Computing the nucleolus of weighted voting games

Edith Elkind[1] Dmitrii Pasechnik[2]

[1]Intelligence, Agents, Multimedia group,
School of Electronics and Computer Science,
University of Southampton

[2]Division of Mathematical Sciences,
School of Physical and Mathematical Sciences,
Nanyang Technological University, Singapore

Workshop on Computational Social Choice, 2008

Outline

1. Introduction
   - Coalitional games
   - Solution concepts
   - The least core and the nucleolus
   - Sequential LPs for nucleolus
2. Solving sequential LPs for WVGs
   - Introduction and related work
   - Our main result
3. Conclusion and future work

Outline

Coalitional games

- Pair (I, ν), where $I = \{1, \ldots, n\}$ - set of *agents*, and $\nu : 2^I \to \mathbb{R}$
  - *Simple games*: $\nu(S) \in \{0, 1\}$ for any $S \subset I$
  - $\nu(S) = 1$ if S is *winning*, otherwise – S is *losing*
- *Payoffs*: $0 \leq p \in \mathbb{R}^n$, normalised: $p(I) := \sum_{i \in I} p_i = 1$
- Want to find “most satisfying” payoffs – *solution concepts*
- Want to be able to specify ν efficiently

Weighted voting games (WVGs)

- $0 \leq w \in \mathbb{R}^n$ – weights, $T > 0$ - threshold
- for $S \subset I$, we have $\nu(S) = \begin{cases} 1 : & w(S) \geq T \\ 0 : & w(S) < T \end{cases}$

Outline

Solution concepts

- Fairness-based, such as Shapley-Shubik index and Banzhaf index
- *Stability*-related, such as core, least core, and nucleolus. Maximising the chances for the grand coalition to stay together, treat each coalition as fairly as possible...

Solution concepts

- Fairness-based, such as Shapley-Shubik index and Banzhaf index
- *Stability*-related, such as core, least core, and nucleolus. Maximising the chances for the grand coalition to stay together, treat each coalition as fairly as possible...

Outline

The ε-core and the least core

Definition

The ε-core of a (I, ν) is the set of all p s.t. $p(S) \geq \nu(S) - \varepsilon$ for all $S \subseteq I$.

In particular, when $\varepsilon = 0$ this is just the *core*, mentioned in an earlier talk today. The core might be empty: let's look at the minimal ε^1 so that the ε^1-core is nonempty (this is called *least core*, $\mathcal{L}_1$) Informally, it minimises, over all the possible p, the unhappiness of the most unhappy coalitions.
What would be the "optimal" payoff in $\mathcal{L}_1$?

The ε-core and the least core

Definition

The ε-core of a (I, ν) is the set of all p s.t. $p(S) \geq \nu(S) - \varepsilon$ for all $S \subseteq I$.

In particular, when $\varepsilon = 0$ this is just the *core*, mentioned in an earlier talk today. The core might be empty: let's look at the minimal ε^1 so that the ε^1-core is nonempty (this is called *least core*, $\mathcal{L}_1$) Informally, it minimises, over all the possible p, the unhappiness of the most unhappy coalitions.
What would be the "optimal" payoff in $\mathcal{L}_1$?

The ε-core and the least core

Definition

The ε-core of a (I, ν) is the set of all p s.t. $p(S) \geq \nu(S) - \varepsilon$ for all $S \subseteq I$.

In particular, when $\varepsilon = 0$ this is just the *core*, mentioned in an earlier talk today. The core might be empty: let's look at the minimal ε^1 so that the ε^1-core is nonempty (this is called *least core*, $\mathcal{L}_1$) Informally, it minimises, over all the possible p, the unhappiness of the most unhappy coalitions.
What would be the "optimal" payoff in $\mathcal{L}_1$?

The ε-core and the least core

Definition

The ε-core of a (I, ν) is the set of all p s.t. $p(S) \geq \nu(S) - \varepsilon$ for all $S \subseteq I$.

In particular, when $\varepsilon = 0$ this is just the *core*, mentioned in an earlier talk today. The core might be empty: let's look at the minimal ε^1 so that the ε^1-core is nonempty (this is called *least core*, $\mathcal{L}_1$) Informally, it minimises, over all the possible p, the unhappiness of the most unhappy coalitions.

What would be the "optimal" payoff in $\mathcal{L}_1$?

The ε-core and the least core

Definition

The ε-core of a (I, ν) is the set of all p s.t. $p(S) \geq \nu(S) - \varepsilon$ for all $S \subseteq I$.

In particular, when $\varepsilon = 0$ this is just the *core*, mentioned in an earlier talk today. The core might be empty: let's look at the minimal ε^1 so that the ε^1-core is nonempty (this is called *least core*, $\mathcal{L}_1$) Informally, it minimises, over all the possible p, the unhappiness of the most unhappy coalitions.
What would be the "optimal" payoff in $\mathcal{L}_1$?

The nucleolus and the deficits

– a particular way to define such an optimal payoff. We try to minimize the unhappiness of all the coalitions, not only the most unhappy ones.

- Let $d_S(p)$, for $S \subset I$ and $p \in \mathcal{L}_1$, be given by $p(S) = \nu(S) + d_S(p)$. This is the *deficit* of S w.r.t. p.
- Sort $S \subset I$ so that $d_{S_1}(p) \leq d_{S_2}(p) \ldots$
- This defines a function $\phi : \mathcal{L}_1 \to \{\text{non-decreasing vectors of length } 2^n\}$
- There will be the lexicographically maximal element d^* in $\phi(\mathcal{L}_1)$.
- The (necessarily unique) $p = \phi^{-1}(d^*)$ is the *nucleolus* of (I, ν)

The nucleolus and the deficits

– a particular way to define such an optimal payoff. We try to minimize the unhappiness of all the coalitions, not only the most unhappy ones.

- Let $d_S(p)$, for $S \subset I$ and $p \in \mathcal{L}_1$, be given by $p(S) = \nu(S) + d_S(p)$. This is the *deficit* of S w.r.t. p.
- Sort $S \subset I$ so that $d_{S_1}(p) \leq d_{S_2}(p) \dots$
- This defines a function $\phi : \mathcal{L}_1 \to \{\text{non-decreasing vectors of length } 2^n\}$
- There will be the lexicographically maximal element d^* in $\phi(\mathcal{L}_1)$.
- The (necessarily unique) $p = \phi^{-1}(d^*)$ is the *nucleolus* of (I, ν)

The nucleolus and the deficits

– a particular way to define such an optimal payoff. We try to minimize the unhappiness of all the coalitions, not only the most unhappy ones.

- Let $d_S(p)$, for $S \subset I$ and $p \in \mathcal{L}_1$, be given by $p(S) = \nu(S) + d_S(p)$. This is the *deficit* of S w.r.t. p.
- Sort $S \subset I$ so that $d_{S_1}(p) \leq d_{S_2}(p) \ldots$
- This defines a function $\phi : \mathcal{L}_1 \to \{\text{non-decreasing vectors of length } 2^n\}$
- There will be the lexicographically maximal element d^* in $\phi(\mathcal{L}_1)$.
- The (necessarily unique) $p = \phi^{-1}(d^*)$ is the *nucleolus* of (I, ν)

The nucleolus and the deficits

– a particular way to define such an optimal payoff. We try to minimize the unhappiness of all the coalitions, not only the most unhappy ones.

- Let $d_S(p)$, for $S \subset I$ and $p \in \mathcal{L}_1$, be given by $p(S) = \nu(S) + d_S(p)$. This is the *deficit* of S w.r.t. p.
- Sort $S \subset I$ so that $d_{S_1}(p) \leq d_{S_2}(p) \ldots$
- This defines a function $\phi : \mathcal{L}_1 \rightarrow \{\text{non-decreasing vectors of length } 2^n\}$
- There will be the lexicographically maximal element d^* in $\phi(\mathcal{L}_1)$.
- The (necessarily unique) $p = \phi^{-1}(d^*)$ is the *nucleolus* of (I, ν)

The nucleolus and the deficits

– a particular way to define such an optimal payoff. We try to minimize the unhappiness of all the coalitions, not only the most unhappy ones.

- Let $d_S(p)$, for $S \subset I$ and $p \in \mathcal{L}_1$, be given by $p(S) = \nu(S) + d_S(p)$. This is the *deficit* of S w.r.t. p.
- Sort $S \subset I$ so that $d_{S_1}(p) \leq d_{S_2}(p) \ldots$
- This defines a function $\phi : \mathcal{L}_1 \to \{\text{non-decreasing vectors of length } 2^n\}$
- There will be the lexicographically maximal element d^* in $\phi(\mathcal{L}_1)$.
- The (necessarily unique) $p = \phi^{-1}(d^*)$ is the *nucleolus* of (I, ν)

The nucleolus and the deficits

– a particular way to define such an optimal payoff. We try to minimize the unhappiness of all the coalitions, not only the most unhappy ones.

- Let $d_S(p)$, for $S \subset I$ and $p \in \mathcal{L}_1$, be given by $p(S) = \nu(S) + d_S(p)$. This is the *deficit* of S w.r.t. p.
- Sort $S \subset I$ so that $d_{S_1}(p) \leq d_{S_2}(p) \ldots$
- This defines a function $\phi : \mathcal{L}_1 \to \{\text{non-decreasing vectors of length } 2^n\}$
- There will be the lexicographically maximal element d^* in $\phi(\mathcal{L}_1)$.
- The (necessarily unique) $p = \phi^{-1}(d^*)$ is the *nucleolus* of (I, ν)

The nucleolus and the deficits

– a particular way to define such an optimal payoff. We try to minimize the unhappiness of all the coalitions, not only the most unhappy ones.

- Let $d_S(p)$, for $S \subset I$ and $p \in \mathcal{L}_1$, be given by $p(S) = \nu(S) + d_S(p)$. This is the *deficit* of S w.r.t. p.
- Sort $S \subset I$ so that $d_{S_1}(p) \leq d_{S_2}(p) \ldots$
- This defines a function $\phi : \mathcal{L}_1 \to \{\text{non-decreasing vectors of length } 2^n\}$
- There will be the lexicographically maximal element d^* in $\phi(\mathcal{L}_1)$.
- The (necessarily unique) $p = \phi^{-1}(d^*)$ is the *nucleolus* of (I, ν)

Outline

LP for the least core

Finding ε^1—what we need for $\mathcal{L}_1$—is a *linear program* (LP)

$$\min_{(p,\varepsilon)} \varepsilon \quad \text{s.t.} \quad \begin{cases} \sum_{i \in I} p_i = 1, \quad p_i \geq 0 \quad \text{for all} \quad i = 1, \ldots, n \\ \sum_{i \in S} p_i \geq \nu(S) - \varepsilon \text{ for all } S \subset I. \end{cases} \tag{1}$$

Let (p^1, ε^1) be an interior optimizer to (1). Let Σ^1 be the set of tight constraints for (p^1, ε^1) : for any $S \in \Sigma^1$ we have $p^1(S) = \nu(S) - \varepsilon^1$. Now we can specify the least core:

$$\mathcal{L}_1 = \begin{cases} p(I) = 1, \quad p \geq 0 \\ p(S) \geq \nu(S) \text{ for all } \Sigma^1 \not\ni S \subset I \\ p(S) = \nu(S) - \varepsilon^1 \text{ for all } S \in \Sigma^1. \end{cases}$$

LP for the least core

Finding ε^1—what we need for $\mathcal{L}_1$—is a *linear program* (LP)

$$\min_{(p,\varepsilon)} \varepsilon \quad \text{s.t.} \quad \begin{cases} \sum_{i \in I} p_i = 1, \quad p_i \geq 0 \quad \text{for all} \quad i = 1, \ldots, n \\ \sum_{i \in S} p_i \geq \nu(S) - \varepsilon \text{ for all } S \subset I. \end{cases} \tag{1}$$

Let (p^1, ε^1) be an interior optimizer to (1). Let Σ^1 be the set of tight constraints for (p^1, ε^1) : for any $S \in \Sigma^1$ we have $p^1(S) = \nu(S) - \varepsilon^1$. Now we can specify the least core:

$$\mathcal{L}_1 = \begin{cases} p(I) = 1, \quad p \geq 0 \\ p(S) \geq \nu(S) \text{ for all } \Sigma^1 \not\ni S \subset I \\ p(S) = \nu(S) - \varepsilon^1 \text{ for all } S \in \Sigma^1. \end{cases}$$

LP for the least core

Finding ε^1—what we need for $\mathcal{L}_1$—is a *linear program* (LP)

$$\min_{(p,\varepsilon)} \varepsilon \quad \text{s.t.} \quad \begin{cases} \sum_{i \in I} p_i = 1, \quad p_i \geq 0 \quad \text{for all} \quad i = 1, \ldots, n \\ \sum_{i \in S} p_i \geq \nu(S) - \varepsilon \text{ for all } S \subset I. \end{cases} \tag{1}$$

Let (p^1, ε^1) be an interior optimizer to (1). Let Σ^1 be the set of tight constraints for (p^1, ε^1) : for any $S \in \Sigma^1$ we have $p^1(S) = \nu(S) - \varepsilon^1$. Now we can specify the least core:

$$\mathcal{L}_1 = \begin{cases} p(I) = 1, \quad p \geq 0 \\ p(S) \geq \nu(S) \text{ for all } \Sigma^1 \not\ni S \subset I \\ p(S) = \nu(S) - \varepsilon^1 \text{ for all } S \in \Sigma^1. \end{cases}$$

LP for the least core

Finding ε^1—what we need for $\mathcal{L}_1$—is a *linear program* (LP)

$$\min_{(p,\varepsilon)} \varepsilon \quad \text{s.t.} \quad \begin{cases} \sum_{i\in I} p_i = 1, \quad p_i \geq 0 \quad \text{for all } \; i = 1, \ldots, n \\ \sum_{i\in S} p_i \geq \nu(S) - \varepsilon \text{ for all } S \subset I. \end{cases} \tag{1}$$

Let (p^1, ε^1) be an interior optimizer to (1). Let Σ^1 be the set of tight constraints for (p^1, ε^1) : for any $S \in \Sigma^1$ we have $p^1(S) = \nu(S) - \varepsilon^1$. Define its lifting to (p, ε)-space:

$$\tilde{\mathcal{L}}_1 = \begin{cases} p(I) = 1, \quad p \geq 0, \quad \varepsilon \geq 0 \\ p(S) \geq \nu(S) - \varepsilon \text{ for all } \Sigma^1 \not\ni S \subset I \\ p(S) = \nu(S) - \varepsilon^1 \text{ for all } S \in \Sigma^1. \end{cases}$$

Sequential LPs for nucleolus

Now we can restrict attention to $\tilde{\mathcal{L}}_1$

$$\varepsilon^2 := \min_{(p,\varepsilon)\in\tilde{\mathcal{L}}_1} \varepsilon. \tag{2}$$

Let (p^2, ε^2) be an interior optimizer to (2). Let Σ^2 be the set of tight constraints for (p^2, ε^2) : for any $S \in \Sigma^2$ we have $p^2(S) = \nu(S) - \varepsilon^2$. Now we can specify the "second" least core:

$$\mathcal{L}_2 = \begin{cases} p(I) = 1, \quad p \geq 0 \\ p(S) \geq \nu(S) \text{ for all } \Sigma^1 \cup \Sigma^2 \not\ni S \subset I \\ p(S) = \nu(S) - \varepsilon^j \text{ for all } S \in \Sigma^j, \quad j = 1, 2. \end{cases}$$

We keep going, specifying $\mathcal{L}_3, \ldots, \mathcal{L}_k = \{p^*\}$. Note that $k < n$, as the dimension goes down.

Sequential LPs for nucleolus

Now we can restrict attention to $\tilde{\mathcal{L}}_1$

$$\varepsilon^2 := \min_{(p,\varepsilon)\in\tilde{\mathcal{L}}_1} \varepsilon. \tag{2}$$

Let (p^2, ε^2) be an interior optimizer to (2). Let Σ^2 be the set of tight constraints for (p^2, ε^2) : for any $S \in \Sigma^2$ we have $p^2(S) = \nu(S) - \varepsilon^2$. Now we can specify the "second" least core:

$$\mathcal{L}_2 = \begin{cases} p(I) = 1, \quad p \geq 0 \\ p(S) \geq \nu(S) \text{ for all } \Sigma^1 \cup \Sigma^2 \not\ni S \subset I \\ p(S) = \nu(S) - \varepsilon^j \text{ for all } S \in \Sigma^j, \quad j = 1, 2. \end{cases}$$

We keep going, specifying $\mathcal{L}_3, \ldots, \mathcal{L}_k = \{p^*\}$. Note that $k < n$, as the dimension goes down.

Sequential LPs for nucleolus

Now we can restrict attention to $\tilde{\mathcal{L}}_1$

$$\varepsilon^2 := \min_{(p,\varepsilon)\in\tilde{\mathcal{L}}_1} \varepsilon. \tag{2}$$

Let (p^2, ε^2) be an interior optimizer to (2). Let Σ^2 be the set of tight constraints for (p^2, ε^2) : for any $S \in \Sigma^2$ we have $p^2(S) = \nu(S) - \varepsilon^2$. Now we can specify the "second" least core:

$$\mathcal{L}_2 = \begin{cases} p(I) = 1, \quad p \geq 0 \\ p(S) \geq \nu(S) \text{ for all } \Sigma^1 \cup \Sigma^2 \not\ni S \subset I \\ p(S) = \nu(S) - \varepsilon^j \text{ for all } S \in \Sigma^j, \quad j = 1, 2. \end{cases}$$

We keep going, specifying $\mathcal{L}_3, \ldots, \mathcal{L}_k = \{p^*\}$. Note that $k < n$, as the dimension goes down.

Sequential LPs for nucleolus

Now we can restrict attention to $\tilde{\mathcal{L}}_1$

$$\varepsilon^2 := \min_{(p,\varepsilon)\in\tilde{\mathcal{L}}_1} \varepsilon. \quad (2)$$

Let (p^2, ε^2) be an interior optimizer to (2). Let Σ^2 be the set of tight constraints for (p^2, ε^2) : for any $S \in \Sigma^2$ we have $p^2(S) = \nu(S) - \varepsilon^2$. Now we can specify the "second" least core:

$$\mathcal{L}_2 = \begin{cases} p(I) = 1, \quad p \geq 0 \\ p(S) \geq \nu(S) \text{ for all } \Sigma^1 \cup \Sigma^2 \not\ni S \subset I \\ p(S) = \nu(S) - \varepsilon^j \text{ for all } S \in \Sigma^j, \quad j = 1, 2. \end{cases}$$

We keep going, specifying $\mathcal{L}_3, \ldots, \mathcal{L}_k = \{p^*\}$. Note that $k < n$, as the dimension goes down.

Oracles for LPs and ellipsoid method

For an (I, ν), these LPs will have $O(2^n)$ constraints, so one cannot, generally speaking, solve them in polynomial time, unless there exists a polynomial-time *separation oracle*

Definition

A separation oracle for a polytope $\mathcal{P} = \{x \in \mathbb{R}^n \mid \langle c_i, x\rangle \leq b_i,\ 1 \leq i \leq k\}$ is an algorithm that, given $y \in \mathbb{R}^n$, checks whether $y \in \mathcal{P}$, and if $y \notin \mathcal{P}$, returns an inequality $\langle c, x\rangle \leq b$ that is valid for $\mathcal{P}$, but $\langle c, y\rangle > b$.

Given such a polytime oracle, one can apply the ellipsoid method to solve LPs over $\mathcal{P}$, as well as e.g. finding vertices, interior points, dimension – all this in polytime.

Oracles for LPs and ellipsoid method

For an (I, ν), these LPs will have $O(2^n)$ constraints, so one cannot, generally speaking, solve them in polynomial time, unless there exists a polynomial-time *separation oracle*

Definition

A separation oracle for a polytope $\mathcal{P} = \{x \in \mathbb{R}^n \mid \langle c_i, x \rangle \leq b_i,\ 1 \leq i \leq k\}$ is an algorithm that, given $y \in \mathbb{R}^n$, checks whether $y \in \mathcal{P}$, and if $y \notin \mathcal{P}$, returns an inequality $\langle c, x \rangle \leq b$ that is valid for $\mathcal{P}$, but $\langle c, y \rangle > b$.

Given such a polytime oracle, one can apply the ellipsoid method to solve LPs over $\mathcal{P}$, as well as e.g. finding vertices, interior points, dimension – all this in polytime.

Oracles for LPs and ellipsoid method

For an (I, ν), these LPs will have $O(2^n)$ constraints, so one cannot, generally speaking, solve them in polynomial time, unless there exists a polynomial-time *separation oracle*

Definition

A separation oracle for a polytope $\mathcal{P} = \{x \in \mathbb{R}^n \mid \langle c_i, x \rangle \leq b_i,\ 1 \leq i \leq k\}$ is an algorithm that, given $y \in \mathbb{R}^n$, checks whether $y \in \mathcal{P}$, and if $y \notin \mathcal{P}$, returns an inequality $\langle c, x \rangle \leq b$ that is valid for $\mathcal{P}$, but $\langle c, y \rangle > b$.

Given such a polytime oracle, one can apply the ellipsoid method to solve LPs over $\mathcal{P}$, as well as e.g. finding vertices, interior points, dimension – all this in polytime.

Oracles for LPs and ellipsoid method

For an (I, ν), these LPs will have $O(2^n)$ constraints, so one cannot, generally speaking, solve them in polynomial time, unless there exists a polynomial-time *separation oracle*

Definition

A separation oracle for a polytope $\mathcal{P} = \{x \in \mathbb{R}^n \mid \langle c_i, x\rangle \leq b_i,\ 1 \leq i \leq k\}$ is an algorithm that, given $y \in \mathbb{R}^n$, checks whether $y \in \mathcal{P}$, and if $y \notin \mathcal{P}$, returns an inequality $\langle c, x\rangle \leq b$ that is valid for $\mathcal{P}$, but $\langle c, y\rangle > b$.

Given such a polytime oracle, one can apply the ellipsoid method to solve LPs over $\mathcal{P}$, as well as e.g. finding vertices, interior points, dimension – all this in polytime.

Outline

Known results.

Polynomial-time algorithms are known for the nucleolus for a number of classes of (I, ν), typically of a combinatorial nature, e.g. flow games, matching games, etc.

For WVG (I, w, T), an algorithm to compute ε_1 is given in [EGGW07]. It runs in time polynomial in n and $\max_i\{w_1, \ldots, w_n\}$, so it is pseudo-polynomial – a truly polynomial-time procedure would depend rather on bitsizes, i.e. on $\log w_1, \ldots, \log w_n$. However, [EGGW07] shows that already computing ε_1 is NP-hard.

Note the parallel with the KNAPSACK problem. It is not a coincidence, as KNAPSACK with weights w is essentially the problem solved by the corresponding separation oracle.

Known results.

Polynomial-time algorithms are known for the nucleolus for a number of classes of (I, ν), typically of a combinatorial nature, e.g. flow games, matching games, etc.

For WVG (I, w, T), an algorithm to compute ε_1 is given in [EGGW07]. It runs in time polynomial in n and $\max_i\{w_1, \ldots, w_n\}$, so it is pseudo-polynomial – a truly polynomial-time procedure would depend rather on bitsizes, i.e. on $\log w_1, \ldots, \log w_n$. However, [EGGW07] shows that already computing ε_1 is NP-hard.

Note the parallel with the KNAPSACK problem. It is not a coincidence, as KNAPSACK with weights w is essentially the problem solved by the corresponding separation oracle.

Known results.

Polynomial-time algorithms are known for the nucleolus for a number of classes of (I, ν), typically of a combinatorial nature, e.g. flow games, matching games, etc.

For WVG (I, w, T), an algorithm to compute ε_1 is given in [EGGW07]. It runs in time polynomial in n and $\max_i\{w_1, \ldots, w_n\}$, so it is pseudo-polynomial – a truly polynomial-time procedure would depend rather on bitsizes, i.e. on $\log w_1, \ldots, \log w_n$. However, [EGGW07] shows that already computing ε_1 is NP-hard.

Note the parallel with the KNAPSACK problem. It is not a coincidence, as KNAPSACK with weights w is essentially the problem solved by the corresponding separation oracle.

Known results.

Polynomial-time algorithms are known for the nucleolus for a number of classes of (I, ν), typically of a combinatorial nature, e.g. flow games, matching games, etc.

For WVG (I, w, T), an algorithm to compute ε_1 is given in [EGGW07]. It runs in time polynomial in n and $\max_i\{w_1, \dots, w_n\}$, so it is pseudo-polynomial – a truly polynomial-time procedure would depend rather on bitsizes, i.e. on $\log w_1, \dots, \log w_n$. However, [EGGW07] shows that already computing ε_1 is NP-hard.

Note the parallel with the KNAPSACK problem. It is not a coincidence, as KNAPSACK with weights w is essentially the problem solved by the corresponding separation oracle.

Outline

Computing the nucleolus of WVGs

Theorem

For a WVG specified by integer weights $w_1, \ldots, w_n$ and a quota T, there exists a procedure that computes its nucleolus in time polynomial in n and $W = \max_i w_i$.

Our algorithm solves the sequence of LPs using the ellipsoid method. The main technical difficulty is thus designing the separation oracles.

Computing the nucleolus of WVGs

Theorem

For a WVG specified by integer weights $w_1, \dots, w_n$ and a quota T, there exists a procedure that computes its nucleolus in time polynomial in n and $W = \max_i w_i$.

Our algorithm solves the sequence of LPs using the ellipsoid method. The main technical difficulty is thus designing the separation oracles.

An oracle for $\tilde{\mathcal{L}}_j$ in WVG

$$\tilde{\mathcal{L}}_j = \begin{cases} \nu(S) = (1 + \mathrm{sign}(w(S) - T))/2, \quad S \subset I \\ p(I) = 1, \quad p \geq 0, \varepsilon \leq \varepsilon^{j-1} \\ p(S) = \nu(S) - \varepsilon^k \text{ for all } S \in \Sigma^k, \quad 1 \leq k \leq j-1 \\ p(S) \geq \nu(S) - \varepsilon \text{ for all } \cup_{k=1}^{j-1} \Sigma^j \not\ni S \subset I \end{cases}$$

An oracle shall be able to tell whether a given (p, ε) belongs to $\tilde{\mathcal{L}}_j$, and return a violated inequality (e.g. just an $S \subset I$). The 2nd and 3rd rows are easy, as one can maintain a "short" equivalent system of linear equations (they can be obtained using the ellipsoid method). The 4th row is complicated – we cannot explicitly list $\Sigma^1, \ldots, \Sigma^{j-1}$.

An oracle for $\tilde{\mathcal{L}}_j$ in WVG

$$\tilde{\mathcal{L}}_j = \begin{cases} \nu(S) = (1 + \text{sign}(w(S) - T))/2, \quad S \subset I \\ p(I) = 1, \quad p \geq 0, \varepsilon \leq \varepsilon^{j-1} \\ p(S) = \nu(S) - \varepsilon^k \text{ for all } S \in \Sigma^k, \quad 1 \leq k \leq j-1 \\ p(S) \geq \nu(S) - \varepsilon \text{ for all } \cup_{k=1}^{j-1} \Sigma^j \not\ni S \subset I \end{cases}$$

An oracle shall be able to tell whether a given (p, ε) belongs to $\tilde{\mathcal{L}}_j$, and return a violated inequality (e.g. just an $S \subset I$). The 2nd and 3rd rows are easy, as one can maintain a "short" equivalent system of linear equations (they can be obtained using the ellipsoid method). The 4th row is complicated – we cannot explicitly list $\Sigma^1, \ldots, \Sigma^{j-1}$.

An oracle for $\tilde{\mathcal{L}}_j$ in WVG

$$\tilde{\mathcal{L}}_j = \begin{cases} \nu(S) = (1 + \mathrm{sign}(w(S) - T))/2, \quad S \subset I \\ p(I) = 1, \quad p \geq 0, \varepsilon \leq \varepsilon^{j-1} \\ p(S) = \nu(S) - \varepsilon^k \text{ for all } S \in \Sigma^k, \quad 1 \leq k \leq j-1 \\ p(S) \geq \nu(S) - \varepsilon \text{ for all } \cup_{k=1}^{j-1} \Sigma^j \not\ni S \subset I \end{cases}$$

An oracle shall be able to tell whether a given (p, ε) belongs to $\tilde{\mathcal{L}}_j$, and return a violated inequality (e.g. just an $S \subset I$). The 2nd and 3rd rows are easy, as one can maintain a "short" equivalent system of linear equations (they can be obtained using the ellipsoid method). The 4th row is complicated – we cannot explicitly list $\Sigma^1, \ldots, \Sigma^{j-1}$.

An oracle for $\tilde{\mathcal{L}}_j$ in WVG

$$\tilde{\mathcal{L}}_j = \begin{cases} \nu(S) = (1 + \mathrm{sign}(w(S) - T))/2, \quad S \subset I \\ p(I) = 1, \quad p \geq 0, \varepsilon \leq \varepsilon^{j-1} \\ p(S) = \nu(S) - \varepsilon^k \text{ for all } S \in \Sigma^k, \quad 1 \leq k \leq j-1 \\ p(S) \geq \nu(S) - \varepsilon \text{ for all } \cup_{k=1}^{j-1} \Sigma^j \not\ni S \subset I \end{cases}$$

An oracle shall be able to tell whether a given (p, ε) belongs to $\tilde{\mathcal{L}}_j$, and return a violated inequality (e.g. just an $S \subset I$). The 2nd and 3rd rows are easy, as one can maintain a “short” equivalent system of linear equations (they can be obtained using the ellipsoid method). The 4th row is complicated – we cannot explicitly list $\Sigma^1, \ldots, \Sigma^{j-1}$.

Naive attempt

We can try to formulate the conditions on $S \subset I$ to provide a separating hyperplane as the following $0-1$ linear feasibility problem:

$$\sum_i p_i^{j-1} x_i > 1 - \varepsilon^{j-1}, \tag{3}$$

$$\sum_i p_i x_i < 1 - \varepsilon, \tag{4}$$

$$\sum_i w_i x_i \geq T, \quad x \in \{0,1\}^n. \tag{5}$$

But this is NP-hard, in general - the bitsizes of p and p^{j-1} are too big!

So off-the-shelf tools won't work here. In (3) we have certainly thrown away a lot of extra information available.

Naive attempt

We can try to formulate the conditions on $S \subset I$ to provide a separating hyperplane as the following $0-1$ linear feasibility problem:

$$\sum_i p_i^{j-1} x_i > 1 - \varepsilon^{j-1}, \tag{3}$$

$$\sum_i p_i x_i < 1 - \varepsilon, \tag{4}$$

$$\sum_i w_i x_i \geq T, \quad x \in \{0,1\}^n. \tag{5}$$

But this is NP-hard, in general - the bitsizes of p and p^{j-1} are too big!

So off-the-shelf tools won't work here. In (3) we have certainly thrown away a lot of extra information available.

Naive attempt

We can try to formulate the conditions on $S \subset I$ to provide a separating hyperplane as the following $0-1$ linear feasibility problem:

$$\sum_i p_i^{j-1} x_i > 1 - \varepsilon^{j-1}, \tag{3}$$

$$\sum_i p_i x_i < 1 - \varepsilon, \tag{4}$$

$$\sum_i w_i x_i \geq T, \quad x \in \{0,1\}^n. \tag{5}$$

But this is NP-hard, in general - the bitsizes of p and p^{j-1} are too big!
So off-the-shelf tools won't work here. In (3) we have certainly thrown away a lot of extra information available.

A counting oracle

compute the top j distinct deficits $d_S(p) := p(S) - \nu(S) + \varepsilon$:

$$m^1 = \max\{d_S(p) \mid S \subseteq I\}$$
$$m^2 = \max\{d_S(p) \mid S \subseteq I, d_S(p) \neq m^1\}$$
$$\dots$$
$$m^j = \max\{d_S(p) \mid S \subseteq I, d_S(p) \neq m^1, \dots, m^{j-1}\}$$

as well as the numbers $n^1, \dots, n^j$ of coalitions that have deficits of $m^1, \dots, m^j$, respectively:

$$n^k = |\{S \mid S \subseteq I, d_S(p) = m^k\}|, \qquad k = 1, \dots, j.$$

Doable by dynamic programming in polynomial in W and n time! If $m^t = \varepsilon^t$ and $n^t = |\Sigma^t|$ for all $t = 1, \dots, j-1$ and $m^j \leq \varepsilon$, then (p, ε) is feasible, otherwise separation-inducing S can be found by a variation of the above.

A counting oracle

compute the top j distinct deficits $d_S(p) := p(S) - \nu(S) + \varepsilon$:

$$m^1 = \max\{d_S(p) \mid S \subseteq I\}$$
$$m^2 = \max\{d_S(p) \mid S \subseteq I, d_S(p) \neq m^1\}$$
$$\dots$$
$$m^j = \max\{d_S(p) \mid S \subseteq I, d_S(p) \neq m^1, \dots, m^{j-1}\}$$

as well as the numbers $n^1, \dots, n^j$ of coalitions that have deficits of $m^1, \dots, m^j$, respectively:

$$n^k = |\{S \mid S \subseteq I, d_S(p) = m^k\}|, \qquad k = 1, \dots, j.$$

Doable by dynamic programming in polynomial in W and n time! If $m^t = \varepsilon^t$ and $n^t = |\Sigma^t|$ for all $t = 1, \dots, j-1$ and $m^j \leq \varepsilon$, then (p, ε) is feasible, otherwise separation-inducing S can be found by a variation of the above.

A counting oracle

compute the top j distinct deficits $d_S(p) := p(S) - \nu(S) + \varepsilon$:

$$
\begin{aligned}
m^1 &= \max\{d_S(p) \mid S \subseteq I\} \\
m^2 &= \max\{d_S(p) \mid S \subseteq I, d_S(p) \neq m^1\} \\
&\dots \\
m^j &= \max\{d_S(p) \mid S \subseteq I, d_S(p) \neq m^1, \dots, m^{j-1}\}
\end{aligned}
$$

as well as the numbers $n^1, \dots, n^j$ of coalitions that have deficits of $m^1, \dots, m^j$, respectively:

$$n^k = |\{S \mid S \subseteq I, d_S(p) = m^k\}|, \qquad k = 1, \dots, j.$$

Doable by dynamic programming in polynomial in W and n time! If $m^t = \varepsilon^t$ and $n^t = |\Sigma^t|$ for all $t = 1, \dots, j-1$ and $m^j \leq \varepsilon$, then (p, ε) is feasible, otherwise separation-inducing S can be found by a variation of the above.

A counting oracle

compute the top j distinct deficits $d_S(p) := p(S) - \nu(S) + \varepsilon$:

$$\begin{aligned} m^1 &= \max\{d_S(p) \mid S \subseteq I\} \\ m^2 &= \max\{d_S(p) \mid S \subseteq I, d_S(p) \neq m^1\} \\ &\dots \\ m^j &= \max\{d_S(p) \mid S \subseteq I, d_S(p) \neq m^1, \dots, m^{j-1}\} \end{aligned}$$

as well as the numbers $n^1, \dots, n^j$ of coalitions that have deficits of $m^1, \dots, m^j$, respectively:

$$n^k = |\{S \mid S \subseteq I, d_S(p) = m^k\}|, \qquad k = 1, \dots, j.$$

Doable by dynamic programming in polynomial in W and n time! If $m^t = \varepsilon^t$ and $n^t = |\Sigma^t|$ for all $t = 1, \dots, j-1$ and $m^j \leq \varepsilon$, then (p, ε) is feasible, otherwise separation-inducing S can be found by a variation of the above.

Conclusion and future work

- Essentially the same procedure provides a pseudo-polynomial time algorithm for the nucleolus of the k-vector WVGs, for a fixed k.
- The oracle developed can be used in a practical implementation of nucleolus computation for WVGs (this, due to poor practical performance of the ellipsoid method, ought to be e.g. a dual simplex cutting plane procedure).
- An approximation algorithm for the nucleolus of WVGs? (For ε_1, this was done in [EGGW07]). This will have to be an *additive* approximation, as it is NP-hard to decide whether the nucleolus payoff of a player is 0.

Conclusion and future work

- Essentially the same procedure provides a pseudo-polynomial time algorithm for the nucleolus of the k-vector WVGs, for a fixed k.
- The oracle developed can be used in a practical implementation of nucleolus computation for WVGs (this, due to poor practical performance of the ellipsoid method, ought to be e.g. a dual simplex cutting plane procedure).
- An approximation algorithm for the nucleolus of WVGs? (For ε_1, this was done in [EGGW07]). This will have to be an *additive* approximation, as it is NP-hard to decide whether the nucleolus payoff of a player is 0.

Conclusion and future work

- Essentially the same procedure provides a pseudo-polynomial time algorithm for the nucleolus of the k-vector WVGs, for a fixed k.
- The oracle developed can be used in a practical implementation of nucleolus computation for WVGs (this, due to poor practical performance of the ellipsoid method, ought to be e.g. a dual simplex cutting plane procedure).
- An approximation algorithm for the nucleolus of WVGs? (For ε_1, this was done in [EGGW07]). This will have to be an *additive* approximation, as it is NP-hard to decide whether the nucleolus payoff of a player is 0.

Conclusion and future work

- Essentially the same procedure provides a pseudo-polynomial time algorithm for the nucleolus of the k-vector WVGs, for a fixed k.
- The oracle developed can be used in a practical implementation of nucleolus computation for WVGs (this, due to poor practical performance of the ellipsoid method, ought to be e.g. a dual simplex cutting plane procedure).
- An approximation algorithm for the nucleolus of WVGs? (For ε_1, this was done in [EGGW07]). This will have to be an *additive* approximation, as it is NP-hard to decide whether the nucleolus payoff of a player is 0.

Conclusion and future work

- Essentially the same procedure provides a pseudo-polynomial time algorithm for the nucleolus of the k-vector WVGs, for a fixed k.
- The oracle developed can be used in a practical implementation of nucleolus computation for WVGs (this, due to poor practical performance of the ellipsoid method, ought to be e.g. a dual simplex cutting plane procedure).
- An approximation algorithm for the nucleolus of WVGs? (For ε_1, this was done in [EGGW07]). This will have to be an *additive* approximation, as it is NP-hard to decide whether the nucleolus payoff of a player is 0.