The nucleolus and related notions in cooperative games

Kincső Kinyó
Supervisor: Tamás Király
May, 2026

Introduction

During my project work, I will deal with cooperative games, which are suitable for modeling many problems like a simple market situation or a bankruptcy problem. We examine how players form a coalition in a given situation, and how the benefits obtained together could be combined later on. If a coalition has already formed, it would be beneficial to ensure that it is not in the interest of the members or even a sub-coalition to break away from the initial one. This is kind of a stable situation, which we can describe with the concept of nucleolus, and we want to approach it as closely as possible. In my project, I will deal with this definition to find this "stable situation" and especially how it can be achieved and calculated in our model in some special cases.

Cooperative games

A cooperative game is defined by two qualities. The **player set** is $\mathbf{N} = \{1, 2, \dots n\}$, where each $S \subseteq N$ subset is called a coalition. We also deal with N as the grand coalition. The other one is the **value function v** : $2^N \to \mathbb{R}$, which describes the value of a coalition such that $v(\emptyset) = 0$. This value is the benefits obtained by the members of the coalition. It is independent from the decisions of those not participating in the coalition. We can define a special type of games called cost games where the players have to pay a certain amount of money during the game. Therefore the value of a coalition will be negative, and we define the **cost function** as $\mathbf{c}(U) = -v(U)$. But the same properties can be used just with some modification. For example if v is superadditive then c is subadditive. We denote a game by (N, v) or (N, c) in the following.

An **allocation** $x \in \mathbb{R}^n$ introduces a method to distribute the obtained benefit by a coalition. This assigns a value to each player but can also be defined for a coalition $x(S) = \sum_{i \in S} x_i$. An allocation x is **efficient** if $\sum_{i \in N} x_i = v(N)$. And x is an **imputation** if it is efficient and $x_i \geq v(\{i\})$ for all i. The set of efficient allocations is denoted by $\mathbf{I}^*(N, v)$ and the imputations with $\mathbf{I}(N, v)$. An imputation is suitable for all one-person coalition. But x can be acceptable for all coalitions: $\sum_{i \in S} x_i \geq v(S)$. If it is true for every coalition $S \subseteq N$ and also efficient, we call it a **core** allocation. With cost games this condition is: $\sum_{i \in S} x_i \leq c(S)$. The set of core allocations is represented by $\mathbf{core}(N, v)$. So, in the case of a core allocation, no coalition would want to leave the big coalition. We also want to find a core distribution that achieve a more stable allocation among the players. For this, we define the **profit** $p(S, x) := x(S) - v(S)$ for each x allocation. The profit vector $\Theta(x) \in \mathbb{R}^{2^n}$ contains all the profit values of different coalitions in non-decreasing order. Profit vectors can be ordered lexicographically. The nucleolus is the only allocation that lexicographically maximizes the profit vector on $I(N, v)$:

$$N(v) = \{x \in I(N, v) | \quad \Theta(y) \preccurlyeq \Theta(x) \quad \forall y \in I(N, v)\}$$

But if we consider cost games the nucleolus maximizes the excess vectors. An **exsess** of a coalition S with respect to x is $exc(S, x) := c(S) - x(S)$. The excess vector is defined the same way as the profit vector just with excesses of coalitions.

The nucleolus always exists and is an element of the core if the core is not empty. Otherwise, it is the allocation that best approximates stability. The lexicographic averaging method is an algorithm ([1]) to find the nucleolus. However, it is a slow process in general.

The standard tree game

Γ tree network is given by a finite (V, E) tree, $|E| = q$. One vertex $o \in V$ has a special role and it is the root of the tree. There is a cost function on the edges: $a : E \to \mathbb{R}$, and also on the nodes of the tree: $b : E \to \mathbb{R}$. So as a cooperative game will consider cost which is the negative of the value of a coalition. We can

imagine this tree as a road system between cities which are represented by the nodes. Each node $p \in V$ has a player set called N_p with n_p residents. The union of all residents will be the player set N. To get a cooperative game we also need to extend the cost function to any coalition. This will require some other notations.

For each node $p \in V \setminus \{o\}$ there exists a unique past from p to the root o. Denote the first edge of this past by e_p and its cost by $a_p = a(e_p)$. The other endpoint of e_p will be $\pi(p)$, the parent of p. And p is parent of d_p many nodes. If $d_p = 0$ then p is a leaf. Define a relation among the nodes. For every node q contained by this unique path from o to p: $p \geqq$ holds. With this relation define the trunks: $T = (V(T), E(T))$ is a subgraph of (V, E), spanned by the nodes $V(T)$ that is closed under the precedence relation.

For trunk T let be the total cost $C(T) = b(o) + \sum_{p \in V(T) \setminus \{o\}} (a_p + b(p))$. With these cost we can define the costs of any coalition $S \subseteq N$. $c(S) = min\{C(T) : T$ is a trunk and $S \subseteq N(T)\}$. Where $N(T)$ contains the union of players in the nodes of $V(T)$. Let T_S denote the trunk spanned by the root and the nodes in which the players of S reside.

Standard tree networks are a specific type of tree networks defined by the following additional properties: $a \geq 0$, $b \geq 0$ and $N_p \neq \emptyset$ for every leaf p. Games generated by standard tree networks are called standard tree games. The following lemma shows that a tree networks where every trunk has non-negative costs coincides the class of standard tree networks.

Lemma([3]): *Let* Γ *be a tree network such that all trunks have non-negative costs. Then there exists a standard tree network* Γ' *that generates the same cost.*

In the proof of this lemma we transfer the costs of the nodes to the edges, and after if there is any edge cost which is negative we transfer this cost along these unique pasts to the root. During this transformation the costs of the trunks doesn't change and the cost of the root also won't be negative because trunks have non-negative costs. And at the end the costs in the root are equal to the initial costs of some trunks. So this proof also gives a method to get a standard tree network, and we will work only with standard tree networks.

Let $\mathcal{E}$ be a certain subset of the coalitions: $N \setminus \{i\}, (i \in N)$ and $N(T_p), (p \in V \setminus \{o\})$. The core and the nucleolus of a standard tree game depend on the coalitions of $\mathcal{E}$ according to the following two lemmas.

Lemma([3]): *Let* Γ *be a standard tree network with player set* N *and let* $x \in \mathbb{R}^n$. *Then* $x \in core(N, c)$ *if and only if* $x(N) = c(N)$ *and* $x(S) \leq c(S)$ *for all* $S \in \mathcal{E}$.

Lemma([3]): *Let* Γ *be a standard tree network then the nucleolus of* (N, c) *is the imputation* x *maximizes lexicographically* $\{exc(S, x) : S \in \mathcal{E}\}$.

Algorithm

During the algorithm in each iteration the costs of the edges are allocated among the players. Players who live in the same village (they are in N_p pf the same p) will pay the same amount of money. We have maximum q iterations because at the end of an iteration at least one edge's cost became zero. And this edge is contracted. To track how much cost are allocated we define $x \in \mathbb{R}^n$ and x_p denotes how much every player residing in p have to pay so far. The output will be x as the nucleolus of the original standard tree game. How to allocate the cost of the edges? Let us introduce the variable $g_p := n_p + d_p - i_p$ grade of $p \in V \setminus \{o\}$ where i_p is 1 if $\pi(p) \neq o$, otherwise it equals to 0. Then calculate $y \in \mathbb{R}^+$ satisfying $a_p - yg_p \geq 0$ for all $p \in V \setminus \{o\}$ with at lest one equality. And those e_p edges will be contracted for which the equality holds for the corresponding p. If we contract e_p we have to move the players residing in p previously and to update the parent map. And we will do it along these unique pasts to the root, so every node of N_p will be residents of $N_{\pi(p)}$. And p became inactive, so we remove it from the set of active nodes V_a which is V at the beginning, the cardinality of V_a is q_a in each iteration. Every player who is contracted to the root are not charged any further. After that increase every x_p by y and decrease the edge costs: $a_p - yg_p$. And we use the function $r : V \setminus \{o\} \to V_a \cup \{o\}$ to keep track of to which verticies the original nodes have been contracted. In every step $\sum_{p \neq o} g_p = \sum_{p \neq o} n_p$ and every edge lost yg_p from its cost and each players cost increase by y. So the allocaiton of the cost of the edges between the playes works.

`Input:` $(V,E); o; \pi; (n_p)_{p\in V}; (a_p)_{p\in V}$

`Initialization:` $x := 0 \in \mathbb{R}^V$; $q_a := q$; $V_a := V\backslash\{o\}$;

for $p \in V_a$ **do**

$d_p := |\{q \in V : \pi(q) = p\}|$

$$i_p := \begin{cases} 1 & \text{if } \pi(p) = o \\ 0 & \text{if } \pi(p) \neq o \end{cases}$$

$r(p) := p$

`Iterations:`

while $q_a > 0$ **do**

for $p \in V_a$ **do** $g_p := n_p + d_p - i_p$

$y := \max\{x \geq 0 : a_p - xg_p \geq 0\}$

for $p \in V\backslash\{o\}$ **do**

if $r(p) \neq o$ **then** $x_p := x_p + y$

for $p \in V_a$ **do** $a_p := a_p - yg_p$

for $p \in V_a$ **do**

if $a_p = 0$ **then**

$V_a := V_a\backslash\{p\}$, $q_a := q_a - 1$

$n_{\pi(p)} := n_{\pi(p)} + n_p$

$d_{\pi(p)} := d_{\pi(p)} + d_p - 1$

for $q \in V_a$ **do if** $\pi(q) = p$ **then** $\pi(q) := \pi(p)$

for $q \in V_a$ **do if** $\pi(q) = p$ **then** $r(q) := \pi(p)$

if $\pi(p) = o$ **then**

for $q \in V_a$ **do if** $\pi(q) = o$ **then** $i_q := 0$

`Output:` **return** x

At the end we get the nucleolus of the standard tree game: $N(v) = x$. In a single iteration every line takes $O(q)$ times and we have $O(q)$ iteration so the running time is $O(q^2)$. It can be improved to $O(g\log(q))$.

Theorem([3]): *The algorithm above computes for a standard tree network* Γ *its nucleolus* $N(v)$*.If* $i \in N_p$ *then* $N(v)_i = x_p$*.*

Painting story

Now we introduce an interpretation of the algorithm. If we consider the tree of the standard tree game as a road system, the nodes are villages, the root is the capital city and the players are the villagers in the cities. And for example the residents are responsible for painting the roads. The cost of an edge is now viewed as the length of the corresponding rode segment which is equal the time of painting that rode. We will describe a way of painting in which each player spends is equal to his nucleolus payoff. Each villager starts painting from the city it lives in to the capital, then back to its residency. If it travels through an unpainted road, it paints it. More players can paint a road simultaneously. If k players paint a road at the same time, they paint it with a speed k km per hour. If a player comes across a painted road, he just skip it and the time to traverse a painted road is negligible.

The difference from the algorithm above is that the algorithm is discrete while in the story, time is continuous. And in the road system we do not contract any edge, just skip the painted roads. But the main idea behind these are similar. If a group of players finish painting a road it can be considered as the end of an iteration in the algorithm.

Future research

In general, determining the nucleolus takes a lot of time, since the number of possible coalitions is exponentially large. As you could see, in the bankruptcy problem, there is an easily understandable and algorithmizable rule for finding the nucleolus. In my upcoming research, I would like to deal with other special cases where the nucleolus can be calculated in similarly simple ways.

References

[1] Forgó Ferenc, Pintér Miklós, Simonovits András, Solymosi Tamás, *Kooperatív játékelmélet*, elektronikus jegyzet, 2006

[2] Solymosi Tamás, *Kooperatív játékok*, Magyar Tudomány, 2009/5, 547-558

[3] Michael Maschler, Jos Potters, Hans Reijnierse, *The nucleolus of a standard tree game revisited: a study of its monotonicity and computational properties*, International Journal of Game Theory 39, 89-104, 2010