $(\mathbf{z}-\mathbf{x})(\mathcal{S}) \geq 0, \forall \mathcal{S} \in T_0$ by the construction of T_0 and because $\mathbf{z} \in \mathbf{I}$. Then since T_1 is T_0-balanced, we have by Lemma 1 that $(\mathbf{z}-\mathbf{x})(\mathcal{S}) = 0$ for all $\mathcal{S} \in T_1$. Using a similar argument, given that $\mathbf{x}$ and $\mathbf{z}$ are lexicographically equivalent on $\text{span}(T_1)$ and since $\mathbf{z}$ is the nucleolus, we also have $(\mathbf{z}-\mathbf{x})(\mathcal{S}) \geq 0, \forall \mathcal{S} \in T_2$. Thus,

$$(\mathbf{z}-\mathbf{x})(\mathcal{S}) \geq 0, \forall \mathcal{S} \in T_1 \cup T_2.$$

Again, given that $(T_1 \cup T_2)$ is T_0-balanced, we have by Lemma 1 that $(\mathbf{z}-\mathbf{x})(\mathcal{S}) = 0$ for all $\mathcal{S} \in T_1 \cup T_2$. We can continue and use an induction argument to show that $(\mathbf{z}-\mathbf{x})(\mathcal{S}) = 0$ for all $\mathcal{S} \in H_{k-1}, k \geq 1$. Given that $\text{rank}(H_{k-1}) = n$, we have $\mathbf{x} = \mathbf{z}$, i.e. $\mathbf{x}$ is the nucleolus. □

Remark 2. *Step 2 in both Algorithms 1 and 2 still involves comparing vectors of exponential lengths. The key finding in Theorem 2, however, is to show that Step 2 of Algorithm 2 is not repeated more than $(n-1)$ times (instead of possibly exponential in the original Kohlberg criterion described in Algorithm 1). There are structured games such as weighted voting games, network flow games and coalitional skill games in which Step 2 can be executed efficiently. We refer the readers to [20] for details.*

We demonstrate the effectiveness of Algorithm 2 in Section 5. Before that, let us discuss how to resolve some other computationally demanding tasks of our algorithm.

3.2 Reducing the sizes of the tight sets

When checking the Kohlberg criterion we might end up having to store an exponentially large number of coalitions. The computational requirements of checking T_0-balancedness depend entirely on the size of the tight sets we encounter. Therefore, it is of particular interest to find compact representations of large tight sets. We provide a method for reducing the size of H_k to at most $(n-1)$. This is achieved by replacing tight sets with their compact representations.

Lemma 3. *The following statements hold:*

(a) The collection T is T_0-balanced if and only if there exists $\gamma \in \mathbb{R}_{\geq 0}^{|T_0|}, \omega \in \mathbb{R}_{>0}^{|T|}, \mu \in \mathbb{R}$ such that

$$\sum_{\mathcal{S} \in T_0} \gamma_{\mathcal{S}} \mathbf{e}(\mathcal{S}) + \sum_{\mathcal{S} \in T} \omega_{\mathcal{S}} \mathbf{e}(\mathcal{S}) + \mu \mathbf{e}(\mathcal{N}) = \mathbf{e}(\mathcal{N}). \tag{7}$$

(b) Suppose T contains a T_0-balanced subcollection Q. Then T is T_0-balanced if and only if there exists $\gamma \in \mathbb{R}_{\geq 0}^{|T_0|}, \omega \in \mathbb{R}_{>0}^{|T \setminus Q|}, \mu \in \mathbb{R}^{|Q|}$ such that

$$\sum_{\mathcal{S} \in T_0} \gamma_{\mathcal{S}} \mathbf{e}(\mathcal{S}) + \sum_{\mathcal{S} \in T \setminus Q} \omega_{\mathcal{S}} \mathbf{e}(\mathcal{S}) + \sum_{\mathcal{S} \in Q} \mu_{\mathcal{S}} \mathbf{e}(\mathcal{S}) = \mathbf{e}(\mathcal{N}). \tag{8}$$

The proof of Lemma 3 is provided in Appendix G of [22].

Lemma 3b allows us to represent each H_k by a collection R_k of size $\text{rank}(H_k) \leq n$ with the following updating procedure. We need to have $\text{span}(R_k) = \text{span}(H_{k-1} \cup T_k)$ in order to guarantee at most $(n-1)$ iterations. Therefore starting from $R_0 = H_0$, we get R_k by expanding R_{k-1} from a T_0-balanced T_k only

with coalitions that increase its rank. As a result, $\text{span}(R_k) = \text{span}(H_k)$, while $\text{rank}(R_k) = |R_k|$. We denote such a subset $R_k = rep(T_k; R_{k-1})$ and call R_k the representative of H_k.

As a result we can modify Algorithm 2 to be an Improved Kohlberg Algorithm *with compact representation* (denoted by $IKAcr$ in the numerical results of Section 5). In Step 3 we can set $R_k = rep(T_k; R_{k-1})$ instead of $H_k = H_{k-1} \cup T_k$ without changing balancedness whatsoever. This means we replace all tight sets T_k and store only a representative R_k of their union for the subsequent steps. Accordingly, as R_{k-1} is a collection of coalitions with full rank, the stopping criterion can be simplified to checking the cardinality of the representative set R_{k-1}. The correctness of the algorithm can be proven very similarly to Theorem 2 using Lemma 3b.

3.3 A fast algorithm for checking balancedness

According to the Kohlberg criterion, to check T_0-balancedness of T we need to check for the existence of $\gamma \in \mathbb{R}_{\geq 0}^{|T_0|}$ and $\omega \in \mathbb{R}_{>0}^{|T|}$ such that

$$\mathbf{e}(\mathcal{N}) = \sum_{\mathcal{S} \in T_0} \gamma_{\mathcal{S}} \mathbf{e}(\mathcal{S}) + \sum_{\mathcal{S} \in T} \omega_{\mathcal{S}} \mathbf{e}(\mathcal{S}).$$

Solymosi and Sziklai [24] [Lemma 3] provide an approach by solving $|T|$ linear programs as follows. For each $\mathcal{C} \in T$, let

$$q_{\mathcal{C}}^* = \left\{ \max \omega_{\mathcal{C}} \ : \ \sum_{\mathcal{S} \in T_0} \gamma_{\mathcal{S}} \mathbf{e}(\mathcal{S}) + \sum_{\mathcal{S} \in T} \omega_{\mathcal{S}} \mathbf{e}(\mathcal{S}) = \mathbf{e}(\mathcal{N}), \ (\gamma, \omega) \in \mathbb{R}_{\geq 0}^{|T_0|+|T|} \right\}.$$

Then T is T_0-balanced if and only if $q_{\mathcal{C}}^* > 0, \forall \mathcal{C} \in T$. Notice, however, that the collection T appearing in the Kohlberg criterion could be exponentially large, and hence solving all the $|T|$ linear programs is not practical for larger games. Solymosi [17] (see Routine 3.2) presents a faster approach that involves at most $\text{rank}(T)$ linear programs. We improve upon these results by exploiting the knowledge of a T_0-balanced *subcollection* in T to reduce the upper bound of $\text{rank}(T)$ in [17].

Exploiting Lemma 3, we can formulate an efficient algorithm that checks T_0-balancedness of a collection $T \subseteq 2^{\mathcal{N}}$ with a known T_0-balanced subcollection $Q \subsetneq T$ (possibly $Q = \emptyset$) by finding the largest balanced subcollection within T, as described in Algorithm 3 below.

When we check the T_0-balancedness of $(\cup_{j=1}^{k} T_j)$, through $(R_{k-1} \cup T_k)$ exploiting Lemma 3 and using Algorithm 3, $(R_{k-1} \cup T_k)$ and R_{k-1} play the role of T and Q respectively. In this case, when we initialize U as $\text{span}(Q) \cap T$, the set U essentially equals its representative set. However, this is not necessary the case any more when we perform the update in Step 4 of Algorithm 3. Moreover, Algorithm 3 can be

Algorithm 3: Algorithm finding largest T_0-balanced subcollection

Input: Collection T with T_0-balanced subcollection $Q \subsetneq T$;

Output: $U \subseteq T$ largest T_0-balanced subcollection;

1. Initialization: Set $U = \text{span}(Q) \cap T$;

while $\text{rank}(U) < \text{rank}(T)$ **do**

2. Find $\gamma^* \in \mathbb{R}_{\geq 0}^{|T_0|}, \omega^* \in \mathbb{R}_{\geq 0}^{|T \setminus U|}, \mu^* \in \mathbb{R}^{|U|}$ that solve

$$\underset{\gamma,\omega,\mu}{\text{argmax}} \left\{ \sum_{\mathcal{S} \in T \setminus U} \omega_{\mathcal{S}} : \sum_{\mathcal{S} \in T_0} \gamma_{\mathcal{S}} \mathbf{e}(\mathcal{S}) + \sum_{\mathcal{S} \in T \setminus U} \omega_{\mathcal{S}} \mathbf{e}(\mathcal{S}) + \sum_{\mathcal{S} \in U} \mu_{\mathcal{S}} \mathbf{e}(\mathcal{S}) = \mathbf{e}(\mathcal{N}) \right\} \tag{9}$$

if $\omega^* = \mathbf{0}$ *or* (9) *is infeasible* **then**

3. Stop the algorithm and output $U \subsetneq T$.;

else

4. Set $U = \text{span}(U \cup \{\mathcal{S} : \omega^*_{\mathcal{S}} > 0\}) \cap T$;

end

end

5. Output $U = T$.

used for general Q, not necessarily only those that are equal to their own representative set. Both cases can be easily treated by replacing U with its representative set in the corresponding occurrences (Steps 1 and/or 4 of Algorithm 3), not effecting balancedness and hence the outcome of the algorithm. In the following, we establish the improvement in the number of iterations required by our balancedness-checking subroutine, Algorithm 3.

Theorem 3. *Collection T is T_0-balanced if and only if Algorithm 3 terminates at Step 5 with $U = T$, and the algorithm terminates after at most* $(\text{rank}(T) - \text{rank}(Q))$ *iterations.*

Proof. The while loop terminates as $\text{rank}(U)$ keeps increasing via the construction of U in Steps 1 and 4; that is, the set U is enlarged by adding coalitions outside its span, starting from $\text{rank}(Q)$. Thus, the algorithm terminates at either Step 3 or 5 and we need to prove that the corresponding conclusions from the output U are correct. Also, notice that since $\text{span}(U) \cap T = U$, we have $U \subsetneq T$ if $\text{rank}(U) < \text{rank}(T)$[3].

If the algorithm terminates at Step 3, then $\omega^* = \mathbf{0}$ or (9) is infeasible and hence T is not T_0-balanced,

[3]Therefore we could replace the stopping condition $\text{rank}(U) = \text{rank}(T)$ with $U = T$ or $|U| = |T|$ as well.

as otherwise we should have found a feasible $\omega^* \neq \mathbf{0}$. If the algorithm terminates at Step 5 then, prior to that, we have $\text{rank}(U) = \text{rank}(T)$ in order for the while loop to terminate. The construction of U in Step 4 ensures that U is a T_0-balanced set by Lemmas 2b, 2c and 3b. Thus, $T = \text{span}(U) \cap T$ is also T_0-balanced by Lemma 2c. □

3.4 Nucleolus-defining coalitions and characterization sets

We conclude the first part of this article on the improved Kohlberg criterion by linking it with an important development in the nucleolus literature on the characterization set introduced by Granot et al. [23] and the $\mathcal{B}$-nucleolus by Reijnierse and Potters [25].

A cooperative game $G(\mathcal{N}, v)$ is represented by $(2^n - 1)$ coalitional values and although the nucleolus is defined as a function of all these values, i.e. lexicographical minimization of all the $(2^n - 2)$ excess values, Granot et al. [23] and Reijnierse and Potters [25] show that the nucleolus can be determined by a subset of coalitions in the sense that lexicographical minimization with those coalitions as admissible ones will determine the nucleolus. Reijnierse and Potters [25] show that there exists a characterization set in every game with a size of at most $2(n-1)$ coalitions. Although the authors emphasize that identifying this characterization set (or the $\mathcal{B}-$set) would be as hard as finding the nucleolus itself, the result is still quite striking since this essentially means that we can ignore $(2^n - 2(n-1))$ other coalitional values in calculating the nucleolus. The authors also show that the characterization set or the $\mathcal{B}$-nucleolus can be identified efficiently in a number of games, including the assignment games, the balanced matching games, standard tree games, etc. We first define the characterization set.

Definition 4. *For a collection of coalitions $\mathcal{F} \in 2^{\mathcal{N}}$, the $\mathcal{F}$-nucleolus of the game $G(\mathcal{N}, v)$, denoted as $\nu(\mathcal{N}, \mathcal{F}, v)$, consists of imputations that lexicographically minimizes the excess values of coalitions in $\mathcal{F}$. A set $\mathcal{F}$ is called a characterization set (or a $\mathcal{B}$-set) if $\nu(\mathcal{N}, \mathcal{F}, v) = \nu(\mathcal{N}, 2^{\mathcal{N}}, v) = \nu(\mathcal{N}, v)$.*

We now investigate how the improved Kohlberg criterion is linked to the concepts in [23, 25]. We prove that the set of coalitions generated from the improved Kohlberg criterion form 'special' characterization sets. We first identify the set of coalitions which are critical in defining the nucleolus.

Definition 5. *A coalition $\mathcal{S}$ is nucleolus-defining in game $G(\mathcal{N}, v)$ if a small perturbation on its coalitional value can lead to a change in the nucleolus. Formally, for all $\delta > 0$, there exists $|\epsilon| < \delta$ such that $\nu(\mathcal{N}, \tilde{v}) \neq \nu(\mathcal{N}, v)$, where $\tilde{v}(\mathcal{S}) = v(\mathcal{S}) + \epsilon$ and $\tilde{v}(\mathcal{S}') = v(\mathcal{S}')$ for all $\mathcal{N} \supset \mathcal{S}' \neq \mathcal{S}$. All remaining coalitions are called non-nucleolus-defining.*

Theorem 4. *The set of all nucleolus-defining coalitions is precisely* $\cup_{r=1}^{k} T_r$*, where* $T_r, r = 1, \ldots, k$ *are the collections of coalitions generated by the improved Kohlberg Algorithm 2 on the nucleolus* $\mathbf{x}$*.*

Proof. We prove two parts: (a) for all $j \leq k$, each $\mathcal{S} \in T_j$ is a nucleolus-defining coalition and (b) all the remaining ones are non-nucleolus-defining.

Let $\mathcal{S}_0 \in T_j$ for some $1 \leq j \leq k$. Suppose on contradiction that $\mathcal{S}_0$ is non-nucleolus-defining, i.e., there exists $\epsilon > 0$ and small enough such that if we change $v(\mathcal{S}_0)$ to $v(\mathcal{S}_0) + \epsilon$ the nucleolus of the new game is still $\mathbf{x}$. By setting $0 < \epsilon < \epsilon_{j-1} - \epsilon_j$[4] we have $\epsilon_j < v(\mathcal{S}_0) - \mathbf{x}(\mathcal{S}_0) < \epsilon_{j-1}$. Therefore the tight sets for $\mathbf{x}$ are $T_1, \ldots, T_{j-1}, \{\mathcal{S}_0\}, T_j \backslash \mathcal{S}_0, T_{j+1}, \ldots, T_k$. Here, note that both $\cup_{i=1}^{j-1} T_i$ and $\cup_{i=1}^{j-1} T_i \cup \mathcal{S}_0$ are balanced due to $\mathbf{x}$ being the nucleolus (according to the Kohlberg criterion). By Lemma 1, there exists $\boldsymbol{\alpha} > 0$ and $\boldsymbol{\beta} > 0$ such that $\boldsymbol{e}(\mathcal{N}) = \sum_{\mathcal{S} \in \cup_{i=1}^{j-1} T_i} \alpha_{\mathcal{S}} \boldsymbol{e}(\mathcal{S}) = \beta_{\mathcal{S}_0} \boldsymbol{e}(\mathcal{S}_0) + \sum_{\mathcal{S} \in \cup_{i=1}^{j-1} T_i} \beta_{\mathcal{S}} \boldsymbol{e}(\mathcal{S})$. Thus,

$$\beta_{\mathcal{S}_0} v(\mathcal{S}_0) = \sum_{\mathcal{S} \in \cup_{i=1}^{j-1} T_i} (\alpha_{\mathcal{S}} - \beta_{\mathcal{S}}) \boldsymbol{e}(\mathcal{S}),$$

that is $\mathcal{S}_0 \in span(\cup_{r=1}^{j-1} T_r)$, contradicting the construction $T_j \cap span(\cup_{r=1}^{j-1} T_r) = \emptyset$ in Algorithm 2. Part (a) of the theorem is proven.

Now let $\mathcal{S}_0 \notin \cup_{j=1}^{k} T_j$. We note, however, that $\mathcal{S}_0 \in span(\cup_{j=1}^{k} T_j)$ since $span(\cup_{j=1}^{k} T_j)$ has full rank. This means there exists a smallest index $r \in \{1, \ldots, k\}$ such that $\mathcal{S}_0 \notin \cup_{j=1}^{r} T_j$ while $\mathcal{S}_0 \in span(\cup_{j=1}^{r} T_j)$. This construction leads to $v(\mathcal{S}_0) - \mathbf{x}(\mathcal{S}_0) > \epsilon_r > \epsilon_j, \forall j < r$. Let us set $\delta = v(\mathcal{S}_0) - \mathbf{x}(\mathcal{S}_0) - \epsilon_r$. Then for any $|\epsilon| < \delta$, if we change $v(\mathcal{S}_0)$ to $v(\mathcal{S}_0) + \epsilon$ the nucleolus of the new game is still $\mathbf{x}$ because according to Algorithm 2, all the steps still lead to the same collection of coalitions $\cup_{j=1}^{k} T_j$. □

While all characterization sets lead to the same unique nucleolus, it can be more desirable if the subset of excess values generated from the restricted game can carry more information about the worst excess values in the original game. For example, consider a game with three players where $v(\{1,2,3\} = 9$, $v(\{1\}) = v(\{2\}) = v(\{3\}) = 0$ and $v(\{1,2\}) = v(\{2,3\}) = v(\{3,1\}) = 5$. It can be verified that both $\{\{1\},\{2\},\{3\}\}$ and $\{\{1,2\},\{2,3\},\{3,1\}\}$ form characterization sets. However, the former characterization set contains all non-nucleolus-defining coalitions while the latter contains all nucleolus-defining ones. It can be seen that the excess values generated from the latter provide more information on the most unhappy coalitions.

We define a *meaningful characterization set* as one that contains nucleolus-defining coalitions only. Following the result from Theorem 4, the next corollary provides us a method to construct these characterization sets.

[4]We require the second inequality only for $j > 1$.

Corollary 1. *A meaningful characterization set can be constructed as* $\cup_{i=1}^{k} F_i$*, where for each* $i = 1, \ldots, k$*,* F_i *is a 'representation' of* T_i*; that is,* $F_i \subset T_i$ *and* $rank(F_i) = rank(T_i)$*. The smallest size of meaningful characterization set is* $n + k - 1$ *which is constructed from minimals* $F_i, i = 1, \ldots, k$*, i.e., when* $rank(F_i) = |F_i| = rank(T_i)$*.*

Theorem 4 and Corollary 1 are related to the results in Granot et al. [23] and Reijnierse and Potters [25], however, we show exactly how some characterization sets are constructed. We skip the proof of Corollary 1 for brevity as it is quite straightforward based on the result of Theorem 4 and it shares analogies with the proof on the size of characterization sets in Reijnierse and Potters [25], which makes use of the nested LP sequence.

4 Lexicographical descent algorithm for finding the nucleolus

Our improved Kohlberg criterion allows us to formulate a constructive algorithm that not only verifies whether a given imputation is the nucleolus, but also gives means to find it, in case the given candidate is not the desired payoff. This new algorithm fits into a general iterative descent framework as follows:

- Starting from any imputation $\mathbf{x} \in \mathbf{I}$ we perform a (local) optimality test.
- If $\mathbf{x}$ fails the test, we generate an improving direction $\mathbf{y}$ and step size α (here, 'improving' is w.r.t. the lexicographical ordering of the corresponding dissatisfactions).
- We update $\mathbf{x} = \mathbf{x} + \alpha\mathbf{y}$ and repeat the procedure until no further improving direction is found.

In this scheme, the optimality test is derived from the new Kohlberg criterion developed in Section 3, improving directions are generated using duality, while step sizes are found exactly to guarantee *necessary* and *sufficient* change in the imputation and its tight collection of coalitions.

Our new algorithm also fits somewhat into the simplex framework for linear programming: improving directions are chosen using considerations similar to reduced costs, and the step size provides the pivoting rule through a sort of minimal ratio test. Indeed, we are moving on the facets of polytopes in Maschlers scheme, but not necessarily from vertex to vertex, like most traditional simplex implementations do.

4.1 Finding improving directions

Algorithm 3 not only handles the tedious strict positivity constraints related to balancedness, it essentially finds the largest T_0-balanced subcollection in T, starting from a previously identified (possibly

empty) balanced subcollection Q. Suppose that Algorithm 2 with compact representation (Algorithm 1 of [22]) terminates in Step 4, which happens precisely when Algorithm 3 exits with $\omega^* = 0$ or (9) is infeasible, while $\text{rank}(U) < \text{rank}(T)$. In the former case, we found the largest T_0-balanced subcollection U in T, but since $T \setminus U \neq \emptyset$, T is not T_0-balanced. In the latter case, there is no T_0-balanced subcollection in T (more precisely, the largest one is the empty set). In both cases we know that precisely the collection $T \setminus U \neq \emptyset$ is responsible for the lack of T_0-balancedness.

Recall that in iteration k of Algorithm 2 (with compact representation), when we check T_0-balancedness with Algorithm 3, input T is $(R_{k-1} \cup T_k)$ while the T_0-balanced subcollection Q is R_{k-1}, and we get the output U. For sake of simplicity we use T as $(R_{k-1} \cup T_k)$ and U as the corresponding output from Algorithm 3.

If T is not T_0-balanced, it is possible to generate an improving direction $\mathbf{y}$, such that moving from $\mathbf{x}$ to $(\mathbf{x} + \alpha\mathbf{y})$ will fulfill all of the following three objectives:

(a) not changing the excess of coalitions in $\text{span}(R_{k-1})$,

(b) remaining in the set of imputations and not increasing the excess of coalitions in U,

(c) decreasing the excess of coalitions in $T \setminus U$.

In other words, the change from $\mathbf{x}$ to $(\mathbf{x} + \alpha\mathbf{y})$ will increase the satisfaction of the most dissatisfied *unbalanced* coalitions, while maintaining the excess of the already settled *balanced* coalitions. In this subsection we focus on how to generate an improving direction while Subsection 4.2 is devoted to the calculation of the optimal step size.

When Algorithm 3 terminates with $\text{rank}(U) < \text{rank}(T)$ the system

$$\begin{aligned} \sum_{\mathcal{S} \in T_0} \gamma_{\mathcal{S}} \mathbf{e}(\mathcal{S}) + \sum_{\mathcal{S} \in T \setminus U} \omega_{\mathcal{S}} \mathbf{e}(\mathcal{S}) + \sum_{\mathcal{S} \in U} \mu_{\mathcal{S}} \mathbf{e}(\mathcal{S}) &= \mathbf{e}(\mathcal{N}) \\ \omega_{\mathcal{Q}} &> 0 \\ \gamma_{\mathcal{S}}, \omega_{\mathcal{P}} &\geq 0 \qquad \forall \mathcal{S} \in T_0, \mathcal{P} \in T \setminus U \\ \mu_{\mathcal{S}} &\in \mathbb{R} \qquad \forall \mathcal{S} \in U \end{aligned} \tag{10}$$

is infeasible for all $\mathcal{Q} \in T \setminus U$.Therefore, using Farkas' lemma we get

$$\{\mathbf{y} \in \mathbb{R}^n : \mathbf{y}(\mathcal{Q}) > 0, \mathbf{y}(\mathcal{P}) \geq 0, \forall \mathcal{P} \in T_0 \cup (T \setminus U), \mathbf{y}(\mathcal{S}) = 0, \forall \mathcal{S} \in U \cup \{\mathcal{N}\}\} \neq \emptyset.$$

Note that the preceding result holds for any $\mathcal{Q} \in T \setminus U$. While the corresponding $\mathbf{y}$ might differ for different $\mathcal{Q}$, we can take the average (or sum) of all these to arrive at a common, normalized $\mathbf{y}$ in

$$\{\mathbf{y} \in \mathbb{R}^n : \mathbf{y}(\mathcal{Q}) \geq 1, \forall \mathcal{Q} \in T \setminus U, \mathbf{y}(\mathcal{P}) \geq 0, \forall \mathcal{P} \in T_0, \mathbf{y}(\mathcal{S}) = 0, \forall \mathcal{S} \in U \cup \{\mathcal{N}\}\} \tag{11}$$

Furthermore, Lemma 3b shows that whenever we iteratively check whether a collection of coalitions $\cup_{j=1}^{k} T_j$ satisfies T_0-balancedness for all k or not, it is sufficient to require strict positivity from the weights of the current new set of coalitions T_k, if we already found that the collection is T_0-balanced up to level $(k-1)$. The lemma is not only useful to make checking of balancedness easier, as shown in Section 3.3, it also yields an improved system via (11). In iteration k, if T is not T_0-balanced, then in (11) we can require $\mathbf{y}(\mathcal{Q}) = 0$ from all coalitions $\mathcal{Q} \in \cup_{j=1}^{k-1} T_j \cup \{\mathcal{N}\}$ and still get a feasible system. Additionally, because for all $\mathcal{S} \in \cup_{j=1}^{k-1} T_j \cup \{\mathcal{N}\}$ there exists $\lambda \in \mathbb{R}^{|R_{k-1}|}$ such that $\mathbf{y}(\mathcal{S}) = \sum_{\mathcal{Q} \in R_{k-1}} \lambda_{\mathcal{Q}} \mathbf{y}(\mathcal{Q})$, the set

$$\{\mathbf{y} \in \mathbb{R}^n : \mathbf{y}(\mathcal{Q}) \geq 1, \forall \mathcal{Q} \in T_k \setminus U, \mathbf{y}(\mathcal{P}) \geq 0, \forall \mathcal{P} \in T_0, \mathbf{y}(\mathcal{S}) = 0, \forall \mathcal{S} \in R_{k-1} \cup (U \cap T_k)\} \tag{12}$$

is non-empty as well. We call vectors $\mathbf{y}$ in (12) *improving directions*. Since improving directions are defined through a feasible set of constraints, there could be many different improving directions, and we have the freedom to choose an objective function to optimize over that set. The following section determines the optimal step size, also shedding light on the most suitable objective function to choose.

4.2 Step size

A feasible point $\mathbf{y}$ in (12) is an improving direction in the sense that moving along $\mathbf{y}$ from our current point (which is not the nucleolus) improves the satisfaction of the coalitions that are currently worst off and causing the lack of balancedness, while still maintaining the satisfaction of previously checked balanced subcollections and ensuring that we stay in the imputation set for small enough step size. When determining a suitable step size $\alpha > 0$ for a given improving direction $\mathbf{y}$, we naturally want to choose α large enough in order to avoid small steps that do not result in changes in T, since T is not T_0-balanced. Also, we want to increase α only until we experience a change in T (or in T_0) in the hope that the new collection is T_0-balanced.

Suppose that, at iteration k, we are currently at imputation $\mathbf{x}$. For all coalitions $\mathcal{S}$, the change of excess as we move in direction $\mathbf{y}$ with step size α is

$$d(\mathcal{S}, \mathbf{x} + \alpha\mathbf{y}) - d(\mathcal{S}, \mathbf{x}) = v(\mathcal{S}) - (\mathbf{x}(\mathcal{S}) + \alpha\mathbf{y}(\mathcal{S})) - (v(\mathcal{S}) - \mathbf{x}(\mathcal{S})) = -\alpha\mathbf{y}(\mathcal{S}).$$

Currently the largest dissatisfaction among coalitions not in $\text{span}(R_{k-1})$ is $\epsilon_k(\mathbf{x}) = d(\mathcal{S}, \mathbf{x})$ for any $\mathcal{S} \in T_k(\mathbf{x})$. Thus, for sufficiently small $\alpha > 0$ the new maximal dissatisfaction is $\epsilon_k(\mathbf{x}+\alpha\mathbf{y}) = d(\mathcal{S}, \mathbf{x}+\alpha\mathbf{y})$ for some (possibly more than one) $\mathcal{S} \in T_k(\mathbf{x})$. Fix one such coalition as $\tilde{\mathcal{S}}$, then the change in the maximal dissatisfaction is $\epsilon_k(\mathbf{x} + \alpha\mathbf{y}) - \epsilon_k(\mathbf{x}) = -\alpha\mathbf{y}(\tilde{\mathcal{S}})$.

We are essentially interested in the *tightness* of coalitions measured as the difference of their excess from the maximal dissatisfaction, that is how far they are from being tight. Specifically, we are interested in the change of their tightness

$$(d(\mathcal{S}, \mathbf{x}+\alpha\mathbf{y}) - \epsilon_k(\mathbf{x}+\alpha\mathbf{y})) - (d(\mathcal{S}, \mathbf{x}) - \epsilon_k(\mathbf{x})) = \alpha(\mathbf{y}(\tilde{\mathcal{S}}) - \mathbf{y}(\mathcal{S})) \geq \alpha(1 - \mathbf{y}(\mathcal{S})), \tag{13}$$

with the last inequality due to $\mathbf{y}(\tilde{\mathcal{S}}) \geq 1$.

This brings us back to the practical question of how to choose improving directions from the cone determined by (12). Since every feasible point of that set is an improving direction we can use, we have the freedom to choose an objective function to optimize over this set. In order to control $\min_{\mathcal{S}\in T_k\setminus U} \mathbf{y}(\mathcal{S})$ as well as to make the bound we used in (13) sharp, we choose to minimize $\sum_{\mathcal{S}\in T_k\setminus U} \mathbf{y}(\mathcal{S})$.

Recall that when we check the T_0-balancedness of $\cup_{j=1}^{k} T_j$ in iteration k, we choose $\mathbf{y}$ solving

$$\begin{aligned} \min_{\mathbf{y}} \quad & \sum_{\mathcal{Q}\in T_k\setminus U} \mathbf{y}(\mathcal{Q}) & & \\ s.t. \quad & \mathbf{y}(\mathcal{Q}) & \geq 1 \quad & \forall \mathcal{Q} \in T_k \setminus U \\ & \mathbf{y}(\mathcal{P}) & \geq 0 \quad & \forall \mathcal{P} \in T_0 \\ & \mathbf{y}(\mathcal{S}) & = 0 \quad & \forall \mathcal{S} \in U \setminus T_k \end{aligned} \tag{$ID(T_0; T_k; U)$}$$

Thus, for every optimal solution $\mathbf{y}$ of $ID(T_0; T_k; U)$, we have $\mathbf{y}(\tilde{\mathcal{S}}) = 1$. As we increase α from 0, we see that the tightness of coalition $\mathcal{S}$ decreases if $\mathbf{y}(\mathcal{S}) > 1$, the tightness does not change if $\mathbf{y}(\mathcal{S}) = 1$, and it increases if $\mathbf{y}(\mathcal{S}) < 1$. By increasing tightness we mean that the difference $\epsilon_k(\mathbf{x}+\alpha\mathbf{y}) - d(\mathcal{S}, \mathbf{x}+\alpha\mathbf{y})$ decreases. Let us denote the collection of coalitions with increasing tightness as $\mathcal{J} = \{\mathcal{S} \notin \text{span}(R_{k-1}) \cup T_k : \mathbf{y}(\mathcal{S}) < 1\}$, the coalitions that are candidates to enter the tight set as we make a step.

We know that $d(\mathcal{S}, x) < \epsilon_k(\mathbf{x})$ for all coalitions $\mathcal{S} \notin \text{span}(R_{k-1}) \cup T_k$. Hence, there exists $\alpha > 0$ sufficiently small such that

$$d(\mathcal{S}, \mathbf{x}) - \alpha\mathbf{y}(\mathcal{S}) = d(\mathcal{S}, \mathbf{x}+\alpha\mathbf{y}) \leq \epsilon_k(\mathbf{x}+\alpha\mathbf{y}) = \epsilon_k(\mathbf{x}) - \alpha\mathbf{y}(\tilde{\mathcal{S}}) \leq \epsilon_k(\mathbf{x}) - \alpha.$$

Rearranging these terms, we get $d(\mathcal{S}, \mathbf{x}) + \alpha(1 - \mathbf{y}(\mathcal{S})) \leq \epsilon_k(\mathbf{x})$. Candidates of coalitions satisfying the latter relation with equality for large enough α are in collection $\mathcal{J}$, thus we increase α until we reach equality for some coalition in $\mathcal{J}$. However, we also need to bound α such that we stay in the imputation set. Taking both constraints into account, and introducing $\mathcal{N}_0 = \{j \in \mathcal{N} \setminus T_0 : y_j < 0\}$, the optimal step size is

$$\alpha = \min\left(\left\{\frac{\epsilon_k(\mathbf{x}) - d(\mathcal{S}, \mathbf{x})}{1 - \mathbf{y}(\mathcal{S})} : \mathcal{S} \in \mathcal{J}\right\} \cup \left\{\frac{x_j - v(\{j\})}{-y_j} : j \in \mathcal{N}_0\right\}\right), \tag{14}$$

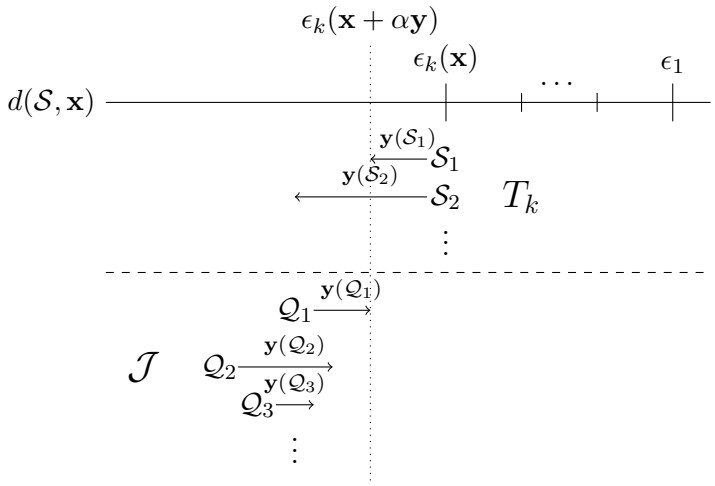

Figure 1: Optimal step size

the smallest step size for which we experience either $T_k(\mathbf{x}) \neq \underset{\mathcal{S}\notin \text{span}(R_{k-1})}{\text{argmax}} \{v(\mathcal{S}) - (\mathbf{x}+\alpha\mathbf{y})(\mathcal{S})\}$ or $T_0 \neq \{\{i\} : x_i + \alpha y_i = v(\{i\}), i \in \mathcal{N}\}$.

Figure 1 captures how the tight set changes as we move from $\mathbf{x}$ to $(\mathbf{x}+\alpha\mathbf{y})$. At $\mathbf{x}$, the largest dissatisfaction outside of the already settled span(R_{k-1}) belongs to coalitions in T_k. Their dissatisfactions decrease with varying rates, depending on $\mathbf{y}$, but with no smaller than 1. The new largest dissatisfaction $\epsilon_k(\mathbf{x}+\alpha\mathbf{y})$ is determined by coalitions in $\text{argmin}_{\mathcal{S}\in T_k\setminus U}\{\mathbf{y}(\mathcal{S})\}$.

In Figure 1, the dissatisfaction of coalitions in $\mathcal{J}$ increases (relative to the moving target of $\epsilon_k(\mathbf{x}+\alpha\mathbf{y})$), again with varying speed depending on $\mathbf{y}$. The coalition first meeting $\text{argmin}_{\mathcal{S}\in T_k\setminus U}\{\mathbf{y}(\mathcal{S})\}$ enters the tight set[5].

4.3 Lexicographical descent algorithm

Now that we have all necessary elements at our disposal, we formulate the new algorithm for calculating the nucleolus of a cooperative game.

Algorithm 4 starts with an arbitrarily chosen imputation. If, at the current point, the tight set T_k fails to pass a balancedness requirement related to the Kohlberg criterion, we generate an improving direction and a step size.

Beside the descent-based nature of the algorithm as presented in the preceeding sections, Algorithm 4 also shares some similarities with the simplex method for linear programming, as finding an improving direction $\mathbf{y}$ and a suitable step size α in Step 3 of the algorithm is similar to a pivot step in the simplex

[5]If there are multiple, all of them enter the tight set.

Algorithm 4: Algorithm computing the nucleolus of a cooperative game.

Input: Game $(\mathcal{N}, v)$ with $\mathbf{I} \neq \emptyset$;

Output: ν nucleolus of game $(\mathcal{N}, v)$;

1. Initialization: Set $\mathbf{x} \in \mathbf{I}$ arbitrary, $R_0 = \{\mathcal{N}\}$ and $k = 1$;

while $|R_{k-1}| < n$ **do**

2. Find $\epsilon_k(\mathbf{x}) = \max_{\mathcal{S} \notin \text{span}(R_{k-1})} d(\mathcal{S}, \mathbf{x})$, $T_k(\mathbf{x}) = \{\mathcal{S} \notin \text{span}(R_{k-1}) : d(\mathcal{S}, \mathbf{x}) = \epsilon_k\}$, $T_0(\mathbf{x}) = \{\{i\} : x_i = v(\{i\}), i \in \mathcal{N}\}$, and $U \subseteq (R_{k-1} \cup T_k)$ generated by Algorithm 3;

if $T_k \setminus U \neq \emptyset$ **then**

3. Find $\mathbf{y}$ solving $ID(T_0; T_k; U)$ and α using (14). Update $\mathbf{x} = \mathbf{x} + \alpha\mathbf{y}$ and go to Step **2**.;

else

4. Set $R_k = rep(T_k; R_{k-1})$, and $k = k + 1$;

end

end

5. $\mathbf{x} = \nu$ is the nucleolus.

algorithm. Inside the **while** loop the algorithm keeps 'pivoting' until T_0-balancedness is achieved, while the iterations of the loop correspond to solving LPs in the sequential LP formulation of the nucleolus (cf. [17]). The overall algorithm can also be interpreted as an active-set or column generation approach, because checking the balancedness of a collection of (primal) tight coalitions is nothing else than solving relaxed dual programs in the aforementioned LP sequence (cf. [19]).

Example 1. *Consider the 3-player game v with coalition values $v(\{1\}) = 1$, $v(\{2\}) = 2$, $v(\{3\}) = 5$, $v(\{1,2\}) = 6$, $v(\{1,3\}) = 7$, $v(\{2,3\}) = 8$, and $v(\mathcal{N}) = 12$. For readability, during this example we use superscripts to distinguish between different imputations, while subscripts of maximum dissatisfaction levels ϵ_k and tight sets T_k are used to keep track of iterations. We are using Algorithm 4 to find the nucleolus ν from a starting imputation $\mathbf{x}^0 = [1, 4, 7]$.*

First, we find that our distance to the boundary of the core *is 1, hence $\epsilon_1(\mathbf{x}^0) = 1$. Also, currently the largest infeasibility among core inequalities belongs to constraint $x_1 + x_2 \geq 6$. Therefore, $T_1(\mathbf{x}^0) = \{\{1,2\}\}$, while $T_0(\mathbf{x}^0) = \{\{1\}\}$ because $x_1^0 = v(\{1\})$. It is easy to see that the current tight set $T_1(\mathbf{x}^0)$ is not $T_0(\mathbf{x}^0)$-balanced. In the algorithm we run into an infeasible system when checking the balancedness, so we can find improving directions by solving $ID(T_1(\mathbf{x}^0); T_0(\mathbf{x}^0); \{\mathcal{N}\})$, for example $\mathbf{y} = [-1, 0, 1]$.*

Notice that we measure the distance from the boundary with a special signed distance*; in the interior*

of the core the distance from the boundary is understood to be negative. Thus, we can move even further along the direction after reaching the core, until we can not decrease *the distance any more. This happens precisely when the distances from both constraints* $(x_3 \geq 5)$ *and* $(x_1 + x_2 \geq 6)$ *are* -0.5*, that is when* $\mathbf{x}^1 = [2.5, 4, 5.5], \epsilon_1(\mathbf{x}^1) = -0.5, T_1(\mathbf{x}^1) = \{\{1,2\},\{3\}\}$ *and* $T_0(\mathbf{x}^1) = \emptyset$*. This is similar to a pivot step in a simplex-like algorithm, where coalition* $\{3\}$ *enters the basis. According to* (14)*, the step size chosen in direction* $\mathbf{y}$ *is* $\alpha = 1.5$*. After this step, we find that* $T_1(\mathbf{x}^1)$ *is* $T_0(\mathbf{x}^1)$*-balanced, therefore we expand* R_0 *with an arbitrary element of* $T_1(\mathbf{x}^1)$ *as* $\operatorname{rank}(R_0 \cup T_1(\mathbf{x}^1)) = 2$*. By doing so we lift those inequality constraints that limit the decrease; that is, throughout the remaining execution of the algorithm those constraints remain satisfied with equality at the current largest excess level of* $\epsilon_1(\mathbf{x}^1) = -0.5$*.*

The set of imputations having the coalitions in $T_1(\mathbf{x}^1)$ *being tight at* $\epsilon_1 = -0.5$ *actually form the least core of the game. At the current point* $\mathbf{x}^1$*, among constraints not in* $\operatorname{span}(R_1)$*, we are closest to violating* $x_1 + x_3 \geq 7$ *with* $\epsilon_2(\mathbf{x}^1) = -1$*. Also we have that* T_0 *remains empty at* $\mathbf{x}^1$ *and the tight set* $T_2(\mathbf{x}^1) = \{\{1,3\}\}$ *is not* T_0*-balanced, so we find an improving direction parallel to the set of least core payoffs. That is, the unique solution of* $ID(T_2(\mathbf{x}^1); T_0; R_1)$ *is* $\mathbf{y} = [1, -1, 0]$*, and we can take a step size of* $\alpha = 1/4$ *until coalition* $\{2,3\}$ *becomes tight as well. The resulting point is* $\mathbf{x}^2 = [2.75, 3.75, 5.5]^\top$ *with largest excess* $\epsilon_2(\mathbf{x}^2) = -1.25$*,* $T_0 = \emptyset$ *and tight set* $T_2(\mathbf{x}^2) = \{\{1,3\},\{2,3\}\}$*. Since* $T_1(\mathbf{x}^2) \cup T_2(\mathbf{x}^2)$ *is* T_0*-balanced and* $\operatorname{rank}(R_1 \cup T_2(\mathbf{x}^2)) = n$ *we found the nucleolus* $\nu = \mathbf{x}^2$*.*

In the followings we establish results regarding the convergence of Algorithm 4 and its connection to sequential LP methods. The results also justify why we call Algorithm 4 a lexicographical descent method.

Lemma 4. *Suppose that at iteration* k*, Algorithm 4 goes through to Step 3 and updates* $\mathbf{x}$ *to* $(\mathbf{x} + \alpha\mathbf{y})$*. Then we have* $\Theta(\mathbf{x} + \alpha\mathbf{y}) <_L \Theta(\mathbf{x})$*. Furthermore,*

(a) if $U \cap T_k = \emptyset$*, then* $\epsilon_k(\mathbf{x} + \alpha\mathbf{y}) < \epsilon_k(\mathbf{x})$*,*

(b) if $U \cap T_k \neq \emptyset$*, then* $\epsilon_k(\mathbf{x} + \alpha\mathbf{y}) = \epsilon_k(\mathbf{x}) = \epsilon_k(\nu)$ *and* $T_k(\mathbf{x} + \alpha\mathbf{y}) = T_k(\nu)$*.*

Proof. Let us start by noting that by the definition of $\mathbf{y}$ solving $ID(T_0; T_k; U)$ and α in (14), the new point $(\mathbf{x} + \alpha\mathbf{y}) \in \mathbf{I}$ if $\mathbf{x} \in \mathbf{I}$. Additionally $\mathbf{x}(\mathcal{S}) = (\mathbf{x} + \alpha\mathbf{y})(\mathcal{S})$ for all $\mathcal{S} \in \operatorname{span}(R_{k-1})$, also due to $\mathbf{y}$ solving $ID(T_0; T_k; U)$. As a result, both $\Theta(\mathbf{x})$ and $\Theta(\mathbf{x} + \alpha\mathbf{y})$ contain (and start with) the same excess values for coalitions $\mathcal{S}$ with excess $d(\mathcal{S}, \nu) \geq \epsilon_{k-1}(\nu)$. Therefore, in order to make the lexicographical comparison, it is sufficient to focus on the truncated ordered excess vectors over the set $2^{\mathcal{N}} \setminus \operatorname{span}(R_{k-1})$, i.e., between $\Theta^{2^{\mathcal{N}} \setminus \operatorname{span}(R_{k-1})}(\mathbf{x})$ and $\Theta^{2^{\mathcal{N}} \setminus \operatorname{span}(R_{k-1})}(\mathbf{x} + \alpha\mathbf{y})$.

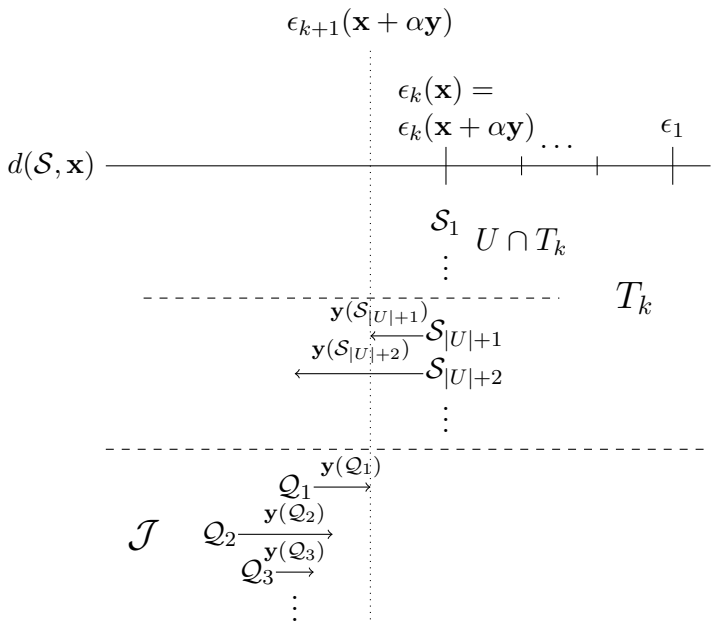

Figure 2: Changes of tight coalitions at a pivot step when $U \neq \emptyset$.

(a) The first component of both truncated ordered excess vectors is a value corresponding to a tight coalition $\mathcal{S} \in T_k(\mathbf{x})$. Since $\mathbf{y}(\mathcal{S}) \geq 1$ for all $\mathcal{S} \in T_k(\mathbf{x})$ and $\alpha > 0$, we have that $\epsilon_k(\mathbf{x}+\alpha\mathbf{y}) < \epsilon_k(\mathbf{x})$ (as Figure 1 demonstrates) and therefore $\Theta(\mathbf{x}+\alpha\mathbf{y}) <_L \Theta(\mathbf{x})$ in this case.

(b) However, if $U \cap T_k \neq \emptyset$, then we have $\mathbf{y}(\mathcal{Q}) = 0$ for all $\mathcal{Q} \in U$ since $\mathbf{y}$ is a solution of $ID(T_0; T_k; U)$, hence $\epsilon_k(\mathbf{x}) = \epsilon_k(\mathbf{x}+\alpha\mathbf{y}) = d(\mathcal{Q},\mathbf{x})$ for $\mathcal{Q} \in U$. Note that at Step 3 of Algorithm 4, we have $T_k \setminus U \neq \emptyset$. Since $\mathbf{y}(\mathcal{S}) \geq 1$ for all $\mathcal{S} \in T_k \setminus U$, we also have $U \cap T_k(\mathbf{x}) = T_k(\mathbf{x}+\alpha\mathbf{y}) \subsetneq T_k(\mathbf{x})$, as Figure 2 demonstrates. Consequently, $\Theta(\mathbf{x}+\alpha\mathbf{y}) <_L \Theta(\mathbf{x})$ because of $\Theta^{2^{\mathcal{N}}\setminus \mathrm{span}(R_{k-1})}(\mathbf{x}+\alpha\mathbf{y})$ starts with less excess values of $\epsilon_k(\mathbf{x})$ than $\Theta^{2^{\mathcal{N}}\setminus \mathrm{span}(R_{k-1})}(\mathbf{x})$; that is $|T_k(\mathbf{x}+\alpha\mathbf{y})| < |T_k(\mathbf{x})|$. On the other hand, since $T_k(\mathbf{x}+\alpha\mathbf{y})$ is T_0-balanced, we have $\epsilon_k(\mathbf{x}+\alpha\mathbf{y}) = \epsilon_k(\nu)$ and $T_k(\mathbf{x}+\alpha\mathbf{y}) = T_k(\nu)$.

□

Remark 3. *We have the following observations by Lemma 4:*

- *Lemma 4 justifies naming Algorithm 4* lexicographical descent. *Starting from an arbitrary imputation, we follow a trajectory of imputations by generating improving directions and step sizes, with the corresponding ordered vector of excesses keep* strictly *lexicographically decreasing in every step of the trajectory. The only other 'descent' method [19] that we are aware of for general cooperative games does not have this property.*

- *Due to the strict lexicographical descent, during Algorithm 4 we never circulate in the imputation set* $\mathbf{x} \in \mathbf{I}$.

- *Furthermore, Algorithm 4 is connected to the classical sequential LP methods by Lemma 4b: as soon as we have* $U \cap T_k \neq \emptyset$*, that is, we have found a balanced (sub)collection in the new tight set, we have solved the* k*-th LP of the sequence due to primal-dual feasibility. Notice, however, that even in that case we could make a further step in the lexicographical descent, since the original tight set was not balanced, allowing us to find the interior of the optimal facet of the LP.*

Theorem 5. *Algorithm 4 stops after a finite number of steps. The* ***while*** *loop is executed at most* $(n-1)$ *iterations before finding the nucleolus* ν*.*

Proof. We start by showing the iteration limit of the **while** loop. Note that $\operatorname{rank}(R_k)$ increases in every iteration as $\emptyset \neq T_k \subseteq 2^{\mathcal{N}} \setminus \operatorname{span}(R_{k-1})$, and since $\operatorname{rank}(R_0) = 1$ the algorithm terminates in at most $(n-1)$ iterations for the **while** loop.

Thus, for finite convergence to ν, we only need to show that, within a single iteration, a finite number of steps from $\mathbf{x}$ to $(\mathbf{x} + \alpha\mathbf{y})$ is sufficient to reach a T_0-balanced tight set, which gets the algorithm out of the current iteration. For that purpose let us fix that iteration to be k, and for notational ease, for the remainder of the proof we omit the iteration subscripts k; i.e. T is used in place of T_k, and so on.

According to Lemma 4, as soon as $U \cap T \neq \emptyset$, we reached a T_0-balanced tight set $U \cap T$ itself. Therefore we suppose that $U \cap T = \emptyset$.

Since $T \subseteq 2^{\mathcal{N}} \setminus \operatorname{span}(R)$, the possible tight sets we can encounter is finite, among which there exists T_0-balanced as well, for instance $T(\nu)$. Thus, the only way not to reach a T_0-balanced tight set is to encounter an infinite series of tight sets $\mathcal{T} = (T^1, T^2, \dots, T^m, T^1, T^{m+1} \dots)$, among which none is T_0-balanced. Here, we use the superscripts to denote the different steps that we encounter under the same iteration k. Again because of the finitely many tight sets, we guaranteed to revisit at least one tight set infinitely many times. W.l.o.g., let that one be T^1 and suppose that we first revisit it after taking m steps. Let us denote the corresponding improving directions, step sizes and maximal dissatisfactions we encounter at each tight set as $(\mathbf{y_1}, \mathbf{y_2}, \dots)$, $(\alpha_1, \alpha_2, \dots)$ and $(\epsilon_1, \epsilon_2, \dots, \epsilon_m, \epsilon_{m+1}, \dots)$, respectively.

Let us suppose that the starting tight set T^1 corresponds to the imputation $\mathbf{x}_1 = \mathbf{x}$. By Lemma 4 we have $\epsilon_1 > \epsilon_m > \epsilon_{m+1}$ and $\mathbf{x} + \sum_{j=1}^{m} \alpha_j \mathbf{y_j} \in \mathbf{I}$. Thus, we generate improving direction $\mathbf{y_1}$ solving the

LP

$$
\begin{aligned}
\min_{\mathbf{y}} \quad & \sum_{\mathcal{Q}\in T^1} \mathbf{y}(\mathcal{Q}) & & & \\
s.t. \quad & \mathbf{y}(\mathcal{Q}) & \geq 1, & \quad \forall \mathcal{Q} \in T^1, & (\widetilde{ID}(T^1)) \\
& \mathbf{y}(\mathcal{P}) & \geq 0, & \quad \forall \mathcal{P} \in T_0, & \\
& \mathbf{y}(\mathcal{S}) & = 0, & \quad \forall \mathcal{S} \in R. &
\end{aligned}
$$

Note that, here we assume $T^1 \cap U = \emptyset$ and hence we obtain a simpler version of $ID(T_0; T^1; U)$.

Since $\sum_{j=1}^{m} \alpha_j \mathbf{y_j}(\mathcal{S}) = \epsilon_1 - \epsilon_{m+1}$ is constant through $\mathcal{S} \in T^1$, $\beta \sum_{j=1}^{m} \alpha_j \mathbf{y_j}$ is feasible in $\widetilde{ID}(T^1)$ for large enough $\beta > 0$, thus we have $\frac{\sum_{j=1}^{m} \alpha_j \mathbf{y_j}}{\epsilon_1 - \epsilon_{m+1}}$ is optimal in $\widetilde{ID}(T^1)$. As a result, $\mathbf{y_1}(\mathcal{S}) = 1$ for all $\mathcal{S} \in T^1$ as $\mathbf{y_1}$ is also an optimal solution of $\widetilde{ID}(T^1)$. Consequently all coalitions remain tight, while some coalition must join, based on the definition of the step size, hence $T^1 \subsetneq T^2$.

Note that it is not always the case that T^1 is followed by the same T^2. However, since T^2 is pooled from the finite power set, there must be at least one set T^2 such that the subsequence (T^1, T^2) is repeated infinitely many times with $T^1 \subsetneq T^2$. Using the same line of arguments, we arrive at longer repeated subsequences $T^1 \subsetneq T^2 \ldots \subsetneq T^j$ for as large j as we wish. This is impossible because the size of T^j is bounded. Thus, the series of tight set $\mathcal{T}$ is finite under each iteration. As the number of iterations is bounded by $(n-1)$, the algorithm converges in finitely many steps. □

Remark 4. *We have the following observations by Theorem 5:*

- *During Algorithm 4, not only in the imputations, but we never circulate in the tight set space either.*
- *Note that requiring a finite number of steps is enough to achieve the result claimed above. Numerical experiments indicate that Algorithm 4 executes a very small number of steps, see Subsections 5.1 and 5.2.*

Remark 5. *The $(n-1)$ linear programs in Maschler's scheme [26], the $(n-1)$ iterations in the improved Kohlberg criterion for verifying the nucleolus (Algorithm 2), the upper bound of n iterations in the balancedness checking algorithm (Algorithm 3), and finally the $(n-1)$ iteration in our lexicographical descent algorithm as stated in Theorem 5 is due to the fact that all of these algorithms are designed to increase the rank of the collection of tight coalitions considered. While eventually in an implementation of Maschler's scheme (e.g. [18]) the same collection of tight coalitions is found as in our lexicographical descent method, the trajectory how the two different methods reaches this point is different, as evidenced by the different number of iterations required (cf. Section 5).*

Finally we note that Algorithm 4 can be easily adapted to the case when we have a characterization set $\mathcal{F} \subseteq 2^{\mathcal{N}}$ at our disposal, by simply changing the search space in $\epsilon_k(\mathbf{x})$ and $T_k(\mathbf{x})$ from $\mathcal{S} \notin \text{span}(R_{k-1})$ to $\mathcal{S} \notin \left(\text{span}(R_{k-1}) \cup \left(2^{\mathcal{N}} \setminus \mathcal{F}\right)\right)$. The computational bottleneck of Algorithm 4 is performing Step 2 and computing the step size in Step 3. However, given a characterization set $\mathcal{F}$ of polynomial size as an additional input, the algorithm runs in polynomial time. That characterization set is available, if we restrict our attention, for example to the class of assignment games ([2]), to balanced matching games ([25]), to standard tree games ([27]), among many other classes ([23]).

5 Numerical results

In the following subsections we present numerical results assessing the performance of the algorithms described above. Both the different versions of Kohlberg algorithms (Algorithms 1, 2 and 1 of [22]) and the constructive algorithm (Algorithm 4) have been tested on 4 different types of games, the player set size ranging from 5 to 30. For games with $n \leq 25$, fifty instances were generated from each type, and we report averages of computational time in seconds, number of iterations, pivot steps and subroutines (wherever applicable), as well as number of coalitions saved from storage by compact representation sets. Similarly, for games with $n > 25$, ten instances were generated from each type. In each category the corresponding minimal values are highlighted with bold (wherever applicable).

For the sake of completeness the Kohlberg algorithms are tested with 4 solution points including the nucleolus, a random imputation, a point in the least core and in the *least-least core* (an element of the least core with T_0-balanced $(T_1 \cup T_2)$). For brevity, we only present here results for the solution being the nucleolus. Results for the other three solutions are presented in Appendix H of [22]. The original and improved Kohlberg algorithms 1 and 2 are denoted with *Kohlberg* and *IKA* respectively, while Algorithm 1 of [22] that includes the compact representation is denoted with *IKAcr*. The lexicographical descent Algorithm 4 (denoted *BFN*) is compared to 4 methods: *SP* (*SD*) are the primal (dual) nested LP algorithms due to Solymosi [17], *DK* is Derks and Kuipers [19]'s algorithm, while *PRA* denotes the prolonged simplex algorithm by [18].

All algorithms were implemented in C++ and computations were carried out on a desktop PC with Intel Core i5-2500 3.30 GHz CPU and 16 Gb RAM[6]. All the LPs involved are solved with CPLEX

[6]In this configuration, the time-efficient implementation of the algorithms run out of memory at $n = 28$ while processing initialization, therefore we used a memory-efficient implementation instead for $n > 28$.

12.7.1's primal simplex method (with default settings[7]). Time limitations were set with 12 hours for $n \leq 25$, 15 hours for $n \leq 28$, and 18 hours for $n > 28$. All of the codes (along with the test instances) used to produce these results are available for free access at the GitHub repository [21].

5.1 Type I and II games

Type I and II games both appear in [18] and [28]. The characteristic function for type I is given by $v(\{i\}) = 0$ for all $i \in \mathcal{N}$, $v(\mathcal{N})$ is a random integer between $100(n-2)$ and $100n$, while $v(\mathcal{S})$ is a random integer between 1 and $100|\mathcal{S}|$ for all other (non-empty) coalitions $\mathcal{S}$. Type II games are generated as $v(\{i\}) = 0$ for all $i \in \mathcal{N}$ and $v(\mathcal{S})$ is a random integer between 1 and $50n$ for all other (non-empty) coalitions $\mathcal{S}$.

Table 1: Original and Improved Kohlberg algorithms on type I games and nucleolus solution

n	Time			Iterations			Subroutines			Repr.
	Kohlberg	IKA	IKAcr	Kohlberg	IKA	IKAcr	Kohlberg	IKA	IKAcr	IKAcr
5	0.014	0.0016	**0.0014**	25.6	**2.6**	**2.6**	43.3	3.3	**2.6**	2.6
10	2.5	**0.0034**	0.0042	964.1	**2.3**	**2.3**	1279.8	3	**2.3**	2.4
15	OoT	**0.068**	0.068	OoT	**1.9**	**1.9**	OoT	2.5	**1.9**	1.9
20	-	3.3	**3.2**	-	**1.7**	**1.7**	-	1.9	**1.7**	1.8
25	-	161	**159**	-	**1.9**	**1.9**	-	2.4	**1.9**	1.9
26	-	257	**250**	-	**1.6**	**1.6**	-	1.7	**1.6**	1.6
27	-	650	**631**	-	**1.9**	**1.9**	-	2.4	**1.9**	1.9
28	-	2827	**2591**	-	**1.9**	**1.9**	-	2.2	**1.9**	1.9
29	-	**2087**	2145	-	**1.7**	**1.7**	-	2	**1.7**	1.7
30	-	**3248**	3323	-	**1.4**	**1.4**	-	1.7	**1.4**	72842.4

In terms of the verifying Kohlberg algorithms we can start with the general observation that the classical Kohlberg algorithm is only usable up to a limited size. From Tables 1, 2, 5 and 6 we find that games of size $n = 15$ provide already a challenge that Algorithm 1 can not tackle, as it runs out of time.

For the remaining two algorithms *IKA* and *IKAcr*, the differences in performance do not seem to be significant. The advantage of having a compact representation only affects a small number of coalitions most of the time, with the notable exception of one type I game with 30 players. This is a random game with quite substantially larger tight set $T_1(\mathbf{x})$ than the other games considered. However, as both *IKA*

[7]In the case of *SP* and *SD*, the average number of pivots reflect the values of CPLEX parameter *iterations* reported in the output. In order to obtain realistic pivot numbers, preprocessing was turned off (which did not change the overall computation time significantly).

Table 2: Original and Improved Kohlberg algorithms on type II games and nucleolus solution

n	Time			Iterations			Subroutines			Repr.
	Kohlberg	IKA	IKAcr	Kohlberg	IKA	IKAcr	Kohlberg	IKA	IKAcr	IKAcr
5	0.014	**0.0016**	**0.0016**	26.1	2.8	**2.6**	48.3	3.9	**2.6**	1.6
10	6.9	**0.0033**	0.0036	951	2.6	**2.5**	2023.5	3.6	**2.6**	2.1
15	OoT	0.11	**0.11**	OoT	**2.5**	**2.5**	OoT	3.6	**2.6**	2.3
20	-	5.3	**5.2**	-	**2.5**	**2.5**	-	3.4	**2.5**	2.5
25	-	295	**289**	-	**3.6**	**3.6**	-	5.8	**3.6**	3.6
26	-	576	**558**	-	**3.1**	**3.1**	-	3.8	**3.1**	3.1
27	-	1061	**1041**	-	**3**	**3**	-	4.2	**3.1**	3.1
28	-	5672	**5271**	-	**4.8**	**4.8**	-	7.9	**4.8**	4.9
29	-	**7253**	7547	-	**4.4**	**4.4**	-	7.3	**4.4**	4.4
30	-	**12766**	13325	-	**3.8**	**3.8**	-	5	**3.8**	3.9

and *IKAcr* terminate after performing only 1 iteration, the advantage of a compact representation is not realised during any subsequent iterations. On the contrary, the small additional workload necessary for finding this compact representation appears to provide a slight disadvantage in computation time.

Table 3: Computing the nucleolus of type I games

n	Time					Iterations					Pivots					Subrout.	
	BFN	DK	SP	SD	PRA	BFN	DK	SP	SD	PRA	BFN	DK	SP	SD	PRA	BFN	SP
10	0.011	0.014	**0.009**	0.016	0.11	**2.02**	2.04	**2.02**	2.06	2.06	**11.8**	14.1	55.6	63.6	41	14.3	**3.1**
15	**0.1**	0.17	0.42	0.17	370	**1.5**	1.52	**1.5**	1.52	1.52	**19.2**	25.9	93.6	153	161	21	**2**
20	**4.4**	7.69	22.1	8.43	OoM	**1.46**	**1.46**	**1.46**	**1.46**	OoM	**28.9**	40.6	159	330	OoM	31.4	**1.9**
25	**227**	425	OoM	OoT	-	**1.78**	1.8	OoM	OoT	-	**42.1**	56.5	OoM	OoT	-	**47.6**	OoM
26	**463**	958	-	-	-	**1.6**	**1.6**	-	-	-	**40.6**	71.8	-	-	-	**43.3**	-
27	**1261**	2047	-	-	-	**1.9**	**1.9**	-	-	-	**57.1**	70.4	-	-	-	**69**	-
28	**4406**	6421	-	-	-	**1.9**	**1.9**	-	-	-	**48.5**	82.1	-	-	-	**53.7**	-
29	**12220**	17796	-	-	-	**1.7**	**1.7**	-	-	-	**58.1**	78	-	-	-	**66.4**	-
30	**19232**	OoT	-	-	-	**1.4**	OoT	-	-	-	**44.4**	OoT	-	-	-	**47.4**	-

Turning to constructive algorithms, we can start with a general observation similar to the one made on verification algorithms. Considering the results presented in Table 3 and 4, one finds that classical sequential LP formulations can not solve games with $n = 25$ players or more; while *SP* runs out of memory, *SD* does not finish within reasonable time restrictions. It is of little surprise, considering that these methods handle exponential sized (either in rows or columns) LPs. As impressive the prolonged simplex method by [18] is, it suffers more having exponential number of *both* rows and columns, thus

Table 4: Computing the nucleolus of type II games

n	Time					Iterations					Pivots					Subrout.	
	BFN	DK	SP	SD	PRA	BNF	DK	SP	SD	PRA	BFN	DK	SP	SD	PRA	BFN	SP
10	0.01	0.013	0.012	**0.007**	0.066	**2.5**	2.6	**2.5**	2.6	2.72	**11**	14.3	44.2	50.1	22.7	13.8	**4**
15	**0.13**	0.18	0.62	0.29	246	**2.5**	**2.5**	**2.5**	2.6	2.72	21.6	**21**	116	103	106	26.4	**4**
20	**5.06**	9.05	38	15	OoM	**2.5**	**2.5**	**2.5**	**2.5**	OoM	34.1	**34**	257	198	OoM	42.5	**4**
25	**294**	611	OoM	OoT	-	**3.6**	**3.6**	OoM	OoT	-	**53.2**	69	OoM	OoT	-	**69.6**	OoM
26	**524**	1155	-	-	-	**3.1**	**3.1**	-	-	-	**42.7**	65.6	-	-	-	**49**	-
27	**1167**	2322	-	-	-	**3**	3.1	-	-	-	**52.3**	69.1	-	-	-	**66.3**	-
28	**5222**	7846	-	-	-	**4.8**	**4.8**	-	-	-	**58.3**	61.7	-	-	-	**77.6**	-
29	**14624**	21429	-	-	-	**4.4**	**4.4**	-	-	-	**69.7**	78.3	-	-	-	**96.7**	-
30	**29593**	OoT	-	-	-	**3.8**	OoT	-	-	-	**68.1**	OoT	-	-	-	**90.7**	-

running out of memory already at 20 players. This is the case for all types of games considered, as Tables 7 and 8 also confirm.

Regarding the number of iterations needed, we see from Table 3 and 4 that type I and II games barely distinguish between primal (*BFN*, *SP*) and dual methods (*DK*, *SD*, *PRA*), the latter requiring at least as many iterations as the former, by nature. Even though the main advantage of primal methods, i. e. having a smaller number of iterations, is barely realised in these types of games, *BFN* still produces the best computing times, outperforming *DK* for every size of games, while the latter becomes unusable at $n = 30$. Furthermore, while *BFN* requires less pivots at the price of invoking subroutine Algorithm 3, this seems to be rarely rewarded with fewer number of iterations, at least for type I and II games.

5.2 Type III and IV games

Derks and Kuipers [19] were interested in games where the number of iterations grows more or less linearly with the number of players, and so they introduced type III games as $v(\mathcal{S}) = 0$ for all $|\mathcal{S}| < n-2$, $v(\mathcal{S}) = 1$ with probability 0.9 for $n - 2 \leq |\mathcal{S}| < n$ and $v(\mathcal{N}) = 1$. According to Table 7 the authors were obviously successful in terms of generating games where their (dual) method struggles, whereas for primal methods these games can be considered as *trivial*. As a result, it is no wonder that the computation times of *DK* are magnitudes higher compared to those of *BFN*.

In order to test the methods on games, which distinguish between the number of iterations required by primal and dual methods more realistically, that is games that are 'somewhere between types I-II and III', we introduce type IV games as $v(\{i\}) = 0$ for all $i \in \mathcal{N}$ and $v(\mathcal{S})$ is a random integer between 1 and n for all other (non-empty) coalitions $\mathcal{S}$.

Table 5: Original and Improved Kohlberg algorithms on type III games and nucleolus solution

n	Time			Iterations			Subroutines			Repr.
	Kohlberg	IKA	IKAcr	Kohlberg	IKA	IKAcr	Kohlberg	IKA	IKAcr	IKAcr
5	0.0021	**0.0006**	0.0011	5.2	**1**	**1**	6.1	**2.0**	**2.0**	5.0
10	0.43	**0.0012**	0.0028	10.7	**1**	**1**	12.7	**2.6**	2.9	30.2
15	OoT	0.026	**0.026**	OoT	**1**	**1**	OoT	**2.2**	2.5	80.9
20	-	**1.00**	1.0	-	**1**	**1**	-	**2**	2.7	152.2
25	-	**38.7**	39.7	-	**1**	**1**	-	**2.1**	2.9	245.2
26	-	81.8	**81.6**	-	**1**	**1**	-	**2.2**	2.6	270.3
27	-	**168**	170	-	**1**	**1**	-	**2.3**	2.7	291.5
28	-	1092	**1076**	-	**1**	**1**	-	**2.2**	2.6	315.7
29	-	**217**	226	-	**1**	**1**	-	**2.2**	2.7	336.0
30	-	**447**	449	-	**1**	**1**	-	**2.2**	2.8	358.5

Table 6: Original and Improved Kohlberg algorithms on type IV games and nucleolus solution

n	Time			Iterations			Subroutines			Repr.
	Kohlberg	IKA	IKAcr	Kohlberg	IKA	IKAcr	Kohlberg	IKA	IKAcr	IKAcr
5	0.010	**0.0009**	0.0009	14.2	**1.6**	**1.6**	24.5	2.6	**1.8**	2.8
10	2.2	**0.0027**	0.0028	285	**2**	**2**	608	3.4	**2.4**	3.7
15	OoT	**0.079**	0.079	OoT	**2**	**2**	OoT	3.6	**2.8**	9.4
20	-	3.7	**3.6**	-	**2.1**	**2.1**	-	4.3	**4**	18.6
25	-	**220**	225	-	**2.9**	**2.9**	-	6.1	**3.9**	15.6
26	-	590	**573**	-	**3.4**	**3.4**	-	6.2	**5**	16.1
27	-	1305	**1269**	-	**3.6**	**3.6**	-	7.1	**4.7**	18.1
28	-	4576	**4344**	-	**3.7**	**3.7**	-	7.4	**5.1**	18.2
29	-	**5464**	5658	-	**3.5**	**3.5**	-	7.2	**5.4**	25.2
30	-	**9647**	9842	-	**2.8**	**2.8**	-	5.9	**4.4**	29.1

Both *IKA* and *IKAcr* solve type III games extremely easily, making them hard to compare with each other[8]. Their performance for type IV games show a similar behaviour as games of types I and II.

Tables 7 and 8 show that as soon as the required number of iterations at least moderately distinguishes between primal and dual methods, the difference in computational time between *BFN* and *DK* greatly increases.

[8]The non-monotonicity in computation time occurring between 28- and 29-player games are due to the two types of implementations, a so-called time-efficient and memory-efficient version. The time-efficient implementation is actually *not efficient in terms of computational time*, as it wastes time at initialization compared to the memory-efficient version.

Table 7: Computing the nucleolus of type III games

n	Time					Iterations					Pivots					Subrout.	
	BFN	DK	SP	SD	PRA	BFN	DK	SP	SD	PRA	BFN	DK	SP	SD	PRA	BFN	SP
10	**0.001**	0.051	0.003	0.007	0.099	**1**	5.8	**1**	2.9	5.72	**0**	52.1	15.5	55.7	28.5	3.1	**2.6**
15	**0.007**	0.73	0.18	0.25	164	**1**	6.7	**1**	2.6	6.22	**0**	110	23.5	102	61.4	3.9	**2.4**
20	**0.18**	47.6	8.44	15.2	OoM	**1**	11	**1**	3	OoM	**0**	234	33.8	933	OoM	3.8	**2.8**
25	**6.48**	2571	OoM	OoT	-	**1**	11.6	OoM	OoT	-	**0**	348	OoM	OoT	-	**4.7**	OoM
26	**13.6**	5705	-	-	-	**1**	13.5	-	-	-	**0**	385	-	-	-	**5.1**	-
27	**29.1**	12208	-	-	-	**1**	12.6	-	-	-	**0**	408	-	-	-	**5.2**	-
28	**729**	29867	-	-	-	**1**	14.7	-	-	-	**0**	471	-	-	-	**5.4**	-
29	**120**	OoT	-	-	-	**1**	OoT	-	-	-	**0**	OoT	-	-	-	**6**	-
30	**239**	-	-	-	-	**1**	-	-	-	-	**0**	-	-	-	-	**6**	-

Table 8: Computing the nucleolus of type IV games

n	Time					Iterations					Pivots					Subrout.	
	BFN	DK	SP	SD	PRA	BFN	DK	SP	SD	PRA	BFN	DK	SP	SD	PRA	BFN	SP
10	**0.007**	0.014	0.008	0.007	0.054	**2**	2.5	**2**	2.5	2.62	**6.6**	15.1	33.5	42.4	18.4	9.3	**3.3**
15	**0.05**	0.2	0.36	0.27	122	**2**	2.9	**2**	2.5	2.72	**8.9**	27.4	71.1	95.7	106	12.1	**3.6**
20	**2.17**	11.4	20.7	19.1	OoM	**2.1**	4.2	**2.1**	3.4	OoM	**10.8**	44.4	130	223	OoM	15.2	**4.5**
25	**102**	563	OoM	OoT	-	**2.9**	4.2	OoM	OoT	-	**16.8**	69.3	OoM	OoT	-	**23.1**	OoM
26	**188**	1375	-	-	-	**3.4**	5.2	-	-	-	**16.4**	70.5	-	-	-	**22.6**	-
27	**486**	3024	-	-	-	**3.6**	5.4	-	-	-	**19.7**	78.3	-	-	-	**31**	-
28	**3128**	9648	-	-	-	**3.7**	6	-	-	-	**19.3**	106	-	-	-	**26.3**	-
29	**4665**	22760	-	-	-	**3.5**	6.1	-	-	-	**21.3**	89	-	-	-	**32.1**	-
30	**8550**	OoT	-	-	-	**2.8**	OoT	-	-	-	**18.9**	OoT	-	-	-	**25.4**	-

5.3 Limitations of our algorithm

We now study the bottleneck of the lexicographical descent algorithm in attempt to find games that our proposed method struggles with. The performance of Algorithm 3 as the balancedness subroutine of Algorithm 4 depends on the size of the tight set, so we now look for games with extremely large tight sets. From a verification point of view we expect that the compact representation of tight sets carries an improvement in these games, therefore also providing significant distinguishment between Algorithms 2 and 1 of [22].

For our purposes we adopt the United Nations (UN) Security Council voting mechanism into weighted voting games with arbitrary size, where there are 5 *big* (veto) players and the rest (originally 10) are *small*. Formal description and results for the verification algorithms can be found in

Appendix H.3 of [22].

Our attempts to find a game our method struggles with were somewhat successful, meaning that while we have a sizeable advantage in computation time over other algorithms for $n \leq 26$, this advantage vanishes at $n = 27$, until eventually *BFN* runs out of memory for $n = 28$. This is due to the fact that these games have extremely large tight sets, which severely affects *BFN* through Algorithm 3 with a very large $|T|$ of exponential size, while by the nature of [19]'s method this does not affect *DK*.

It should be noted that finding the nucleolus of these games is trivial, i.e. one can easily find analytically that the 5 veto players share the total payoff of 1 amongst themselves in an egalitarian way, while all the small players get 0. Therefore anyone interested in finding the nucleolus of such a game would never turn to any of the aforementioned algorithms. Instead, since these games are of a very peculiar nature from an algorithmic perspective, they carry a *theoretical interest* from a computational point of view. For games with structures like this, we expect further improvement by exploiting the structure in a similar way to [20]. However, within the scope of this paper, we want to provide a like-for-like comparison and hence leave further improvements for future research.

5.4 Comparing Kohlberg algorithms on different solutions

Finally, we consider further numerical tests of the various Kohlberg algorithms (Algorithms 1, 2 and 1 of [22]) that verify whether a particular solution of a game is the nucleolus or not. We test these algorithms on four kinds of solutions: a random imputation, an element each of the least core and the least-least core (i.e. $T_1 \cup T_2$ is T_0-balanced), and the nucleolus. Results for the latter were presented above in Sections 5.1-5.2, while we cover the former three in Appendix H of [22].

Naturally, our expectations are that random imputations are probably in no relation with the nucleolus, therefore should be rejected straight away, while as we 'go deeper' into the least core, more effort is needed to reject solutions that are not the nucleoli themselves.

As a general observation, our first expectation is met, regardless of the type of the game. Tables 1, 2, 7, 8 and 11 of [22] show that all of the algorithms reject random solutions without any significant effort (and therefore we omit these cases from further analysis). Our other expectations seem to be met as well, as we clearly notice increases in time, iterations and subroutine calls when moving towards more involved solutions.

Another observation is that the original Kohlberg algorithm is again not able to solve instances with more than 10 players as soon as we consider a solution from the least core, or more than 15 players and the least-least core in case of the UN Security Council game cf. Tables 12-13 of [22]. Thus, as before,

algorithms *IKA* and *IKAcr* provide the only option for most games and the solutions to be verified. Hence, in our further analysis we again restrict ourselves to comparing these two algorithms, with the details provided in Appendix H of [22].

6 Conclusion

In this paper, we present both an Improved algorithmic approach for verifying whether a payoff vector is the nucleolus and a novel constructive method for finding it. In the first part, we develop an Improved Kohlberg criterion in which the number of iterations is bounded by at most $(n-1)$ instead of possibly exponentially large in the original Kohlberg criterion. This also comes with introducing representative sets for more efficient storage of the coalitions and a faster algorithm for checking balancedness. In the second part, we develop a novel descent-based algorithm for computing the nucleolus that exploits the new and Improved Kohlberg criterion. We compare the performance of our new algorithms with existing methods and demonstrate their effectiveness through numerical testing with a number of games proposed in the literature. Finally, we provide our algorithms, as well as the relevant literature's in an online open-source code repository, which we believe is an important step forward, that the cooperative game theory community can build upon.

Acknowledgement

The authors would like to thank the Editor and the two anonymous reviewers for their valuable comments and detailed suggestions on how to improve the manuscript. The first author acknowledges that the research reported in this paper has been supported by the National Research, Development and Innovation Fund (TUDFO/51757/2019-ITM, Thematic Excellence Program). The third author acknowledges the funding support from the Engineering and Physical Sciences Research Council (grant EP/P021042/1).

References

[1] D. Schmeidler. The nucleolus of a characteristic function game. *SIAM Journal of Applied Mathematics*, 17(6):1163–1170, 1969. URL http://dx.doi.org/10.1137/0117107.

[2] T. Solymosi and T.E.S. Raghavan. An algorithm for finding the nucleolus of assignment games. *International Journal of Game Theory*, 23(2):119–143, 1994.

[3] H. Hamers, F. Klijn, T. Solymosi, S. Tijs, and D. Vermeulen. On the nucleolus of neighbor games. *European Journal of Operational Research*, 146(1):1–18, 2003.

[4] T. Solymosi, T. Raghavan, and S. Tijs. Computing the nucleolus of cyclic permutation games. *European journal of operational research*, 162(1):270–280, 2005.

[5] J. Potters, H. Reijnierse, and A. Biswas. The nucleolus of balanced simple flow networks. *Games and Economic Behavior*, 54(1):205–225, 2006.

[6] X. Deng, Q. Fang, and X. Sun. Finding nucleolus of flow game. *Journal of combinatorial optimization*, 18(1):64–86, 2009.

[7] W. Kern and D. Paulusma. On the core and f-nucleolus of flow games. *Mathematics of Operations Research*, 34(4):981–991, 2009.

[8] E. Elkind, L.A. Goldberg, P. Goldberg, and M. Wooldridge. Computational complexity of weighted threshold games. In *Proceeding of the National Conference On Artificial Intelligence*, volume 22, page 718, 2007.

[9] E. Kohlberg. On the nucleolus of a characteristic function game. *SIAM Journal on Applied Mathematics*, 1(20):62–66, 1971.

[10] A. Kopelowitz. Computation of the kernels of simple games and the nucleolus of n-person games. Technical report, DTIC Document, 1967.

[11] K. Kido. A modified kohlberg criterion and a nonlinear method to compute the nucleolus of a cooperative game. *Taiwanese Journal of Mathematics*, pages 1581–1590, 2008.

[12] E. Kohlberg. The nucleolus as a solution of a minimization problem. *SIAM Journal on Applied Mathematics*, 23(1):34–39, 1972.

[13] G. Owen. A note on the nucleolus. *International Journal of Game Theory*, 3(2):101–103, 1974.

[14] J. Puerto and F. Perea. Finding the nucleolus of any n-person cooperative game by a single linear program. *Computers and Operations Research*, 40(10):2308–2313, 2013.

[15] M. Maschler, B. Peleg, and L.S. Shapley. Geometric properties of the kernel, nucleolus, and related solution concepts. *Mathematics of operations research*, 4(4):303–338, 1979.

[16] J.K. Sankaran. On finding the nucleolus of an n-person cooperative game. *International Journal of Game Theory*, 19(4):329–338, 1991.

[17] T. Solymosi. On computing the nucleolus of cooperative games. *Ph.D. Thesis, University of Illinois*, 1993. doi: 10.13140/RG.2.2.28952.80642.

[18] J.A.M. Potters, J.H. Reijnierse, and M. Ansing. Computing the nucleolus by solving a prolonged simplex algorithm. *Mathematics of operations research*, 21(3):757–768, 1996.

[19] J. Derks and J. Kuipers. Implementing the simplex method for computing the prenucleolus of transferable utility games. 1997.

[20] T. Nguyen and L. Thomas. Finding the nucleoli of large cooperative games. *European Journal of Operational Research*, 3(248):1078–1092, 2016.

[21] M. Benedek. Nucleolus. https://github.com/blrzsvrzs/nucleolus, 2018.

[22] M. Benedek, J. Fliege, and T.D. Nguyen. Finding and verifying the nucleolus of cooperative games – technical report, 2019.

[23] Daniel Granot, Frieda Granot, and Weiping R Zhu. Characterization sets for the nucleolus. *International Journal of Game Theory*, 27(3):359–374, 1998.

[24] T. Solymosi and B. Sziklai. Characterization sets for the nucleolus in balanced games. *Operations Research Letters*, 44(4):520–524, 2016.

[25] Hans Reijnierse and Jos Potters. The b-nucleolus of tu-games. *Games and Economic Behavior*, 24 (1-2):77–96, 1998.

[26] M. Maschler, J.A.M. Potters, and S.H. Tijs. The general nucleolus and the reduced game property. *International Journal of Game Theory*, 21(1):85–106, 1992.

[27] Daniel Granot, Michael Maschler, Guillermo Owen, and Weiping R Zhu. The kernel/nucleolus of a standard tree game. *International Journal of Game Theory*, 25(2):219–244, 1996.

[28] J.H. Reijnierse. Games, graphs and algorithms. *Ph.D. Thesis, Nijmegen*, 1995.