The Complexity of the Nucleolus in Compact Games

Gianluigi Greco

Dipartimento di Matematica e Informatica, University of Calabria, Italy
ggreco@mat.unical.it

Enrico Malizia*, Luigi Palopoli, and Francesco Scarcello

DIMES, University of Calabria, Italy
{emalizia,palopoli,scarcello}@dimes.unical.it

Abstract

The nucleolus is a well-known solution concept for coalitional games to fairly distribute the total available worth among the players. The nucleolus is known to be **NP**-hard to compute over *compact coalitional games*, that is, over games whose functions specifying the worth associated with each coalition are encoded in terms of polynomially computable functions over combinatorial structures. In particular, hardness results have been exhibited over minimum spanning tree games, threshold games, and flow games. However, due to its intricate definition involving reasoning over exponentially many coalitions, a non-trivial upper bound on its complexity was missing in the literature and looked for.

The paper faces this question and precisely characterizes the complexity of the nucleolus, by exhibiting an upper bound that holds on *any* class of compact games, and by showing that this bound is tight even on the (structurally simple) class of *graph games*. The upper bound is established by proposing a variant of the standard linear-programming based algorithm for nucleolus computation and by studying a framework for reasoning about succinctly specified linear programs, which are contributions of interest in their own. The hardness result is based on an elaborate combinatorial reduction, which is conceptually relevant for it provides a "measure" of the computational cost to be paid for guaranteeing voluntary participation to the distribution process. In fact, the *pre*-nucleolus is known to be efficiently computable over graph games, with this solution concept being defined as the nucleolus but without guaranteeing that each player is granted with it at least the worth she can get alone, i.e., without collaborating with the other players.

Finally, the paper identifies relevant tractable classes of coalitional games, based on the notion of *type* of a player. Indeed, in most applications where many players are involved, it is often the case that such players do belong in fact to a limited number of classes, which is known in advance and may be exploited for computing the nucleolus in a fast way.

1 Introduction

1.1 Compact Coalitional Games and Solution Concepts

Coalitional games were introduced by von Neumann and Morgenstern [67] as tools to reason about scenarios where players can collaborate by forming coalitions with the aim of obtaining higher worths than by acting in isolation. Formally, a coalitional game $\mathcal{G}$ is a pair $\langle N, v\rangle$, where N is a finite set of players, and v is a function associating with each non-empty set of players $S \subseteq N$, called *coalition*, the worth $v(S) \in \mathbb{R}$ that players in S can obtain by collaborating with each other. The outcome of $\mathcal{G}$ is an *imputation* x, i.e., a vector of payoffs $(x_i)_{i\in N}$ meant to specify the distribution among players in N of the total worth $v(N)$. An imputation x is required to be *efficient*, i.e., $\sum_{i\in N} x_i = v(N)$ (so that the whole available worth is distributed), and *individually rational*, i.e., $x_i \geq v(\{i\})$, for each $i \in N$ (so that each player voluntarily opts for participating to the game). The set of all imputations of $\mathcal{G}$ will be hereinafter denoted by $X(\mathcal{G})$.

 The definitive version was published in ACM Transactions on Computation Theory, {Vol. 7, Iss. 1, (Dec. 2014)} DOI: 10.1145/2692372.2692374

*Part of Enrico Malizia's work was supported by the European Commission through the European Social Fund and by Calabria Region. Part of his work was carried out while visiting the Department of Computer Science of the University of Oxford, UK.

Example 1.1. Consider the production processes of three factories: p, s, and g.

Factory p produces ceramic pipes, with an annual profit of 30M dollars. The industrial process here yields small ceramic cylinders as the production waste resulting from cutting rough long ceramic pipes to market established sizes.

Factory s produces house objects made from large pieces of synthetic sponges, with an annual profit of 10M dollars. In this case, production waste comes in the form of small pieces of synthetic sponge.

Moreover, production wastes of p and s might be used to produce mechanical filters for liquids to be used, for instance, to filter small fish tank's water. In fact, if the two factories decided to cooperate to open a common production line, then their synergy would yield an additional profit of 20M dollars per year.

Factory g produces instead sun glasses, with a profit of 10M dollars per year. In this case, for g to cooperate with p and/or s does not produce any economic advantage.

The situation above is easily described using a coalitional game $\mathcal{G}_0 = \langle \{p, s, g\}, v \rangle$, whose players are the three factories p, s, and g, and where the worth function is such that: $v(\{p\}) = 30$, $v(\{s\}) = 10$, $v(\{g\}) = 10$, $v(\{p, s\}) = v(\{p\}) + v(\{s\}) + 20 = 60$, $v(\{p, g\}) = v(\{p\}) + v(\{g\}) = 30 + 10 = 40$, $v(\{s, g\}) = v(\{s\}) + v(\{g\}) = 10 + 10 = 20$, $v(\{p, s, g\}) = v(\{p, s\}) + v(\{g\}) = 60 + 10 = 70$.

An imputation of $\mathcal{G}_0$ is any vector x of payoffs such that: $x_p + x_s + x_g = 70$ (efficiency); and $x_p \geq 30$, $x_s \geq 10$, $x_g \geq 10$ (individual rationality). That is, an imputation encodes a way to distribute to total available worth $v(\{p, s, g\}) = 70$ among the factories, in a way that each factory gets at least the worth it can get without collaborating with the two other ones. Hence, in this example, x encodes a way to distribute the surplus of 20M, coming from the cooperation (of p and s). ◁

In many real-world applications, computing just an arbitrary imputation is often not careful enough for the worth distribution purposes. In fact, a fundamental problem for coalitional games is to single out the most desirable elements of $X(\mathcal{G})$ in terms of appropriate notions of *fairness* and *stability* of worth distributions, which are usually called *solution concepts* (see, e.g., [46]). As an example, the *core* of a game $\mathcal{G} = \langle N, v \rangle$ is a well-known solution concept singling out the set of all imputations $x \in X(\mathcal{G})$ that are "stable" because there is no coalition $S \subset N$ whose members can receive a higher payoff than in x by leaving the grand coalition (which is the set of all players in the game) [22]. Formally, $x \in X(\mathcal{G})$ is in the core of the game $\mathcal{G}$ if there is no coalition $S \subseteq N$ and vector $(y_i)_{i \in S}$ such that $\sum_{i \in S} y_i = v(S)$ and $y_i > x_i$, for all $i \in S$. Equivalently, an imputation x is in the core if $\sum_{i \in S} x_i \geq v(S)$ holds, for each coalition $S \subseteq N$ (see, e.g., [46]).

Example 1.2. In the coalitional game $\mathcal{G}_0$ defined in Example 1.1, consider a payoff vector x' such that $x'_p = 30$, $x'_s = 10$, and $x'_g = 30$. Basically, the surplus of 20M is entirely provided to g, which however does not play any role in the cooperation between p and s. Hence, while being an imputation in $X(\mathcal{G}_0)$, the vector x' does not appear to be an appropriate outcome of the game. Indeed, this is formalized by the fact that x' does not belong to the core, because there exist a coalition $\{p, s\}$ and a vector y, with $y_p = 40$ and $y_s = 20$, such that $y_p + y_s = v(\{p, s\})$, $y_p > x'_p$, and $y_s > x'_s$. In words, x' is not stable, because both p and s are better off by excluding g from the collaboration.

On the other hand, consider the imputation x'' such that $x''_p = 50$, $x''_s = 10$, and $x''_g = 10$, and notice that x'' belongs to the core of $\mathcal{G}_0$. ◁

It is easily seen that, in many cases, the stability notion embodied by the core is not sufficient (alone) to define an appropriate method for worth distribution. Consider, for instance, the imputation x'' in the above example. According to x'', the surplus of 20M dollars (gained via the collaboration between p and s) is entirely provided to p. Hence, while this worth distribution is "stable" according to the core, it is hardly acceptable by s, because it completely ignores its contribution to the surplus. In words, s may perceive that the imputation x'' is not *fair*. In fact, addressing fairness issues in the worth distribution process is crucial problem in coalitional game theory, which motivated the definition of specialized solution concepts, ideally coming as a refinement of the core.

The focus of this paper is on the *nucleolus*, which is a solution concept formalized by Schmeidler [56] based on the idea that a fair distribution of the total available worth should lexicographically minimize the sorted vector of the *excesses* associated with all possible coalitions. Its (somehow involved) definition is recalled next.

Let us denote by $e(S, x)$ the excess of a coalition $S \subseteq N$ at the imputation $x \in X(\mathcal{G})$, i.e., the value $e(S, x) = v(S) - x(S)$ measuring the dissatisfaction of S at x, with $x(S)$ being a shorthand for $\sum_{i \in S} x_i$. Moreover, let us define $\theta(x)$ as the $2^{|N|} - 2$ dimensional vector where the various excesses of all coalitions $S \subset N$ are arranged in non-increasing order. Let $\theta(x)[i]$ be the i-th element of $\theta(x)$ and, for any imputation $y \in X(\mathcal{G})$, let $\theta(y) \prec \theta(x)$ denote that $\theta(y)$ is *lexicographically smaller* than $\theta(x)$, i.e., there exists a positive integer q such that $\theta(y)[i] = \theta(x)[i]$ for all $i < q$ and $\theta(y)[q] < \theta(x)[q]$. Then, the *nucleolus* of $\mathcal{G}$ is defined[1] as the set

$$\mathcal{N}(\mathcal{G}) = \{x \in X(\mathcal{G}) \mid \nexists y \in X(\mathcal{G}) \text{ s.t. } \theta(y) \prec \theta(x)\}.$$

[1]The nucleolus can be equivalently defined as the set of all imputations lexicographically *maximizing* the non-*decreasingly* sorted vector over the (satisfaction) values $x(S) - v(S)$, for each coalition $S \subset N$.

Example 1.3. Consider, again, the setting of Example 1.1, and the imputation $\bar{x}$ such that $\bar{x}_p = v(\{p\})+10 = 40$, $\bar{x}_s = v(\{s\}) + 10 = 20$, and $\bar{x}_g = v(\{g\}) = 10$. We claim that $\bar{x}$ is the only imputation in the nucleolus of $\mathcal{G}_0$.

To prove the claim, observe first that the maximum excess at $\bar{x}$ over all coalitions is $\max_{S\subseteq\{p,s,g\}} e(S,\bar{x}) = 0$. In particular, $e(\{p,s\},\bar{x}) = v(\{p,s\}) - \bar{x}_p - \bar{x}_s = 60 - 40 - 20 = 0$ and $e(\{g\},\bar{x}) = v(\{g\}) - \bar{x}_g = 10 - 10 = 0$, while $e(S,\bar{x}) = -10$ holds, for each coalition $S \subset \{p,s,g\}$ with $S \neq \{p,s\}$ and $S \neq \{g\}$. Let now $\bar{x}'$ be an imputation such that $\bar{x}' \neq \bar{x}$, and consider the following two cases. In the case where $\bar{x}'(\{p,s\}) < \bar{x}(\{p,s\})$, we get that $e(\{p,s\},\bar{x}') > e(\{p,s\},\bar{x}')$. Therefore, $\max_{S\subset\{p,s,g\}} e(S,\bar{x}') > \max_{S\subset\{p,s,g\}} e(S,\bar{x})$, which implies that $\theta(\bar{x}) \prec \theta(\bar{x}')$, so that $\bar{x}'$ does not belong to the nucleolus. On the other hand, in the case where $\bar{x}'(\{p,s\}) > \bar{x}(\{p,s\})$, we have that $\bar{x}'_g < \bar{x}_g$, because $\bar{x}'_p + \bar{x}'_s + \bar{x}'_g = 70$ holds, by the efficiency of the imputation $\bar{x}'$. Thus, $e(\{g\},\bar{x}') > e(\{g\},\bar{x}')$ holds. Therefore, $\max_{S\subset\{p,s,g\}} e(S,\bar{x}') > \max_{S\subset\{p,s,g\}} e(S,\bar{x})$, and again $\bar{x}'$ does not belong to the nucleolus. It remains to consider possible worth-redistributions between p and s such that $\bar{x}'_s + \bar{x}'_p = 60$ holds. Recall that $e(S,\bar{x}) = -10$, for each coalition $S \subset \{p,s,g\}$ with $S \neq \{p,s\}$ and $S \neq \{g\}$. In particular, p and s have the same excess at $\bar{x}$. Therefore, every worth assignment that does not equally divide the surplus 20 between p and s (as in $\bar{x}$) would alter this equal-excess state, so that either player would increase its excess, leading to a vector of excesses that is lexicographically greater than $\theta(\bar{x})$. It follows that $\bar{x}$ is in fact the only imputation in the nucleolus. ◁

Note that, in the above example, the only imputation $\bar{x}$ in the nucleolus of the game $\mathcal{G}_0$ belongs to the core of the game, too. This is not by chance: It is well-known and easy to see that, whenever the core of a game is not empty, the nucleolus is a subset of the core [56]. The following examples elaborate slightly more involved scenarios for nucleolus computation, where the core of the game is empty.

Example 1.4. Consider a coalitional game $\mathcal{G}_1 = \langle N, v\rangle$ over the players in $N = \{a,b,c\}$, and such that $v(\{a\}) = v(\{b\}) = v(\{c\}) = 0$, $v(\{a,b\}) = 1$, $v(\{a,c\}) = -2$, $v(\{b,c\}) = 2$, and $v(\{a,b,c\}) = 1$. Note that the game is not monotone, in that adding some player to a given coalition might cause a loss of the available worth. For instance, for b and c to collaborate with a is not beneficial and in fact leads to a loss of the available worth, since $v(\{b,c\}) = 2$ while $v(\{a,b,c\}) = 1$.

Assume that $\bar{x}$ is an imputation in the nucleolus of $\mathcal{G}_1$, and consider the following expressions for the excesses at $\bar{x}$:

- $e(\{a\},\bar{x}) = v(\{a\}) - \bar{x}_a = -\bar{x}_a$;
- $e(\{b\},\bar{x}) = v(\{b\}) - \bar{x}_b = -\bar{x}_b$;
- $e(\{c\},\bar{x}) = v(\{c\}) - \bar{x}_c = -\bar{x}_c$;
- $e(\{a,b\},\bar{x}) = v(\{a,b\}) - \bar{x}_a - \bar{x}_b = 1 - \bar{x}_a - \bar{x}_b$;
- $e(\{a,c\},\bar{x}) = v(\{a,c\}) - \bar{x}_a - \bar{x}_c = -2 - \bar{x}_a - \bar{x}_c$;
- $e(\{b,c\},\bar{x}) = v(\{b,c\}) - \bar{x}_b - \bar{x}_c = 2 - \bar{x}_b - \bar{x}_c$.

Since $\bar{x}$ is an imputation, we have $\bar{x}_a + \bar{x}_b + \bar{x}_c = 1$, $\bar{x}_a \geq 0$, $\bar{x}_b \geq 0$, and $\bar{x}_c \geq 0$. Therefore, $e(\{b,c\},\bar{x}) = v(\{b,c\}) - \bar{x}_b - \bar{x}_c = 2 - \bar{x}_b - \bar{x}_c \geq 1 \geq v(S) \geq e(S,\bar{x})$ holds, for each coalition $S \neq \{b,c\}$. In words, the coalition $\{b,c\}$ is the one where the maximum excess is necessarily achieved at $\bar{x}$, even if the whole available worth is distributed only between b and c. In order to minimize the maximum excess, it follows that $\bar{x}$ is such that $\bar{x}_a = 0$ and $\bar{x}_b + \bar{x}_c = 1$. We now distinguish two cases.

In the case where $\bar{x}_b > \bar{x}_c$, then the above expressions for the excesses at $\bar{x}$ plus the equalities $\bar{x}_a = 0$ and $\bar{x}_b + \bar{x}_c = 1$ lead to the vector $\theta(\bar{x}) = (1, 1-\bar{x}_b, 0, -\bar{x}_c, -\bar{x}_b, -2-\bar{x}_c)$. Therefore, in order to lexicographically minimize $\theta(\bar{x})$ and recalling that $\bar{x}_b \geq 0$, $\bar{x}_c \geq 0$ and $\bar{x}_b + \bar{x}_c = 1$, it must be the case that $\bar{x}_b = 1$ and, hence, $\bar{x}_a = \bar{x}_c = 0$.

In the case where $x_b \leq x_c$, we have instead $\theta(\bar{x}) = (1, 1-\bar{x}_b, 0, -\bar{x}_b, -\bar{x}_c, -2-\bar{x}_c)$, but again we have to set $\bar{x}_b = 1$ in order to lexicographically minimize $\theta(\bar{x})$.

In both cases, we have shown that $\bar{x}$ is such that $\bar{x}_a = \bar{x}_c = 0$ and $\bar{x}_b = 1$, and that this is hence the only imputation in the nucleolus. Therefore, the nucleolus suggests us that in the game $\mathcal{G}_1$ it is fair that player b takes the whole worth for herself. On the other hand, it can be checked that the core of the game is empty. For instance, $\bar{x}$ is not "stable" according to the core, because players b and c find it convenient to leave the grand coalition and share the worth $v(\{b,c\}) = 2$ in a way that each of them receives more than in $\bar{x}$ (e.g., they can get $1 + \frac{1}{2}$ and $\frac{1}{2}$, respectively).

The above is of course an extreme scenario for worth distribution, with a player getting the whole worth for itself. For an example going into the opposite direction, consider the game $\mathcal{G}'_1 = \langle N, v'\rangle$ where $v'(\{a\}) = v'(\{b\}) = v'(\{c\}) = 0$, and $v'(\{a,b\}) = v'(\{a,c\}) = v'(\{b,c\}) = v'(\{a,b,c\}) = 1$. In this case, by the symmetry of the worth function, it can be easily checked that the nucleolus consists of the vector $\bar{x}'$ with $\bar{x}'_a = \bar{x}'_b = \bar{x}'_c = \frac{1}{3}$,

which is a very natural outcome for this game. However, the core of $\mathcal{G}'_1$ is again empty. In particular, $\bar{x}'$ is not stable because $\bar{x}'_b + \bar{x}'_c < v'(\{b, c\})$. $\lhd$

In all the examples discussed above, the nucleolus happened to be a singleton. In fact, this is not by chance and it is actually always the case for any game $\mathcal{G}$ such that $X(\mathcal{G}) \neq \varnothing$ [56]. Accordingly, for any such game $\mathcal{G}$, by $\mathscr{N}(\mathcal{G})$ we directly mean the imputation (as it is commonly done in the literature), rather than the set of which it is the unique element.

Therefore, the nucleolus provides a "one-solution" deterministic method to fairly distribute the total worth among the players. Moreover, we have already recalled that it is a stable imputation whenever it is possible, in that it is a subset of the core of the game whenever the latter is not empty. This relationship with the core, together with its uniqueness property, makes the nucleolus the solution concept of choice in modeling several and relevant application scenarios (cf. [15]). Some of these scenarios are next discussed.

The Talmund rule A quite long-standing scenario, whose interpretation puzzled people for two millennia, comes from the Talmund. The story is one of a man who died, by leaving debts 100, 200, and 300 (zuz) to three distinct creditors c_1, c_2, and c_3, respectively, totalling more than his estate E. The question is how the estate has to be divided among the creditors. The Talmud stipulates the following division for three values of E. For $E = 100$, the estate has to be equally divided. For $E = 200$, c_1, c_2, and c_3 should receive 50, 75, and 75, respectively. And, for $E = 300$, creditors should receive 50, 100, and 150, respectively. This obscure rule has spawned a large literature, until Aumann and Maschler [4] presented a coalitional game that can be naturally associated with this bankruptcy problem and showed that the nucleolus of the game precisely prescribes the numbers reported in the Talmud—whereby the Talmund was since then credited to anticipate cooperative game theory. In fact, since the work of Aumann and Maschler [4], the nucleolus is often considered as an appropriate solution concept for fair distribution in bankruptcy problems, as opposed to the naïve proportional division.

Airport pricing policies Airport pricing policies have been illustrated by Littlechild and Thompson [40] in terms of game theoretic concepts. In particular, an airport pricing policy is discussed where the common costs of runway construction are shared among the different aircraft types according to a club principle. Optimal runway size and fair and efficient landing fees are then analyzed in the perspective of game theory and rules are suggested for charging costs based on the nucleolus. The model is applied to Birmingham Airport. Airport related games and their relations with the nucleolus are also discussed by Brânzei et al. [10].

Water supply management Water supply management policies are the application subject discussed by Young et al. [68]. In particular, the scenario is about the allocation of limited supplies of water and related land resources for several mutually conflicting purposes, e.g., municipal, industrial, agricultural. A concrete application scenario refers to water distribution management in a Swedish region, where a main problem is how to determine a fair charging policy of joint costs. Various cost allocation methods based on the nucleolus and supposedly well-suited to model the analyzed situation are discussed, from both the theoretical and the practical standpoints.

Computer networks A file sharing service scenario in a computer network is dealt with by Militano et al. [44]. In the described scenario, there is a service provider offering P2P group-options to its customers, and acting as a cooperation server coordinator. To reach its economic goals, the service coordinator implements a policy minimizing user costs, while allowing a cost distribution that they judge to be fair. These are compulsory conditions for the cooperative process to be accepted by all parties. Users may specify some constraints over their contribution to the proposed P2P framework. For instance, any user may impose a limit on the amount of data to be downloaded in the P2P application. A suitable pricing function must be defined for every non-empty group of users, which may take into account frequent buyers, customers with high-feedbacks, and so on. Noteworthy, because of the group-discount modeled by the pricing function, the cost assigned to each node will be always not higher than the cost that node would sustain for a stand-alone download. In fact, the precise cost is determined according to the nucleolus of a coalitional game naturally induced by the application.

Job assignment A further natural application modeling that can be attained by employing the nucleolus is described by Solymosi et al. [63]. Suppose you have n jobs, with each job to be processed by one of n available machines. Each machine can process any of the jobs, but the efficiency in doing that varies with the machine. Suppose, further, that jobs and machines are owned by n distinct owners. In the simplest assignment, each owner might decide to process his job on his machine. However, this may not correspond to the best solution, which might be counter-wisely attained if involved people shared their resources in

job-machines assignments. Being the involved agents self-motivated, a suitable solution to the described problem is obtained by Solymosi et al. [63] by modeling the application scenario in the form of a suitable coalitional game and then computing its nucleolus.

Bohm-Bawerk's horse market Finally, we mention that the nucleolus has been also applied by Núñez and Rafels [45] to the context of Bohm-Bawerk's horse market games, that are, two-sided market games without product differentiation. In this context, the nucleolus emerged to enjoy several desirable economic properties. Results on related settings have been provided by Granot and Granot [23].

1.2 Research Questions and Contributions

Looking at players' decision processes about worth distributions, it is sensible to assume players' reasoning resources not to come unbounded and to use the tools of computational complexity as a viable mean to model and reason about this bounded rationality principle [14]. In particular, it is easily noted that computational questions arising from coalitional games are of interest whenever the function specifying the worth associated with each possible coalition is encoded in some succinct way, in particular, when it is given in terms of polynomially computable functions over some combinatorial structure. Indeed, all problems trivialize if we explicitly represent the worth of every coalition, which requires exponential space in the number of involved players. Coalitional games whose worth functions can be computed in polynomial time on top of an underlying succinct encoding will be hereinafter called *compact (coalitional) games* (see [27]).

A non-exhaustive list of well known-classes of compact games includes graph and hypergraph games [14], marginal contribution nets [30], games in multi-issue domains [12], weighted voting games [19], minimum cost spanning tree games [43], flow games [33], linear production games [48], multi-attribute games [31], read-once (and general) marginal contribution nets [18], skill games [6], matching games [34, 58], path disruption games [5], (vertex) connectivity games [7], cover and clique games [11].

Coalitional games gained popularity in the context of multi-agent systems and artificial intelligence research since the nineties, when they had been recognized in these research communities as natural models to understand and reason about cooperative actions. In particular, inspired by the seminal paper of Deng and Papadimitriou [14], the questions of finding representation schemes to compactly encode worth functions and assessing over them the complexity of solution concepts have motivated most of the research on coalitional games in the artificial intelligence field. However, despite several efforts have been spent and remarkable results have been achieved, a clear picture of the complexity of reasoning problems involving the nucleolus over compact games was missing.

The goal of this paper is precisely to shed lights on these complexity issues, in particular, by answering two open research questions. The starting point of our research is the observation that various classes of compact games have been studied over the years, and for many of them, such as *minimum spanning tree games* (over arbitrary graphs) [20], *threshold games* [19], *flow and linear production games* [15], it turned out that computing the nucleolus is an intractable problem, formally **NP**-hard.

However, prior to our work, a non-trivial upper bound was missing that holds on all compact games and is possibly tight for some relevant well-known representation schemes. In fact, establishing this bound is technically challenging as a consequence of the intricate definition of the nucleolus involving reasoning over exponentially large vectors, and has been mentioned as an open issue in the literature [19]. Our first contribution is to address this issue. Indeed,

(1) We focus on the "standard" approaches by Kopelowitz [36] and by Maschler et al. [42] for computing the nucleolus of coalitional games, both based on solving a succession of linear programs. We shed lights on the properties of their methods, by showing that the former requires in some cases exponentially many programs to be solved (w.r.t. the number of players), as opposed to the latter which is known to converge after linearly many iterations (see, e.g., [34]). This analysis is of independent interest to the theory of coalitional games, given that the two methods have been sometimes (erroneously) considered as interchangeable in the literature.

(2) We define a computation method for the nucleolus that is an adaptation of the approach by Maschler et al. [42]. While each linear program needs to take into account exponentially many coalitions, we show that only polynomially many representative ones have to be taken into account and computed when moving from one step to the next. In particular, we provide analytical characterizations for such representatives (in terms of appropriate concepts of polyhedral geometry) as well as algorithms for their computation.

(3) We show that the above computation method can be executed in polynomial time by a deterministic

Turing machine equipped with an **NP** oracle.[2] In terms of complexity classes (of functions), this shows that computing the nucleolus of any compact coalitional game is feasible in $\mathbf{F}\boldsymbol{\Delta}_{\mathbf{2}}^{\boldsymbol{P}}$. To establish the result, we need to develop a theory of "succinctly" specified linear programs, that is, roughly speaking, of linear programs with n variables and $O(\exp(n))$ inequalities, which can be encoded in $O(n^c)$ space, for some constant c. This technical contribution is of independent interest, and may be useful in different fields of research, because it is about such a basic mathematical tool as the systems of linear inequalities.

(4) We show that the above result is tight even on the simple class of *graph games*, which is a well-known class of compact games defined by Deng and Papadimitriou in their seminal paper [14]. Indeed, deciding whether an outcome is the nucleolus of a graph game turns out to be $\boldsymbol{\Delta}_{\mathbf{2}}^{\boldsymbol{P}}$-complete, with this class being the counterpart of $\mathbf{F}\boldsymbol{\Delta}_{\mathbf{2}}^{\boldsymbol{P}}$ for decision problems. The hardness result is based on a rather elaborate combinatorial reduction, and it is interesting in the light of the fact that the *pre-nucleolus* of graph games is tractable, instead. Notably, this latter notion is defined precisely as the nucleolus, but getting rid of the individual rationality constraints. Thus, the complexity jump between these two solution concepts may be viewed as the cost to be paid for guaranteeing the voluntary participation to the distribution process.

The complexity analysis carried out on compact coalitional games demonstrated that computing the nucleolus is intractable in general, thus calling for identifying (possibly large) tractability islands. In particular, by inspecting our $\boldsymbol{\Delta}_{\mathbf{2}}^{\boldsymbol{P}}$-hardness proofs (as well as **NP**-hardness proofs available in the literature), it emerges that reductions exploit settings where each player in the game may have a distinctive behavior. On the contrary, it is everyday life experience that agents, in reasoning within a specific decision context, behave according to a few behavioral schemes and hence can be classified in a limited number of categories, usually called *types*. Therefore, it is natural to ask whether focusing on classes of games with a bounded number of distinct player types is beneficial as far as the requirements for nucleolus computation are concerned.

The study of coalitional games w.r.t. the number of their players' types has been recently initiated by Shrot et al. [59], who mainly focused on graph games and games with *synergies among coalitions* [13], and then put forward by Ueda et al. [64] and by Aadithya et al. [1], who extended the analysis to classes of games with polynomial-time computable functions. Following these approaches, we say that a coalitional game is *k-typed* if its players can be partitioned into at most k types, with k being a fixed natural number. Moreover, we say that a game is (given) in *type-based* form if the type of each player is known a-priori, i.e., it is part of the input in the reasoning problems.

Prior to our work, the current knowledge was that problems related to the core, such as determining whether the core is not empty and checking whether an imputation belongs to the core, are feasible in polynomial time over compact k-typed games that are moreover given in type-based form (or for which determining players' types is feasible in polynomial time) [1, 64]. However, extending the analysis to further solution concepts, and in particular to the nucleolus, has been left as an open research issue [64], which is faced in this paper. Indeed,

(5) We show that the good computational properties proved for the core hold on the nucleolus, too: computing the nucleolus is feasible in polynomial time over compact k-typed games that are either given in type-based form, or for which determining players' types is feasible in polynomial time. As a noticeable specialization, we get that it is tractable to compute the nucleolus of k-typed graph games.

(6) We go further beyond this result by answering the open question of Ueda et al. [64], thereby shedding lights on coalitional games where players' types are considered:

- First, we exhibit a class of compact coalitional games over which even deciding whether two players have the same type is intractable, formally co-**NP**-complete. This implies that there is no efficient method (unless $\mathbf{P} = \mathbf{NP}$) to transform an arbitrary compact coalitional game into a game in type-based form.
- Second, we show that the above intractability result about transforming an arbitrary compact game into a game in type-based form holds even on games whose number of types is known to be at most 2. However, hardness is here established under *randomized reductions* (see [65]). Finally we show that, under this same complexity model, the result mentioned in **(5)** is essentially tight, since computing the nucleolus is intractable even on 2-typed compact games, if the classification of players by types is not known.

1.3 Organization

The rest of the paper is organized as follows. The setting of compact games and the $\boldsymbol{\Delta}_{\mathbf{2}}^{\boldsymbol{P}}$-hardness result that holds even on the class of graph games is illustrated in Section 2. The linear programming tools for nucleolus

[2] We assume that the reader is familiar with the basic notions of complexity theory, such as polynomial-time reductions and complexity classes in the polynomial hierarchy (see, e.g., [32]).

computation are illustrated in Section 3, while the analysis of problems about succinct linear programs is reported in Section 4. These two ingredients are then put together in Section 5 to establish the corresponding membership result that hold over any class of compact games. The analysis of k-typed compact games is provided in Section 6, while Section 7 reports the conclusions. There are also three appendices providing, respectively, proofs of some relevant properties for the reduction in the $\boldsymbol{\Delta_2^P}$-hardness result, the analysis of a different linear programming approach to nucleolus computation that is described in the literature, and the formal proof of a result about computational problems on succinct linear programs.

2 Compact Representations and Nucleolus Computation: Graph Games

Finding compact representation schemes for coalitional games and assessing over them the complexity of various solution concepts have attracted much research in the artificial intelligence field, especially in the last few years (see, for instance, the work by Ågotnes et al. [2] for a classification and discussions about existing approaches).

Following the framework recently proposed by Greco et al. [27], a *compact representation* $\mathcal{R}$ is viewed in this paper as a method to encode a class of coalitional games, denoted by $\mathcal{C}(\mathcal{R})$. Formally, any representation $\mathcal{R}$ is associated with an encoding function $\xi^{\mathcal{R}}$ and a worth function $v^{\mathcal{R}}$ such that, for any coalitional game $\mathcal{G} = \langle N, v\rangle$ in $\mathcal{C}(\mathcal{R})$, $\xi^{\mathcal{R}}(\mathcal{G})$ is the encoding of the game $\mathcal{G}$, and the function $v^{\mathcal{R}}(\xi^{\mathcal{R}}(\mathcal{G}), S)$ returns the worth associated with any coalition S, according to $\mathcal{G}$. Moreover, we assume that the game encoding includes the list of players, so that $|N| \leq ||\xi^{\mathcal{R}}(\mathcal{G})||$ holds.[3] We say that $\mathcal{R}$ is a *polynomial-time* compact representation (short: **P**-representation) if $v^{\mathcal{R}}$ can be computed by a deterministic transducer in time being polynomial in the size of the encoding of the game. Whenever a compact representation $\mathcal{R}$ is understood, we shall write just $\mathcal{G}$ instead of $\xi^{\mathcal{R}}(\mathcal{G})$, and $v(S)$ instead of $v^{\mathcal{R}}(\xi^{\mathcal{R}}(\mathcal{G}), S)$.

In the paper, we analyze the computational complexity of reasoning problems over games encoded according to compact representations. In the analysis, we follow the classical complexity theory based on standard Turing machines. As usual in this framework, numerical computations actually deal with numbers given in the fractional form p/q, where p (resp., q) is an integer (resp., natural number) encoded in binary. Therefore, in particular, the worth associated with any coalition is assumed to be a *rational number*, rather than an arbitrary (possibly irrational) real number. For models of computations tailored to work with real numbers (as well as with other fields), we refer the interested reader to [8]).

In this section, we start by studying the complexity of computing the nucleolus over the class of *graph games* defined by Deng and Papadimitriou [14]. Before presenting our results, we next recall some basic notions and facts on graph games.

2.1 Graph Games and (Pre-)Nucleolus Computation

Let us denote by $\texttt{GGR}$ the *graph-game* compact representation defined by Deng and Papadimitriou [14]. According to this representation, a coalitional game $\mathcal{G} \in \mathcal{C}(\texttt{GGR})$ is encoded as a weighted graph $\xi^{\texttt{GGR}}(\mathcal{G}) = \langle (N, E), w\rangle$, whose nodes in N denote the players, and where the list w encodes the edge weighting function, so that $w(e) \in \mathbb{Q}$ is the weight associated with the edge $e \in E$. Then, the worth $v^{\texttt{GGR}}(\xi^{\texttt{GGR}}(\mathcal{G}), S)$ is computed for every coalition $S \subseteq N$ by taking the sum of the weights of all edges of $\xi^{\texttt{GGR}}(\mathcal{G})$ included in S, i.e., $v^{\texttt{GGR}}(\xi^{\texttt{GGR}}(\mathcal{G}), S) = \sum_{e \in E | e \subseteq S} w(e)$. Note that $\texttt{GGR}$ is a **P**-representation.

Example 2.1. Consider the graph game (encoded by the weighted graph) depicted on the top-left part of Figure 1 over the set of players $\{a, b, c\}$. It is easily seen that: the coalition $\{a, b, c\}$ gets a worth $v(\{a, b, c\}) = 2-2+1 = 1$; coalitions $\{a, b\}$, $\{a, c\}$, and $\{b, c\}$ get worth $v(\{a, b\}) = 1$, $v(\{a, c\}) = -2$, and $v(\{b, c\}) = 2$, respectively, and all coalitions formed by one player get worth 0. Thus, the worth function is precisely the one of the game discussed in Example 1.4. However, note that the graph encodes 2^3 coalition worths, via 3 weights only. Indeed, in this representation, $O(n^2)$ weights succinctly encode the 2^n coalition worths, where n is the number of players. $\lhd$

Deng and Papadimitriou [14] studied the computational complexity of a number of solution concepts over graph games, such as the core and the *pre-nucleolus*. This latter solution concept is defined precisely as the nucleolus, but by getting rid of the requirement that the outcome must be individually rational: Formally, we say that a vector of payoffs x is a *pre-imputation* of a coalitional game $\mathcal{G} = \langle N, v\rangle$ if $x(N) = v(N)$. Then, the pre-nucleolus of $\mathcal{G}$ is the set of all pre-imputations lexicographically minimizing the sorted vector of *excesses* associated with all possible coalitions. In fact, as for the nucleolus, the pre-nucleolus is always a singleton (and will be therefore identified with its unique element).

[3] Note that we use the standard notation where $|\cdot|$ denotes the cardinality of a set or a list, and $||\cdot||$ denotes the size of any object encoding.

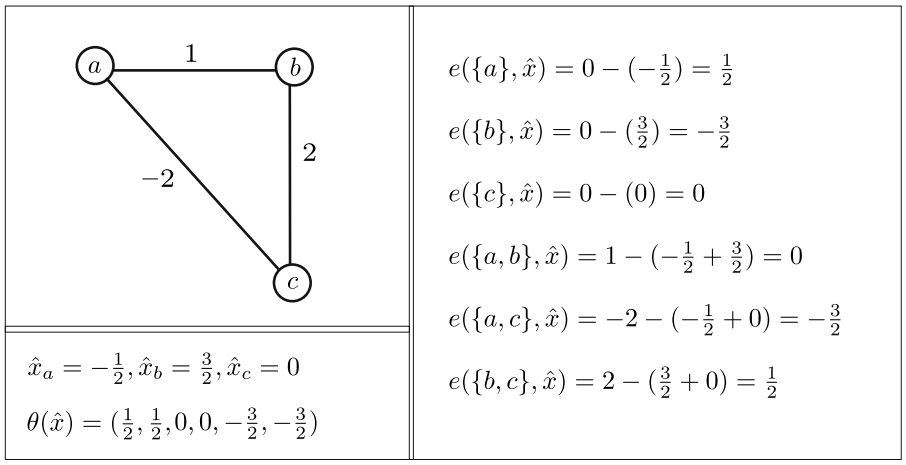

Figure 1: The graph game in Example 1.4.

Deng and Papadimitriou [14] observed that the pre-nucleolus of a graph game can be computed in polynomial time, by showing that it can be characterized by a simple closed form: For any graph game $G = \langle (N, E), w \rangle$, the pre-nucleolus $\hat{x}$ is such that

$$\hat{x}_i = \frac{1}{2} \sum_{j \in N \mid \{i,j\} \in E} w(\{i, j\}), \text{ for each } i \in N.$$

Example 2.2. Consider again the graph game discussed in Example 2.1, and the pre-imputation $\hat{x}$ such that $\hat{x}_a = -\frac{1}{2}$, $\hat{x}_b = \frac{3}{2}$, and $\hat{x}_c = 0$ (hence $\hat{x}_a + \hat{x}_b + \hat{x}_c = v(\{a, b, c\}) = 1$). Figure 1 reports the excess $e(S, \hat{x})$, for each coalition $S \subset \{a, b, c\}$, plus the vector $\theta(\hat{x}) = (\frac{1}{2}, \frac{1}{2}, 0, 0, -\frac{3}{2}, -\frac{3}{2})$ associated with $\hat{x}$, where such excesses are lexicographically ordered in non-increasing order.

According to the result by Deng and Papadimitriou [14], $\hat{x}$ is the pre-nucleolus of the game, and hence there is no pre-imputation x such that $\theta(x) \prec \theta(\hat{x})$. For instance, consider the pre-imputation x such that $x_a = 1$ and $x_b = x_c = 0$, and check that $\theta(x) = (2, 0, 0, 0, -1, -3)$ holds, whence $\theta(\hat{x}) \prec \theta(x)$. ◁

Note that if all individual rationality constraints are satisfied by the pre-nucleolus $\hat{x}$, then $\hat{x}$ is also the nucleolus of the game—indeed, if by contradiction there is an imputation x such that $\theta(x) \prec \theta(\hat{x})$ holds, then x would also witness that $\hat{x}$ is not the pre-nucleolus. Thus, the above discussed closed form for the pre-nucleolus might provide us (even) with the nucleolus in some lucky situations. In general, however, this is not the case. For instance, the pre-nucleolus $\hat{x}$ of the game discussed in Example 2.1 (and Example 1.4) is such that $\hat{x}_a = -\frac{1}{2}$, $\hat{x}_b = \frac{3}{2}$, and $\hat{x}_c = 0$ (in particular, it is not an imputation because of $\hat{x}_a < v(\{a\}) = 0$), whereas we know by Example 1.4 that the nucleolus $\bar{x}$ is such that that $\bar{x}_a = 0$, $\bar{x}_b = 1$, and $\bar{x}_c = 0$.

Given that differences between the nucleolus and the pre-nucleolus are confined to the individual rationality constraints, it is natural to expect that the nucleolus of graph games is tractable in its turn, possibly again with a simple closed-form characterization. However, the question of whether the nucleolus of graph games is efficiently computable was not answered by Deng and Papadimitriou [14]. Moreover, despite that the setting of graph games has been tremendously influential in the study of compact encodings for coalitional games and that our knowledge of the complexity issues arising there is now fairly complete, this question remained without an answer so far. Next, we shall answer this question by surprisingly highlighting that, to guarantee individual rationality, a significant computational cost has to be paid.

2.2 Hardness on Graph Games: The Cost of Individual Rationally

In this section, we show that given a game $\mathcal{G} \in \mathcal{C}(\texttt{GGR})$ encoded as a graph game, deciding whether a vector is the nucleolus of $\mathcal{G}$ is $\boldsymbol{\Delta}_{\mathbf{2}}^{\boldsymbol{P}}$-hard. To show this hardness result, the reduction is based on encoding **3CNF** Boolean formulae in terms of suitable graph games,[4] where recall that such formulae are in conjunctive normal form and each clause contains at most three literals, i.e., positive or negated variables.

Hereinafter, assume that a **3CNF** Boolean formula $\phi = c_1 \wedge \cdots \wedge c_m$ is given over a set $\{\alpha_1, \ldots, \alpha_n\}$ of variables, with $n \geq 2$.

Based on the formula ϕ, we first define the graph $(N_{\boldsymbol{N}}, E_{\boldsymbol{N}})$ as follows—an illustration is reported in Figure 2. The set $N_{\boldsymbol{N}}$ of its nodes/players is such that $N_{\boldsymbol{N}} = N_k \cup \overline{N}_k \cup N_r$, where N_k is the set of players containing:

[4]This is a non-trivial extension of a reduction that can be found in the work by Greco et al. [27].

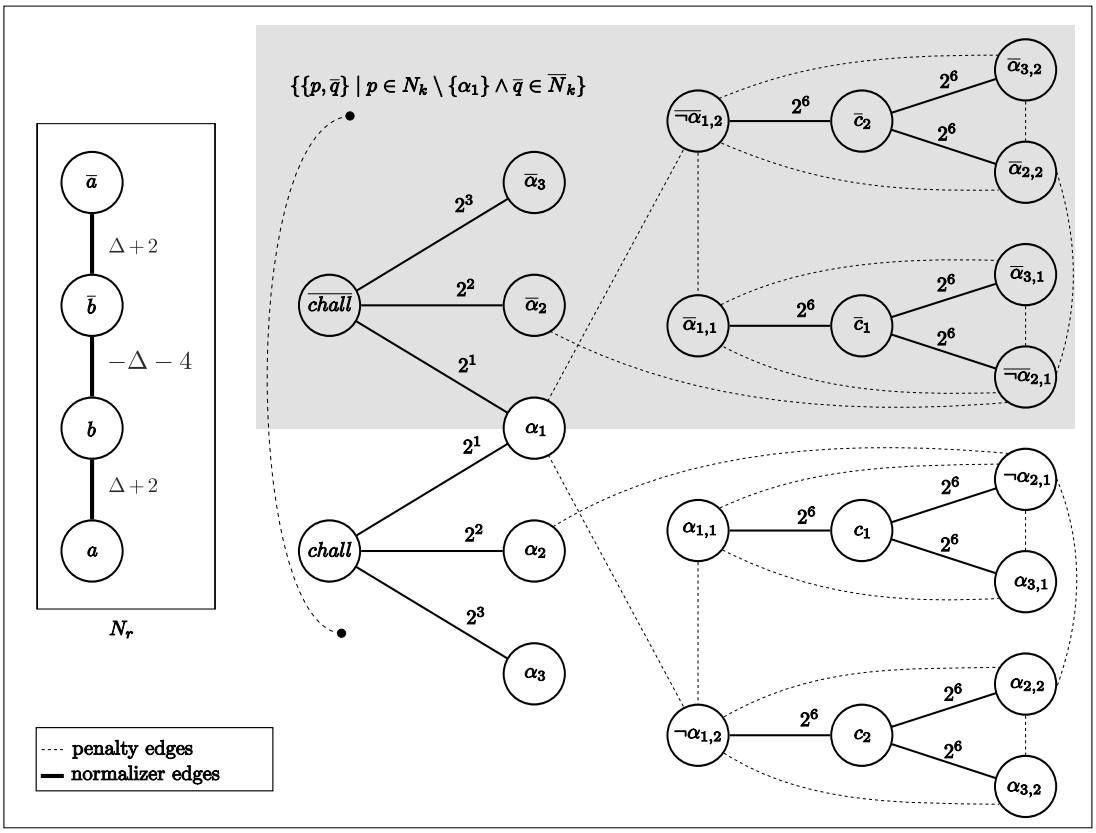

Figure 2: The game $\boldsymbol{N}(\widehat{\phi})$, where $\widehat{\phi} = (\alpha_1 \vee \neg\alpha_2 \vee \alpha_3) \wedge (\neg\alpha_1 \vee \alpha_2 \vee \alpha_3)$. Nodes in $\overline{N}_k$ are reported in the gray area.

- a *variable player* α_i, for each variable α_i in ϕ;
- a *clause player* c_j, for each clause c_j in ϕ;
- a *literal player* $\ell_{i,j}$ (either $\ell_{i,j} = \alpha_{i,j}$ or $\ell_{i,j} = \neg\alpha_{i,j}$), for each clause c_j and each literal ℓ_i ($\ell_i = \alpha_i$ or $\ell_i = \neg\alpha_i$, respectively) occurring in it;
- a special player "*chall*";

where $\overline{N}_k = \{\overline{p} \mid p \in N_k \wedge p \neq \alpha_1\}$ is the set containing a (over-lined) copy of each player in N_k but α_1, and where $N_r = \{a, \overline{a}, b, \overline{b}\}$.

The set $E_{\boldsymbol{N}}$ of its edges is such that $E_{\boldsymbol{N}} = E_k \cup \overline{E}_k \cup E_r \cup E_b$, where (i) E_k is the set of edges containing:

- an edge $\{chall, \alpha_i\}$, for each variable α_i in ϕ;
- an edge $\{c_j, \ell_{i,j}\}$, for each clause c_j and literal ℓ_i occurring in it;
- an edge $\{\ell_{i,j}, \ell_{i',j}\}$, for each clause c_j, and for each pair of distinct literals ℓ_i and $\ell_{i'}$ occurring in it;
- an edge $\{\alpha_{i,j}, \neg\alpha_{i,j'}\}$, for each pair of distinct clauses c_j and $c_{j'}$, and for each variable α_i occurring positively in c_j and negated in $c_{j'}$;
- an edge $\{\alpha_i, \neg\alpha_{i,j}\}$, for each variable α_i and each literal player of the form $\neg\alpha_{i,j}$ (encoding that α_i occurs negated in the clause c_j).

(ii) $\overline{E}_k = \{\{\overline{p}, \overline{q}\} \mid \{p, q\} \in E_k \wedge \{\overline{p}, \overline{q}\} \subseteq \overline{N}_k\} \cup \{\{\alpha_1, \overline{q}\} \mid \{\alpha_1, q\} \in E_k \wedge \overline{q} \in \overline{N}_k\}$ is the set containing a copy of each edge in E_k over the nodes in $\overline{N}_k \cup \{\alpha_1\}$ corresponding to those in N_k, (iii) $E_r = \{\{a, b\}, \{\overline{a}, \overline{b}\}, \{b, \overline{b}\}\}$, and (iv) $E_b = \{\{p, \overline{q}\} \mid p \in N_k \setminus \{\alpha_1\} \wedge \overline{q} \in \overline{N}_k\}$ is the set of edges between each node in $N_k \setminus \{\alpha_1\}$ and each node in $\overline{N}_k$.

We now define the weighted graph $\boldsymbol{N}(\phi) = \langle (N_{\boldsymbol{N}}, E_{\boldsymbol{N}}), w \rangle$ in a way that edges in $E_{\boldsymbol{N}}$ can be partitioned into the following three groups based on their weights.

(Positive edges)

- $w(\{c_j, \ell_{i,j}\}) = w(\{\overline{c}_j, \overline{\ell}_{i,j}\}) = 2^{n+3}$, for each clause c_j and literal ℓ_i occurring in it;
- $w(\{chall, \alpha_i\}) = w(\{\overline{chall}, \overline{\alpha}_i\}) = 2^i$, for each $2 \leq i \leq n$;
- $w(\{chall, \alpha_1\}) = w(\{\overline{chall}, \alpha_1\}) = 2^1$.

("Penalty" edges)

- $w(\{\ell_{i,j}, \ell_{i',j}\}) = w(\{\overline{\ell}_{i,j}, \overline{\ell}_{i',j}\}) = -2^{m+n+7}$, for each clause c_j and pair of distinct literals ℓ_i and $\ell_{i'}$ occurring in it;
- $w(\{\alpha_{i,j}, \neg\alpha_{i,j'}\}) = w(\{\overline{\alpha}_{i,j}, \overline{\neg\alpha}_{i,j'}\}) = -2^{m+n+7}$, for each pair of distinct clauses c_j and $c_{j'}$, and for each variable α_i occurring positively in c_j and negated in $c_{j'}$;
- $w(\{\alpha_i, \neg\alpha_{i,j}\}) = w(\{\overline{\alpha}_i, \overline{\neg\alpha}_{i,j}\}) = -2^{m+n+7}$, for each clause c_j and variable α_i with $i \neq 1$ occurring negated in it;
- $w(\{\alpha_1, \neg\alpha_{1,j}\}) = w(\{\alpha_1, \overline{\neg\alpha}_{1,j}\}) = -2^{m+n+7}$, for each clause c_j where α_1 negatively occurs;
- $w(\{p, \overline{q}\}) = -2^{m+n+7}$, for each pair of players $p \neq \alpha_1$ and $\overline{q}$, with $p \in N_k$ and $\overline{q} \in \overline{N}_k$.

("Normalizer" edges) Let $\Delta = 1 - \sum_{e \in E_{\boldsymbol{N}} | e \subseteq N_k \cup \overline{N}_k} w(e)$. Then, let $w(\{a, b\}) = \Delta + 2$, $w(\{\overline{a}, \overline{b}\}) = \Delta + 2$, and $w(\{b, \overline{b}\}) = -\Delta - 4$.

Note that the size of the representation of all weights is polynomial in the number of variables and clauses of ϕ and that, given the formula ϕ, the corresponding weighted graph $\boldsymbol{N}(\phi)$ can be built in polynomial time. In addition, the construction features three important properties, which are stated below.

Lemma 2.3. *Let $\boldsymbol{N}(\phi) = \langle (N_{\boldsymbol{N}}, E_{\boldsymbol{N}}), w \rangle$ be the graph game associated with a* **3CNF** *formula ϕ defined over the set $\{\alpha_1, \ldots, \alpha_n\}$ of variables. Assume that $n > 1$. Then,*

(A) $\Delta > 1$*;*

(B) $D + w(e) < 0$*, for each penalty edge e, where $D = \max_{S \subseteq N_k \cup \overline{N}_k} v(S)$ denotes the maximum worth over all the coalitions of players in $N_k \cup \overline{N}_k$.*

(C) $v(N_{\boldsymbol{N}}) = 1$.

Proof. Let us first show that (A) holds. Consider the values P^+ and P^- such that

- $P^+ = \sum_{e \in E_{\boldsymbol{N}}, e \subseteq N_k \cup \overline{N}_k, w(e) > 0} w(e)$, i.e., P^+ is the sum of the weights associated with the positive edges, and
- $P^- = \sum_{e \in E_{\boldsymbol{N}}, e \subseteq N_k \cup \overline{N}_k, w(e) < 0} w(e)$, i.e., P^- is the sum of the weights associated with the penalty edges,

and observe that $\Delta = 1 - P^+ - P^-$. In particular, as $n > 1$ holds, $\boldsymbol{N}(\phi)$ contains at least one penalty edge, and hence $-P^- \geq 2^{m+n+7}$. It follows that $\Delta \geq 1 + 2^{m+n+7} - P^+$.

Consider now the value P^+. By looking at the definition of positive edges, it is immediate to check that $P^+ \leq 2\left(3m2^{n+3} + \sum_{i=1}^{n} 2^i\right)$. Therefore, we get:

$$P^+ \quad \leq \quad 2\left(3m2^{n+3} + \sum_{i=1}^{n} 2^i\right) \quad \leq \quad 2\left(2^{m+n+5} + 2^{n+1}\right) \quad \leq \quad 2^{m+n+6} \; + \; 2^{n+2} \quad < \quad 2^{m+n+7}.$$

By combining the above relationship with the fact that $\Delta \geq 1 + 2^{m+n+7} - P^+$, we obtain that $\Delta > 1$ holds, which proves (A).

In order to conclude the proof of (B), let us observe that a trivial upper bound for $D = \max_{S \subseteq N_k \cup \overline{N}_k} v(S)$ is precisely the value P^+ of the sum of the weights associated with the positive edges. Thus, $D \leq P^+ < 2^{m+n+7}$ holds, i.e, $D - 2^{m+n+7} < 0$. Eventually, (B) follows by observing that $w(e) = -2^{m+n+7}$ holds, for each penalty edge.

Finally, concerning (C), note that $v(N_{\boldsymbol{N}}) = \sum_{e \in E_{\boldsymbol{N}}} w(e) = 1$ holds, because $w(\{a, b\}) + w(\{\overline{a}, \overline{b}\}) + w(\{b, \overline{b}\}) = \Delta = 1 - \sum_{e \in E_{\boldsymbol{N}} | e \subseteq N_k \cup \overline{N}_k} w(e)$ and given that $E_{\boldsymbol{N}} = \{e \in E_{\boldsymbol{N}} \mid e \subseteq N_k \cup \overline{N}_k\} \cup E_r$ with $\{e \in E_{\boldsymbol{N}} \mid e \subseteq N_k \cup \overline{N}_k\} \cap E_r = \varnothing$. □

By exploiting the above construction and the properties in Lemma 2.3, the main result of this section can be shown.

Theorem 2.4. *On the class $\mathcal{C}(\texttt{GGR})$ of graph games, deciding whether a vector is the nucleolus is $\boldsymbol{\Delta_2^P}$-hard (even if the vector is known to be an imputation).*[5]

Proof. Let $\phi = c_1 \wedge \cdots \wedge c_m$ be a *satisfiable* **3CNF** formula over a set $\{\alpha_1, \ldots, \alpha_n\}$ of variables that are lexicographically ordered (according to their indices). Deciding whether α_1 (that is, the lexicographically least significant variable) is true in the *lexicographically maximum satisfying assignment* for ϕ is a well-known $\boldsymbol{\Delta_2^P}$-complete problem [37]. Assume, w.l.o.g., that $n > 1$, i.e., there are at least two variables, and that the assignment mapping all variables to *false* is not satisfying.

Consider the graph game $\boldsymbol{N}(\phi) = \langle (N_{\boldsymbol{N}}, E_{\boldsymbol{N}}), w \rangle$ built based on ϕ. We shall show that deciding whether an imputation is the nucleolus of $\boldsymbol{N}(\phi)$ is as hard as deciding whether α_1 is true in the lexicographically maximum satisfying assignment for ϕ. We start by stating a useful property, whose proof is in Appendix A.

Property 2.4.(1). *Let S be a coalition such that $S \subseteq N_k \cup \overline{N}_k$ and $v(S) > 0$. Then, $S \cap (N_k \setminus \{\alpha_1\}) \neq \varnothing$ if, and only if, $S \cap \overline{N}_k = \varnothing$.*

A coalition solely built from players in N_k is said *primal*, whereas a coalition built from players in $\overline{N}_k \cup \{\alpha_1\}$ is said *dual*. In fact, from the definition of the game $\boldsymbol{N}(\phi)$, it is immediate to observe that there is a one-to-one correspondence between primal and dual coalitions. In particular, for each primal coalition S, let $\overline{S} = \{\overline{p} \mid p \in S \wedge p \neq \alpha_1\} \cup \{\alpha_1 \mid \alpha_1 \in S\}$ denote its corresponding dual coalition, and observe that $v(S) = v(\overline{S})$ holds. Moreover, for each dual coalition $\overline{S}$, let $S = \{p \mid \overline{p} \in \overline{S}\} \cup \{\alpha_1 \mid \alpha_1 \in \overline{S}\}$ denote its corresponding primal coalition, and observe that $v(\overline{S}) = v(S)$ holds.

Below, we shall state further properties of the construction, by focusing, w.l.o.g., on primal coalitions only. Again, proofs are reported in Appendix A.[6]

For any assignment σ, we denote by $\sigma \models \phi$ the fact that σ satisfies ϕ, and by $\sigma(\alpha_i) = \texttt{true}$ (resp., $\sigma(\alpha_i) = \texttt{false}$) the fact that α_i evaluates to true (resp., false) in σ. Moreover, for any coalition S, let σ_S denote the assignment such that $\sigma_S(\alpha_i) = \texttt{true}$ (resp., $\sigma_S(\alpha_i) = \texttt{false}$) if α_i occurs (resp., does not occur) in S.

Property 2.4.(2). $\max_{S \subseteq N_k} v(S) = m2^{n+3} + \max_{\sigma \models \phi} \sum_{\alpha_i \mid \sigma(\alpha_i) = \texttt{true}} 2^i$. *Moreover, let $S_* \subseteq N_k$ be a coalition such that $v(S_*) = \max_{S \subseteq N_k} v(S)$, and let σ_* be the lexicographically maximum satisfying assignment. Then, $\mathit{chall} \in S_*$ and $\sigma_{S_*} = \sigma_*$.*

Property 2.4.(3). *Let $S_* \subseteq N_k$ be a coalition with $v(S_*) = \max_{S \subseteq N_k} v(S)$. Then, for each coalition $S \subseteq N_k \cup \overline{N}_k$ with $S \neq S_*$ and $S \neq \overline{S}_*$, $v(S_*) = v(\overline{S}_*) \geq v(S) + 2$ holds. Moreover, for each imputation y, $e(S_*, y) \geq e(S, y) + 1$ and $e(\overline{S}_*, y) \geq e(S, y) + 1$ hold.*

Property 2.4.(4). *For any coalition S, $v(S) = v(S \cap N_r) + v(S \cap (N_k \cup \overline{N}_k))$.*

Property 2.4.(5). *Let $S_* \subseteq N_k$ be a coalition with $v(S_*) = \max_{S \subseteq N_k} v(S)$. Then, the eight coalitions $S_1 = S_* \cup \{a, b\}$, $S_2 = S_* \cup \{\overline{a}, \overline{b}\}$, $S_3 = \overline{S}_* \cup \{a, b\}$, $S_4 = \overline{S}_* \cup \{\overline{a}\,\overline{b}\}$, $S_5 = S_1 \cup \{\overline{a}\}$, $S_6 = S_2 \cup \{a\}$, $S_7 = S_3 \cup \{\overline{a}\}$, and $S_8 = S_4 \cup \{a\}$ are such that $v(S_1) = \cdots = v(S_8) = \max_{S \subseteq N_k \cup \overline{N}_k \cup N_r} v(S) = v(S_*) + \Delta + 2$.*

Property 2.4.(6). *For each imputation y and each coalition $S \notin \{S_1, \ldots, S_8\}$, it holds that $e(S_i, y) > e(S, y)$, for each $i \in \{1, \ldots, 8\}$.*

Armed with the above properties, consider the payoff vector x that assigns 0 to all players of $\boldsymbol{N}(\phi)$, but to α_1, which receives 1. Note that x is an imputation. Indeed, x is individually rational as $v(\{p\}) = 0$, for each player $p \in N_{\boldsymbol{N}}$, and it is efficient since $v(N_{\boldsymbol{N}}) = 1$, by Lemma 2.3.(C).

We now show that: *α_1 is true in the lexicographically maximum satisfying assignment σ^* for $\phi \Leftrightarrow x$ is the nucleolus.*

($\Rightarrow$) Assume that α_1 is true in σ^*, and let S_* be such that $v(S_*) = \max_{S \subseteq N_k} v(S)$. Recall that, given the correspondence between primal and dual coalitions, $v(S_*) = v(\overline{S}_*)$ holds. Consider the coalitions $S_1, \ldots, S_8$ defined in the statement of Property 2.4.(5) and note that, by Property 2.4.(2), $S_1 \cap \cdots \cap S_8 = \{\alpha_1\}$ holds. Hence, $e(S_i, x) = v(S_i) - 1$ holds, for each $i \in \{1, \ldots, 8\}$. Consider now any imputation $y \neq x$ (so that $y_{\alpha_1} < 1$) and note that, for each coalition S_i, with $i \in \{1, \ldots, 8\}$, $y(S_i) \leq 1$ holds. Indeed, $y(N_{\boldsymbol{N}}) = v(N_{\boldsymbol{N}}) = 1$ because of the efficiency of y and by Lemma 2.3.(C); moreover, y must assign to each player a non-negative payoff because of the individual rationality constraints (recall that $v(\{p\}) = 0$, for each player $p \in N_{\boldsymbol{N}}$). Eventually, $e(S_i, x) = v(S_i) - 1$ and $y(S_i) \leq 1$ lead to conclude that $e(S_i, y) \geq e(S_i, x)$ holds, for each $i \in \{1, \ldots, 8\}$.

[5] In fact, from the tractability of computing the pre-nucleolus of graph-games [14], this theorem also entails that even the knowledge of the pre-nucleolus does not help, in general.

[6] Some properties are only needed to prove subsequent properties. Yet, they are listed here to provide a clear picture of the flow of the proof.

Now, we claim that there is a coalition S_h, with $h \in \{1, \ldots, 8\}$, such that $e(S_h, y) > e(S_h, x)$. Indeed, assume for the sake of contradiction that $e(S_i, y) = e(S_i, x)$, for each $i \in \{1, \ldots, 8\}$. In particular, assume that $e(S_1, y) = e(S_1, x)$ and $e(S_4, y) = e(S_4, x)$ hold. From these relationships, we can conclude that $e(S_1, y) = e(S_4, y)$, because $e(S_1, x) = e(S_4, x)$ holds by Property 2.4.(5) and the fact that $e(S_1, x) = v(S_1) - 1$ and $e(S_4, x) = v(S_4) - 1$. In its turn, $e(S_1, y) = e(S_4, y)$ implies that $y(S_1) = y(S_4)$, so that we have in fact $y(S_1) = y(S_4) = 1$, because $v(S_1) - y(S_1) = e(S_1, y) = e(S_1, x) = v(S_1) - 1$. To conclude, observe that $S_1 \cap S_4 = \{\alpha_1\}$ and recall that y is an imputation. Hence, $y(S_1) = y(S_4) = 1$ implies that $y_{\alpha_1} = 1$, which is impossible.

We thus know that, for any imputation $y \neq x$, there is a coalition S_h, with $h \in \{1, \ldots, 8\}$, such that $e(S_h, y) > e(S_h, x)$. In fact, this immediately entails that $\theta(x) \prec \theta(y)$ holds. Indeed, $e(S_1, x) = \ldots = e(S_8, x)$ holds by Property 2.4.(5) and given that $e(S_i, x) = v(S_i) - 1$, for each $i \in \{1, \ldots, 8\}$. By Property 2.4.(6), the coalitions in $\{S_1, \ldots, S_8\}$ are those over which the maximum excess is achieved, for any specific imputation being considered. It follows that x is the nucleolus.

($\Leftarrow$) Assume that α_1 is false in the lexicographically maximum satisfying assignment σ^* for ϕ, and let S_* be such that $v(S_*) = v(\overline{S}_*) = \max_{S \subseteq N_k} v(S)$. Because of Property 2.4.(2), it is the case that $\alpha_1 \notin S_*$ and $\alpha_1 \notin \overline{S}_*$. Thus, $e(S_*, x) = e(\overline{S}_*, x) = v(S_*) = v(\overline{S}_*)$. Moreover, by Property 2.4.(5), $e(S_i, x) = e(S_*, x) + \Delta + 2 = e(\overline{S}_*, x) + \Delta + 2$ holds, for each $i \in \{1, \ldots, 8\}$. Consider now the imputation y such that $y_{chall} = y_{\overline{chall}} = \frac{1}{2}$ and $y_p = 0$, for any other player $p \in N_{\mathcal{N}} \setminus \{chall, \overline{chall}\}$. Note that, by the same property, we have $e(S_i, y) = e(S_*, y) + \Delta + 2$. Due to Property 2.4.(2), $chall \in S_*$ (and $\overline{chall} \in \overline{S}_*$, by duality). Thus, $e(S_*, y) = e(\overline{S}_*, y) = v(S_*) - \frac{1}{2} = v(\overline{S}_*) - \frac{1}{2} = e(S_*, x) - \frac{1}{2} = e(\overline{S}_*, x) - \frac{1}{2}$. That is, $e(S_*, y) < e(S_*, x)$ and $e(\overline{S}_*, y) < e(\overline{S}_*, x)$. Therefore, $e(S_i, y) = e(S_*, y) + \Delta + 2 < e(S_*, x) + \Delta + 2 = e(S_i, x)$ holds, for each $i \in \{1, \ldots, 8\}$. To conclude, recall from Property 2.4.(6) that the maximum excess is achieved over a coalition in $\{S_1, \ldots, S_8\}$, for any specific imputation being considered. Hence, x is not the nucleolus. □

Before leaving the section, following the formal setting in [27], we note that the above result immediately generalizes to all those classes of games that are at least as expressive as graph games.

Let $\mathcal{R}_1$ and $\mathcal{R}_2$ be a pair of game representations. We say that $\mathcal{R}_2$ is *at least as expressive (and succinct) as* $\mathcal{R}_1$, denoted by $\mathcal{R}_1 \precsim_e \mathcal{R}_2$, if there exists a function f in **FP** that translates a game $\xi^{\mathcal{R}_1}(\mathcal{G})$ represented in $\mathcal{R}_1$ into an equivalent game $\xi^{\mathcal{R}_2}(\mathcal{G})$ represented in $\mathcal{R}_2$, that is, into a game with the same players and the same worth function as the former one. More precisely, we require that $\xi^{\mathcal{R}_2}(\mathcal{G}) = f(\xi^{\mathcal{R}_1}(\mathcal{G}))$ and $v^{\mathcal{R}_1}(\xi^{\mathcal{R}_1}(\mathcal{G}), S) = v^{\mathcal{R}_2}(\xi^{\mathcal{R}_2}(\mathcal{G}), S)$, for each coalition of players S in the game $\mathcal{G}$.

With the above notions in place, we can now show the following.

Corollary 2.5. *Let $\mathcal{R}$ be any compact representation such that $\mathtt{GGR} \precsim_e \mathcal{R}$. On the class $\mathcal{C}(\mathcal{R})$, deciding whether a vector is the nucleolus is $\boldsymbol{\Delta}_{\mathbf{2}}^{\boldsymbol{P}}$-hard.*

Proof. From the $\boldsymbol{\Delta}_{\mathbf{2}}^{\boldsymbol{P}}$-hardness for graph games, we know that there is a polynomial-time reduction f_1 from any $\boldsymbol{\Delta}_{\mathbf{2}}^{\boldsymbol{P}}$ problem Υ to the problem of deciding whether a vector is the nucleolus of graph games. Moreover, recall that $\mathtt{GGR} \precsim_e \mathcal{R}$ means that there exists a polynomial-time function f_2 that translates any graph game $\xi^{\mathtt{GGR}}(\mathcal{G})$ into an equivalent game $f_2(\xi^{\mathtt{GGR}}(\mathcal{G}))$ belonging to $\mathcal{C}(\mathcal{R})$, that is, into a game with the same worth function and, thus, the same nucleolus as the former one. Therefore, the composition of f_1 and f_2 is a polynomial-time reduction from any $\boldsymbol{\Delta}_{\mathbf{2}}^{\boldsymbol{P}}$ problem to the the problem of deciding whether a vector is the nucleolus of games in $\mathcal{C}(\mathcal{R})$. □

As an example, the above $\boldsymbol{\Delta}_{\mathbf{2}}^{\boldsymbol{P}}$-hardness result applies to the class of games defined by *marginal contribution nets* [30], which is a well-known representation scheme at least as expressive as graph games [27].

In fact, in the following, we shall show that the bound provided by Corollary 2.5 is tight, in the sense that computing the nucleolus of *any* compact coalitional game is feasible in $\mathbf{F}\boldsymbol{\Delta}_{\mathbf{2}}^{\boldsymbol{P}}$.

3 Linear Programming Tools for Computing the Nucleolus

Several algorithms to compute the nucleolus of coalitional games have been proposed in the literature, but none of them has been designed to deal with the issues arising from compact games. The goal of this section is to illustrate how the classical "linear programming approach" [25, 34] may be adapted by using a bunch of new technical tools developed for this purpose.

3.1 Elements of Polyhedral Geometry

We first illustrate some elements of polyhedral geometry that will be relevant to our end. We refer the reader to the works by Papadimitriou and Steiglitz [49] and Grötschel et al. [28] for further background.

Basic Notions. Unless otherwise specified, any vector $x \in \mathbb{R}^n$ is viewed in this context as a column vector (i.e., an $n \times 1$ matrix). For a row vector z, the corresponding (column) vector is obtained as the *transposition* of z, denoted by z^T (and vice versa). For any pair of vectors $x, y \in \mathbb{R}^n$, $x^T y$ is their *inner product* $\sum_{i=1}^{n} x_i y_i$.

For any pair of natural numbers $m, n > 0$, we denote by $\mathbb{R}^{m \times n}$ the set of all $m \times n$ matrices with entries in $\mathbb{R}$. The i-th row of a matrix $A \in \mathbb{R}^{m \times n}$ is denoted by $A_{i,\bullet}$ (note that $A_{i,\bullet}^T \in \mathbb{R}^n$), the j-th column of A is denoted by $A_{\bullet,j}$ (note that $A_{\bullet,j} \in \mathbb{R}^m$), and the i-th entry of $A_{\bullet,j}$ (in fact, the j-th entry of $A_{i,\bullet}^T$) is denoted by $A_{i,j}$ (note that $A_{i,j} \in \mathbb{R}$).

A vector $x \in \mathbb{R}^n$ is a *linear combination* of the vectors $v^1, \ldots, v^k \in \mathbb{R}^n$, if there are k real numbers $\lambda^1, \ldots, \lambda^k$ such that $x = \sum_{h=1}^{k} \lambda^h v^h$; the combination is *proper* if neither $\lambda^h = 0$ for each $h \in \{1, \ldots, k\}$, nor $\lambda^{\bar{h}} = 1$ for some $\bar{h} \in \{1, \ldots, k\}$ and $\lambda_{h'} = 0$ for each $h' \neq \bar{h}$. A subset $S \subseteq \mathbb{R}^n$ is *linearly independent* if none of its members is a proper linear combination of the vectors in S; the *rank* of S, denoted by $rank(S)$, is the cardinality of the largest linearly independent subset of S.

A vector $x \in \mathbb{R}^n$ is an *affine combination* of the vectors $v^1, \ldots, v^k \in \mathbb{R}^n$, if there are k real numbers $\lambda^1, \ldots, \lambda^k$ such that $x = \sum_{h=1}^{k} \lambda^h v^h$ and $\sum_{h=1}^{k} \lambda^h = 1$. The *affine hull* of a non-empty set $S \subseteq \mathbb{R}^n$, denoted by $aff(S)$, is the set of all the vectors in $\mathbb{R}^n$ that are affine combinations of finitely many vectors of S.

Systems of Linear Inequalities. If $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, and $x = [x_1 \ldots x_n]$ is a vector of n variables, then the set $\{A_{i,\bullet} x \leq b_i \mid i \in \{1, \ldots, m\}\}$ is called a *system of linear inequalities* and is shortly denoted by $Ax \leq b$. Any vector $\bar{x} \in \mathbb{R}^n$ such that $A\bar{x} \leq b$ (i.e., $A_{i,\bullet}\bar{x} \leq b_i$, for each $i \in \{1, \ldots, m\}$) is a *(feasible) solution* to the system. The set of all solutions is a *polyhedron*, which will be denoted by $\Omega(Ax \leq b)$ in the following.

A polyhedron $\Omega(Ax \leq b)$ is *bounded* if there is a real number $k \in \mathbb{R}$ such that, for each $x \in \Omega(Ax \leq b)$ and each $j \in \{1, \ldots, n\}$, it holds that $-k \leq x_j \leq k$. A bounded polyhedron is also called a *polytope*.

Example 3.1 (Imputations of Coalitional Games)**.** Given a coalitional game $\mathcal{G} = \langle N, v \rangle$, the set $X(\mathcal{G})$ of all imputations is a polytope. Indeed, $X(\mathcal{G})$ can be characterized as the set of all the solutions to the follow system of linear inequalities:

$$A(\mathcal{G})x \leq b(\mathcal{G}) = \begin{cases} -x(N) \leq -v(N) \\ x(N) \leq v(N) \\ -x_i \leq -v(\{i\}), \ \forall i \in N \end{cases}$$

where the first two inequalities enforce the efficiency of x, i.e., $x(N) = v(N)$, while the remaining ones enforce the fact that x is individually rational. ◁

Let $Ax \leq b$ be a system of linear inequalities. For a vector $\bar{x} \in \Omega(Ax \leq b)$, we denote by $act(\bar{x}) = \{`A_{i,\bullet}x \leq b_i' \mid A_{i,\bullet}\bar{x} = b_i\}$ the set of the inequalities that are satisfied as equalities at $\bar{x}$, and we say that they are *active* at $\bar{x}$. For any inequality $`A_{i,\bullet}x \leq b_i'$, the vector $A_{i,\bullet}^T$ is called its *characteristic vector*. For operations on sets of vectors, by slightly abusing notation, we freely use inequalities to mean their characteristic vectors. Thus, for a set of inequalities I in the system, $rank(I)$ denotes $rank(\{A_{i,\bullet}^T \mid `A_{i,\bullet}x \leq b_i' \in I\})$. A vector $\bar{x} \in \Omega(Ax \leq b)$ is a *basic solution* if $rank(act(\bar{x})) = n$. It is well-known that basic solutions can be geometrically interpreted as the *vertices* of the polyhedron $\Omega(Ax \leq b)$, and that every polytope has at least one basic solution.

For a system $Ax \leq b$, we say that the inequality $`A_{i,\bullet}x \leq b_i'$ is an *implied equality* if, for each $\bar{x} \in \Omega(Ax \leq b)$, $`A_{i,\bullet}x \leq b_i' \in act(\bar{x})$. We denote by $ie(Ax \leq b)$ the set of all the implied equalities of the system.

The affine hull $aff(Ax \leq b)$ of $\Omega(Ax \leq b)$ is known to coincide with the set $\{x \in \mathbb{R}^n \mid A_{i,\bullet}x = b_i, \text{ for each implied equality } `A_{i,\bullet}x \leq b_i' \in ie(Ax \leq b)\}$, which can be geometrically interpreted as the smallest affine subspace of $\mathbb{R}^n$ containing $\Omega(Ax \leq b)$.

The *dimension* of a polyhedron $\Omega(Ax \leq b)$ is -1 if it is empty; otherwise, it is the non-negative number $dim(Ax \leq b) = n - rank(ie(Ax \leq b))$.

Linear Programming. Let $Ax \leq b$, with $A \in \mathbb{R}^{m \times n}$ (and $b \in \mathbb{R}^m$), be a system of linear inequalities. A (feasible) solution $\bar{x} \in \Omega(Ax \leq b)$ is *optimal* w.r.t. a vector $c \in \mathbb{R}^n$ if its associated *value* $c^T\bar{x}$ is such that $c^T\bar{x} = \min\{c^T\hat{x} \mid \hat{x} \in \Omega(Ax \leq b)\}$.

The input to a *linear programming problem* is given as an expression of the following form, called a *linear program*:

$$\min c^T x \ \mid \ Ax \leq b.$$

By solving a linear program we mean computing an optimal solution $\bar{x} \in \Omega(Ax \leq b)$ w.r.t. c (or, alternatively, check that there is no solution, i.e., $\Omega(Ax \leq b) = \varnothing$, or that there is no optimal solution, i.e., for each

$\bar{x} \in \Omega(Ax \leq b)$, there is $\bar{x}' \in \Omega(Ax \leq b)$ such that $c^T\bar{x}' < c^T\bar{x}$). The fundamental theorem of linear programming states that, whenever a linear program has an optimal solution, it has an optimal solution that is a basic solution to $Ax \leq b$. In particular, note that if $\Omega(Ax \leq b)$ is a non-empty polytope, then the linear program has always an optimal solution.

In the paper we consider, as usual, linear programs such that $A \in \mathbb{Q}^{m\times n}$, $b \in \mathbb{Q}^m$, and $c \in \mathbb{Q}^n$, and we assume the (standard) encoding where such rational numbers are given in the fractional form. Then, the set of solutions of such a linear program forms a so-called *rational polyhedron* of $\mathbb{R}^n$, and the program can be solved in polynomial time (in the program size $||c|| + ||A|| + ||b||$) (see, e.g., [57]). In particular, if a solution exists and $\Omega(Ax \leq b)$ is a non-empty polytope, then an optimal basic solution that can be represented with polynomially many bits exists, too.

3.2 Cutting Polyhedra by Linear Programs

The linear programming approach for nucleolus computation consists of a succession of linear programs that monotonically shrinks the space of all (candidate) imputations until the nucleolus is singled out. Roughly speaking, at each step t, some coalitions cannot reduce their excess (dissatisfaction) according to the current imputation space, while other coalitions have some degree of freedom. Then, a linear program $\mathtt{LP}_t$ is evaluated to minimize the maximum excess over these latter coalitions. By using this excess, further inequalities are added to the program to cut the search space. The process continues until all excesses are eventually fixed and the imputation space consists of one point only, which is the nucleolus.

All algorithms in the literature for nucleolus computation come as variants of this scheme (see, e.g., [17, 25, 34, 35, 47, 51, 55, 60] and the references therein), whose idea is rooted in the procedure proposed by Kopelowitz [36] and in its geometrical interpretation provided by Maschler et al. [42]. The salient results in these two seminal papers are next discussed, by (basically) following the formalization of Granot et al. [25], Kern and Paulusma [34], and Paulusma [50].

Definition 3.2. Let $\mathcal{G} = \langle N, v\rangle$ be a coalitional game. Define $\mathbf{LP}(\mathcal{G})$ to be the sequence of linear programs $\{\mathtt{LP}_t(\mathcal{G}) \mid t \geq 1\}$ such that

$$\begin{array}{lll} \mathtt{LP}_t(\mathcal{G}) = \min \varepsilon \;\mid\; v(S) - x(S) \leq \varepsilon & & \forall S \subseteq N, S \notin \mathcal{F}_{t-1} \\ \qquad\qquad\quad v(S) - x(S) \leq \varepsilon_{t-1} & & \forall S \subseteq N, S \notin \mathcal{F}_{t-2} \\ \qquad\qquad\quad \vdots & & \\ \qquad\qquad\quad v(S) - x(S) \leq \varepsilon_2 & & \forall S \subseteq N, S \notin \mathcal{F}_1 \\ \qquad\qquad\quad v(S) - x(S) \leq \varepsilon_1 & & \forall S \subseteq N, S \notin \mathcal{F}_0 = \{\varnothing, N\} \\ \qquad\qquad\quad A(\mathcal{G})x \leq b(\mathcal{G}), \text{ i.e., } x \in X(\mathcal{G}), & & \end{array}$$

where, for each $r \in \{1, \ldots, t-1\}$, ε_r is the value of an optimal solution to $\mathtt{LP}_r(\mathcal{G})$ and $\mathcal{F}_r = \{S \subseteq N \mid x(S) = y(S)$, for each pair x, y such that (x, ε_r) and (y, ε_r) are optimal solutions to $\mathtt{LP}_r(\mathcal{G})\}$; and where $A(\mathcal{G})x \leq b(\mathcal{G})$ is the system defining the imputations of $\mathcal{G}$, by enforcing efficiency and individual rationality (as described in Example 3.1). □

Let us provide some intuition about $\mathbf{LP}(\mathcal{G})$, with the help of a running example. By means of the first program $\mathtt{LP}_1(\mathcal{G})$, having the form

$$\begin{array}{lll} \mathtt{LP}_1(\mathcal{G}) = \min \varepsilon \;\mid\; v(S) - x(S) \leq \varepsilon & & \forall S \subseteq N, S \notin \mathcal{F}_0 = \{\varnothing, N\} \\ \qquad\qquad\quad A(\mathcal{G})x \leq b(\mathcal{G}), \text{ i.e., } x \in X(\mathcal{G}), & & \end{array}$$

we look for all those imputations that minimize the maximum excess, and we actually compute the value ε_1 of this excess. In particular, the associated—say optimal—imputations are those in the set $\{x \in X(\mathcal{G}) \mid (x, \varepsilon_1)$ is an optimal solution to $\mathtt{LP}_1(\mathcal{G})\}$.

Example 3.3. Let $\mathcal{G} = \langle N, v\rangle$ be the coalitional game of Example 1.4. Then,

$$\begin{array}{ll} \mathtt{LP}_1(\mathcal{G}) = \min \varepsilon \;\mid\; v(\{a\}) - x(\{a\}) = 0 - x_a \leq \varepsilon \\ \qquad\qquad\quad v(\{b\}) - x(\{b\}) = 0 - x_b \leq \varepsilon \\ \qquad\qquad\quad v(\{c\}) - x(\{c\}) = 0 - x_c \leq \varepsilon \\ \qquad\qquad\quad v(\{a,b\}) - x(\{a,b\}) = 1 - x_a - x_b \leq \varepsilon \\ \qquad\qquad\quad v(\{a,c\}) - x(\{a,c\}) = -2 - x_a - x_c \leq \varepsilon \\ \qquad\qquad\quad v(\{b,c\}) - x(\{b,c\}) = 2 - x_b - x_c \leq \varepsilon \\ \qquad\qquad\quad A(\mathcal{G})x \leq b(\mathcal{G}), \text{ i.e., } x_a + x_b + x_c = 1, x_a \geq 0, x_b \geq 0, x_c \geq 0 \end{array}$$

Because of the inequality $2 - x_b - x_c \leq \varepsilon$, it can be checked (see also the discussion in Example 1.4) that the optimal value is $\varepsilon_1 = 1$, and that the imputations over which this (excess) value is obtained are those in the set $\{x \in X(\mathcal{G}) \mid x_b + x_c = 1\}$. $\lhd$

By means of the second program, having the form

$$\begin{array}{rll} \mathtt{LP}_2(\mathcal{G}) = \min \varepsilon \;\mid\; & v(S) - x(S) \leq \varepsilon & \forall S \subseteq N, S \notin \mathcal{F}_1 \\ & v(S) - x(S) \leq \varepsilon_1 & \forall S \subseteq N, S \notin \mathcal{F}_0 = \{\varnothing, N\} \\ & A(\mathcal{G})x \leq b(\mathcal{G}), \text{ i.e., } x \in X(\mathcal{G}), & \end{array}$$

we then focus on the set $\{x \in X(\mathcal{G}) \mid (x, \varepsilon_1) \text{ is an optimal solution to } \mathtt{LP}_1(\mathcal{G})\}$. Indeed, notice that the set of all solutions to $\mathtt{LP}_2(\mathcal{G})$ coincides with the set of all optimal solutions to $\mathtt{LP}_1(\mathcal{G})$. Over this set, we put aside those coalitions (in $\mathcal{F}_1$) whose excess is *constant*, and we minimize the maximum excess over the remaining coalitions, in order to single out those imputations minimizing the second maximum excess.

Example 3.4. Consider again Example 3.3, and note that $\mathcal{F}_1$ consists of the coalitions $\{a, b, c\}$, $\{b, c\}$ and $\{a\}$. In particular observe that, for each imputation x in the set $\{x \in X(\mathcal{G}) \mid x_b + x_c = 1\}$, we have that $x_a + x_b + x_c = 1$, $x_b + x_c = 1$ and, hence, $x_a = 0$. Therefore, the second program is as follows:

$$\begin{array}{rl} \mathtt{LP}_2(\mathcal{G}) = \min \varepsilon \;\mid\; & v(\{b\}) - x(\{b\}) = 0 - x_b \leq \varepsilon \\ & v(\{c\}) - x(\{c\}) = 0 - x_c \leq \varepsilon \\ & v(\{a, b\}) - x(\{a, b\}) = 1 - x_a - x_b \leq \varepsilon \\ & v(\{a, c\}) - x(\{a, c\}) = -2 - x_a - x_c \leq \varepsilon \\ & v(\{a\}) - x(\{a\}) = 0 - x_a \leq \varepsilon_1 \\ & v(\{b\}) - x(\{b\}) = 0 - x_b \leq \varepsilon_1 \\ & v(\{c\}) - x(\{c\}) = 0 - x_c \leq \varepsilon_1 \\ & v(\{a, b\}) - x(\{a, b\}) = 1 - x_a - x_b \leq \varepsilon_1 \\ & v(\{a, c\}) - x(\{a, c\}) = -2 - x_a - x_c \leq \varepsilon_1 \\ & v(\{b, c\}) - x(\{b, c\}) = 2 - x_b - x_c \leq \varepsilon_1 \\ & A(\mathcal{G})x \leq b(\mathcal{G}), \text{ i.e., } x_a + x_b + x_c = 1, x_a \geq 0, x_b \geq 0, x_c \geq 0 \end{array}$$

Recall that the solutions to $\mathtt{LP}_2(\mathcal{G})$ are the optimal solutions to $\mathtt{LP}_1(\mathcal{G})$. Hence, for any (x, ε) satisfying the inequalities of $\mathtt{LP}_2(\mathcal{G})$, we known that $x \in X(\mathcal{G})$ and $x_b + x_c = 1$ hold, hence $x_a = 0$. Therefore, because of the inequality $1 - x_a - x_b = 1 - x_b \leq \varepsilon$, it is easily seen that the optimal value for $\mathtt{LP}_2(\mathcal{G})$ is $\varepsilon_2 = 0$, and that the imputations over which this (excess) value is obtained are those in the set $\{x \in X(\mathcal{G}) \mid x_a = 0 \wedge x_b = 1\}$. $\lhd$

After the second step, we get again a subset of the previous set of candidate imputations, as well as some new coalitions whose excess is constant over it. These coalitions in turn are put aside, and the process is repeated until a step t such that the set of the optimal solutions to $\mathtt{LP}_t(\mathcal{G})$ is a singleton, in fact coinciding with the nucleolus.

Example 3.5. In our running example, note that $(\bar{x}, \varepsilon_2)$, where $\varepsilon_2 = 0$ and $\bar{x}_a = 0$, $\bar{x}_b = 1$, and $\bar{x}_c = 0$, is the unique optimal solution to $\mathtt{LP}_2(\mathcal{G})$. As observed in Example 1.4, $\bar{x}$ is the nucleolus of $\mathcal{G}$. $\lhd$

In the above example, the process converged in just two steps. More generally, it can be noticed that, by putting aside all coalitions in $\mathcal{F}_{t-1}$, the dimension of the current set of optimal solutions decreases at least of 1 at each minimization step and, hence, in at most $|N|$ steps we obtain a region containing one point that is the nucleolus.

Formally, define $V_0(\mathcal{G}) = X(\mathcal{G})$ and, for each $r \in \{1, \ldots, t-1\}$, $V_r(\mathcal{G}) = \{x \mid (x, \varepsilon_r) \text{ is an optimal solution to } \mathtt{LP}_r(\mathcal{G})\}$. Note that

$$\varepsilon_r = \min\{\varepsilon \mid x \in V_{r-1}(\mathcal{G}) \wedge \; \forall S \subseteq N, S \notin \mathcal{F}_{r-1}, \; v(S) - x(S) \leq \varepsilon\}. \tag{1}$$

Then, the following holds.

Proposition 3.6. (CF. [34, 42, 50]) *Let $\mathcal{G} = \langle N, v \rangle$ be a coalitional game. Then, there is a natural number $t^* \leq |N|$ such that $V_{t^*}(\mathcal{G})$ consists of a unique vector that is the nucleolus $\mathcal{N}(\mathcal{G})$.*

REMARK. We point out that the the "original" procedure by Kopelowitz [36] is based on a sequence of linear programs different from the sequence $\mathbf{LP}(\mathcal{G})$ of Definition 3.2. Indeed, in the former case, at each step t, only the coalitions minimizing the maximum excess are excluded by the subsequent optimization steps. This is in

contrast with our approach, based on $\mathbf{LP}(\mathcal{G})$, where all coalitions having fixed excesses (not necessarily the minimum one), i.e., those belonging to $\mathcal{F}_t$, are excluded from the subsequent optimization steps. As a matter of fact, this subtle issue was source of confusion in the literature where the two approaches were often considered interchangeable, in spite of the observation by Maschler et al. [42], who argued (without formal statements and proofs) that the procedure based on the sequence $\mathbf{LP}(\mathcal{G})$ is a great enhancement over Kopelowitz's procedure.

In order to shed some light on this subject, we precisely characterize the computational difference between the two approaches. Indeed we formally prove in Appendix B that Kopelowitz's procedure needs $\Omega(2^{|N|})$ steps in the worst case, which is exponentially worse than the procedure based on $\mathbf{LP}(\mathcal{G})$, after Proposition 3.6. While the proof of this result is provided in the appendix for the sake of presentation, we believe it may be a useful reading for a deeper understanding of the computational issues behind the computation of the nucleolus. □

3.3 Dealing with Compact Games

Recall from Section 2 that deciding whether a payoff vector is the nucleolus is hard for the complexity class $\mathbf{\Delta_2^P}$ (even over the class of graph games). A major goal of the paper is to show that this result is in fact tight, by proving that the nucleolus can be computed in $\mathbf{F\Delta_2^P}$, the functional version of $\mathbf{\Delta_2^P}$, over every class of games encoded according to any polynomial-time compact representation. To this end, recall first that $\mathbf{F\Delta_2^P}$ is the class of all functions that can be computed by deterministic Turing transducers in polynomial time and by using an **NP** Turing machine as an oracle. Therefore, to show a membership result in $\mathbf{F\Delta_2^P}$ we must design a polynomial time algorithm (which thus may use at most polynomial space) that may call at each step an oracle to solve **NP** combinatorial tasks.

The basic idea is to use the linear programming approach suggested by Maschler et al. [42], that is, the sequence of linear programs of Definition 3.2, which allows us to compute the nucleolus of a given game $\mathcal{G}$ after a number of steps not greater than the number $|N|$ of game players. However, in order to get a polynomial-time algorithm (though with the possibility of calling oracles) two issues should be carefully taken into account while dealing with each program $\mathtt{LP}_t(\mathcal{G})$ in this sequence ($1 \leq t \leq |N|$):

- First, it might happen that exponentially many coalitions get a constant excess, so that it becomes unfeasible (in $\mathbf{F\Delta_2^P}$) to explicitly store the whole set $\mathcal{F}_t$ to be used in the subsequent iteration. Therefore, the question comes into play about whether it is possible to "implicitly" represent the set of all coalitions with constant excesses, moreover in a way that checking membership in the set is still feasible in $\mathbf{F\Delta_2^P}$.
- Second, the system of linear inequalities in $\mathtt{LP}_t(\mathcal{G})$ involves exponentially many inequalities w.r.t. the number of players of $\mathcal{G}$ (and hence w.r.t. the size of the game encoding, in general). In this case, it is not difficult to foresee that there is no need to "materialize" these inequalities as each of them can, in fact, be computed in polynomial time based on the encoding of the compact game $\mathcal{G}$. On the other hand, the problem in this case is that the resulting system of linear inequalities is not given explicitly, but it is specified in its turn in succinct form. Hence, standard results on linear programming cannot be applied to analyze the complexity of reasoning with such a program. This calls for a theory of succinctly specified linear programs that we develop (for what is relevant to this paper) in Section 4.

In the rest of this section, the first issue will be addressed by introducing a suitable concept of *basis* over a system of linear inequalities.

Basis of Implied Equalities. Let $Ax \leq b$, with $A \in \mathbb{Q}^{m \times n}$ (and $b \in \mathbb{Q}^m$), be a system of linear inequalities. For each natural number $h \in \{1, \ldots, m\}$, let $[A_{h,\boldsymbol{\cdot}}, b_h]^T$ denote the vector in $\mathbb{Q}^{n+1}$ whose first n components are those of the characteristic vector $A^T_{h,\boldsymbol{\cdot}}$, and the last component is b_h.

Definition 3.7. A *basis of the implied equalities* of a given system $Ax \leq b$ is any set $\mathcal{B} \subseteq ie(Ax \leq b)$ satisfying the following two conditions:

(i) $|\mathcal{B}| = rank(\mathcal{B})$;

(ii) $|\mathcal{B}| = rank(ie(Ax \leq b))$. □

In words, a basis is a maximal set of implied equalities whose characteristic vectors are linearly independent. The intuition underlying this notion is that a basis provides us with a convenient method to encode the whole set of all implied equalities, in a way that membership to this latter set can be reduced to checking linear independence among suitably defined vectors. This is formalized next.

Theorem 3.8. *Let $\mathcal{B}$ be a basis of the implied equalities of a system of linear inequalities $Ax \leq b$. Then, '$A_{\bar{i},\boldsymbol{\cdot}}x \leq b_{\bar{i}}$' $\in ie(Ax \leq b)$ if, and only if,*

$$rank(\{[A_{i,\boldsymbol{\cdot}}, b_i]^T \mid \text{`}A_{i,\boldsymbol{\cdot}}x \leq b_i\text{'} \in \mathcal{B}\}) = rank(\{\ [A_{\bar{i},\boldsymbol{\cdot}}, b_{\bar{i}}]^T\} \cup \{[A_{i,\boldsymbol{\cdot}}, b_i]^T \mid \text{`}A_{i,\boldsymbol{\cdot}}x \leq b_i\text{'} \in \mathcal{B}\}).$$

Proof. To begin with, assume that '$A_{\bar{i},\bullet}x \leq b_{\bar{i}}$' $\in ie(Ax \leq b)$. Since $\mathcal{B} \subseteq ie(Ax \leq b)$ and $|\mathcal{B}| = rank(\mathcal{B}) = rank(ie(Ax \leq b))$, we immediately have that $|\mathcal{B}| = rank(\mathcal{B} \cup \{`A_{\bar{i},\bullet}x \leq b_{\bar{i}}'\})$. Therefore, the characteristic vector $A^T_{\bar{i},\bullet}$ can be expressed as a linear combination $\sum_{`A_{i,\bullet}x \leq b_i' \in \mathcal{B}} \lambda_i A^T_{i,\bullet}$ of the characteristic vectors of the inequalities in $\mathcal{B}$. Because all inequalities in $\mathcal{B}$ are, in fact, implied equalities, the system must satisfy the additional equality '$A_{\bar{i},\bullet}x = \sum_{`A_{i,\bullet}x \leq b_i' \in \mathcal{B}} \lambda_i b_i$', which entails that the latter summation must give $b_{\bar{i}}$, for otherwise '$A_{\bar{i},\bullet}x \leq b_{\bar{i}}$' $\notin ie(Ax \leq b)$. We thus obtained that $[A_{\bar{i},\bullet}, b_{\bar{i}}]^T$ can be written as a linear combination of the vectors $\{[A_{i,\bullet}, b_i]^T \mid `A_{i,\bullet}x \leq b_i' \in \mathcal{B}\}$.

To conclude, let us show that $rank(\{[A_{i,\bullet}, b_i]^T \mid `A_{i,\bullet}x \leq b_i' \in \mathcal{B}\}) = rank(\{\ [A_{\bar{i},\bullet}, b_{\bar{i}}]^T\} \cup \{[A_{i,\bullet}, b_i]^T \mid `A_{i,\bullet}x \leq b_i' \in \mathcal{B}\})$ implies '$A_{\bar{i},\bullet}x \leq b_{\bar{i}}$' $\in ie(Ax \leq b)$. Indeed, because of the above equality, we can rewrite $[A_{\bar{i},\bullet}, b_{\bar{i}}]^T$ as a linear combination of the vectors in $\{[A_{i,\bullet}, b_i]^T \mid `A_{i,\bullet}x \leq b_i' \in \mathcal{B}\}$. That is, $[A_{\bar{i},\bullet}, b_{\bar{i}}]^T = \sum_{`A_{i,\bullet}x \leq b_i' \in \mathcal{B}} \lambda_i [A_{i,\bullet}, b_i]^T$, for suitable coefficients λ_i. In particular, $A_{\bar{i},\bullet} = \sum_{`A_{i,\bullet}x \leq b_i' \in \mathcal{B}} \lambda_i A_{i,\bullet}$ and $b_{\bar{i}} = \sum_{`A_{i,\bullet}x \leq b_i' \in \mathcal{B}} \lambda_i b_i$. Again, since $\mathcal{B}$ is a set of implied equalities, for each $\bar{x} \in \Omega(Ax \leq b)$ and for each '$A_{i,\bullet}x \leq b_i$' $\in \mathcal{B}$, $A_{i,\bullet}\bar{x} = b_i$ holds. Therefore, for each $\bar{x} \in \Omega(Ax \leq b)$, $A_{\bar{i},\bullet}\bar{x} = \sum_{`A_{i,\bullet}x \leq b_i' \in \mathcal{B}} \lambda_i A_{i,\bullet}\bar{x} = \sum_{`A_{i,\bullet}x \leq b_i' \in \mathcal{B}} \lambda_i b_i = b_{\bar{i}}$ holds, and hence '$A_{\bar{i},\bullet}x \leq b_{\bar{i}}$' $\in ie(Ax \leq b)$. □

Example 3.9. Let $Ax \leq b$, with $A \in \mathbb{Q}^{10\times 3}$, be a system of linear inequalities and assume that $\mathcal{B}_2 = \{`x_1 + x_2 \leq 1', `x_2 + x_3 \leq 1'\}$ is a basis of its implied equalities. Therefore, $dim(Ax \leq b) = 3 - rank(\mathcal{B}_2) = 3 - |\mathcal{B}_2| = 1$.

Moreover, assume that the system contains the following inequalities $\alpha =$ '$2x_1 + 3x_2 + x_3 \leq 3$' and $\beta =$ '$x_1 + 2x_2 + x_3 \leq 3$'. Note that α can be written as a linear combination of the implied equalities in $\mathcal{B}_2$. Hence, by Theorem 3.8, α is an implied equality, too. The same does not apply to β, i.e., β is not a linear combination of the implied equalities in $\mathcal{B}_2$. However, the characteristic vector of β can be written as a linear combination of the characteristic vectors associated with the inequalities in $\mathcal{B}_2$. Since $\mathcal{B}_2$ is a basis of implied equalities, this means that every solution to the system must satisfy the equality '$x_1 + 2x_2 + x_3 = 1 + 1 = 2$'. In other words, β is not an implied equality of the system, but its characteristic vector identifies an (implied) equality, though with the constant term 2 instead of the value 3 occurring in the inequality β. ◁

Application to Compact Coalitional Games. The property of bases of inequalities described in the previous example is crucial for the purpose of this paper. The idea is indeed to use bases for representing in a succinct way all coalitions having constant excesses in the sequence $\mathbf{LP}(\mathcal{G})$ of Definition 3.2, as formalized by the following result.

Let $\mathcal{G} = \langle N, v\rangle$ be a coalitional game. For each coalition $S \subseteq N$, we denote by $\mathbb{1}_S$ the $|N|$-dimensional vector whose i-th component, for each $i \in \{1, \ldots, |N|\}$, is 1 (resp., 0) if $i \in S$ (resp., $i \notin S$). Moreover, consider the linear program $\mathtt{LP}_t(\mathcal{G})$, for $t \geq 1$, and let $A[t]x \leq b[t]$ denote the system of linear inequalities obtained from it by setting $\varepsilon = \varepsilon_t$, with ε_t denoting the value of variable ε in any optimal solution to $\mathtt{LP}_t(\mathcal{G})$. Note that $\Omega(A[t]x \leq b[t])$ coincides with $V_t(\mathcal{G})$. In addition, observe that each inequality '$A_{i,\bullet}[t]x \leq b_i[t]$' in this system identifies a coalition, in the sense that '$A_{i,\bullet}[t]x \leq b_i[t]$' is of the form $\alpha \mathbb{1}_S^T x \leq \beta$, for some coalition S, and where $\alpha \in \{1, -1\}$ and $\beta \in \mathbb{Q}$.

Theorem 3.10. *Let $\mathcal{G} = \langle N, v\rangle$ be a coalitional game such that $X(\mathcal{G}) \neq \varnothing$, let $t \geq 1$ be a natural number, let $\mathcal{B}_t$ be a basis of the implied equalities of the system $A[t]x \leq b[t]$, and let $S \subseteq N$ be a coalition. Then, the following are equivalent:*

(1) $S \in \mathcal{F}_t$, where $\mathcal{F}_t = \{S \subseteq N \mid x(S) = y(S)$, for each pair x, y such that (x, ε_t) and (y, ε_t) are optimal solutions to $\mathtt{LP}_t(\mathcal{G})\}$ (cf. Definition 3.2);

(2) $rank(\mathcal{B}_t) = rank(\mathcal{B}_t \cup \{\mathbb{1}_S\})$.[7]

Proof. We first show that (2) ⇒ (1). Assume that $rank(\mathcal{B}_t) = rank(\mathcal{B}_t \cup \{\mathbb{1}_S\})$, and let $\{W_1, \ldots, W_q\}$ be the characteristic vectors of the inequalities in $\mathcal{B}_t$. Then, there exist real numbers $\lambda_1, \ldots, \lambda_q$ such that $\mathbb{1}_S = \sum_{i=1}^q \lambda_i W_i$. This equation implies that

$$x(S) = \mathbb{1}_S^T x = \sum_{i=1}^{q} (\lambda_i {W_i}^T x).$$

Observe that, since $\mathcal{B}_t$ is a basis of the implied equalities of $A[t]x \leq b[t]$, ${W_i}^T x$ is a constant over each point $x \in V_t(\mathcal{G})$, for each $i \in \{1, \ldots, q\}$. Equivalently, we have that ${W_i}^T \bar{x} = {W_i}^T \bar{y}$ for each pair $\bar{x}, \bar{y}$ in $V_t(\mathcal{G}) = \Omega(A[t]x \leq b[t])$ and for each $i \in \{1, \ldots, q\}$.

[7] Recall that, in vector operations such as the rank, we identify inequalities with their characteristic vectors, so that $\mathcal{B}_t$ denotes the set of characteristic vectors of the basis inequalities.

Therefore, from the fact that $x(S) = \sum_{i=1}^{q}(\lambda_i {W_i}^T x)$, we conclude that $\bar{x}(S) = \bar{y}(S)$, for each pair $\bar{x}, \bar{y}$ in $V_t(\mathcal{G})$. This immediately entails the statement $S \in \mathcal{F}_t$, by recalling that $\mathcal{F}_t = \{S \subseteq N \mid x(S) = y(S)$, for each pair x, y such that (x, ε_r) and (y, ε_r) are optimal solutions to $\mathtt{LP}_r(\mathcal{G})\} = \{S \subseteq N \mid x(S) = y(S)$, for each pair x, y in $V_t(\mathcal{G})\}$.

Second, we show that (1) $\Rightarrow$ (2). Note first that $V_t(\mathcal{G})$ has the following form:

$$V_t(\mathcal{G}) = \{x \mid \forall k \in \{1, \ldots, t\}, \forall S \subseteq N, S \notin \mathcal{F}_{k-1},\ v(S) - x(S) \leq \varepsilon_k\},$$

and that $V_t(\mathcal{G}) = \Omega(A[t]x \leq b[t])$.

Let $\bar{S} \in \mathcal{F}_t$. Then, this coalition has a constant excess, say $\varepsilon_{\bar{S}}$, over all points in $V_t(\mathcal{G})$, hence over all solutions to the system $A[t]x \leq b[t]$. Consider now the system of linear inequalities $\bar{A}[t]x \leq \bar{b}[t]$ obtained from $A[t]x \leq b[t]$ by adding the inequality $\mathbb{1}_{\bar{S}}^T x \leq \varepsilon_{\bar{S}} - v(\bar{S})$. From the hypothesis on coalition $\bar{S}$, $\forall \bar{x} \in \Omega(A[t]x \leq b[t]), \mathbb{1}_{\bar{S}}^T \bar{x} = \varepsilon_{\bar{S}} - v(\bar{S})$, and therefore the following two facts hold for the new inequality $\mathbb{1}_{\bar{S}}^T x \leq \varepsilon_{\bar{S}} - v(\bar{S})$:

1. it does not change the set of solutions, i.e., $\Omega(A[t]x \leq b[t]) = \Omega(\bar{A}[t]x \leq \bar{b}[t])$, and
2. it is in fact an implied equality in $ie(\bar{A}[t]x \leq \bar{b}[t])$.

The above Fact (1) entails that $dim(A[t]x \leq b[t]) = dim(\bar{A}[t]x \leq \bar{b}[t])$ and thus, by definition of dimension, that $rank(ie(A[t]x \leq b[t])) = rank(ie(\bar{A}[t]x \leq \bar{b}[t]))$. Because the former term is equal to $rank(\mathcal{B}_t)$, we get

$$rank(\mathcal{B}_t) = rank(ie(\bar{A}[t]x \leq \bar{b}[t])).$$

However, $\mathcal{B}_t \subseteq ie(A[t]x \leq b[t]) \subseteq ie(\bar{A}[t]x \leq \bar{b}[t])$ clearly holds. Therefore, $\mathcal{B}_t$ is also a basis for $ie(\bar{A}[t]x \leq \bar{b}[t])$, and the characteristic vector of any of these inequalities can be rewritten as a linear combination of the characteristic vectors of inequalities in $\mathcal{B}_t$. From Fact (2), this applies to the characteristic vector $\mathbb{1}_{\bar{S}}$, too. □

We leave the section by noticing that it is known in the literature that a small set of coalitions (often called *essentials*) suffices to determine the nucleolus of any given coalitional game (see, e.g., [9, 53, 54, 62]). However, this is generally not helpful to compute the nucleolus, as no method has been described to compute such coalitions (without having already the nucleolus), and it is not clear how to identify them within the complexity bounds we are interested in. In abstract terms, the above result combined with the results presented in the subsequent sections can be viewed as leading to a method that synergically computes the nucleolus and its (small) set of characterizing coalitions.

4 Reasoning on Succinctly Specified Linear Programs

In this section, we analyze the complexity of a number of reasoning tasks about succinctly specified linear programs. On the one hand, these results play a crucial role in establishing membership in $\mathbf{F\Delta_2^P}$ of the nucleolus computation. On the other hand, they are interesting in their own, as the framework of succinct linear programming smoothly fits many real-world settings modeled as programs involving exponentially many inequalities w.r.t. the number of variables (or, equivalently, by duality theorems in linear programming, where problems are considered with exponentially many variables w.r.t. the number of inequalities).

Example 4.1. Let $G = (N, E)$ be a graph, and let s and t be two nodes in N. Consider a problem where we need to assign non-negative weights to the edges of G such that the weights of edges in any path connecting s and t sum at least 1, but the total weight of E does not exceed a given number K.

This problem and variants thereof frequently occur in the design of flow networks, and can be modeled via linear programming. In fact, the set of all solutions to this problem coincides with the set of all solutions to the following system of linear inequalities, where $\mathcal{P}$ denotes the set of all paths connecting s and t (with a path being just viewed below as a set of edges) and where x_e is the weight assigned to edge e:

$$\begin{cases} \sum_{e \in E} x_e \leq K & \\ -x_e \leq 0 & \forall e \in E \\ -\sum_{e \in p} x_e \leq -1 & \forall p \in \mathcal{P} \end{cases}$$

Note that the system is defined over $|E|$ variables, while the number $|\mathcal{P}|$ of all paths connecting s and t is exponentially larger than $|E|$, in general. ◁

4.1 Problems, Computational Setting, and Overview of the Results

We consider the following decision problems, all of them defined over a given system $Ax \leq b$ of linear inequalities, with $A \in \mathbb{Q}^{m \times n}$, plus further possible inputs:

MEMBERSHIP: Given a system $Ax \leq b$, with $A \in \mathbb{Q}^{m \times n}$, and given a vector $\bar{x} \in \mathbb{Q}^n$, is $\bar{x} \in \Omega(Ax \leq b)$?

NONEMPTINESS: Given a system $Ax \leq b$, with $A \in \mathbb{Q}^{m \times n}$, is $\Omega(Ax \leq b) \neq \varnothing$?

DIMENSION: Given a system $Ax \leq b$, with $A \in \mathbb{Q}^{m \times n}$, and a non-negative number $k \leq n+1$, is $dim(Ax \leq b) \leq n - k$?

Moreover, we shall consider the following computational problems:

IMPLIEDEQUALITIES-COMPUTATION: Given a system $Ax \leq b$, with $A \in \mathbb{Q}^{m \times n}$ and $\Omega(Ax \leq b) \neq \varnothing$, compute a basis $\mathcal{B}$ of the implied equalities of $Ax \leq b$.

OPTIMALVALUE-COMPUTATION: Given a system $Ax \leq b$, with $A \in \mathbb{Q}^{m \times n}$ and where $\Omega(Ax \leq b) \neq \varnothing$ is a non-empty polytope,[8] and given a vector $c \in \mathbb{Q}^n$, compute the value $\min\{c^T \hat{x} \mid \hat{x} \in \Omega(Ax \leq b)\}$.

SOLUTION-COMPUTATION: Given a system $Ax \leq b$, with $A \in \mathbb{Q}^{m \times n}$ and where $\Omega(Ax \leq b) \neq \varnothing$ is a non-empty polytope, compute a vector $x \in \Omega(Ax \leq b)$.

OPTIMALSOLUTION-COMPUTATION: Given a system $Ax \leq b$, with $A \in \mathbb{Q}^{m \times n}$ and where $\Omega(Ax \leq b) \neq \varnothing$ is a non-empty polytope, and given a vector $c \in \mathbb{Q}^n$, compute a vector $\bar{x} \in \Omega(Ax \leq b)$ such that $c^T \bar{x} = \min\{c^T \hat{x} \mid \hat{x} \in \Omega(Ax \leq b)\}$.

In the classical computational setting of linear programming, it is assumed that all the coefficients involved in the definition of the systems of linear inequalities are explicitly provided as input. Under this standard representation scheme, all the above reasoning tasks are known to be feasible in polynomial time (see e.g., [28, 49, 57]).

Next, we re-consider these problems in a different setting, where the input system $Ax \leq b$ is given in a compact form. For instance, we have noticed that the linear program in Example 4.1, is univocally determined by the underlying graph, which in fact can be used as a succinct implicit way to encode all the (exponentially many) paths between s and t.

The implicit encoding approach we propose below shares the spirit of the compact game encoding discussed in Section 2, in that exponentially many elements (here, the inequalities) are encoded via some polynomial space representation associated with a polynomial time function to deal with it.

Definition 4.2. A *compact representation* Λ for systems of linear inequalities of the form $Ax \leq b$, with $A \in \mathbb{Q}^{m \times n}$ and $m \in O(\exp(n))$, defines a class $\mathcal{C}(\Lambda)$ by means of two functions γ^Λ and $\mathcal{L}^\Lambda$. For each system $Ax \leq b \in \mathcal{C}(\Lambda)$,

(1) $\gamma^\Lambda(Ax \leq b)$ is an object, whose size is not smaller than n, that encodes the system (roughly, it is a compact representation that contains at least one bit for each variable occurring in the system); and

(2) $\mathcal{L}^\Lambda$ is a polynomial-time function that, given the encoding $E = \gamma^\Lambda(Ax \leq b)$ and any natural number i, computes the coefficients of some inequality associated with i, if any. More precisely, $\mathcal{L}^\Lambda(E, i) = (A_{i',\boldsymbol{.}}, b_{i'})$ for some $i' \leq m$, or the empty output if no inequality is associated with i. Observe that this entails that the size of such coefficients is bounded by a polynomial function of $||E||$. It is required that the size of any input to $\mathcal{L}^\Lambda$ is bounded by some polynomial of $||E||$ (i.e., the input indices i are at most exponentially larger than $||E||$), and that all inequalities of the system are in its codomain (i.e., the encoding is complete: every inequality is associated with some input to $\mathcal{L}^\Lambda$).

Whenever a compact representation Λ is understood, we write for short $\gamma(Ax \leq b)$ and $\mathcal{L}(i)$ instead of $\gamma^\Lambda(Ax \leq b)$ and $\mathcal{L}^\Lambda(\gamma^\Lambda(Ax \leq b), i)$, respectively. Moreover, we often abuse the notation and use $\mathcal{L}(i)$ to denote the inequality $A_{i',\boldsymbol{.}} x \leq b_{i'}$ associated with number i via $\mathcal{L}$, rather than just the coefficients $(A_{i',\boldsymbol{.}}, b_{i'})$. □

Example 4.3. Consider again the class of systems of inequalities encoding the flow problem in Example 4.1. Every system $Ax \leq b$ in this class is such that $A \in \mathbb{Q}^{m \times n}$, with n being the number of edges of some graph $G = (N, E)$ and $m = |\mathcal{P}| + n + 1$, where $\mathcal{P}$ is the set of all paths connecting two distinguished nodes $s, t \in N$, w.l.o.g., the first two nodes in N (so that we need no further effort to distinguish them).

[8] For the purpose of this paper, it suffices to study these computational problems over linear programs whose solutions form non-empty polytopes; however, our techniques might easily be extended to deal with arbitrary (possibly empty or unbounded) polyhedra.

This class has a compact encoding FLOW, defined as follows. Each system in $\mathcal{C}(\texttt{FLOW})$ is encoded as a pair $F = (G, K)$, where K is the bound on the total weight and $G = (N, E)$ is just (a suitable encoding of) the graph associated with each instance of the flow problem. Moreover, $\mathcal{L}$ is a polynomial time function taking as its input the encoding F and any (natural) binary number $i \leq 2^n + n + 1$. Each number i up to 2^n is interpreted as a bit vector encoding the set of edges $S_i = \{e \in E \mid \text{ the } e\text{-th bit of } i \text{ is } 1\}$, which represents a candidate path from s to t. The numbers above 2^n encode the further $n + 1$ inequalities of the system. Moreover, we assume that edges are encoded as natural numbers (from 1 to n). Then, the function $\mathcal{L}$ behaves as follows:

- $\mathcal{L}(F, i) = -\sum_{e \in S_i} x_e \leq -1$, if $i \leq 2^n$ and the edges in S_i encode a path in $\mathcal{P}$;
- $\mathcal{L}(F, i) = -x_e \leq 0$, if $i = 2^n + e$ (e is an edge in E);
- $\mathcal{L}(F, i) = \sum_{e \in E} x_e \leq K$, if $i = 2^n + n + 1$; and
- $\mathcal{L}(F, i)$ returns the empty output in all other cases, i.e., whenever $i \leq 2^n$ but the edges in S_i does not encode a path in $\mathcal{P}$.

Note that $\mathcal{C}(\texttt{FLOW})$ is the class of systems described in Example 4.1, and that FLOW is in fact a compact representation. In particular, observe that $||F|| = ||G|| + ||K|| \geq |E| = n$, and that $\mathcal{L}$ can be evaluated in polynomial time by a deterministic Turing machine. ◁

The reader that is familiar with linear programming might have already recognized that, while being encoded via a system over exponentially many inequalities, the above problem is actually solvable in polynomial time by using the *ellipsoid method* with a polynomial-time *separation oracle* (see, e.g., Grötschel et al. [28]). Here, we recall that a separation oracle is an algorithm that given a point x, either establishes the feasibility of x or produces a violated inequality. In Example 4.1, the following polynomial-time algorithm is a separation oracle: First, for every $2^n + 1 \leq i \leq 2^n + n + 1$, it can explicitly check whether the given weight assignment x satisfies the inequality identified by $\mathcal{L}(F, i)$. If this is not the case, then a violated inequality is computed as its output. Otherwise, i.e., if all such inequalities are satisfied by x, then the algorithm continues by computing a path p from s to t having minimum weight according to x. If the weight of path p is smaller than 1, then x is not feasible and the inequality associated with p is violated. In this case, the algorithm outputs the inequality $\mathcal{L}(F, i)$, where i is the bit vector encoding the set of edges S_i occurring in p. Otherwise, x is feasible, as the weight of all other paths in $\mathcal{P}$ cannot be less than the weight of p.

Of course, there are classes of systems having no polynomial-time separation oracle. We are interested in finding upper bounds for the evaluation of any classes of systems having a compact representation, including those classes encoding hard combinatorial problems such as most problems about (compact) coalitional games (see, e.g., [26, 27] and the references therein).

Example 4.4 (Core non-emptiness in Graph Games)**.** Consider the class of graph games $\mathcal{C}(\texttt{GGR})$, and the class of systems of inequalities encoding the problem of checking whether the core of any given game $\mathcal{G}$ in this class is not empty. Recall that, if the set of players of $\mathcal{G}$ is N, such a system consists of the following $2^{|N|} + 1$ inequalities: for each coalition $S \subseteq N$, $-\sum_{i \in S} x_i \leq -v(S)$; and $\sum_{i \in N} x_i \leq v(N)$.

This class of systems has a compact encoding CoreGG, defined as follows. Each system in $\mathcal{C}(\texttt{CoreGG})$ is encoded as a weighted graph $G = \langle (N, E), w \rangle$, and $\mathcal{L}$ is a polynomial time function taking as its input G and any (natural) binary number $i \leq 2^{|N|} + 1$. In a similar way as in the previous example, any value of i up to $2^{|N|}$ is interpreted as a bit vector encoding the coalition of players $S_i = \{p \in N \mid \text{ the } p\text{-th bit of } i \text{ is } 1\}$, while the last value $2^{|N|} + 1$ encodes the last inequality of the system (the efficiency constraint for the grand-coalition). Formally, define

- $\mathcal{L}(G, i) = -\sum_{i \in S} x_i \leq -v(S_i)$, if $i \leq 2^{|N|}$ (where $v(S_i) = \sum_{e \in E | e \subseteq S_i} w(e)$); and
- $\mathcal{L}(G, i) = \sum_{i \in N} x_i \leq v(N)$, if $i = 2^{|N|} + 1$.

Note that CoreGG is a compact representation. In particular, $\mathcal{L}$ can be evaluated in polynomial time by a deterministic Turing machine. Moreover, any system in the class $\mathcal{C}(\texttt{CoreGG})$ has some solution if, and only if, its associated game has a non-empty core. ◁

We are now ready to study such succinctly specified systems of linear inequalities. For each problem P illustrated at the beginning of this section, let Succint-P be its succinct variant where the input system of linear inequalities is encoded via some fixed compact representation $\mathcal{C}(\Lambda)$. A summary of our complexity results for these problems is illustrated in Figure 3.

Note that completeness results are given in Figure 3 for the three decision problems we have considered. In particular, membership results hold over every class $\mathcal{C}(\Lambda)$, with Λ being any compact representation, whereas

Problem	Result
Succint-Membership	co-**NP**-complete
Succint-NonEmptiness	co-**NP**-complete
Succint-Dimension	**NP**-complete
Succint-ImpliedEqualities-Computation	in $\mathbf{F\Delta_2^P}$
Succint-OptimalValue-Computation	in $\mathbf{F\Delta_2^P}$
Succint-Solution-Computation	in $\mathbf{F\Delta_2^P}$
Succint-OptimalSolution-Computation	in $\mathbf{F\Delta_2^P}$

Figure 3: Summary of Complexity Results in Section 4.

hardness results[9] mean that there are classes of instances that encode hard problems (of course, there are also "easy" classes, such as the one described in Example 4.3).

Theorem 4.5. *There is a class of systems of linear inequalities (encoded in some compact way) over which* Succint-Membership *and* Succint-NonEmptiness *are* co-**NP**-*hard, while* Succint-Dimension *is* **NP**-*hard.*

Proof. Consider the class $\mathcal{C}(\texttt{CoreGG})$ in Example 4.4 and let $Ax \leq b \in \mathcal{C}(\texttt{CoreGG})$, with $A \in \mathbb{Q}^{(2^n+1)\times n}$, be a system succinctly encoded as a weighted graph G. Every solution to such a system is clearly an imputation belonging to the core of the graph game encoded by G. Then, the co-**NP**-hardness of Succint-NonEmptiness and Succint-Membership immediately follow, respectively, from the co-**NP**-hardness of checking non-emptiness of the core and of checking whether a point belongs to the core, over the class of graph games [14]. In the same way, the **NP**-hardness of Succint-Dimension follows: $(G, n+1)$ is a "yes" instance of Succint-Dimension if, and only if, the core of the game encoded by G is empty. □

In the rest of the section, we detail the proofs for the membership results illustrated in Figure 3. In fact, while complexity issues arising with specific forms of succinctly specified linear programs have been already analyzed in the literature (e.g., to derive tractability results when a separation oracle is available), the analysis of the general theory was missing so far. In particular, proofs of the problems Succint-OptimalValue-Computation, Succint-Solution-Computation, and Succint-OptimalSolution-Computation use standard arguments adjusted to fit the general framework, and hence they are reported in Appendix C, for the sake of completeness only. Proofs of the remaining problems require novel technical elaborations, which are detailed in the following. Eventually, we also stress that a number of complexity results are known for different settings of succinct linear programming that cannot be recast in terms of Definition 4.2, e.g., when coefficients involved in the linear programs are encoded as circuits (see [21]).

4.2 Membership Results for Decision Problems

Note that Figure 3 also reports the complexity of the four computational problems that we have considered, and that are used as procedures in the proposed algorithm for the computation of the nucleolus. In their turns, such procedures, which belong to $\mathbf{F\Delta_2^P}$, use **NP** oracles based on the following membership results for the decision problems.

Theorem 4.6. *Let* Λ *be any compact representation for systems of linear inequalities. On the class* $\mathcal{C}(\Lambda)$*,* Succint-Membership *is in* co-**NP**.

Proof. Let $\gamma(Ax \leq b)$ be the encoding of a system $Ax \leq b \in \mathcal{C}(\Lambda)$, with $A \in \mathbb{Q}^{m\times n}$, provided as input to Succint-Membership, together with a vector $\bar{x} \in \mathbb{Q}^n$. We have to decide whether $\bar{x} \in \Omega(Ax \leq b)$. Note that the size of the input is $||\gamma(Ax \leq b)|| + ||\bar{x}||$.

Consider then the complementary problem of checking whether $\bar{x} \notin \Omega(Ax \leq b)$. This problem can be solved by guessing a natural number i', by computing $\mathcal{L}(i') = (A_{i,\bullet}, b_i)$, and by checking that $A_{i,\bullet}\bar{x} > b_i$. We claim that these tasks are feasible in polynomial time using a non-deterministic Turing machine (hence, the problem belongs to **NP**). Indeed, by Definition 4.2, it is sufficient to guess polynomially many bits, because any useful number i' is at most exponentially larger than the size of the system encoding.

Moreover, by the same definition, $\mathcal{L}(i') = (A_{i,\bullet}, b_i)$ can be computed in polynomial time w.r.t. $||\gamma(Ax \leq b)||$. Eventually, after that $(A_{i,\bullet}, b_i)$ is computed (whose encoding is polynomial w.r.t. $||\gamma(Ax \leq b)||$), we can trivially check whether $A_{i,\bullet}\bar{x} > b_i$ holds in time polynomial w.r.t. $||\gamma(Ax \leq b)|| + ||\bar{x}||$. □

[9]Hardness results are provided for the sake of completeness only, as they are not relevant w.r.t. our main goal of characterizing the complexity of nucleolus computation on compact games.

In order to deal with the complexity of Succint-NonEmptiness, we recall a classical combinatorial result about convex sets, which is due to Helly (see also [27], for a nice geometric interpretation of this result for systems of linear inequalities defininig cores of coalitional games).

Proposition 4.7 ([29, 52])**.** *Let $\mathcal{C} = \{c_1, \ldots, c_h\}$ be a collection of convex subsets of $\mathbb{R}^n$. If $\bigcap_{c_i \in \mathcal{C}} c_i = \varnothing$, then there is a non-empty collection $\mathcal{C}' \subseteq \mathcal{C}$ such that $|\mathcal{C}'| \leq n+1$ and $\bigcap_{c_i \in \mathcal{C}'} c_i = \varnothing$.*

Theorem 4.8. *Let Λ be any compact representation for systems of linear inequalities. On the class $\mathcal{C}(\Lambda)$,* Succint-NonEmptiness *is in* co-**NP**.

Proof. Let $E = \gamma(Ax \leq b)$ be the input to Succint-NonEmptiness, i.e., the encoding of a system $Ax \leq b \in \mathcal{C}(\Lambda)$, with $A \in \mathbb{Q}^{m \times n}$, and consider the complementary problem of deciding whether $\Omega(Ax \leq b) = \varnothing$. Observe that the set $\{x \in \mathbb{R}^n \mid A_{i,\bullet}x \leq b_i\}$ is convex, for each natural number $i \in \{1, \ldots, m\}$. By Helly's Theorem, we have that $\Omega(Ax \leq b) = \varnothing$ if, and only if, there is a non-empty set of (at most) $n+1$ indices $I = \{i_1, \ldots, i_{n+1}\} \subseteq \{1, \ldots, m\}$ such that the set $\{x \in \mathbb{R}^n \mid \forall i \in I,\ A_{i,\bullet}x \leq b_i\}$ is empty.

By the above observation, it follows that the complementary problem can be solved in polynomial time by a non-deterministic Turing machine (hence, the problem belongs to **NP**), by guessing a set I' of $n+1$ natural numbers associated with such inequalities via function $\mathcal{L}$, and by checking that $P = \{x \in \mathbb{R}^n \mid \forall i' \in I',\ A_{i,\bullet}x \leq b_i,\ \text{where } \mathcal{L}(i') = (A_{i,\bullet}, b_i)\}$ is empty. In particular, the size of I' is $O((n+1) \times ||E||^c)$ and hence $O(||E||^{c+1})$, for some constant c, because—as observed in the previous proof—it is sufficient to guess numbers whose size is polynomial in the system encoding (as these numbers are at most exponentially larger than such an encoding). Moreover, observe that P is a polyhedron defined by (at most) $n+1$ inequalities and, hence, its non-emptiness can be checked in polynomial time w.r.t. the size of its (explicit) representation, by standard results on linear programming (see, e.g., [28, 49]). The proof is then completed by noticing that, in the light of Definition 4.2, the size of each inequality defining P is polynomial w.r.t. the size $||E||$ of the input to Succint-NonEmptiness. □

Let us now study the problem of checking the dimension of a given polytope. To this end, we find it useful to state a simple characterization for implied equalities.

Lemma 4.9. *Let $Ax \leq b$ be a system of linear inequalities, with $A \in \mathbb{R}^{m \times n}$ and $\Omega(Ax \leq b) \neq \varnothing$, and let $\bar{i} \in \{1, \ldots, m\}$. Then, '$A_{\bar{i},\bullet}x \leq b_{\bar{i}}$' is an implied equality if, and only if, there is a set W of inequalities occurring in $Ax \leq b$ such that $|W| \leq n$, and*

$$\{x \in \mathbb{R}^n \mid (A_{\bar{i},\bullet}x < b_{\bar{i}}) \wedge \bigwedge_{\text{`}A_{i,\bullet}x \leq b_i\text{'} \in W} A_{i,\bullet}x \leq b_i\} = \varnothing.$$

Proof. Recall that '$A_{\bar{i},\bullet}x \leq b_{\bar{i}}$' is an implied equality if for each $\bar{x} \in \Omega(Ax \leq b)$, $A_{\bar{i},\bullet}\bar{x} = b_{\bar{i}}$ holds, and consider the collection $\mathcal{C}$ of convex sets precisely including a set $c_h = \{x \in \mathbb{R}^n \mid A_{h,\bullet}x \leq b_h\}$, for each $h \in \{1, \ldots, m\} \setminus \{\bar{i}\}$, plus the set $c_{\bar{i}} = \{x \in \mathbb{R}^n \mid A_{\bar{i},\bullet}x < b_{\bar{i}}\}$. We first claim that $A_{\bar{i},\bullet}x \leq b_{\bar{i}}$ is an implied equality if, and only if, $\bigcap_{c_i \in \mathcal{C}} c_i = \varnothing$. Indeed, if $A_{\bar{i},\bullet}x \leq b_{\bar{i}}$ is an implied equality, then we immediately have that $\bigcap_{c_i \in \mathcal{C}} c_i = \varnothing$. For the other side, because $\Omega(Ax \leq b) \neq \varnothing$, the only possibility to have $\bigcap_{c_i \in \mathcal{C}} c_i = \varnothing$ is that, for each $\bar{x} \in \Omega(Ax \leq b)$, $A_{\bar{i},\bullet}\bar{x} = b_{\bar{i}}$ holds, i.e., $A_{\bar{i},\bullet}x \leq b_{\bar{i}}$ is an implied equality. In the light of the above observation and of Proposition 4.7, it follows that $A_{\bar{i},\bullet}x \leq b_{\bar{i}}$ is an implied equality if, and only if, there is a collection $\mathcal{C}' \subseteq \mathcal{C}$ such that $|\mathcal{C}'| \leq n+1$ and $\bigcap_{c_i \in \mathcal{C}'} c_i = \varnothing$, and where $c_{\bar{i}} \in \mathcal{C}'$ must hold. □

In the rest of this section, a set W of at most n inequalities with the properties stated in Lemma 4.9 will be called a *supporting set* (for the inequality $A_{\bar{i},\bullet}\bar{x} \leq b_{\bar{i}}$). Note that checking whether a set of inequalities is a supporting set is feasible in polynomial time by standard arguments from linear programming—in particular, it is well-known [49] that any strict inequality of the form $A_{\bar{i},\bullet}x < b_{\bar{i}}$ can equivalently be replaced by the inequality $A_{\bar{i},\bullet}x \leq b_{\bar{i}} + \varepsilon$, for a suitable constant ε such that $||\varepsilon||$ is polynomial in the size of the given system (here, in the size of the set of inequalities W to be evaluated).

With the above notations and results in place, we can now conclude the picture of the membership results for our decision problems.

Theorem 4.10. *Let Λ be any compact representation for systems of linear inequalities. On the class $\mathcal{C}(\Lambda)$,* Succint-Dimension *is in* **NP**.

Proof. Recall that the input to Succint-Dimension is an encoding $E = \gamma(Ax \leq b)$ of a system $Ax \leq b \in \mathcal{C}(\Lambda)$, with $A \in \mathbb{Q}^{m \times n}$, plus a non-negative number $k \leq n+1$.

We describe a non-deterministic Turing machine M deciding whether $dim(Ax \leq b) \leq n-k$ in polynomial-time w.r.t. the size of E. First, M non-deterministically guesses one bit to choose whether moving to a state that deals with an empty $\Omega(Ax \leq b)$ or to a state that deals with a non-empty polytope.

In the former case, M continues by guessing in polynomial-time a witness that $\Omega(Ax \leq b) = \varnothing$, and then by checking in deterministic polynomial time that this is actually the case, as described in the proof of Theorem 4.8. If the check is successful, M halts and accepts, because $dim(Ax \leq b) \leq n - k$ holds for every $0 \leq k \leq n + 1$.

In the latter case ($\Omega(Ax \leq b) \neq \varnothing$), $dim(Ax \leq b) \leq n - k$ holds if, and only if, there is a set of inequalities I occurring in the system, with $|I| = k$, such that: *(1)* $I \subseteq ie(Ax \leq b)$ and *(2)* $rank(I) = k$. Then, the machine M guesses a set of natural numbers I' with $|I'| = k$ identifying the inequalities in I via $\mathcal{L}$, together with k sets of numbers $W_1, \ldots, W_k$, and then checks that: *(1)* for each $i \in I'$, $\{\mathcal{L}(j) \mid j \in W_i\}$ is a supporting set for the inequality $\mathcal{L}(i)$ (which is equivalent to checking that $\mathcal{L}(i) \in ie(Ax \leq b)$, by Lemma 4.9); and *(2)* that $rank(\{\mathcal{L}(i) \mid i \in I'\}) = k$.

In particular, by definition of supporting set and by Definition 4.2, it follows that $||I'|| + ||W_1|| + \cdots + ||W_k|| \leq k \times ||E||^c + kn \times ||E||^c$, where $k \leq n$ and c is some fixed constant. Thus, guessing all the needed sets of numbers is actually feasible by the machine M in polynomial time w.r.t. $||E||$. Moreover, once the above numbers are known, computing the inequalities by using the function $\mathcal{L}$ and checking whether W_i is actually a supporting set for $\mathcal{L}(i)$, for each $i \in I'$, is feasible in (deterministic) polynomial time w.r.t. $||E||$. Finally, observe that checking whether $rank(\{\mathcal{L}(i) \mid i \in I'\}) = k$ is feasible in polynomial time task, too. Indeed, both computing the rank of a set of vectors, and computing the characteristic vector of the inequality associated with each $i \in I'$ via $\mathcal{L}$ are polynomial time tasks. □

4.3 Computational Problems

For all computational problems studied in this section, we present algorithms running as deterministic polynomial-time Turing transducers equipped with **NP** oracles, which characterize function problems in $\mathbf{F\Delta_2^P}$. We point out that the requirement that the input system $Ax \leq b$ for any considered problem has some solution (to get rid of the trivial empty-case) is easy to check for such machines. Indeed, recall that the emptiness test is an **NP** task from Theorem 4.8, hence a Turing machine with an **NP** oracle can easily check the above property with a simple preliminary call to its oracle, and then it may return the empty output if the required condition is not met.

We start with the complexity of computing a basis of the implied equalities for a given system $Ax \leq b$, represented in a compact way, such that $\Omega(Ax \leq b) \neq \varnothing$ (otherwise, all inequalities would trivially be implied equalities). Recall from Section 3 that a basis of the implied equalities of a system $Ax \leq b$ is any set $\mathcal{B} \subseteq ie(Ax \leq b)$ satisfying the following two conditions: *(i)* $|\mathcal{B}| = rank(\mathcal{B})$ and *(ii)* $|\mathcal{B}| = rank(ie(Ax \leq b))$.

Theorem 4.11. *Let Λ be any compact representation for systems of linear inequalities. On the class $\mathcal{C}(\Lambda)$,* Succint-ImpliedEqualities-Computation *is in $\mathbf{F\Delta_2^P}$.*

Proof. Let $\gamma(Ax \leq b)$ be the encoding of a system $Ax \leq b \in \mathcal{C}(\Lambda)$, with $A \in \mathbb{Q}^{m \times n}$, and such that $\Omega(Ax \leq b) \neq \varnothing$. We shall show that, on input $\gamma(Ax \leq b)$, Succint-ImpliedEqualities-Computation can be solved by a polynomial time algorithm that uses an **NP** oracle. The algorithm works in two phases.

In the first phase, the algorithm computes the value $rank(ie(Ax \leq b))$. In particular, since $dim(Ax \leq b) = n - rank(ie(Ax \leq b))$, the algorithm actually computes first the dimension of $\Omega(Ax \leq b)$, in order to subsequently evaluate $n - dim(Ax \leq b)$. To compute the dimension, recall that by Theorem 4.10 an **NP** Turing machine can decide, for any given natural number k, whether $dim(Ax \leq b) \leq n - k$. Thus, the actual dimension $n - k^*$ can be computed via a binary search in the range $[0 \ldots n]$, by requiring a number of calls to the oracle that is logarithmic w.r.t. n, hence polynomial in $||\gamma(Ax \leq b)||$, by Definition 4.2.

In the second phase, the $k^* = rank(ie(Ax \leq b))$ basis inequalities are computed one-by-one according to a binary-search procedure, by repeatedly calling an oracle for the following problem Base-Check: Assume that a natural number h and a set I of inequalities occurring in the system, with $|I| < k^*$, are given in input, together with the encoding $\gamma(Ax \leq b)$. We have to decide whether there exists a basis $\mathcal{B}$ of the implied equalities of $Ax \leq b$ and a natural number $\bar{i} \geq h$, with $\mathcal{L}(\bar{i}) \notin I$, such that $\mathcal{B} \supseteq I \cup \{\mathcal{L}(\bar{i})\}$.

Claim 4.11.(1): Base-Check *is feasible in* **NP**.

Proof. The problem can be solved in polynomial time by a non-deterministic Turing machine that first guesses a set $W_{\mathcal{B}}$ of k^* natural numbers, associated via $\mathcal{L}$ with the inequalities $\mathcal{B} = \{\mathcal{L}(i) \mid i \in W_{\mathcal{B}}\}$ (intuitively, some basis); then guesses k^* sets of numbers encoding supporting sets for inequalities in $\mathcal{B}$ (as in the proof of Theorem 4.10); and eventually checks that: (1) $I \subseteq \mathcal{B}$, (2) there is a number $\bar{i} \in W_{\mathcal{B}}$ with $\bar{i} \geq h$ and $\mathcal{L}(\bar{i}) \notin I$, (3) $\mathcal{B} \subseteq ie(Ax \leq b)$, and (4) $\mathcal{B}$ is a basis of the implied equalities. Note that, by construction and by Definition 4.2, $||W_{\mathcal{B}}|| \leq n \times ||\gamma(Ax \leq b)||^c$ for some constant c, and hence guessing the set $W_{\mathcal{B}}$ is feasible in polynomial time. Moreover, as described in the proof of Theorem 4.10, the same holds for the guess of the k^* sets of numbers, each one of cardinality at most n, encoding the supporting sets for the inequalities in $\mathcal{B}$.

We next point out that all checks above are feasible in (deterministic) polynomial time. Conditions (1) and (2) can be trivially checked in polynomial time. Checking condition (3) is equivalent to checking whether, for

Input: A coalitional game $\mathcal{G} = \langle N, v \rangle$;
Output: The nucleolus $\mathcal{N}(\mathcal{G})$ of $\mathcal{G}$;

---

1. **if** $X(\mathcal{G}) = \varnothing$ **then return** $\varnothing$;
2. **let** $t := 0$, and $\mathcal{B}_0 = \varnothing$;
3. **do**
4. **let** $t := t + 1$;
5. **let** $A[t](x, \varepsilon) \leq b[t]$ be the following system of linear inequalities:
   $v(S) - x(S) \leq \varepsilon$, $\forall S \subseteq N$ s.t. $rank(\mathcal{B}_{t-1}) \neq rank(\mathcal{B}_{t-1} \cup \{\mathbb{1}_S\})$
   $v(S) - x(S) \leq \varepsilon_r$, $\forall r \in \{1, \ldots, t-1\}, \forall S \subseteq N$ s.t. $rank(\mathcal{B}_{r-1}) \neq rank(\mathcal{B}_{r-1} \cup \{\mathbb{1}_S\})$
   $A(\mathcal{G})x \leq b(\mathcal{G})$, i.e., $x \in X(\mathcal{G})$
   and **let** $\mathtt{LP}_t(\mathcal{G})$ be the linear program $\mathtt{LP}_t(\mathcal{G}) = \min \varepsilon \mid A[t](x, \varepsilon) \leq b[t]$;
6. **compute** the value ε_t of an optimal solution to $\mathtt{LP}_t(\mathcal{G})$;
7. **let** $A[t](x, \varepsilon = \varepsilon_t) \leq b[t]$ be the modified system where variable ε is fixed to ε_t.
8. **compute** a basis $\mathcal{B}_t$ of the implied equalities of $A[t](x, \varepsilon = \varepsilon_t) \leq b[t]$;
9. **while** $|\mathcal{B}_t| < |N|$;
10. **compute** an element $\bar{x}$ in $\Omega(A[t]x \leq b[t])$;
11. **return** $\bar{x}$;

Figure 4: **Algorithm** ComputeNucleolus.

each natural number $i \in W_{\mathcal{B}}$, $\mathcal{L}(i)$ is an implied equality of $Ax \leq b$, which is easily done having guessed a supporting set for $\mathcal{L}(i)$. Finally, in order to check that condition (4) holds, we can easily check in polynomial time that $|\mathcal{B}| = k^*$ and that $|\mathcal{B}| = rank(\mathcal{B})$, as the characteristic vectors of these inequalities are obtained in polynomial time by applying the function $\mathcal{L}$ to the guessed numbers in $W_{\mathcal{B}}$. ⋄

As previously stated, the second phase of the algorithm uses the oracle solving Base-Check. Recall that the value $k^* = rank(ie(Ax \leq b))$ is available when invoking this oracle, because it has been computed in the first phase of the algorithm via a computation in $\mathbf{F\Delta_2^P}$. The oracle is then used in a loop that starts with $I = \varnothing$ and then performs the following tasks for k^* iterations: At each iteration, it performs a binary search over the exponentially many natural numbers h that may identify an inequality of the system (via the function $\mathcal{L}$), which is an implied equality not included in the current set I. More precisely, by this binary search we compute bit by bit, through a polynomial number of oracle calls, a number $\bar{h}$ such that $\mathcal{L}(\bar{h}) \in ie(Ax \leq b) \setminus I$. Then, I is updated by adding the new inequality $\mathcal{L}(\bar{h})$, and the loop continues with the next iteration. Eventually, after $k^* \leq n$ iterations, the set I is a basis of implied equalities of the given system, so that the second phase requires at most polynomially many calls of the Base-Check oracle. Putting this together with the complexity of the first phase, we have that Succint-ImpliedEqualities-Computation is in $\mathbf{F\Delta_2^P}$. □

From what we have seen so far, it is rather easy to prove that, for any compact representation, there is an $\mathbf{F\Delta_2^P}$ separation oracle: given the encoding $\gamma(Ax \leq b)$ of a system and some vector $\bar{x}$, an inequality violated by $\bar{x}$ (if any) can be identified in **NP** by guessing polynomially many bits, and hence can be computed in $\mathbf{F\Delta_2^P}$ by a standard self-reducibility argument. Therefore, the following result about the remaining computational problems can be obtained from known results from linear programming (see, e.g., Grötschel et al. [28]), by noting that any function computable by a polynomial-time deterministic Turing machine using an $\mathbf{F\Delta_2^P}$ procedure belongs in fact to $\mathbf{F\Delta_2^P}$. Nevertheless, we provide for completeness a direct proof in Appendix C.

Theorem 4.12. *Let Λ be any compact representation for systems of linear inequalities. On the class $\mathcal{C}(\Lambda)$,* Succint-OptimalValue-Computation, Succint-Solution-Computation, *and* Succint-OptimalSolution-Computation *are in $\mathbf{F\Delta_2^P}$.*

5 Putting It All Together: Nucleolus Computation is in $\mathbf{F\Delta_2^P}$

Now that we have analyzed the complexity issues arising from reasoning about succinctly specified linear programs, we come back to the problem of computing the nucleolus and analyze its complexity over compact games.

We first present an algorithm that computes the nucleolus of any coalitional game ($\mathcal{G}$) based on the sequence of linear programs $\mathbf{LP}(\mathcal{G})$ of Definition 3.2.

The algorithm, named ComputeNucleolus and reported in Figure 4, returns the empty set if $X(\mathcal{G}) = \varnothing$. Otherwise, it constructs the linear program $\mathtt{LP}_t(\mathcal{G})$, which is precisely the one illustrated in Section 3.2, but for the fact that, rather than to directly check whether a coalition S belongs to a set of the form $\mathcal{F}_r$, we use the characterization provided by Theorem 3.10 to detect coalitions having constant excesses. The value ε_t of an

optimal solution is then computed, so that we immediately obtain the system $A[t](x, \varepsilon = \varepsilon_t) \leq b[t]$, where the variable ε is fixed to the constant value ε_t, and for which $\Omega(A[t](x, \varepsilon = \varepsilon_t) \leq b[t]) = V_t(\mathcal{G})$ holds. Then, the algorithm computes a basis $\mathcal{B}_t$ of the implied equalities of this program, which will be used in the subsequent iteration. All these steps are repeated until the rank of the basis eventually reaches $|N|$, so that the dimension of the non-empty set of imputations $V_t(\mathcal{G})$ becomes 0, which means it contains just one imputation, which is the nucleolus.

In fact, the number of iterations required by the algorithm can be determined immediately from Proposition 3.6 and Theorem 3.10.

Theorem 5.1. ComputeNucleolus *computes the nucleolus $\mathcal{N}(\mathcal{G})$ of $\mathcal{G} = \langle N, v\rangle$, with no more than $|N|$ iterations of the main loop (steps 3–9).*

Let $\mathcal{R}$ be a polynomial-time compact representation, and let $\mathcal{G}$ be a coalitional game representable through $\mathcal{R}$. A finer analysis of the algorithm is given next by assuming that $\xi^{\mathcal{R}}(\mathcal{G})$ is the encoding of the input game, and where the worth function is the (deterministic) polynomial-time function $v^{\mathcal{R}}$. Recall that well-known classes of **P**-representations are graph and hypergraph games [14], marginal contribution nets [30], games in multi-issue domains [12], weighted voting games [19], minimum cost spanning tree games [43], flow games [33], linear production games [48], multi-attribute games [31], read-once (and general) marginal contribution nets [18], skill games [6], matching games [34, 58], path disruption games [5], and (vertex) connectivity games [7]. Our analysis applies to all of them.

Based on $\mathcal{R}$, we preliminarily define a compact representation $\Lambda_{\mathcal{R}}$ for systems of inequalities, in order to encode the systems defined in the loop of Algorithm ComputeNucleolus. The class $\mathcal{C}(\Lambda_{\mathcal{R}})$ contains a number of systems for every coalitional game $\mathcal{G} = \langle N, v\rangle \in \mathcal{C}(\mathcal{R})$. Any system $Ax \leq b$ is encoded by the game encoding $\xi^{\mathcal{R}}(\mathcal{G})$, plus $t-1$ sets of inequalities $\mathcal{B}_0, \ldots, \mathcal{B}_{t-1}$, plus $t-1$ real values $\varepsilon_1, \ldots, \varepsilon_{t-1}$, and a further element ε_t, which can be either a real value or a variable.

The representation is designed in such a way that, according to what ε_t is (either a value or a variable), we encode the systems $(A[t](x, \varepsilon = \varepsilon_t) \leq b[t])$ or $(A[t](x, \varepsilon) \leq b[t])$ computed during a run of Algorithm ComputeNucleolus. The number of variables, say n, is $|N|$ in the former case, and $|N|+1$ in the latter one. The number of inequalities, say m, is at most $|t| \times 2^{|N|} + 2 + |N|$. Moreover, the polynomial time function $\mathcal{L}^{\Lambda}$ mapping natural numbers to (coefficients) of inequalities in the system is defined as follows.

Let τ be the (bijective) function associating every natural number i in the set $\{1, \ldots, |t| \times 2^{|N|}\}$ with the pair $\tau(i) = (r, S)$ such that $r = \lceil i/2^{|N|} \rceil$ (hence, $r \leq t$); and $p \in S$ if, and only if, the p-th least significant bit of the remainder of the integer division $i/2^{|N|}$ is 1. Then, the output identified by $\mathcal{L}^{\Lambda}(i)$, with $i \in \{1, \ldots, |t| \times 2^{|N|}\}$ and $\tau(i) = (r, S)$, is $v(S) - x(S) \leq \varepsilon_r$, if $\mathit{rank}(\mathcal{B}_{r-1}) \neq \mathit{rank}(\mathcal{B}_{r-1} \cup \{\mathbb{1}_S\})$; or the empty output, otherwise (no inequality is associated with i).

In particular, note that when $\tau(i) = (t, S)$ (i.e., $r = t$) and ε_t encodes a variable, $\mathcal{L}^{\Lambda}(i)$ computes the inequalities of the system $A[t](x, \varepsilon) \leq b[t]$ associated with those coalitions S over which the excess ε has to be minimized (cf. instruction 5 in Figure 4). Moreover, the remaining $2 + |N|$ inequalities encode individual rationality and efficiency. Formally, for any $i = p + |t| \times 2^{|N|}$, $1 \leq p \leq |N|$, $\mathcal{L}^{\Lambda}(i)$ is the inequality $-x_p \leq -v(\{p\})$; while $\mathcal{L}^{\Lambda}(|t| \times 2^{|N|} + |N| + 1)$ is $-\sum_{p \in N} x_p \leq -v(N)$ and $\mathcal{L}^{\Lambda}(|t| \times 2^{|N|} + |N| + 2)$ is $\sum_{p \in N} x_p \leq v(N)$. Note that $\mathcal{L}^{\Lambda}$ is actually a polynomial time function, as all the operations involved in its computation are feasible in polynomial time. In particular, recall that computing the rank of a given set of vectors is feasible in polynomial time, and computing the worth $v(S)$ of a given coalition $S \subseteq N$ is in polynomial time, too, because $\mathcal{R}$ is a **P**-representation.

Assuming the compact encoding being now understood, to keep the notation simple we often omit hereafter the indication of the representation Λ, and refer directly to the systems described in the algorithm.

Theorem 5.2. *Let $\mathcal{R}$ be a* **P***-representation. On the class $\mathcal{C}(\mathcal{R})$, computing the nucleolus is feasible in $\mathbf{F}\boldsymbol{\Delta}_{\mathbf{2}}^{\boldsymbol{P}}$.*

Proof. Let $\mathcal{R}$ be a **P**-representation for coalitional games. Consider the algorithm shown in Figure 4 with the encoding $\xi^{\mathcal{R}}(\mathcal{G})$ of a coalitional game $\mathcal{G}$ as its input. Recall that, for each coalition $S \subseteq N$, the worth $v^{\mathcal{R}}(\xi^{\mathcal{R}}(\mathcal{G}), S)$ can be computed in polynomial time w.r.t. $||\xi^{\mathcal{R}}(\mathcal{G})||$, because $\mathcal{R}$ is a **P**-representation.

After Theorem 5.1, it suffices to show that each step of the algorithm is feasible in $\mathbf{F}\boldsymbol{\Delta}_{\mathbf{2}}^{\boldsymbol{P}}$ (w.r.t. the size $||\xi^{\mathcal{R}}(\mathcal{G})||$). This is clearly true for the polynomial-time steps 1 and 2. Let us focus on the main loop (steps 3–9), recalling that the systems of linear inequalities involved in the loop operations are encoded according to the compact representation $\Lambda_{\mathcal{R}}$, as observed above. Therefore, all the results presented in Section 4 may be applied to the class of systems represented this way. In particular, given any input system encoded according to $\Lambda_{\mathcal{R}}$, every needed operation is feasible in $\mathbf{F}\boldsymbol{\Delta}_{\mathbf{2}}^{\boldsymbol{P}}$ (w.r.t. the size of this input system).

Now, observe that, at each $t \geq 1$, the size of the encodings for the systems $A[t](x, \varepsilon) \leq b[t]$ and $A[t](x, \varepsilon = \varepsilon_t) \leq b[t]$ are $O(||\xi^{\mathcal{R}}(\mathcal{G})||^c)$, for some constant c. This follows from the definition of the compact representation $\Lambda_{\mathcal{R}}$ and from the following facts:

- From Theorem 5.1, we know that $t \leq |N|$.
- The size of the nucleolus is polynomially bounded in the size of the game [50], as well as the size of ε_t, for each ε_t in the sequence of linear programs $\mathbf{LP}(\mathcal{G})$ of Definition 3.2 and hence in the linear programs in Algorithm ComputeNucleolus (see, e.g., the proof of Theorem 2.2 in Paulusma [50]).

In order to conclude, we next show that each of these (at most $|N|$) encodings can be computed in $\mathbf{F\Delta_2^P}$ w.r.t. $||\xi^{\mathcal{R}}(\mathcal{G})||$.

First step: In the base case where $t = 1$, the encoding for $A[t](x, \varepsilon) \leq b[t]$ is just given by $\xi^{\mathcal{R}}(\mathcal{G})$, hence it can be built in linear time. Therefore, by noting that the solutions of such a system form a non-empty polytope and by applying Theorem 4.12 on the representation $\Lambda_{\mathcal{R}}$, we derive that we can compute in $\mathbf{F\Delta_2^P}$ the value ε_1 (see step 6). As noted above, the size of ε_1 and of the encoding for $A[1](x, \varepsilon = \varepsilon_1) \leq b[1]$ are polynomially bounded in the size of the game $\xi^{\mathcal{R}}(\mathcal{G})$. Thus, we are in the position of applying Theorem 4.11 in order to conclude that a basis $\mathcal{B}_1$ of the implied equalities of $A[1](x, \varepsilon = \varepsilon_1) \leq b[1]$ can be computed in $\mathbf{F\Delta_2^P}$ (see step 8).

Generic step: At the generic step t', both $\mathcal{B}_{t'-1}$ and the encoding of $A[t'-1](x, \varepsilon = \varepsilon_{t'-1}) \leq b[t'-1]$ have been computed (in $\mathbf{F\Delta_2^P}$), with the size of this system being polynomial w.r.t. $||\xi^{\mathcal{R}}(\mathcal{G})||$. As the encoding of $A[t'](x, \varepsilon) \leq b[t']$ just contains these two encodings, plus the encoding of the variable $\varepsilon_{t'}$, it is computable in polynomial time from them. Therefore, we are precisely in the same scenario as in the first step above, and we can apply the same line of reasoning to conclude that the computation of the optimal value $\varepsilon_{t'}$ is feasible in $\mathbf{F\Delta_2^P}$. Moreover, the size of the encoding of $A[t'](x, \varepsilon = \varepsilon_{t'}) \leq b[t']$ is polynomial w.r.t. $||\xi^{\mathcal{R}}(\mathcal{G})||$, and the basis $\mathcal{B}_{t'}$ can be computed in $\mathbf{F\Delta_2^P}$ (w.r.t. $||\xi^{\mathcal{R}}(\mathcal{G})||$).

To conclude the proof just note that, by Theorem 4.12, step 10 is feasible in $\mathbf{F\Delta_2^P}$ with respect to the size of the last computed system $A[t](x, \varepsilon = \varepsilon_t) \leq b[t]$, and hence with respect to $||\xi^{\mathcal{R}}(\mathcal{G})||$. □

Note that the complexity derived above matches the hardness result given for graph games in Section 2. Therefore, as an immediate consequence of the above result and Corollary 2.5, we get the following.

Corollary 5.3. *Let $\mathcal{R}$ be any $\mathbf{P}$-representation such that $\texttt{GGR} \precsim_e \mathcal{R}$. On the class $\mathcal{C}(\mathcal{R})$, deciding whether a vector is the nucleolus is $\mathbf{\Delta_2^P}$-complete.*

6 Tractable Classes of Compact Games

Several efforts have been spent in searching for classes of compact games over which computing the nucleolus is tractable. For instance, efficient algorithms have been singled out for computing the nucleolus of tree games [24, 43], convex games [38], assignment games [61], airport games [39], certain classes of routing games [16], cardinality matching games [34], cyclic permutation games [63], veto-rich games [3], games with permission structure [66], flow games games with unitary arc capacities [15], connected games [62], and peer group games [9].

Our approach to identify tractability islands, which we are illustrating below, complements this literature, and follows instead the recent studies by Shrot et al. [59], Ueda et al. [64], and Aadithya et al. [1]. Indeed, guided by the observation that obstructions to tractability in coalitional games emerge in scenarios where most players are "different", rather than focusing on some specific class of games we consider arbitrary classes where the number of distinct player *types* is small (bounded by some fixed constant).

Definition 6.1 (Shrot et al. [59])**.** Let $\mathcal{G} = \langle N, v \rangle$ be a coalitional game. We say that two players $i, j \in N$ are *strategically equivalent* in $\mathcal{G}$ (or, simply, have the same *type*) if $v(S \cup \{i\}) = v(S \cup \{j\})$ holds, for each coalition $S \subseteq N$ such that $S \cap \{i, j\} = \varnothing$.

The game $\mathcal{G}$ is *k-typed* if its players can be partitioned into at most k pairwise disjoint classes of strategically equivalent players. □

For any compact representation $\mathcal{R}$ and fixed natural number $k > 0$, we define $\mathcal{C}_k(\mathcal{R})$ as the subclass of $\mathcal{C}(\mathcal{R})$ of all the k-typed games defined by $\mathcal{R}$.

In the rest of this section, the complexity of computing the nucleolus is studied over classes of k-typed compact games.

6.1 Results for Games in Type-Based Form

Let k be some natural number, let $\mathcal{R}$ be a compact representation for coalitional games, and let $C_k(\mathcal{R})$ be the class of k-typed games defined by $\mathcal{R}$. We say that a game $\mathcal{G} \in C_k(\mathcal{R})$ is in *type-based form* if its encoding $\xi^{\mathcal{R}}(\mathcal{G})$ comprises a type classification of players in $\mathcal{G}$, i.e., a list $N_1, \ldots, N_k$ of disjoint sets of players, with all players in N_i having the same type, and such that $\bigcup_{i=1}^{k} N_i$ is the set of all players of $\mathcal{G}$.[10] Any game $\mathcal{G} \in C_k(\mathcal{R})$ in type-based form can conveniently be denoted as a tuple $\langle (N_1, \ldots, N_k), v \rangle$.

It is well known that the worth function v of k-typed games is such that the value $v(S)$ assigned to each coalition S depends only on how many players of each type belong to S (see, e.g., [1, 64]), as stated below.

Proposition 6.2 ([59])**.** *Let $\langle N, v \rangle$ be a coalitional game, and let $(N_1, \ldots, N_k)$ be a partition of N into k sets of strategically equivalent players. Given any two coalitions $S, T \subseteq N$, if $|S \cap N_i| = |T \cap N_i|$, for each $i \in \{1, \ldots, k\}$, then $v(S) = v(T)$.*

Our goal is now to characterize the complexity of computing the nucleolus of k-typed games given in type-based form. To this end, it is relevant to characterize the "structure" of this solution concept over k-typed coalitional games. The following result shows that, as intuitively expected, the nucleolus treats equals in the same way, so that it is "symmetric" w.r.t. player types.

Theorem 6.3. *Let $\mathcal{G} = \langle N, v \rangle$ be a coalitional game with $X(\mathcal{G}) \neq \varnothing$, and let $\bar{x}$ be the unique imputation in $\mathcal{N}(\mathcal{G})$. Then, $\bar{x}_i = \bar{x}_j$ holds, for each pair of players i and j in N having the same type.*

Proof. Let $\mathcal{G} = \langle N, v \rangle$ be a coalitional game with $X(\mathcal{G}) \neq \varnothing$ (hence $|\mathcal{N}(\mathcal{G})| = 1$), and assume by contradiction that there are two players i and j in N having the same type and such that $\bar{x}_i \neq \bar{x}_j$ (w.l.o.g., we can assume that $\bar{x}_i > \bar{x}_j$). We claim that $\{\bar{x}\} \neq \mathcal{N}(\mathcal{G})$.

Indeed, let $\bar{x}'$ be the worth assignment where the values assigned to i and j are swapped, that is, such that $\bar{x}'_i = \bar{x}_j$, $\bar{x}'_j = \bar{x}_i$, and $\bar{x}'_p = \bar{x}_p$, for each $p \in N \setminus \{i, j\}$. Note that, for any coalition S such that $S \cap \{i, j\} = \varnothing$ or $\{i, j\} \subseteq S$, the total worth does not change, and hence $e(S, \bar{x}) = s(S, \bar{x}')$. It remains to consider all pairs of "symmetric" coalitions T, T' such that $i \in T$ and $j \notin T$, $i \notin T'$ and $j \in T'$, and coinciding as for the rest, i.e., for $T'' = T \setminus \{i, j\} = T' \setminus \{i, j\}$. Note that for each $p \in T''$, $\bar{x}_p = \bar{x}'_p$, and that $v(T) = v(T'' \cup \{i\}) = v(T'' \cup \{j\}) = v(T')$, as i and j have the same type. It follows that, for every such pair of coalitions, $e(T', \bar{x}') = e(T, \bar{x})$ and $e(T, \bar{x}') = e(T', \bar{x})$; that is, their excesses are just swapped. Therefore, the vector of excesses does not change when considering $\bar{x}'$ in place of $\bar{x}$, and we get $\theta(\bar{x}) = \theta(\bar{x}')$, which is impossible as the nucleolus is a singleton. □

Just for completeness, note that the converse of Theorem 6.3 does not hold. For instance, on the game $\mathcal{G} = \langle \{a, b, c\}, v \rangle$ such that $v(\{a\}) = v(\{b\}) = v(\{c\}) = 1$, $v(\{a, b, c\}) = 3$, $v(\{a, b\}) = 1$, $v(\{a, c\}) = 2$, and $v(\{b, c\}) = 3$, the vector x with $x_a = x_b = x_c = 1$ is the only imputation and hence belongs to $\mathcal{N}(\mathcal{G})$, but the three players have different types.

With the above result in place, let us analyze the complexity of computing the nucleolus. The good news is that the problem is no longer intractable, if the k-types of players are known and the worth function can be computed in polynomial time.

Theorem 6.4. *Let $\mathcal{R}$ be a* **P***-representation and let k be a fixed natural number. Given any game $\mathcal{G} \in \mathcal{C}_k(\mathcal{R})$ in type-based form, computing the nucleolus of $\mathcal{G}$ is feasible in polynomial time.*

Proof. Let $\mathcal{G} = \langle (N_1, \ldots, N_k), v \rangle \in \mathcal{C}_k(\mathcal{R})$ be a coalitional game in type-based form. Assume that an arbitrary ordering of players in N is fixed, and define the *characteristic-coalitions set* $\mathcal{D}_{\mathcal{G}} \subseteq 2^N$ as the set of coalitions $\{P_1 \cup P_2 \cup \cdots \cup P_k \mid S \subseteq N$, and P_i contains the first $|S \cap N_i|$ players from set $N_i, 1 \leq i \leq k\}$. Note that the size of $\mathcal{D}_{\mathcal{G}}$ is polynomial w.r.t. the size of $\mathcal{G}$, as it contains at most $|N_1| \times |N_2| \times \cdots \times |N_k|$ coalitions. Moreover, consider the convex set $\widehat{X}(\mathcal{G}) = \{x \in X(\mathcal{G}) \mid x_i = x_j$, for each pair i, j of players having the same type$\}$.

By Theorem 6.3, $\mathcal{N}(\mathcal{G}) \subseteq \widehat{X}(\mathcal{G})$, and thus $\mathcal{N}(\mathcal{G})$ can be computed by the sequence of linear programs shown in Section 3.2, by constraining the imputations to belong to $\widehat{X}(\mathcal{G})$ (see Lemma 6.5 in [42]). In fact, having restricted the feasible regions of these programs to $\widehat{X}(\mathcal{G})$, it follows that every inequality associated with some coalition S entails every other inequality obtained by replacing any variable x_i (associated with a player) of a certain type by any other variable x_j (associated with a player) of the same type. As a consequence, it is sufficient to consider only inequalities associated with the coalitions in the characteristic set $\mathcal{D}_{\mathcal{G}}$, in place of all subsets of N.

[10] Without loss of generality, we assume that the compact game encoding $\xi^{\mathcal{R}}$ according to representation $\mathcal{R}$ may include a partition of players (otherwise, one may easily define a modified representation $\mathcal{R}'$, which is the same as $\mathcal{R}$ but is able to encode the type information, too).

Now, observe that any linear program therefore contains just polynomially many distinct inequalities and that the coalitions to be considered (in $\mathcal{D}_{\mathcal{G}}$) are polynomially many ones, given that k is a fixed natural number. In particular, because $\mathcal{G}$ is in type-based form, all these inequalities can be computed in polynomial time by iterating over all possible combinations of numbers of players per type. Thus, by standard result in linear programming, solving the succession of the linear programs and, hence, computing the nucleolus of $\mathcal{G}$ are feasible in polynomial time. □

Note that as a corollary of the above general result, we can get the tractability of well-known classes of compact games whose worth functions are computable in **FP**, and for which determining player types is feasible in polynomial time. In particular, recall that, for any fixed k, a k-typed graph game can be represented in type-based form (i.e., the clustering of its players can be found) in polynomial time [59]. Hence, the following is immediately established.

Corollary 6.5. *For any fixed natural number k, on the class $\mathcal{C}_k(\texttt{GGR})$ of k-typed graph games, computing the nucleolus is feasible in polynomial time.*

Another class of games over which determining player types is feasible in polynomial time is the class of k-typed games with *synergies among coalitions* [13, 59]. According to this representation scheme, a game $\mathcal{G} = \langle N, v\rangle$ is encoded as a set $\{(B_1, v(B_1)), \ldots, (B_h, v(B_h))\}$ of pairs, where each coalition B_i, with $i \in \{1, \ldots, h\}$, is explicitly associated with its worth. These pairs form the basis for determining the worth associated with the remaining coalitions. Indeed, for any coalition S, $v(S)$ is defined as the maximum aggregate value it can be obtained by partitioning itself into sub-coalitions taken from the given set, i.e., as the value:

$$\max\{\sum_{j\in P} v(B_j) \mid P \subseteq \{1, \ldots, h\}, \cup_{j\in P} B_j = S, \text{ and } B_j \cap B_{j'} = \varnothing, \forall j, j' \in P \text{ with } j \neq j'\}.$$

For games with synergies among coalitions, given a fixed natural number k, it is tractable to decide whether they are k-typed and to eventually represented them in type-based form [59]. Moreover, while it is easy to see that the worth function is in general **NP**-hard to compute [13], over k-typed games, it can be computed in polynomial time. In fact, the problem is fixed parameter tractable, with k being the parameter [59]. Therefore, the following is again immediately established.

Corollary 6.6. *For any fixed natural number k, on the class of k-typed games with synergies among coalitions, computing the nucleolus is feasible in polynomial time.*

6.2 On The Hardness of Finding Player Types

The general tractability result derived in Theorem 6.4 is useful whenever games are given in type-based form. As already discussed, in some cases this is not a real obstruction to tractability, as types can be efficiently recognized over some classes of games. In this section, we explore more in general the intrinsic complexity of this latter task.

Note that it has been observed by Shrot et al. [59] that deciding whether two players have the same type in *games with synergies among coalitions* [13] is an **NP**-hard problem over games that are not k-typed—as discussed above, the problem is instead tractable if the number of agent types is known to be bounded by a constant k. In fact, this **NP**-hardness result is hardly surprising as the associated worth function is **NP**-hard to compute [13]. Hence, the intrinsic difficulty of the worth function actually obscures here the complexity of the problem defined on top of it. Our first result is to strengthen this analysis, by showing that the problem remains intractable even on games defined by **P**-representations, in general. In particular, we next show that the problem is complete for the class co-**NP**.

Before stating the result, we give some definitions that will be used in the following. For any Boolean formula ϕ over a set X of variables, consider the game $\mathcal{G}_\phi = \langle X, v_\phi\rangle$, whose players coincide with the variables in ϕ, and where, for each coalition $S \subseteq X$,

$$v_\phi(S) = \begin{cases} 1, \text{ if } \sigma(S) \models \phi, \text{ i.e., } \sigma(S) \text{ is a satisfying assignment for } \phi, \text{ and} \\ 0, \text{ otherwise;} \end{cases}$$

with $\sigma(S)$ denoting the truth assignment where a variable x_i evaluates to true if, and only if, the corresponding player x_i belongs to S.

Let Φ be a compact representation for coalitional games, such that for each Boolean formula ϕ, the game $\mathcal{G}_\phi$ is in $\mathcal{C}(\Phi)$. In particular, the encoding $\xi^\Phi(\mathcal{G}_\phi)$ is the formula ϕ itself, and the worth function is the above polynomial-time computable function v_ϕ. Note that Φ is in fact a **P**-representation.

Moreover, consider the following problem *Critical Swap (CS)*: Given a tuple $\langle \phi, x_i, x_j\rangle$, where ϕ is a Boolean formula over a set X of variables and $\{x_i, x_j\} \subseteq X$, decide whether $\{x_i, x_j\}$ is a *critical pair* (w.r.t. ϕ), i.e.,

decide whether there is a satisfying truth assignment $\bar{\sigma}$ such that: (1) $\bar{\sigma}[x_i] \neq \bar{\sigma}[x_j]$ and (2) the assignment σ', where $\sigma'[x_k] = \bar{\sigma}[x_k]$, for each $x_k \in X \setminus \{x_i, x_j\}$, $\sigma'[x_i] = \bar{\sigma}[x_j]$, and $\sigma'[x_j] = \bar{\sigma}[x_i]$, is not satisfying. It is easy to see that CS is **NP**-hard, by a reduction from the satisfiability of Boolean formulae: For any Boolean formula γ, let $\phi = \gamma \wedge x_a \wedge \neg x_b$ be a new Boolean formula where x_a and x_b are fresh variables (i.e., not in γ). It is immediate to check that γ is satisfiable if, and only if, $\langle \phi, x_a, x_b \rangle$ is a "yes" instance of *CS*.

Theorem 6.7. *Let $\mathcal{R}$ be a* **P***-representation and let $\mathcal{G}$ be any game in $\mathcal{C}(\mathcal{R})$. Deciding whether two players of $\mathcal{G}$ have the same type is in* co-**NP**. *Moreover, there is a* **P***-representation $\bar{\mathcal{R}}$ such that the problem is* co-**NP***-complete on the class $\mathcal{C}(\bar{\mathcal{R}})$.*

Proof. Let $\mathcal{R}$ be a ***P***-representation and $\mathcal{G}$ any game in $\mathcal{C}(\mathcal{R})$. Consider the complementary problem of deciding whether two players p and q of $\mathcal{G}$ do not have the same type. Note that membership in **NP** is easily seen, as we can guess a coalition S with $S \cap \{p, q\} = \varnothing$, and then check in polynomial time that $v(S \cup \{p\}) \neq v(S \cup \{q\})$.

Hardness is next proven via a reduction from *Critical Swap* to the problem of checking whether a pair of players have the same-type over games encoded according to the ***P***-representation Φ defined above. Let ϕ be a Boolean formula over a set X of variables with $\{x_i, x_j\} \subseteq X$, and consider the game $\mathcal{G}_\phi = \langle X, v_\phi \rangle$, belonging to $\mathcal{C}(\Phi)$, which can be indeed constructed in polynomial time. We show that $\langle \phi, x_i, x_j \rangle$ *is a "yes" instance of* *CS* $\Leftrightarrow$ x_i *and* x_j *do not have the same type in* $\mathcal{G}_\phi$.

($\Rightarrow$) Let $\bar{\sigma}$ be an assignment witnessing that $\langle \phi, x_i, x_j \rangle$ is a "yes" instance. Assume, w.l.o.g., that $\bar{\sigma}[x_i] =$ `true` and $\bar{\sigma}[x_j] =$ `false`. Let $\overline{S} \subseteq X$ be the coalition such that $\sigma(\overline{S}) = \bar{\sigma}$, and note that $x_i \in \overline{S}$ and $x_j \notin \overline{S}$. Consider the coalition $T = \overline{S} \setminus \{x_i\}$, hence such that $\sigma(T \cup \{x_i\}) \models \phi$. By definition of a solution to *CS*, $\sigma(T \cup \{x_j\}) \not\models \phi$. Hence, $v_\phi(T \cup \{x_i\}) = 1$ while $v_\phi(T \cup \{x_j\}) = 0$. Thus, x_i and x_j do not have the same type.

($\Leftarrow$) Assume that $\langle \phi, x_i, x_j \rangle$ is a "no" instance. We consider two cases. (1) ϕ *is unsatisfiable.* In this case, $v_\phi(S) = 0$ holds, for each coalition $S \subseteq X$, and x_i and x_j have trivially the same type. (2) ϕ *is satisfiable.* In this case, for each set $T \subseteq X \setminus \{x_i, x_j\}$, we have that either $\sigma(T \cup \{x_i\}) \not\models \phi$ and $\sigma(T \cup \{x_j\}) \not\models \phi$, or $\sigma(T \cup \{x_i\}) \models \phi$ and $\sigma(T \cup \{x_j\}) \models \phi$. Hence, $v_\phi(T \cup \{x_i\}) = v_\phi(T \cup \{x_j\})$ holds, and x_i and x_j have the same type. □

The above is very bad news, but it does not immediately imply that determining whether the number of player types is bounded by some given constant is a difficult problem. Our second result is to characterize the complexity of this problem.

Theorem 6.8. *Let $\mathcal{R}$ be a* **P***-representation and let k be a natural number. Deciding whether a coalitional game $\mathcal{G} \in \mathcal{C}(\mathcal{R})$ belongs to $\mathcal{C}_k(\mathcal{R})$ (i.e., it is k-typed) is in* co-**NP**. *Moreover, there is a* **P***-representation $\bar{\mathcal{R}}$ such that the problem is* co-**NP***-complete even on the class $\mathcal{C}_1(\bar{\mathcal{R}})$.*

Proof. Let $\mathcal{R}$ be a **P**-representation and k a natural number. First note that, for any game $\mathcal{G} \in \mathcal{C}(\mathcal{R})$, deciding whether there are at least $k' = k + 1$ player types is in **NP**. Indeed, it suffices to guess a set P of k' players together with $k'(k'-1)/2$ coalitions, and then check in polynomial time that such coalitions witness that players in P are pairwise not strategically equivalent.

For the hardness part, we use again the class of games $\mathcal{C}(\Phi)$ defined by the compact representation Φ. Consider the problem *Exists Critical Swap (ECS)*, in which given a Boolean formula ϕ over a set X of variables, we have to decide whether there exists a critical pair $\{x_i, x_j\}$ w.r.t. ϕ. It is easily seen that ECS is **NP**-hard. Indeed, for any Boolean formula γ, let $\phi = \gamma \wedge x_a \wedge \neg x_b$ be a new Boolean formula where x_a and x_b are fresh variables. Then, γ is satisfiable if and only if $\langle \phi \rangle$ is a "yes" instance of *ECS*.

Our result then follows by showing that: ϕ *is a "yes" instance of* *ECS* $\Leftrightarrow$ $\mathcal{G}_\phi$ *has at least two players with different type (hence* $k > 1$*).*

($\Rightarrow$) Assume that x_i and x_j are two variables in X such that $\langle \phi, x_i, x_j \rangle$ is a "yes" instance of CS. By the same line of reasoning as in the proof of Theorem 6.7, we have that x_i and x_j are not strategically equivalent, and hence in $\mathcal{G}_\phi$ there are at least 2 different types of players.

($\Leftarrow$) Assume now that, for each pair of variables x_i and x_j of ϕ, the tuple $\langle \phi, x_i, x_j \rangle$ is a "no" instance of CS. In the case where ϕ is unsatisfiable, $v_\phi(S) = 0$ holds, for each coalitions S. Hence, every player in $\mathcal{G}_\phi$ have the same type. Consider then the case where ϕ is satisfiable, but there is no critical pair $\{x_i, x_j\}$ w.r.t. ϕ. In this latter case, for any chosen pair x_i and x_j, we can apply the same line of reasoning as in the proof of Theorem 6.7 (case (2) of the ($\Leftarrow$)-part), and conclude that x_i and x_j are strategically equivalent. As this holds for each pair of players, we have that all players have the same type. □

6.3 Shedding Light on the "Gray Area"

So far, we have shown tractability results for the classes of k-typed games given in type-based form, and we have pointed out that deciding whether a game is actually k-typed is an intractable problem. There is still a missing piece: What happens if a game is known to be k-typed, but it is not given in type-based form (i.e., the classification of players is actually unknown)? This question is analyzed next.

Our first result is to show that identifying player types is likely to be intractable even on k-typed games having such a bounded number of types. The proofs of intractability results are based here on a complexity-theory setting developed to study problems that are believed to be difficult but could not be classified using the most common reductions (i.e., Karp or Turing reductions).

Consider the problem SAT_1, where we have to decide the satisfiability of a Boolean formula ϕ, under the *promise* that ϕ admits *at most one* satisfying assignment. This is the prototypical **NP**-hard problem *under randomized reductions* [65]. It is widely believed that such problems are not solvable in polynomial time. For our aims here, it is not necessary to expand on the concept of randomized reductions, and we refer the interested reader, for instance, to the work by Mahmoody and Xiao [41]. Indeed, the promise of dealing with a fixed number of player types is next related to SAT_1 via "standard" reductions from this problem, in order to prove the analogue of Theorem 6.7 and Theorem 6.8 for classes of games having bounded types.

Theorem 6.9. *Let $\mathcal{R}$ be a* **P***-representation and let k be a natural number. Consider the k-typed subclass $\mathcal{C}_k(\mathcal{R})$ of the games representable through the representation $\mathcal{R}$. Given a game $\mathcal{G} \in \mathcal{C}_k(\mathcal{R})$, deciding whether two players of $\mathcal{G}$ have the same type, as well as deciding whether $\mathcal{G}$ is actually k'-typed for some $k' < k$, are* co-**NP** *problems. Moreover, there is a* **P***-representation $\bar{\mathcal{R}}$ for which these problems are* co-**NP***-hard under randomized reductions, even on the class $\mathcal{C}_2(\bar{\mathcal{R}})$.*

Proof. Membership results in co-**NP** follow by Theorem 6.7 and Theorem 6.8. Concerning the hardness part, we use the class of 2-typed games $\mathcal{C}_2(\Phi)$ encoded according to the compact representation Φ, and exhibit a polynomial-time reduction from SAT_1 to the considered problem in this class.

Let ϕ' be a Boolean formula over the set X' of variables having one satisfying assignment at most, and define $\phi = \phi' \wedge x_\alpha \wedge \neg x_\beta$ as a Boolean formula over the set $X = X' \cup \{x_\alpha, x_\beta\}$. Note that ϕ has one satisfying assignment at most where, in particular, x_α (resp., x_β) evaluates to true (resp., false). Consider the associated game $\mathcal{G}_\phi$, and observe that if ϕ is unsatisfiable, then $v_\phi(S) = 0$ holds, for each coalition $S \subseteq X$. Thus, in this case, there is only one type of players, and $\mathcal{G}_\phi$ is 1-typed.

Assume now that ϕ is satisfiable, and let $\widetilde{\sigma}$ be its (unique) satisfying truth assignment. Let $\widetilde{S}$ be the coalition such that $\sigma(\widetilde{S}) = \widetilde{\sigma}$, and let x_i and x_j be two arbitrary players. Then, two cases have to be considered:

(1) Assume that x_i and x_j are two players such that $x_i \in \widetilde{S}$ and $x_j \notin \widetilde{S}$. Consider the coalition $\widetilde{T} = \widetilde{S} \setminus \{x_i\}$, and note that $v_\phi(\widetilde{T} \cup \{x_i\}) = 1$ and $v_\phi(\widetilde{T} \cup \{x_j\}) = 0$. Hence, x_i and x_j have two different types.

(2) Assume that either $\{x_i, x_j\} \subseteq \widetilde{S}$ or $\{x_i, x_j\} \cap \widetilde{S} = \varnothing$. Let T be any coalition such that $\{x_i, x_j\} \cap T = \varnothing$. We claim that $v_\phi(T \cup \{x_i\}) = 0$ and $v_\phi(T \cup \{x_j\}) = 0$ hold. Indeed, first observe that $\widetilde{T} \cup \{x_i\} \neq \widetilde{S}$ and $\widetilde{T} \cup \{x_j\} \neq \widetilde{S}$. Then, the claim follows by simply noticing that $\widetilde{S}$ is the one coalition for which $v_\phi(\widetilde{S}) = 1$. Hence, in this case, x_i and x_j have the same type.

By combining the above two cases, we have that players of $\mathcal{G}_\phi$ can be partitioned into exactly two different strategic types: Players in $\widetilde{S}$, and players outside $\widetilde{S}$. Therefore, $\mathcal{G}_\phi$ is 2-typed, but it is not 1-typed. It follows that $\mathcal{G}_\phi$ is 1-typed if and only if ϕ (and, hence, the original formula ϕ') is unsatisfiable. From the **NP**-hardness of SAT_1 under randomized reductions, it follows that, on the class $\mathcal{C}_2(\Phi)$, deciding whether a game is actually 1-typed is co-**NP**-complete under randomized reductions.

Finally, in order to show that deciding whether two players have the same type is co-**NP**-complete under randomized reductions (again, on $\mathcal{C}_2(\Phi)$), it suffices to observe that x_α and x_β have the same type if and only if ϕ (and, hence, ϕ') is unsatisfiable. □

Similarly, we shall next show that computing the nucleolus is hard even on 2-typed games, if the classification of players is not provided in input. Nevertheless, note that the problem is trivial for 1-typed games, because of the equal-treatment of equals by the nucleolus stated in Theorem 6.3.

Theorem 6.10. *There is a* **P***-representation $\bar{\mathcal{R}}$ such that, even on the class $\mathcal{C}_2(\bar{\mathcal{R}})$, computing the nucleolus is* co-**NP***-hard under randomized reductions.*

Proof. Consider again the class of 2-typed games $\mathcal{C}_2(\Phi)$. Moreover, recall the reduction in the proof of Theorem 6.9, based on the Boolean formulae ϕ' over variables in X' and ϕ over $X = X' \cup \{x_\alpha, x_\beta\}$. Define a new game $\bar{\mathcal{G}}_\phi = \langle X, \bar{v}_\phi \rangle$ where $\bar{v}_\phi(X) = 1$ and $\bar{v}_\phi(S) = v_\phi(S)$, for each $S \subset X$. Let z be the imputation assigning the

worth $1/|X|$ to each player. We claim that $z \in \mathscr{N}(\bar{\mathcal{G}}_\phi)$ holds if and only if ϕ is not satisfiable. Indeed, if ϕ is not satisfiable, then $\bar{v}_\phi(S) = 0$, for each $S \subset X$, and it can be easily checked that symmetrically distributing the worth of $\bar{v}_\phi(X)$ over all players leads to the nucleolus. Instead, if ϕ is satisfiable, then there is a coalition S (with $x_\alpha \in S$ and $x_\beta \notin S$) such that $\bar{v}_\phi(S) = 1$. Consider the imputation z' where each player in S (resp., outside S) gets worth $1/|S|$ (resp., 0). Then, $\theta(z') \prec \theta(z)$, and hence $z \notin \mathscr{N}(\bar{\mathcal{G}}_\phi)$. □

7 Conclusions

We studied the problem of computing the nucleolus of coalitional games represented in any polynomial-time compact form. It turns out that this problem is mildly harder than **NP** and co-**NP**, as the nucleolus can be computed in polynomial time by a deterministic Turing transducer exploiting an **NP**-oracle. It is worthwhile noting that we focus on **P**-representations just for the sake of presentation. Indeed, this upper bound extends rather easily to any **FNP**-representation $\mathcal{R}$ (where the worth function $v^{\mathcal{R}}$ is computable in **FNP**, the functional version of **NP**), e.g., to games with synergies among coalitions. To illustrate, just note that in our algorithms, whenever an oracle "guesses" some coalition S, it can also guess its worth $w = v^{\mathcal{R}}(S)$, together with a polynomial-time checkable concise certificate that w is actually the correct value (we refer the interested reader to [27], for more information about **FNP**-representations and the techniques to deal with them).

We then completed the picture about the complexity of the nucleolus in compact games by showing that the above result is tight, because hardness for $\boldsymbol{\Delta}_2^{\boldsymbol{P}}$ holds even for the simple graph-game representation, and hence for any compact representation at least as expressive as graph-games (e.g., for marginal contribution networks).

Besides rather classical combinatorial arguments used for the hardness proofs, the technical ingredients used in the paper comprise a novel theory of succinct systems of linear inequalities (and succinct linear programs), suitably introduced and studied in the paper. We believe that the results obtained for this framework, being defined for such a basic mathematical tool as the systems of linear inequalities, may be quite useful also in contexts and applications very different from game theory.

Finally, we have identified relevant tractable classes of coalitional games (w.r.t. the nucleolus computation), based on the notion of type of a player. Indeed, in most applications where many players are involved, it is often the case that such players do belong in fact to a limited number of classes, which are known in advance and may be exploited for computing solution concepts in a fast way (e.g., think of applications to networking, where players correspond to hardware devices with a limited numbers of possible features, so that we typically have many devices but a few types).

In a future work, we plan to identify further tractable classes of coalitional games where we additionally consider special forms of interactions among players or specific limitations in the ability of forming coalitions and collaborating with each other.

References

[1] K. Aadithya, T. Michalak, and N. Jennings. Representation of coalitional games with algebraic decision diagrams. Technical Report UCB/EECS-2011-8, Department of Electrical Engineering and Computer Sciences, The University of California at Berkeley, 2011.

[2] T. Ågotnes, W. van der Hoek, and M. Wooldridge. Reasoning about coalitional games. *Artificial Intelligence*, 173(1):45–79, 2009.

[3] J. Arin and V. Feltkamp. The nucleolus and kernel of veto-rich transferable utility games. *International Journal of Game Theory*, 26(1):61–73, 1997.

[4] R. J. Aumann and M. Maschler. Game-theoretic analysis of a bankruptcy problem from the Talmud. *Journal of Economic Theory*, 36(2):195–213, 1985.

[5] Y. Bachrach and E. Porat. Path disruption games. In M. Luck, S. Sen, W. van der Hoek, and G. A. Kaminka, editors, *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, pages 1123–1130, Toronto, Canada, 2010.

[6] Y. Bachrach and J. S. Rosenschein. Coalitional skill games. In L. Padgham, D. C. Parkes, J. Müller, and S. Parsons, editors, *Proceedings of the 7th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2008)*, pages 1023–1030, Estoril, Portugal, 2008.

[7] Y. Bachrach, J. S. Rosenschein, and E. Porat. Power and stability in connectivity games. In L. Padgham, D. C. Parkes, J. Müller, and S. Parsons, editors, *Proceedings of the 7th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2008)*, pages 999–1006, Estoril, Portugal, 2008.

[8] L. Blum, F. Cucker, M. Shub, and S. Smale. *Complexity and Real Computation*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1998. ISBN 0-387-98281-7.

[9] R. Brânzei, T. Solymosi, and S. Tijs. Strongly essential coalitions and the nucleolus of peer group games. *International Journal of Game Theory*, 33(3):447–460, 2005.

[10] R. Brânzei, E. Iñarra, S. Tijs, and J. M. Zarzuelo. A simple algorithm for the nucleolus of airport profit games. *International Journal of Game Theory*, 34(2):259–272, 2006.

[11] N. Chen, P. Lu, and H. Zhang. Computing the nucleolus of matching, cover and clique games. In B. S. Jörg Hoffmann, editor, *Proceedings of the 26th National Conference on Artificial Intelligence (AAAI-12)*, pages 1319–1325, Toronto, Ontario, Canada, 2012.

[12] V. Conitzer and T. Sandholm. Computing shapley values, manipulating value division schemes, and checking core membership in multi-issue domains. In D. L. McGuinness and G. Ferguson, editors, *Proceedings of the 19th National Conference on Artificial Intelligence (AAAI-04)*, pages 219–225, San Jose, CA, USA, 2004.

[13] V. Conitzer and T. Sandholm. Complexity of constructing solutions in the core based on synergies among coalitions. *Artificial Intelligence*, 170(6–7):607–619, 2006.

[14] X. Deng and C. H. Papadimitriou. On the complexity of cooperative solution concepts. *Mathematics of Operations Research*, 19(2):257–266, 1994.

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

[16] J. Derks and J. Kuipers. On the core and the nucleolus of routing games. Technical Report 92-07, University of Limburg, Maastricht, Netherlands, 1992.

[17] I. Dragan. A procedure for finding the nucleolus of a cooperativen person game. *Mathematical Methods of Operations Research*, 25(5):119–131, 1981.

[18] E. Elkind, L. A. Goldberg, P. W. Goldberg, and M. Wooldridge. A tractable and expressive class of marginal contribution nets and its applications. *Mathematical Logic Quarterly*, 55(4):362–376, 2009.

[19] E. Elkind, L. A. Goldberg, P. W. Goldberg, and M. Wooldridge. On the computational complexity of weighted voting games. *Annals of Mathematics and Artificial Intelligence*, 56(2):109–131, 2009.

[20] U. Faigle, W. Kern, and J. Kuipers. Computing the nucleolus of min-cost spanning tree games is np-hard. *International Journal of Game Theory*, 27(3):443–450, 1998.

[21] L. Fortnow, R. Impagliazzo, V. Kabanets, and C. Umans. On the complexity of succinct zero-sum games. *Computational Complexity*, 17(3):353–376, 2008. ISSN 1016-3328.

[22] D. B. Gillies. Solutions to general non-zero-sum games. In A. W. Tucker and R. D. Luce, editors, *Contributions to the Theory of Games, Volume IV*, volume 40 of *Annals of Mathematics Studies*, pages 47–85. Princeton University Press, Princeton, NJ, USA, 1959.

[23] D. Granot and F. Granot. On some network flow games. *Mathematics of Operations Research*, 17(4): 792–841, 1992.

[24] D. Granot, M. Maschler, G. Owen, and W. R. Zhu. The kernel/nucleolus of a standard tree game. *International Journal of Game Theory*, 25(2):219–244, 1996.

[25] D. Granot, F. Granot, and W. R. Zhu. Characterization sets for the nucleolus. *International Journal of Game Theory*, 27(3):359–374, 1998.

[26] G. Greco, E. Malizia, L. Palopoli, and F. Scarcello. Non-transferable utility coalitional games via mixed-integer linear constraints. *Journal of Artificial Intelligence Research*, 38:633–685, 2010.

[27] G. Greco, E. Malizia, L. Palopoli, and F. Scarcello. On the complexity of core, kernel, and bargaining set. *Artificial Intelligence*, 175(12–13):1877–1910, 2011.

[28] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*, volume 2 of *Algorithms and Combinatorics*. Springer-Verlag, Berlin Heidelberg, Germany, 2nd edition, 1993.

[29] E. Helly. Über Mengen konvexer Körper mit gemeinschaftlichen Punkten. *Jahresbericht der Deutschen Mathematiker-Vereinigung*, 32:175–176, 1923.

[30] S. Ieong and Y. Shoham. Marginal contribution nets: a compact representation scheme for coalitional games. In J. Riedl, M. J. Kearns, and M. K. Reiter, editors, *Proceedings of the 6th ACM Conference on Electronic Commerce (EC'05)*, pages 193–202, Vancouver, BC, Canada, 2005.

[31] S. Ieong and Y. Shoham. Multi-attribute coalitional games. In J. Feigenbaum, J. Chuang, and D. M. Pennock, editors, *Proceedings of the 7th ACM Conference on Electronic Commerce (EC'06)*, pages 170–179, Ann Arbor, MI, USA, 2006.

[32] D. S. Johnson. A catalog of complexity classes. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, Volume A: Algorithms and Complexity*, pages 67–161. The MIT Press, Cambridge, MA, USA, 1990.

[33] E. Kalai and E. Zemel. On totally balanced games and games of flow. Discussion Paper 413, Northwestern University, Center for Mathematical Studies in Economics and Management Science, Evanston, IL, USA, 1980.

[34] W. Kern and D. Paulusma. Matching games: The least core and the nucleolus. *Mathematics of Operations Research*, 28(2):294–308, 2003.

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

[36] A. Kopelowitz. Computations of the kernels of simple games and the nucleolus of n-person games. Technical Report RM-31, The Hebrew University of Jerusalem, Jerusalem, Israel, 1967.

[37] M. W. Krentel. The complexity of optimization problems. In *Proceedings of the 18th Annual ACM Symposium on Theory of Computing (STOC'86)*, pages 69–76, Berkeley, CA, USA, 1986.

[38] J. Kuipers. A polynomial time algorithm for computing the nucleolus of convex games. Report M 96-12, Maastricht University, Maastricht, The Netherlands., 1996.

[39] S. C. Littlechild. A simple expression for the nucleolus in a special case. *International Journal of Game Theory*, 3(1):21–29, 1974.

[40] S. C. Littlechild and F. Thompson. Aircraft landing fees: A game theory approach. *The Bell Journal of Economics*, 8(1):186–204, 1977.

[41] M. Mahmoody and D. Xiao. On the power of randomized reductions and the checkability of SAT. In D. van Melkebeek, editor, *Proceedings of the 25th Annual IEEE Conference on Computational Complexity (CCC '10)*, pages 64–75, Cambridge, MA, USA, 2010.

[42] 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.

[43] N. Megiddo. Computational complexity of the game theory approach to cost allocation for a tree. *Mathematics of Operations Research*, 3(3):189–196, 1978.

[44] L. Militano, A. Iera, and F. Scarcello. A fair cooperative content-sharing service. *Computer Networks*, 2013.

[45] M. Núñez and C. Rafels. The Böhm-Bawerk horse market: a cooperative analysis. *International Journal of Game Theory*, 33(3):421–430, 2005.

[46] M. J. Osborne and A. Rubinstein. *A Course in Game Theory*. The MIT Press, Cambridge, MA, USA, 1994.

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

[48] G. Owen. On the core of linear production games. *Mathematical Programming*, 9(1):358–370, 1975.

[49] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Dover Publications, 2nd edition, 1998.

[50] D. Paulusma. *Complexity Aspects of Cooperative Games*. PhD thesis, University of Twente, Enschede, The Netherlands, 2001.

[51] 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.

[52] M. Rabin. A note on Helly's theorem. *Pacific Journal of Mathematics*, 5(3):363–366, 1955.

[53] H. Reijnierse. *Games, Graphs and Algorithms*. PhD thesis, University of Nijmegen, the Netherlands, 1995.

[54] H. Reijnierse and J. Potters. The b-nucleolus of tu-games. *Games and Economic Behavior*, 24(1):77–96, 1998.

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

[56] D. Schmeidler. The nucleolus of a characteristic function game. *SIAM Journal of Applied Mathematics*, 17 (6):1163–1170, 1969.

[57] A. Schrijver. *Theory of linear and integer programming*. John Wiley & Sons, New York, NY, USA, 1998.

[58] L. S. Shapley and M. Shubik. The assignment game I: The core. *International Journal of Game Theory*, 1 (1):111–130, 1971.

[59] T. Shrot, Y. Aumann, and S. Kraus. On agent types in coalition formation problems. In M. Luck, S. Sen, W. van der Hoek, and G. A. Kaminka, editors, *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, pages 757–764, Toronto, Canada, 2010.

[60] T. Solymosi. *On computing the nucleolus of cooperative games.* PhD thesis, University of Illinois, Chicago, USA, 1993.

[61] 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.

[62] T. Solymosi, H. Aarts, and T. Driessen. On computing the nucleolus of a balanced connected game. *Mathematics of Operations Research*, 23(4):983–1009, 1998.

[63] T. Solymosi, T. E. S. Raghavan, and S. Tijs. Computing the nucleolus of cyclic permutation games. *European Journal of Operational Research*, 162(1):270–280, 2005.

[64] S. Ueda, M. Kitaki, A. Iwasaki, and M. Yokoo. Concise characteristic function representations in coalitional games based on agent types. In T. Walsh, editor, *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI-11)*, pages 393–399, Barcelona, Spain, 2011.

[65] L. G. Valiant and V. V. Vazirani. NP is as easy as detecting unique solutions. *Theoretical Computer Science*, 47:85–93, 1986.

[66] R. van den Brink, I. Katsev, and G. van der Laan. Computation of the nucleolus for a class of disjunctive games with a permission structure. Technical Report TI 2008-060/3, Tinbergen Institute, Amsterdam, Netherlands, 1998.

[67] J. von Neumann and O. Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, Princeton, NJ, USA, 3rd edition, 1953.

[68] H. P. Young, N. Okada, and T. Hashimoto. Cost allocation in water resources development. *Water Resources Research*, 18(3):463–475, 1982.

A Proofs of Properties in Theorem 2.4

PROPERTY 2.4.(1). *Let S be a coalition such that $S \subseteq N_k \cup \overline{N}_k$ and $v(S) > 0$. Then, $S \cap (N_k \setminus \{\alpha_1\}) \neq \varnothing$ if, and only if, $S \cap \overline{N}_k = \varnothing$.*

Proof. In the light of Lemma 2.3.(B), S cannot include any penalty edges, for otherwise we would have $v(S) < 0$. Therefore we conclude that there is no pair of players $\{p, \overline{q}\} \subseteq S$, with $p \neq \alpha_1$, such that $p \in N_k$ and $\overline{q} \in \overline{N}_k$. Then, we get that either $S \subseteq N_k$ or $S \subseteq \overline{N}_k \cup \{\alpha_1\}$. The result follows since S necessarily includes a positive edge and, hence, at least one player different from α_1. □

PROPERTY 2.4.(2). $\max_{S \subseteq N_k} v(S) = m2^{n+3} + \max_{\sigma \models \phi} \sum_{\alpha_i | \sigma(\alpha_i) = \texttt{true}} 2^i$. *Moreover, let $S_* \subseteq N_k$ be a coalition such that $v(S_*) = \max_{S \subseteq N_k} v(S)$, and let σ_* be the lexicographically maximum satisfying assignment. Then, chall $\in S_*$ and $\sigma_{S_*} = \sigma_*$.*

Proof. Let S_* be the coalition having maximum worth over all the coalitions with players in N_k. Because of Lemma 2.3.(B), S_* cannot cover any penalty edge. Thus, S_* includes only positive edges and, since ϕ is satisfiable, we have $v(S_*) = |C| \times 2^{n+3} + \sum_{\{chall,\alpha_i\}\subseteq S_*} 2^i$, where C is the set of the clause players $c_j \in S_*$ for which exactly one literal player $\ell_{i,j}$ is in S_*; in particular, recall that 2^i is the weight associated with the edge $\{chall, \alpha_i\}$, while 2^{n+3} is the weight associated with each edge of the form $\{c_j, \ell_{i,j}\}$.

Eventually, since ϕ is satisfiable and since $2^{n+3} > \sum_{i=1}^{n} 2^i$, S_* will certainly contain all the m clause players, and hence $|C| = m$. In particular, observe that, for each pair of distinct clauses c_j and $c_{j'}$, and for each variable α_i occurring positively in c_j and negated in $c_{j'}$, $|\{\alpha_{i,j}, \neg\alpha_{i,j'}\} \cap S_*| \leq 1$ holds, in order that no penalty edge is covered. That is, the selection of the literal players induces a satisfying truth assignment (which is possibly partial). Therefore, $v(S_*) = m2^{n+3} + \sum_{\{chall,\alpha_i\}\subseteq S_*} 2^i$. Moreover, the assignment σ_{S_*} associated with S_* is satisfying. Indeed, consider any clause player $c_j \in S_*$, and let $\ell_{i,j}$ be the literal player in S_*. If $\ell_{i,j} = \neg\alpha_{i,j}$, then α_i does not belong to S_*, for otherwise a penalty edge would be covered. Instead, if $\ell_{i,j} = \alpha_{i,j}$, then α_i belongs to S_*, for otherwise we would have $v(S_* \cup \{\alpha_i\}) = v(S_*) + 2^i > v(S_*)$. So, the assignment σ_{S_*} conforms with the truth assignment induced by the selection of the literal players that satisfy all the clauses of ϕ. Hence, σ_{S_*} is satisfying.

Eventually, by the assumption that the assignment mapping all variables to false is not a satisfying one for ϕ, we always have $chall \in S_*$. Moreover, it holds that:

$$v(S_*) = m2^{n+3} + \sum_{\alpha_i|\sigma_{S_*}(\alpha_i)=\texttt{true}} 2^i \leq m2^{n+3} + \max_{\sigma\models\phi} \sum_{\alpha_i|\sigma(\alpha_i)=\texttt{true}} 2^i.$$

To conclude, we claim that σ_{S_*} is the lexicographically maximum satisfying assignment so that the above relationship holds by equality. Indeed, assume, for the sake of contradiction, that a satisfying assignment σ' exists for ϕ such that $v(S_*) < m2^{n+3} + \sum_{\alpha_i|\sigma'(\alpha_i)=\texttt{true}} 2^i$. Based on σ', we can build a coalition S' such that $\{chall, c_1, \ldots, c_m\} \subseteq S'$; $\alpha_i \in S'$, for each α_i such that $\sigma'(\alpha_i) = \texttt{true}$; exactly one literal $\ell_{i,j}$ is in S', for each clause c_j that is satisfied by $\ell_{i,j}$ according to the truth values defined in σ'; and no further player is in S'. By construction and given that σ' is satisfying, no penalty edge is covered by S'. In particular, $v(S') = m2^{n+3} + \sum_{\alpha_i\in S'} 2^i$ and, hence, $v(S') = m2^{n+3} + \sum_{\alpha_i|\sigma'(\alpha_i)=\texttt{true}} 2^i$, which is impossible as we would have a coalition $S' \subseteq N_k$ such that $v(S') > v(S_*) = \max_{S\subseteq N_k} v(S)$. □

PROPERTY 2.4.(3). *Let $S_* \subseteq N_k$ be a coalition with $v(S_*) = \max_{S\subseteq N_k} v(S)$. Then, for each coalition $S \subseteq N_k \cup \overline{N}_k$ with $S \neq S_*$ and $S \neq \overline{S}_*$, $v(S_*) = v(\overline{S}_*) \geq v(S) + 2$ holds. Moreover, for each imputation y, $e(S_*, y) \geq e(S, y) + 1$ and $e(\overline{S}_*, y) \geq e(S, y) + 1$ hold.*

Proof. We first show that the property holds for any coalition S with $S\cap(N_k\setminus\{\alpha_1\}) \neq \varnothing$ and $S\cap\overline{N}_k \neq \varnothing$. Indeed, by Lemma 2.3.(B), it must be the case that $v(S) < 0$, while by Property 2.4.(2) we know that $v(S_*) = v(\overline{S}_*) > 0$, and actually that $v(S_*) = v(\overline{S}_*) > 2$, therefore $v(S_*) = v(\overline{S}_*) \geq v(S) + 2$. Moreover, $e(S, y) = v(S) - y(S) < 0$ holds, for each imputation y. Indeed, y must assign to each player a non-negative payoff because of the individual rationality constraints (recall that $v(\{p\}) = 0$, for each player $p \in N_{\boldsymbol{N}}$). Therefore, we have that $y(S) \geq 0$. Instead, note that $y(S_*) \leq y(N_{\boldsymbol{N}}) = 1$, because of the efficiency of y and given that $v(N_{\boldsymbol{N}}) = 1$ holds, by Lemma 2.3.(C). Then, since $v(S_*) > 2$, we derive that $e(S_*, y) > 1$, and similarly that $e(\overline{S}_*, y) > 1$. By this, since $e(S, y) < 0$, $e(S_*, y) \geq e(S, y) + 1$, and $e(\overline{S}_*, y) \geq e(S, y) + 1$.

Consider, now, a primal coalition $S \subseteq N_k$ with $S \neq S_*$—the same arguments apply to the case of dual coalitions. Let σ_* denote the lexicographically maximum satisfying assignment for ϕ, and let us distinguish two cases. If a satisfying assignment for ϕ exists which is different from σ_*, by exploiting the same line of reasoning as in the proof of Property 2.4.(2), we can derive that the worth of S is bounded by the value $m2^{n+3} + \max_{\sigma\models\phi,\sigma\neq\sigma_*} \sum_{\alpha_i|\sigma(\alpha_i)=\texttt{true}} 2^i$. Then, $v(S) \leq m2^{n+3} + \max_{\sigma\models\phi,\sigma\neq\sigma_*} \sum_{\alpha_i|\sigma(\alpha_i)=\texttt{true}} 2^i \leq m2^{n+3} + \max_{\sigma\models\phi} \sum_{\alpha_i|\sigma(\alpha_i)=\texttt{true}} 2^i - 2 = v(S_*) - 2 = v(\overline{S}_*) - 2$. In particular, note that $\max_{\sigma\models\phi} \sum_{\alpha_i|\sigma(\alpha_i)=\texttt{true}} 2^i$ coincides with $\sum_{\alpha_i|\sigma_*(\alpha_i)=\texttt{true}} 2^i$, because σ_* is the lexicographically maximum satisfying assignment, and observe that $\sum_{\alpha_i|\sigma_*(\alpha_i)=\texttt{true}} 2^i \geq \sum_{\alpha_i|\sigma(\alpha_i)=\texttt{true}} 2^i + 2$ holds, for each assignment $\sigma \neq \sigma_*$ (which is lexicographically smaller). Moreover, by recalling that, for each imputation y and coalition S, $0 \leq y(S) \leq 1$ holds, we get $e(S_*, y) = v(S_*) - y(S_*) \geq v(S_*) - 1 \geq v(S) + 1 \geq e(S, y) + 1$. Similarly, we get $e(\overline{S}_*, y) \geq e(S, y) + 1$.

In order to conclude the proof, consider the second case where σ_* is the only satisfying assignment for ϕ. In this case, by looking again at the arguments in the proof of Property 2.4.(2), it emerges that $v(S) \leq (m-1)2^{n+3} + \sum_{i=1}^{n} 2^i$. Again, we derive that $v(S) \leq v(S_*) - 2 = v(\overline{S}_*) - 2$, which suffices to prove that $e(S_*, y) \geq e(S, y) + 1$ and $e(\overline{S}_*, y) \geq e(S, y) + 1$ hold, for each imputation y, as we have illustrated above. □

PROPERTY 2.4.(4). *For any coalition S, $v(S) = v(S \cap N_r) + v(S \cap (N_k \cup \overline{N}_k))$.*

Proof. The property trivially follows from the fact that $S \subseteq N_k \cup \overline{N}_k \cup N_r$ and since there are no edges between any node in $N_r = \{a, b, \overline{a}, \overline{b}\}$ and any node in $N_k \cup \overline{N}_k$. □

PROPERTY 2.4.(5). *Let $S_* \subseteq N_k$ be a coalition with $v(S_*) = \max_{S \subseteq N_k} v(S)$. Then, the eight coalitions $S_1 = S_* \cup \{a, b\}$, $S_2 = S_* \cup \{\overline{a}, \overline{b}\}$, $S_3 = \overline{S}_* \cup \{a, b\}$, $S_4 = \overline{S}_* \cup \{\overline{a}\,\overline{b}\}$, $S_5 = S_1 \cup \{\overline{a}\}$, $S_6 = S_2 \cup \{a\}$, $S_7 = S_3 \cup \{\overline{a}\}$, and $S_8 = S_4 \cup \{a\}$ are such that $v(S_1) = \cdots = v(S_8) = \max_{S \subseteq N_k \cup \overline{N}_k \cup N_r} v(S) = v(S_*) + \Delta + 2$.*

Proof. Observe that $v(\{a, b\}) = v(\{\overline{a}, \overline{b}\}) = v(\{a, b, \overline{a}\}) = v(\{\overline{a}, \overline{b}, a\}) = \Delta + 2$ and, in fact, $\max_{S \subseteq N_r} v(S) = \Delta + 2$. Therefore, by Property 2.4.(4), $v(S_1) = \cdots = v(S_8) = v(S_*) + \Delta + 2 = v(\overline{S}_*) + \Delta + 2$ and $\max_{S \subseteq N_k \cup \overline{N}_k \cup N_r} v(S) = \max_{S \subseteq N_k \cup \overline{N}_k} v(S) + \Delta + 2$. The result then follows because $v(S_*) = v(\overline{S}_*) = \max_{S \subseteq N_k \cup \overline{N}_k} v(S)$ by Property 2.4.(3). □

PROPERTY 2.4.(6). *For each imputation y and each coalition $S \notin \{S_1, \ldots, S_8\}$, it holds that $e(S_i, y) > e(S, y)$, for each $i \in \{1, \ldots, 8\}$.*

Proof. Consider a generic coalition S and note that, due to Property 2.4.(4), $e(S, y) = e(S \cap N_r, y) + e(S \cap (N_k \cup \overline{N}_k), y)$, for each imputation y. For notational convenience, we denote by W_1 the set $S \cap N_r$, and by W_2 the set $S \cap (N_k \cup \overline{N}_k)$. Thus, $e(S, y) = e(W_1, y) + e(W_2, y)$. Moreover, we denote by S' *any* of the coalitions in the set $\mathcal{S}' = \{\{a, b\}, \{a, b, \overline{a}\}, \{\overline{a}, \overline{b}\}, \{\overline{a}, \overline{b}, a\}\}$, and by S'' *any* of the coalitions in $\mathcal{S}'' = \{S_*, \overline{S}_*\}$. Observe that, for each pair $A, B \in \mathcal{S}'$ of coalitions, $v(A) = v(B) = \max_{S \subseteq N_r} v(S)$ holds. Moreover, recall that $v(S_*) = v(\overline{S}_*) = \max_{S \subseteq N_k \cup \overline{N}_k} v(S)$ and that, for each coalition W, $0 \leq y(W) \leq 1$ holds. Eventually, observe that for each coalition $T \subseteq N_r$ with $T \notin \mathcal{S}'$, $v(T) \leq v(S') - 2$ holds.

At first, we claim that, given a coalition S and any imputation y, $e(W_1, y) \leq e(S', y) + 1$, and $e(W_2, y) \leq e(S'', y) + 1$. Indeed, $e(W_1, y) = v(W_1) - y(W_1) \leq v(W_1) \leq v(S') = v(S') - 1 + 1 \leq v(S') - y(S') + 1 = e(S', y) + 1$. On the other hand, $e(W_2, y) = v(W_2) - y(W_2) \leq v(W_2) \leq v(S'') = v(S'') - 1 + 1 \leq v(S'') - y(S'') + 1 = e(S'', y) + 1$.

Assume that S does not belong to $\{S_1, \ldots, S_8\}$. Then, we have to analyze three cases:

(*i*) Assume that $W_1 \notin \mathcal{S}'$ *and* $W_2 \notin \mathcal{S}''$. In this case, $e(W_1, y) = v(W_1) - y(W_1) \leq v(W_1) \leq v(S') - 2 \leq v(S') - y(S') - 1 = e(S', y) - 1$. Moreover, by Property 2.4.(3), $e(W_2, y) \leq e(S'', y) - 1$. Therefore, $e(S, y) = e(W_1, y) + e(W_2, y) \leq e(S', y) - 1 + e(S'', y) - 1 < e(S', y) + e(S'', y) = e(S_i, y)$, for each $i \in \{1, \ldots, 8\}$.

(*ii*) Assume that $W_1 \notin \mathcal{S}'$ *and* $W_2 \in \mathcal{S}''$. From the above, we already know that, when $W_1 \notin \mathcal{S}'$, $e(W_1, y) \leq e(S', y) - 1$. Recall also that $e(W_2, y) \leq e(S'', y) + 1$ holds. Then, in the case where $e(W_2, y) \leq e(S'', y)$, we immediately get $e(S, y) = e(W_1, y) + e(W_2, y) \leq e(S', y) - 1 + e(S'', y) < e(S', y) + e(S'', y) = e(S_i, y)$, for each $i \in \{1, \ldots, 8\}$.

Consider then the case where $e(W_2, y) > e(S'', y)$, i.e., $v(W_2) - y(W_2) > v(S'') - y(S'')$. Because of $W_2 \in \mathcal{S}''$, $v(W_2) = v(S'')$ and so $y(W_2) < y(S'')$. This implies that $y(S'') > 0$, and hence that $y(N_r) < 1$ because $S'' \subseteq (N_k \cup \overline{N}_k)$ and $N_r \cap (N_k \cup \overline{N}_k) = \varnothing$. Since $S' \subseteq N_r$, we derive that $y(S') < 1$. Therefore, we get $e(W_1, y) = v(W_1) - y(W_1) \leq v(W_1) \leq v(S') - 2 < v(S') - y(S') - 1 = e(S', y) - 1$. So, $e(S, y) = e(W_1, y) + e(W_2, y) < e(S', y) - 1 + e(S'', y) + 1 = e(S', y) + e(S'', y) = e(S_i, y)$, for each $i \in \{1, \ldots, 8\}$.

(*iii*) Assume that $W_1 \in \mathcal{S}'$ *and* $W_2 \notin \mathcal{S}''$. From the above, we know that $e(W_1, y) \leq e(S', y) + 1$ holds. Moreover, by Property 2.4.(3), we know that, when $W_2 \notin \mathcal{S}''$, $e(W_2, y) \leq e(S'', y) - 1$. Then, in the case where $e(W_1, y) \leq e(S', y)$, we immediately get $e(S, y) = e(W_1, y) + e(W_2, y) \leq e(S', y) + e(S'', y) - 1 < e(S', y) + e(S'', y) = e(S_i, y)$, for each $i \in \{1, \ldots, 8\}$.

Consider then the case where $e(W_1, y) > e(S', y)$, i.e., $v(W_1) - y(W_1) > v(S') - y(S')$. Because of $W_1 \in \mathcal{S}'$, $v(W_1) = v(S')$ and so $y(W_1) < y(S')$. This implies that $y(S') > 0$, and hence that $y(N_k \cup \overline{N}_k) < 1$ because $S' \subseteq N_r$ and $N_r \cap (N_k \cup \overline{N}_k) = \varnothing$. As $S'' \subseteq (N_k \cup \overline{N}_k)$, we derive $y(S'') < 1$. Therefore, we get $e(W_2, y) = v(W_2) - y(W_2) \leq v(W_2) \leq v(S'') - 2 < v(S'') - y(S'') - 1 = e(S'', y) - 1$ where, in particular, the inequality $v(W_2) \leq v(S'') - 2$ holds by Property 2.4.(3) and the fact that $W_2 \notin \mathcal{S}''$. So, $e(S, y) = e(W_1, y) + e(W_2, y) < e(S', y) + 1 + e(S'', y) - 1 = e(S', y) + e(S'', y) = e(S_i, y)$, for each $i \in \{1, \ldots, 8\}$.

The proof is completed by observing that the only missing case where $W_1 \in \mathcal{S}'$ *and* $W_2 \in \mathcal{S}''$ is not possible, whenever S does not belong to $\{S_1, \ldots, S_8\}$. □

B On Kopelowitz's approach to Nucleolus Computation

The idea of putting aside all coalitions with constant excesses in the sequence $\mathbf{LP}(\mathcal{G})$ in Definition 3.2 has been described by Maschler et al. [42], who argued (without formal statements) to be a great enhancement over the "original" procedure by Kopelowitz [36], where only coalitions minimizing the maximum excess are considered. In this Appendix we show that, indeed, the latter technique requires in the worst case exponentially more steps than the technique based on $\mathbf{LP}(\mathcal{G})$.

For the sake of completeness, we recall here that the original procedure by Kopelowitz [36] can be formalized via the following succession of linear programs $\mathtt{LP}'_t(\mathcal{G})$, for $t \geq 1$:

$$\begin{array}{lll} \mathtt{LP}'_t(\mathcal{G}) = \min \varepsilon' \;\mid & v(S) - x(S) \leq \varepsilon' & \forall S \subseteq N, S \notin \Lambda_{t-1} \\ & v(S) - x(S) \leq \varepsilon'_{t-1} & \forall S \subseteq N, S \notin \Lambda_{t-2} \\ & \vdots & \\ & v(S) - x(S) \leq \varepsilon'_2 & \forall S \subseteq N, S \notin \Lambda_1 \\ & v(S) - x(S) \leq \varepsilon'_1 & \forall S \subseteq N, S \notin \Lambda_0 = \{\varnothing, N\} \\ & A(\mathcal{G})x \leq b(\mathcal{G}), \text{ i.e., } x \in X(\mathcal{G}), & \end{array}$$

where, for each $r \in \{1, \ldots, t-1\}$, ε'_r is the value of an optimal solution to $\mathtt{LP}'_r(\mathcal{G})$ and $\Lambda_r = \Lambda_{r-1} \cup \{S \subseteq N \mid x(S) = v(S) - \varepsilon'_r$, for each x such that (x, ε'_r) is an optimal solution to $\mathtt{LP}'_r(\mathcal{G})\}$.[11]

While the arguments illustrating the convergence of the above sequence of linear programs to the nucleolus of $\mathcal{G}$ have been provided by Kopelowitz [36], the convergence rate has been formally studied neither by Kopelowitz [36] nor by subsequent works in the literature. In fact, a few authors (see, e.g., [51]) have argued (without proofs) that the approach by Kopelowitz [36] can require, in the worst case, solving up to exponentially many linear programs w.r.t. the number of involved players. However, the fact that a formal analysis of the convergence rate was missing (and the—superficial—similarities with the variant suggested by Maschler et al. [42]) caused confusion in the literature, with the original method and the variant being sometimes used interchangeably.

Note that computing all coalitions whose excesses is constant is an extra effort required in the variant suggested by Maschler et al. [42], which can be quite challenging in the context of compact coalitional games. Therefore, it is relevant to shed lights on the difference between the two approaches.

Specifically, our result below shows that the extra work in the approach by Maschler et al. [42] is unavoidable if we want to have a polynomial (actually, linear) bound on the number of iterations. In fact, we exhibit a class of games over which the basic approach of putting aside only those coalitions minimizing the maximum excess requires exponentially many linear programs to be solved. We argue that this result is of interest on its own to the theory of coalitional games. To formalize our result, define $V'_0(\mathcal{G}) = X(\mathcal{G})$ and, for each $r \in \{1, \ldots, t-1\}$, let $V'_r(\mathcal{G}) = \{x \mid (x, \varepsilon'_r)$ is an optimal solution to $\mathtt{LP}'_r(\mathcal{G})\}$. Note that the following holds:

$$\varepsilon'_r = \min\{\varepsilon' \mid x \in V'_{r-1}(\mathcal{G}) \wedge \; \forall S \subseteq N, S \notin \Lambda_{r-1},\; v(S) - x(S) \leq \varepsilon'\}. \tag{2}$$

Theorem B.1. *There is a class $\{\mathcal{G}_n = \langle N, v\rangle \mid n = |N| \geq 3\}$ of coalitional games such that $V'_t(\mathcal{G}_n) \neq \mathscr{N}(\mathcal{G}_n)$, for each game $\mathcal{G}_n$ and each natural number $t \leq 2^{n-2} - n$.*

Proof. Let $n \geq 3$ and $\mathcal{G}_n = \langle N, v\rangle$ be the n^{th}-game of the class where $N = \{1, \ldots, m, m+1, m+2\}$, with $n = m+2$, and where the worth function v is defined as follows. Let $W_1, W_2, \ldots, W_h$ be an arbitrary fixed ordering over all the subsets of $\{1, \ldots, m\}$ such that $2 \leq |W_i| < m$, for each $i \in \{1, \ldots, h\}$. Note that $h = 2^m - m - 2$ holds. Moreover, let $\mathcal{C}$ be the set of all coalitions S such that $\varnothing \subset S \subset N$, $\{m+1, m+2\} \cap S \neq \varnothing$, and $\{1, \ldots, m\} \cap S \neq \varnothing$, and let

- $v(N) = m + 2$,
- $v(\{j\}) = 1$, for each $j \in \{1, \ldots, m\}$,
- $v(\{m+1\}) = v(\{m+2\}) = 0$,
- $v(\{1, \ldots, m\}) = m$,
- $v(\{m+1, m+2\}) = 2$,

[11] In some works (see, e.g., [25]), the formalization includes the equality $v(S) - x(S) = \varepsilon'_r$, for each $r \in \{1, \ldots, t-1\}$ and coalition $S \in \Lambda_r \setminus \Lambda_{r-1}$. Adding these equalities to the proposed formulation is immaterial, as they are implied by the optimality of ε'_r and the definition of Λ_r.

- $v(W_i) = |W_i| - 1 + 2^{-i}$, $\forall i \in \{1, \ldots, h\}$, and
- $v(S) = -2$ in all other cases, i.e., $\forall S \in \mathcal{C}$.

Consider the linear program associated with the first iteration:

$$\begin{aligned}
\mathtt{LP}'_1(\mathcal{G}_n) = \min \varepsilon' \;\mid\; & 1 - x_j \leq \varepsilon' && \forall j \in \{1, \ldots, m\} \\
& 0 - x_{m+1} \leq \varepsilon' \\
& 0 - x_{m+2} \leq \varepsilon' \\
& m - (x_1 + \cdots + x_m) \leq \varepsilon' \\
& 2 - x_{m+1} - x_{m+2} \leq \varepsilon' \\
& (|W_i| - 1 + 2^{-i}) - x(W_i) \leq \varepsilon' && \forall i \in \{1, \ldots, h\} \\
& -2 - x(S) \leq \varepsilon' && \forall S \in \mathcal{C} \\
& A(\mathcal{G}_n)x \leq b(\mathcal{G}_n), \text{ i.e., } x \in X(\mathcal{G}_n).
\end{aligned}$$

Observe that if x is an imputation, i.e., $x \in X(\mathcal{G}_n)$, then $x_j \geq 1$, for each $j \in \{1, \ldots, m\}$, $x_{m+1} \geq 0$, $x_{m+2} \geq 0$, and $x_1 + \cdots + x_m + x_{m+1} + x_{m+2} = m + 2$ hold. Therefore, the two inequalities $x_1 + \cdots + x_m \geq m - \varepsilon'$ and $x_{m+1} + x_{m+2} \geq 2 - \varepsilon'$ in the definition of $\mathtt{LP}'_1(\mathcal{G}_n)$ and the fact that $x \in X(\mathcal{G}_n)$ immediately imply that the value of any optimal solution to $\mathtt{LP}'_1$ is $\varepsilon'_1 = 0$. By substituting this value in the above inequalities, it follows that $V'_1(\mathcal{G}_n) = \{x \in X(\mathcal{G}_n) \mid x_1 = \cdots = x_m = 1, x_{m+1} \geq 0, x_{m+2} \geq 0, x_{m+1} + x_{m+2} = 2\}$. Moreover, the coalitions achieving the maximum excess in every optimal solution are those in $\Lambda_1 \setminus \Lambda_0 = \{\{1\}, \ldots, \{m\}, \{1, \ldots, m\}, \{m+1, m+2\}\}$. To see that this is the case, just check that for each coalition $S \in \Lambda_1 \setminus \Lambda_0$, it holds that $v(S) - x(S) = \varepsilon'_1 = 0$, for each $x \in V'_1(\mathcal{G}_n)$. Instead, for each coalition $S \notin \Lambda_1 \setminus \Lambda_0$, there is some point $x \in V'_1(\mathcal{G}_n)$ such that $v(S) - x(S) < \varepsilon'_1 = 0$; in particular, observe that for each $i \in \{1, \ldots, h\}$, $x(W_i) = |W_i|$ and, hence, $(|W_i| - 1 + 2^{-i}) - x(W_i) = -1 + 2^{-i} < 0$, for each $x \in V'_1(\mathcal{G}_n)$.

We now claim that at each subsequent step $t = i + 1$, for each $i \in \{1, \ldots, h\}$, it holds that $\varepsilon'_{i+1} = -1 + 2^{-i}$, that $\Lambda_{i+1} \setminus \Lambda_i = \{W_i\}$, and that $V'_{i+1}(\mathcal{G}_n) = V'_1(\mathcal{G}_n)$. To prove the claim, we proceed by induction on the steps of the succession, and to derive the result we make use of Equation 2.

Base Case: Consider, first, the base case where $i = 1$, where Equation 2 leads to the following relationship:

$$\begin{aligned}
\varepsilon'_2 = \{ \min \varepsilon' \;\mid\; & x \in V'_1(\mathcal{G}_n) \wedge \\
& 0 - x_{m+1} \leq \varepsilon' \wedge \\
& 0 - x_{m+2} \leq \varepsilon' \wedge \\
& (|W_i| - 1 + 2^{-i}) - x(W_i) \leq \varepsilon', \forall i \in \{1, \ldots, h\} \wedge \\
& -2 - x(S) \leq \varepsilon', \forall S \in \mathcal{C} \quad \}.
\end{aligned}$$

Since $x_{m+1} + x_{m+2} = 2$, $x_{m+1} \geq 0$, and $x_{m+2} \geq 0$ hold, for each point $x \in V'_1(\mathcal{G}_n)$, the first two inequalities above entail that the optimal value ε'_2 is such that $\varepsilon'_2 \geq -1$. Moreover, note that, for each $x \in V'_1(\mathcal{G}_n)$, $\varepsilon' \geq e(W_i, x) = -1 + 2^{-i}$, $\forall i \in \{1, \ldots, h\}$, and $e(S, x) = -2 - x(S) \leq -2$, $\forall S \in \mathcal{C}$. It follows that $\varepsilon'_2 = -1 + 2^{-1}$ and that W_1 is the coalition achieving this maximum excess. In particular, $e(W_1, x)$ is constant for each point $x \in V'_1(\mathcal{G}_n)$, so that all these points are still optimal solutions, and in this step the region of the candidate imputations is not altered, i.e., $V'_2(\mathcal{G}_n) = V'_1(\mathcal{G}_n)$.

Induction Step: Assume that the claim holds at some step i', with $i' \in \{1, \ldots, h - 1\}$, and consider the case where $i = i' + 1$:

$$\begin{aligned}
\varepsilon'_{i'+2} = \{ \min \varepsilon' \;\mid\; & x \in V'_{i'+1}(\mathcal{G}) = V'_1(\mathcal{G}) \wedge \\
& 0 - x_{m+1} \leq \varepsilon' \wedge \\
& 0 - x_{m+2} \leq \varepsilon' \wedge \\
& (|W_i| + 1 - 2^{-i}) - x(W_i) \leq \varepsilon', \forall i \in \{i', \ldots, h\} \wedge \\
& -2 - x(S) \leq \varepsilon', \forall S \in \mathcal{C} \quad \}.
\end{aligned}$$

Note that we can apply precisely the same line of reasoning as in the base case above in order to conclude that $\varepsilon'_{i'+2} \geq -1$ holds and that $\varepsilon'_{i'+2} = -1 + 2^{-(i'+1)}$ (which in fact is such that $\varepsilon'_{i'+2} < \varepsilon'_{i'+1} = -1 + 2^{-i'}$). Eventually, $W_{i'+1}$ is the coalition achieving the maximum excess, and again we can note that this excess is constant over $V'_{i'+1}(\mathcal{G}_n)$, so that $V'_{i'+2}(\mathcal{G}_n) = V_{i'+1}(\mathcal{G}_n) = V'_1(\mathcal{G}_n)$.

In the light of the above claim, we conclude that, at each step after the initial one, the approach processes some W_i, with $i \in \{1, \ldots, h\}$, without shrinking the set of the candidate imputations. Therefore, $V'_t(\mathcal{G}_n) \neq \mathscr{N}(\mathcal{G}_n)$, for each game $\mathcal{G}_n$ and each natural number $t \leq 2^m - m - 2 = 2^{n-2} - n$. □

C Computational Problems on Compact Representations

Recall the statement of Theorem 4.12: *Let Λ be any compact representation for systems of linear inequalities. On the class $\mathcal{C}(\Lambda)$,* Succint-OptimalValue-Computation, Succint-Solution-Computation, *and* Succint-OptimalSolution-Computation *are in* $\mathbf{F\Delta_2^P}$.

Proof of Theorem 4.12. Let $\gamma^{\Lambda}(Ax \leq b)$ be the encoding of a system $Ax \leq b \in \mathcal{C}(\Lambda)$, with $A \in \mathbb{Q}^{m \times n}$, and such that $\Omega(Ax \leq b)$ is a non-empty polytope.

Succint-OptimalValue-Computation: Recall that the problem asks for computing the value $\min\{c^T\hat{x} \mid \hat{x} \in \Omega(Ax \leq b)\}$, i.e., an optimal solution w.r.t. c, where $c \in \mathbb{Q}^n$ is a vector provided as input together with $\gamma^{\Lambda}(Ax \leq b)$. Note that, since $\Omega(Ax \leq b)$ is a non-empty polytope, we are guaranteed about the existence of an optimal solution $\bar{x}$ that is a basic feasible solution. By Lemma C.1 (reported below), $||\bar{x}||$ is polynomial in the size of the encoding $\gamma^{\Lambda}(Ax \leq b)$, and hence, the value $\min\{c^T\hat{x} \mid \hat{x} \in \Omega(Ax \leq b)\}$ is polynomial too. Therefore, $\min\{c^T\hat{x} \mid \hat{x} \in \Omega(Ax \leq b)\}$ can be computed by means of a binary search over the range of all the possible exponentially-many values, where at each iteration we use an oracle for the problem of checking whether $\min\{c^T\hat{x} \mid \hat{x} \in \Omega(Ax \leq b)\} \leq k$, with k being the current value. We now claim that this problem can be solved in co-**NP**. Then, since the binary search converges after a polynomial number of steps, the fact that Succint-OptimalValue-Computation is feasible in $\mathbf{F\Delta_2^P}$ immediately follows.

To see that the claim holds, first observe a very simple property of compact representations: If Λ is a compact representation for systems of linear inequalities, then the following representation Λ' is compact as well: the systems in the class $\mathcal{C}(\Lambda')$ are encoded as pairs of systems—or sets of linear inequalities—$(\gamma^{\Lambda}(Ax \leq b), (A'x \leq b'))$, with the former compactly encoded according to Λ and the latter listed in some standard extensive way, and where $A \in \mathbb{Q}^{m \times n}$ and $A' \in \mathbb{Q}^{m' \times n}$ with $m' \in O(n^{O(1)})$. Therefore, the number of inequalities listed explicitly are polynomially many, and thus the polynomial time function $\mathcal{L}^{\Lambda'}$ can be easily defined to behave precisely as $\mathcal{L}^{\Lambda}$ over the domain of this function, and to use m' additional numbers to manage a one-to-one correspondence with the inequalities of $A'x \leq b'$.

Then, in order to decide whether $\min\{c^T\hat{x} \mid \hat{x} \in \Omega(Ax \leq b)\} \leq k$, given $\gamma^{\Lambda}(Ax \leq b)$, c, and k, we can build in linear time a new system $\bar{A}x \leq \bar{b}$ encoded as a pair $(\gamma^{\Lambda}(Ax \leq b), \{(c^Tx \leq k)\})$, according to the modified compact representation Λ', as described above. The claim now easily follows from Theorem 4.8, which states that checking whether $\Omega(\bar{A}x \leq \bar{b})$ is not empty is feasible in co-**NP**, and from the fact that $\min\{c^T\hat{x} \mid \hat{x} \in \Omega(Ax \leq b)\} \leq k$ holds if, and only if, $\Omega(\bar{A}x \leq \bar{b}) \neq \varnothing$.

Succint-Solution-Computation: Consider the vector $\bar{x} \in \mathbb{Q}^n$ whose components are defined as follows: $\bar{x}_1 = \min\{x_1 \mid x \in \Omega(Ax \leq b)\}$ and $\bar{x}_j = \min\{x_j \mid x_j \in \Omega(Ax \leq b) \wedge x_1 = \bar{x}_1 \wedge \cdots \wedge x_{j-1} = \bar{x}_{j-1}\}$ for each $2 \leq j \leq n$. Note that $\bar{x}$ is in fact a feasible solution, and that the various components can be incrementally (i.e., from $\bar{x}_1$ to $\bar{x}_n$) computed in $\mathbf{F\Delta_2^P}$ according to the procedure discussed above for Succint-OptimalValue-Computation. Thus, the whole computation is again feasible in $\mathbf{F\Delta_2^P}$.

Succint-OptimalSolution-Computation: In order to solve the problem, we can first compute in $\mathbf{F\Delta_2^P}$ the value $v^* = \min\{c^T\hat{x} \mid \hat{x} \in \Omega(Ax \leq b)\}$ according to the above procedure for Succint-OptimalValue-Computation. Then, consider the system $A'x \leq b'$ encoded by the pair $(\gamma^{\Lambda}(Ax \leq b), \{(c^Tx \leq v^*), (-c^Tx \leq -v^*)\})$, according to the modified compact representation Λ'. Clearly enough, any feasible vector in $\Omega(A'x \leq b')$ is an optimal solution for the input linear program, i.e., a solution to $Ax \leq b$ minimizing c^Tx. It follows that computing such a solution is feasible in $\mathbf{F\Delta_2^P}$, by the above result on Succint-Solution-Computation. □

For the above result, we need to provide a bound on the size of basic feasible solutions. This is a simple result, reported for the sake of completeness only.

Lemma C.1. *Let Λ be any compact representation for systems of linear inequalities. Then, there is a constant $k > 0$ such that, for each system $Ax \leq b \in \mathcal{C}(\Lambda)$ and each basic feasible solution $\bar{x} \in \Omega(Ax \leq b)$, $||\bar{x}|| \leq ||\gamma(Ax \leq b)||^k$.*

Proof. Since $\bar{x}$ is a basic feasible solution, there is a set $I \subseteq \{1, \ldots, m\}$ with $|I| = n$ and such that: $\{\bar{x}\} = \{x \in \mathbb{R}^n \mid A_{i,}x = b_i,\ \forall i \in I\}$. Therefore, the encoding length of $\bar{x}$ is bounded by a polynomial in the size of the inequalities defining the polyhedron $\{x \in \mathbb{R}^n \mid A_{i,}x \leq b_i,\ \forall i \in I\}$ (cf., [49]). In turn, the size of each inequality is polynomially bounded in the size of the encoding, by Definition 4.2. □