Санкт-Петербургский государственный университет
Кафедра математической теории игр и статистических решений

Ролдугина Анна Андреевна

Выпускная квалификационная работа бакалавра

Коалиционная устойчивость эксцессоподобных решений в выпуклых кооперативных играх

Направление 01.03.02

Прикладная математика и информатика

Заведующий кафедрой,
доктор физ.-мат. наук,
профессор
Петросян Л. А.

Научный руководитель,
кандидат физ.-мат. наук,
доцент
Тарашнина С. И.

Санкт-Петербург
2018

Оглавление

Введение

Кооперативные игры — класс игр, в которых игрокам позволено заключать союзы для увеличения потенциальных выигрышей. В играх с трансферабельными полезностями данные выигрыши должны оцениваться в единицах, общих для всех игроков, и могут быть произвольно поделены между игроками. Способы распределения между всеми участниками игры общего выигрыша, полученного в результате кооперации, многочисленны и поэтому являются основным предметом изучения кооперативной теории. Эти способы распределения общего выигрыша называются решениями кооперативной игры. Наиболее изученными концепциями решения в кооперативной теории игр являются C-ядро (Scarf, [11]), N-ядро (Schmeidler, [12]) и вектор Шепли (Shapley, [13]). Сравнительно недавно введены новые концепции решения, такие как SM-ядро (Tarashnina, [15]) и множество α-N-ядер (Smirnova, Tarashnina, [5]). Такая многочисленность решений объясняется различными наборами свойств решений, благодаря которым каждое из решений более предпочтительно в некоторых аспектах в сравнении с другими.

В качестве меры предпочтения в данной работе исследуется принадлежность решения C-ядру, как множеству недоминируемых дележей. Этот подход обусловлен устойчивостью решения в том смысле, что игрокам, целью которых является максимизация собственного выигрыша, невыгодно отказываться от такого распределения из-за возможности получить меньше предложенного.

В общем случае C-ядро может содержать бесконечно много решений, а значит выбор наиболее приемлемого из них становится дополнительной проблемой. Такое решение игры как вектор Шепли позволяет построить наиболее справедливый дележ, учитывающий все возможные вклады игроков в общую прибыль коалиций. Однако в общем случае он может не обладать свойством коалиционной устойчивости.

На классе выпуклых игр C-ядро всегда непусто, и в таком случае появляется возможность изучения известных решений на принадлежность C-ядру. Целью настоящей работы является исследование на коалиционную устойчивость множества α-N-ядер $\mathcal{N}(v)$ и разработка про-

граммных средств для нахождения данного решения. Рассматриваемое решение учитывает как конструктивную, так и блокирующую силу коалиций. Также интересен тот факт, что при $\alpha = 1$ оно совпадает с пред-N-ядром, при $\alpha = 0{,}5$ — с SM-ядром, а при $\alpha = 0$ — с анти-пред-N-ядром.

В связи с этим возникает ряд задач:

- проанализировать различные концепции решения кооперативной ТП-игры;
- изучить свойство коалиционной устойчивости решений выпуклых кооперативных игр;
- разработать программный продукт для нахождения α-N-ядер в игре n лиц и их проверки на коалиционную устойчивость;
- разработать программный продукт для нахождения множества α-N-ядер игры трех лиц с учетом его геометрической структуры.

В ходе исследования получены результаты, позволяющие определять коалиционную устойчивость точечных решений. Также рассмотрены примеры выпуклых кооперативных игр, иллюстрирующие полученные выводы.

В главе 1 даются основные понятия и определения, касающиеся кооперативной теории игр и ее решений, сформулированы и доказаны теоремы и определены классы игр с коалиционно устойчивыми решениями. В главе 2 приведены описания функций программной реализации алгоритмов поиска решений и проверки выполнения условий выпуклости и коалиционной устойчивости.

Обзор литературы

Вопрос выбора приемлемого решения в кооперативной игре исследуется многими известными учеными в области теории игр. Наиболее известные способы распределения выигрыша в кооперативной игре, такие как C-ядро, пред-N-ядро и вектор Шепли, были введены в [11, 12, 13]. В работе [14] доказывается теорема о непустоте C-ядра выпуклой игры, что позволяет проверять коалиционную устойчивость решений кооперативной игры на данном классе игр. Работы [2, 4] содержат результаты, позволяющие определить геометрическую структуру C-ядра как выпуклого замкнутого множества и находить его вершины для выпуклых игр.

Введенное в [12] пред-N-ядро основывается на понятии эксцесса коалиции, которое приводится в [3, 9]. Теорема, доказанная Э. Колбергом в работе [8], позволяет находить пред-N-ядро игры с помощью сбалансированных наборов коалиций.

На основе понятия пред-N-ядра в работах [5, 15] были введены новые концепции решения, такие как SM-ядро и множество α-N-ядер, учитывающие разные соотношения конструктивной и блокирующей сил коалиций. Далее, в работах [6, 7] Н. Смирнова и С. Тарашнина исследуют свойства новых решений, в том числе доказывается теорема о геометрической структуре множества α-N-ядер как последовательно соединенных своими концами отрезков. На исследование именно этого решения делается акцент в данной работе.

Поскольку основой множества α-N-ядер является пред-N-ядро, построение алгоритма его нахождения является важной задачей теории кооперативных игр. Существует множество способов нахождения пред-N-ядра. В основе большинства из них лежит метод, предложенный в работе [9]. Метод состоит в последовательном решении не более чем 4^n задач линейного программирования с 2^n ограничениями относительно 2^n переменных. В работе С. Бритвина и С. Тарашниной [1] предложен новый алгоритм нахождения пред-N-ядра, посредством решения всего одной задачи линейного программирования с $(n+1)$ ограничениями относительно 2^n переменных. В данной работе был программно реализован

алгоритм, предложенный Х. Майнхардом в [10]. Также, был реализован результат работы Н. Смирновой и С. Тарашниной, состоящий в аналитически найденном множестве α-N-ядер кооперативной игры трех лиц.

Глава 1. Кооперативная игра

1.1. Основные понятия и определения

Приведем основные определения и понятия кооперативной теории игр, которые потребуются в дальнейшем. Рассмотрим класс кооперативных игр n лиц с трансферабельными полезностями (ТП-игр). Обозначим через $N = \{1, 2, \ldots, n\}$ конечное непустое множество игроков. Коалицией будем называть любое подмножество S из N. Под характеристической функцией игры будем понимать вещественнозначную функцию $v : 2^N \to R^1 : v(\varnothing) = 0$. Тогда пара (N, v) задает кооперативную ТП-игру.

Множество всех ТП-игр с фиксированным множеством игроков N обозначим через G^N. Пусть игра $(N, v) \in G^N$. Полагая, что игроки сформировали максимальную коалицию N, рассмотрим задачу распределения величины $v(N)$ между всеми игроками.

Будем использовать обозначение $x(S) = \sum\limits_{i \in S} x_i,\ S \subseteq N$.

Определение 1. *Множество допустимых векторов выигрышей в игре* (N, v) *есть множество*

$$X^*(v) = \{x \in R^N : x(N) \leqslant v(N)\}.$$

Определение 2. *Множеством эффективно-рациональных распределений в игре* (N, v) *называется множество*

$$X^0(v) = \{x \in R^N : x(N) = v(N)\}.$$

Определение 3. *Вектор* $x = (x_1, \ldots, x_N)$ *будем называть дележом, если он удовлетворяет условиям:*

$$x_i \geqslant v(\{i\}),\ i \in N \quad (\textit{индивидуальная рациональность}),$$

$$x(N) = v(N) \quad (\textit{групповая рациональность / Парето-оптимальность}).$$

Множество всех дележей в игре (N, v) будем обозначать $I(v)$.

Далее определим понятие решения кооперативной ТП-игры.

Определение 4. *Решением на множестве игр* G^N *называется отображение* f*, которое каждой игре* $(N, v) \in G^N$ *ставит в соответствие подмножество* $f(v)$ *множества* $X^*(v)$.

Решения кооперативных игр могут быть одноточечными или множественными. Например, такие решения как вектор Шепли, пред-N-ядро, α-N-ядро и SM-ядро являются одноточечными, а множество α-N-ядер и C-ядро относятся к множественным.

1.2. C-ядро

Одним из вариантов понимания оптимальности решения в кооперативной ТП-игре является понятие C-ядра [13]. Введем определение C-ядра, основанное на понятии доминируемости дележей.

Определение 5. *Будем говорить, что дележ x доминирует дележ y по коалиции S ($x \succ^S y$), если*

$$x_i > y_i, \quad i \in S, \tag{1}$$

$$x(S) \leqslant v(S). \tag{2}$$

Первое условие означает, что дележ x лучше дележа y для всех членов коалиции S, второе — отражает реализуемость дележа x коалицией S.

Заметим, что доминирование невозможно по одноэлементным и максимальной (множеству всех игроков) коалициям, так как возникают противоречия с условиями (1) и (2), соответственно.

Определение 6. *Будем говорить, что дележ x доминирует дележ y ($x \succ y$), если существует такая коалиция $S \subset N$, для которой $x \succ^S y$.*

Доминирование возможно одновременно по разным коалициям, например, дележ x доминирует дележ y по коалиции S, а дележ y доминирует дележ x по коалиции T. При этом доминирование не обладает свойством транзитивности, т.е. из того, что $x \succ y$ и $y \succ z$, не следует $x \succ z$.

Определение 7. *Множество недоминируемых дележей кооперативной игры (N, v) называется ее C-ядром.*

C-ядро игры (N, v) будем обозначать $C(v)$.

Следующая теорема дает нам необходимое и достаточное условия принадлежности дележа C-ядру.

Теорема Бондаревой—Шепли. *Для того, чтобы дележ x принадлежал C-ядру игры (N, v), необходимо и достаточно выполнение следующих неравенств для всех $S \subset N$:*

$$x(S) \geqslant v(S).$$

Из теоремы Бондаревой—Шепли следует, что C-ядро любой кооперативной ТП-игры является замкнутым выпуклым многогранником — подмножеством множества дележей.

Одним из недостатков C-ядра является его пустота для некоторых классов игр. Поэтому имеет смысл проводить проверку C-ядра на пустоту.

Определение 8. *Кооперативная игра (N, v) называется выпуклой (супермодулярной), если для любых коалиций $S, T \subset N$ выполняется условие*

$$v(S) + v(T) \leqslant v(S \cup T) + v(S \cap T). \tag{3}$$

Приведем здесь теорему, доказанную Л. Шепли в [14] для выпуклых игр.

Теорема 1. *Если кооперативная игра (N, v) выпуклая, то ее C-ядро непусто.*

Для более точного представления структуры C-ядра как выпуклого замкнутого многогранника напомним некоторые результаты.

Угловыми или *крайними* точками выпуклого множества называются точки, которые не являются выпуклой линейной комбинацией произвольных точек этого же множества.

Выпуклым многоугольником называется выпуклое, замкнутое, ограниченное множество на плоскости, имеющее конечное число угловых точек.

Выпуклым многогранником называется часть пространства, расположенная по одну сторону от плоскости каждой его грани и ограниченная совокупностью конечного числа плоских многоугольников, соединенных таким образом, что каждая сторона любого многогранника является стороной ровно одного многоугольника.

Для точного определения положения многогранника в пространстве и взаимного расположения его вершин, необходимо найти способ вычисления координат вершин C-ядра игры (N, v) по ее характеристической функции v.

Рассмотрим некоторую перестановку множества игроков $\{1, 2, \ldots, n\}$: $p = (i_1, \ldots, i_n)$ и приведем теорему, доказанную в [2].

Теорема 2. *Вершиной C-ядра будет являться вектор $\pi(p) = (\pi_{i_1}, \ldots, \pi_{i_n})$, компонентами которого являются вклады игроков*

$$\pi_{i_k} = v(i_1, \ldots, i_k) - v(i_1, \ldots, i_{k-1}),\ k = \overline{1, n}, \tag{4}$$

где $v(i_1, \ldots, i_k)$ — выигрыш k-элементной коалиции, участниками которой являются первые k игроков перестановки p.

Отсюда следует, что C-ядро выпуклой игры может иметь $n!$ вершин. Выпишем достаточное условие [4], при котором C-ядро имеет ровно $n!$ вершин:

Теорема 3. *Если характеристическая функция игры (N, v) удовлетворяет условию*

$$v(S) + v(T) < v(S \cup T) + v(S \cap T), \tag{5}$$

где $S \backslash T \neq \emptyset$, $T \backslash S \neq \emptyset$, то все дележи (4) различны.

Проиллюстрируем несколько возможных ситуаций расположения вершин C-ядра на множестве эффективно-рациональных дележей $X^0(v)$.

Пример 1. *Рассмотрим выпуклую игру трех лиц:*

$$v(1) = 1, v(2) = 3, v(3) = 2, v(1,2) = 4, v(1,3) = 3, v(2,3) = 5, v(N) = 6.$$

C-ядро этой игры определяется системой

$$\begin{cases} x_1 & \geqslant & 1 \\ x_2 & \geqslant & 3 \\ x_3 & \geqslant & 2 \\ x_1 + x_2 & \geqslant & 4 \\ x_1 + x_3 & \geqslant & 3 \\ x_2 + x_3 & \geqslant & 5 \\ x_1 + x_2 + x_3 & = & 6 \end{cases}$$

Пусть $p_1 = (1, 2, 3)$ *— некоторая перестановка множества* $N = \{1, 2, 3\}$. *Тогда из (4) получаем*

$$\pi_1^1 = v(1) - v(\emptyset) = 1,\ \pi_2^1 = v(1,2) - v(1) = 3,\ \pi_3^1 = v(N) - v(1,2) = 2.$$

Следовательно, перестановке p_1 *соответствует вершина* $\pi^1 = (1, 3, 2)$. *Рассмотрев другую перестановку* $p_2 = (2, 1, 3)$, *получаем*

$$\pi_2^2 = v(2) - v(\emptyset) = 3, \pi_1^2 = v(1,2) - v(2) = 1, \pi_3^2 = v(N) - v(1,2) = 2,$$

т.е. $\pi^1 = \pi^2$. *Перебирая остальные перестановки, получаем что все они определяют одну и ту же точку. Следовательно,* C*-ядро состоит из единственного дележа* $(1, 3, 2)$.

Пример 2. *Пусть* $N = \{1, 2, 3\}$. *Характеристическая функция игры имеет вид:*

$$v(1) = v(2) = v(3) = 1,\ v(1,2) = v(1,3) = v(2,3) = 3,\ v(N) = 7.$$

Эта игра удовлетворяет условию строгой выпуклости (5). Приведем все возможные перестановки и соответствующие им вершины C*-ядра:*

$$\begin{aligned}
p_1 &= (1,2,3), \quad \pi_1 = (1,2,4);\\
p_2 &= (1,3,2), \quad \pi_2 = (1,4,2);\\
p_3 &= (2,1,3), \quad \pi_3 = (2,1,4);\\
p_4 &= (2,3,1), \quad \pi_4 = (4,1,2);\\
p_5 &= (3,1,2), \quad \pi_5 = (2,4,1);\\
p_6 &= (3,2,1), \quad \pi_6 = (4,2,1).
\end{aligned}$$

Таким образом, получили 6 различных вершин и убедились, что при выполнении условия строгой выпуклости игры C*-ядро имеет* $n!$ *вершин.*

1.3. Пред-N-ядро

В работе рассматриваются эксцессоподобные решения кооперативной ТП-игры. Введем здесь понятие N-ядра игры, основанное на понятии эксцесса коалиции [3, 9]. Эксцесс коалиции показывает меру неудовлетворенности коалиции своим выигрышем.

Определение 9. *Эксцессом коалиции $S \subseteq N$ для произвольного $x \in X^0(v)$ будем называть величину*

$$e(x, v, S) = v(S) - x(S).$$

Определение N-ядра [12] впервые было введено Д. Шмайдлером (D. Schmeidler) в 1969 году.

Определение 10. *N-ядром относительно множества $X \subset X^0(v)$ называется множество*

$$N(v) = \{x \in X : \theta(e(x, v, S)_{S \subseteq N}) \preceq_{lex} \theta(e(y, v, S)_{S \subseteq N}) \text{ для всех } y \in X\},$$

где $\theta(e(x, v, S)_{S \subseteq N})$ — вектор эксцессов, расположенных в порядке невозрастания. Если $X = X^0(v)$, то $N(v)$ называется пред-N-ядром игры (N, v).

Определение 10 можно интерпретировать так: N-ядро есть дележ, обеспечивающий коалициям наименьшую степень неудовлетворенности, выраженную в виде эксцесса. В [12] Д. Шмайдлером было доказано, что пред-N-ядро игры (N, v) всегда существует и состоит из единственного вектора. Обозначим это единственное решение через $\nu^1(v)$. Результат, позволяющий характеризовать пред-N-ядро с помощью сбалансированных наборов коалиций, был получен Е. Колбергом в 1971 году [8].

Определение 11. *Сбалансированным набором коалиций называется набор коалиций $\mathcal{B}$, если существуют такие числа $\lambda_S > 0, S \in \mathcal{B}$, что для всех $i \in N$ выполняется:*

$$\sum_{S \in \mathcal{B}: i \in S} \lambda_S = 1.$$

Для произвольной игры (N, v), ее вектора выигрышей $x \in X^0(v)$ и числа $\gamma \in R^1$ через $\mathcal{B}_\gamma(x)$ обозначим следующий набор коалиций:

$$\mathcal{B}_\gamma(x) = \{S \subsetneqq N : e(x, v, S) \geqslant \gamma\}.$$

Тогда справедлива следующая теорема, позволяющая находить пред-N-ядро в игре n лиц.

Теорема 4 (Колберг). *Для того чтобы* $x = \nu^1(v)$ *необходимо и достаточно, чтобы наборы* $\mathcal{B}_\gamma(x)$ *были пусты или сбалансированны для всех* γ.

Теорема означает, что только когда для любого $\gamma \in R^1$ набор коалиций с эксцессами не меньшими γ является сбалансированным, решение будет являться пред-N-ядром.

Теорема 5. *Если в произвольной игре* (N, v) C*-ядро непусто, то пред-*N*-ядро содержится в нем, т.е.* $\nu^1(v) \in C(v)$.

Вернемся теперь к выпуклым играм и рассмотрим некоторые результаты, связанные с понятием пред-N-ядра. По теореме 1 C-ядро выпуклой игры непусто. Связывая понятие C-ядра с понятием эксцесса можно сделать вывод, что C-ядро состоит из всех дележей x таких, что $e(x, v, S) \leqslant 0$ для всех коалиций S.

Теорему 5 можно дополнить результатами из [2]: если игра (N, v) выпуклая, то пред-N-ядро занимает центральное положение в C-ядре.

1.4. α-N-ядро

Определим теперь понятие α-N-ядра [6]. Будем оценивать силу коалиции $S \subseteq N$ в игре (N, v) двойственным образом. Будем считать, что коалиция S обладает конструктивной силой, гарантированно обеспечивая себе выигрыш $v(S)$. В то же время коалиция S может препятствовать образованию максимальной коалиции N в игре (N, v). В таком случае будем говорить, что коалиция S обладает блокирующей силой, оцениваемой величиной $v^*(S) = v(N) - v(N\backslash S)$. Эту величину $v^*(S)$ можно понимать как вклад коалиции S в максимальную коалицию N при ее объединении с игроками из $(N\backslash S)$. Иными словами, $v^*(S)$ есть ценность коалиции S для всего сообщества игроков. Таким образом, α-N-ядро позволяет учитывать произвольные соотношения конструктивной и блокирующей сил коалиций с весами α и $(1 - \alpha)$, соответственно.

Рассмотрим игру $(N, v) \in G^N$. *Двойственная* к (N, v) игра (N, v^*) задается по правилу

$$v^*(S) = v(N) - v(N\backslash S), \quad S \subseteq N.$$

Введем определение α-эксцесса коалиции, на котором базируются понятия α-N-ядра и множества α-N-ядер $\mathcal{N}(v)$.

Определение 12. *α-эксцессом коалиции $S \subseteq N$ относительно $x \in X^0(v)$ для фиксированного $\alpha \in R^1$ называется величина*

$$e^\alpha(x, v, S) = \alpha e(x, v, S) + (1 - \alpha)e(x, v^*, S). \tag{6}$$

В соответствии с результатами [5] формулу (6) можно записать в виде:

$$e^\alpha(x, v, S) = \big(\alpha v(S) + (1 - \alpha)v^*(S)\big) - x(S) = e(x, v^\alpha, S), \quad S \subseteq N,$$

где $v^\alpha(S) = \alpha v(S) + (1 - \alpha)v^*(S)$.

Определение 13. *Для фиксированного $\alpha \in R^1$ α-N-ядром игры (N, v) называется множество векторов:*

$$\mathcal{N}^\alpha(v) = \{x \in X^0(v) : \theta(e^\alpha(x, v, S)_{S\subseteq N}) \preceq_{lex} \theta(e^\alpha(y, v, S)_{S\subseteq N}),\ y \in X^0(v)\},$$

где $\theta(e^\alpha(x, v, S)_{S\subseteq N})$ — вектор, компоненты которого (α-эксцессы) расположены в порядке невозрастания.

Определение 14. *$[0, 1]$-N-ядром игры (N, v) на множестве $X^0(v)$ будем называть множество всех α-N-ядер игры (N, v) для $\alpha \in [0, 1]$ и обозначать $\mathcal{N}^{[0,1]}(v)$, т.е.*

$$\mathcal{N}^{[0,1]}(v) = \bigcup_{\alpha\in[0,1]} \mathcal{N}^\alpha(v).$$

В [5] было показано, что следующие теоремы верны не только для $\alpha \in [0, 1]$, но и для $\alpha \in R^1$.

Теорема 6. *α-N-ядро кооперативной игры (N, v) совпадает с пред-N-ядром игры (N, v^α) для любого $\alpha \in R^1$, где*

$$v^\alpha(S) = \alpha v(S) + (1 - \alpha)v^*(S).$$

Теорема 7. *Для фиксированного $\alpha \in R^1$ α-N-ядро кооперативной игры (N, v) непусто и состоит из единственной точки $\nu^\alpha(v)$.*

С учетом результата теоремы 7 формулировка теоремы 6 может быть переписана в следующем виде.

Замечание 1. *Пусть* (N, v) *и* (N, v^α) *— две кооперативные ТП-игры, причем* $v^\alpha(S) = \alpha v(S) + (1-\alpha)v^*(S)$, $S \subseteq N$, $\alpha \in R^1$. *Тогда* $\nu^\alpha(v) = \nu^1(v^\alpha)$.

Данные результаты позволяют находить α-N-ядро кооперативной игры (N, v) через пред-N-ядро вспомогательной игры (N, v^α), что позволяет существенно упростить задачу поиска точечного решения.

Теперь дадим определение множества α-N-ядер кооперативной ТП-игры, введенное в [5].

Определение 15. *Множеством* α-N-*ядер игры* (N, v) *будем называть все* α-N-*ядра игры* (N, v) *для* $\alpha \in R^1$ *и обозначать*

$$\mathcal{N}(v) = \bigcup_{\alpha \in R^1} \mathcal{N}^\alpha(v). \tag{7}$$

Очевидно, что в пространстве R^n множество α-N-ядер игры (N, v) представляет собой множество, в общем случае состоящее более чем из одной точки. Заметим, что при некоторых α элементы множества $\mathcal{N}(v)$ совпадают с уже известными решениями: при $\alpha = 1$ получим пред-N-ядро, при $\alpha = 0$ имеем анти-пред-N-ядро, при $\alpha = 0{,}5$ получаем SM-ядро, а множество распределений векторов при $\alpha \in [0, 1]$ представляет собой $[0, 1]$-N-ядро.

Поскольку $\mathcal{N}(v)$ состоит из множества точек, необходимо исследовать его геометрические свойства. Приведем здесь результат из работы [6] о геометрической структуре $[0, 1]$-N-ядра, верный также для $\alpha \in R^1$.

Теорема 8. *В кооперативной игре* (N, v) *множество* α-N-*ядер* $\mathcal{N}(v)$ *представляет собой связное множество в* R^n, *состоящее из отрезков, последовательно соединенных своими концами.*

Замечание 2. *В пространстве* R^n *множество* α-N-*ядер* $\mathcal{N}(v)$ *является кусочно-выпуклым множеством.*

1.5. Коалиционная устойчивость множества α-N-ядер

Пусть задана выпуклая игра (N, v). Проанализировав C-ядро и множество α-N-ядер кооперативной игры и их взаимное расположение, сформулируем теоремы для класса выпуклых игр, позволяющие расширить понятие коалиционной устойчивости с одноточечного решения — пред-N-ядра — на множественное.

Напомним определения некоторых понятий, которые понадобятся для доказательства.

Определение 16. *Игрок i называется болваном в игре (N, v), если для любой коалиции S такой, что $i \notin S$, выполнено*

$$v(S \cup \{i\}) = v(S) + v(\{i\}).$$

***Свойство болвана**: если игрок i является болваном в игре (N, v), то для любого $x \in f(v)$ справедливо $x_i = v(\{i\})$.*

Точка множества называется *внутренней*, если существует сколь угодно малая окрестность этой точки, в которой содержатся только точки данного множества. *Граничными* называют точки, в любой окрестности которых содержатся как точки, принадлежащие данному множеству, так и не принадлежащие ему.

Замкнутое множество содержит все свои граничные точки.

Выпуклым называется такое множество, в котором все точки отрезка, образуемого любыми двумя точками данного множества, принадлежат этому множеству. Выпуклая комбинация точек выпуклого множества также будет принадлежать этому множеству. *Угловые* точки выпуклого множества — точки, которые нельзя представить в виде выпуклой комбинации точек множества.

Теорема 9. *Пусть множество α-N-ядер представляет собой одну точку. Тогда множество α-N-ядер $\mathcal{N}(v)$ выпуклой игры (N, v) коалиционно устойчиво и пересекается с C-ядром этой игры в единственной точке $\nu^1(v)$.*

Доказательство. Рассмотрим выпуклую игру (N, v). По результатам из [2] пред-N-ядро данной игры является центром C-ядра. Так как

множество α-N-ядер представляет собой одну точку, оно будет совпадать с пред-N-ядром, т.е. $\nu^{\alpha}(v) = \nu^{1}(v)$, $\forall \alpha \in R^1$, тогда $\mathcal{N}(v) = \bigcup\limits_{\alpha \in R^1} \nu^{\alpha}(v) = \nu^{1}(v)$. По теореме 5 пред-$N$-ядро произвольной игры (N, v) принадлежит C-ядру. Следовательно, $\mathcal{N}(v) \in C(v)$. □

Теорема 10. *Пусть множество α-N-ядер представляет собой множество такое, что $|\mathcal{N}(v)| > 1$. Тогда множество α-N-ядер $\mathcal{N}(v)$ выпуклой игры (N, v) пересекается с C-ядром этой игры более чем в одной точке.*

Доказательство. Рассмотрим выпуклую игру (N, v). По результатам из [2] пред-N-ядро данной игры является центром C-ядра. Из того, что $|\mathcal{N}(v)| > 1$ следует, что существует $\alpha \neq 1 : \nu^{\alpha}(v) \neq \nu^{1}(v)$.

C-ядро выпуклой игры представляет собой непустое выпуклое замкнутое множество, имеющее вид выпуклого многогранника. Тогда, по определению, угловыми точками C-ядра будут являться все его вершины. Пусть их будет k. Обозначим вершины C-ядра через $\pi_1, \pi_2, \ldots, \pi_k$.

Поскольку пред-N-ядро выпуклой игры может быть представлено в виде центра C-ядра, то его можно найти по следующей формуле:

$$\nu^{1}(v) = \frac{1}{k} \sum_{i=1}^{k} \pi_i.$$

В работе [6] доказана теорема 8, согласно которой в кооперативной игре (N, v) множество α-N-ядер представляет собой связное множество в R^n, состоящее из отрезков, последовательно соединенных своими концами.

Пусть $k < 3$. Тогда возможны несколько ситуаций. В первой игра (N, v) — игра двух лиц и ее решением будет единственная точка, всегда принадлежащая C-ядру (C-ядро будет представлять собой точку либо прямую). Второй ситуации соответствует игра с числом игроков $n > 2$, в которой есть игроки-болваны, никак не влияющие на игру. В таком случае игра может быть сведена к игре двух лиц, т.е. к первой ситуации.

Пусть $k \geqslant 3$. Зная, что пред-N-ядро $\nu^{1}(v)$ выпуклой игры (N, v) является центром C-ядра, рассмотрим ε-окрестность точки ν^{1} такую,

что все ее точки являются внутренними точками C-ядра, т.е. $\exists\ \varepsilon > 0$: $U_\varepsilon(\nu^1) = \{x \in X^0(v) : \rho(x, \nu^1) \leqslant \varepsilon\}$ принадлежит $C(v)$.

Зная, что множество α-N-ядер есть связное множество, которое можно представить в виде (7), рассмотрим усеченное множество α-N-ядер $\widetilde{\mathcal{N}}(v) = \bigcup\limits_{\alpha\in[1,\alpha_1(\varepsilon)]} \nu^\alpha(v)$, где $\alpha_1(\varepsilon)$ соответствует первой точке пересечения ломаной решений $\widetilde{\mathcal{N}}(v)$ с границей ε-окрестности $U_\varepsilon(\nu^1)$, т.е. $\nu^{\alpha_1}(v) \in U_\varepsilon(\nu^1) \cap \widetilde{\mathcal{N}}(v)$. Так как $\widetilde{\mathcal{N}}(v)$ является подмножеством $\mathcal{N}(v)$, то $\widetilde{\mathcal{N}}(v)$ обладает аналогичной структурой, т.е. имеет вид ломаной, концами которой будут ν^1 и ν^{α_1}, принадлежащие C-ядру.

Тогда по построению, ломаная $\widetilde{\mathcal{N}}(v)$ будет целиком лежать в ε-окрестности $U_\varepsilon(\nu^1)$, т.е. $\widetilde{\mathcal{N}}(v) \in U_\varepsilon(\nu^1)$. Следовательно, $\widetilde{\mathcal{N}}(v) \in C(v)$. Таким образом, доказали, что $\mathcal{N}(v)$ и $C(v)$ имеют в пересечении не менее одной точки. □

Проиллюстрируем результаты теорем примерами.

Пример 3. *Рассмотрим выпуклую игру трех лиц, заданную характеристической функцией:*

$$v(1) = v(2) = 1, v(3) = v(1,2) = 2, v(1,3) = 3, v(2,3) = 4, v(N) = 5.$$

Найдем вершины C-ядра: $\pi_1 = (1,1,3),\ \pi_2 = (1,2,2)$.
Можно заметить, что игрок 1 является болваном:

$$v(1,2) = v(1) + v(2),\ v(1,3) = v(1) + v(3),\ v(N) = v(1) + v(2,3).$$

Тогда его часть дележа уже будет известна, так как никакого дополнительного вклада в коалиции его присутствие не вносит: $x_1 = 1$.

Тогда, отстранив этого игрока, получаем игру двух лиц, в которой $v(2) = 1,\ v(3) = 2,\ v(2,3) = 4$. *Тогда единственным решением этой игры будет точка, соответствующая* $x_2 = 1{,}5$ *и* $x_3 = 2{,}5$. *Возвращая игрока 1, получаем дележ* $x = (1; 1{,}5; 2{,}5)$, *с которым совпадает множество α-N-ядер. Таким образом, все решение $\mathcal{N}$ принадлежит C-ядру, а значит, коалиционно устойчиво.*

Пример 4. *Пусть дана кооперативная игра трех лиц с характеристической функцией v:*

$$v(1) = 1, v(2) = 0, v(3) = 1, v(1,2) = 4, v(1,3) = 3, v(2,3) = 5, v(N) = 10.$$

Данная игра удовлетворяет условию выпуклости (3). Значит ее C-ядро непусто и является выпуклым многоугольником, для которого можно однозначно определить все вершины с помощью формулы (4):

$\pi_1 = (1,3,6), \pi_2 = (1,7,2), \pi_3 = (2,7,1),$

$\pi_4 = (4,0,6), \pi_5 = (5,0,5), \pi_6 = (5,4,1).$

Так как игра выпуклая, можем найти пред-N-ядро игры по формуле:

$$\nu^1(v) = \frac{1}{6}\sum_{i=1}^{6}\pi_i = (3; 3{,}5; 3{,}5).$$

Найдем $\mathcal{N}^\alpha(v)$ для некоторых $\alpha \in [-1;1]$ и представим решения в виде таблицы 1:

α	$\mathcal{N}^\alpha$
-1	$(3; 3{,}5; 3{,}5)$
$-0{,}5$	$(3; 3{,}5; 3{,}5)$
0	$(3; 3{,}5; 3{,}5)$
$0{,}5$	$(3; 3{,}5; 3{,}5)$
1	$(3; 3{,}5; 3{,}5)$

Таблица 1. α-N-ядра к примеру 4

Видим, что в данной игре при заданных α решения совпадают. Расписав α-эксцессы для всех коалиций и проверив их на сбалансированных наборах, можно сделать вывод, что для данной игры множество α-N-ядер вырождается в единственную точку, совпадающую с пред-N-ядром. Поскольку $\mathcal{N}^\alpha(v) = \nu^1(v)$ для всех $\alpha \in R^1$, то $\mathcal{N}$ коалиционно устойчиво.

Пример 5. *Рассмотрим теперь игру пяти лиц, в которой характеристическая функция имеет вид*

$$v(S) = \begin{cases} 1, & S \in \big\{\{2,3,4,5\},\{1,3,4,5\},\{1,2,4,5\}\big\}, \\ 4, & S \in \big\{\{1,2,3\},\{1,2,3,5\},\{1,2,3,4\}\big\}, \\ 5, & S = N, \\ 0, & \textit{иначе}. \end{cases}$$

Проверив игру на выполнение условия (3), можем сделать вывод, что игра выпуклая. Тогда для нее имеет место теорема 10. Покажем это.

Найдем вершины C-ядра с помощью формулы (4):

$$\pi_1 = (0,0,4,0,1), \quad \pi_2 = (0,0,4,1,0), \quad \pi_3 = (0,1,4,0,0),$$
$$\pi_4 = (0,4,0,0,1), \quad \pi_5 = (0,4,0,1,0), \quad \pi_6 = (0,4,1,0,0),$$
$$\pi_7 = (1,0,4,0,0), \quad \pi_8 = (1,4,0,0,0), \quad \pi_9 = (4,0,0,0,1),$$
$$\pi_{10} = (4,0,0,1,0), \quad \pi_{11} = (4,0,1,0,0), \quad \pi_{12} = (4,1,0,0,0).$$

Далее получим множество α-N-ядер, которое можно определить следующими соотношениями:

1) для $\alpha \in \big(-\infty; 0\big]$ $\nu^\alpha(v) = \big(1;1;1;1;1\big)$;

2) для $\alpha \in \left[0; \frac{3}{5}\right]$ $\nu^\alpha(v) = \Big(1+\frac{2}{3}\alpha; 1+\frac{2}{3}\alpha; 1+\frac{2}{3}\alpha; 1-\alpha; 1-\alpha\Big)$;

3) для $\alpha \in \left[\frac{3}{5}; \frac{23}{36}\right]$ $\nu^\alpha(v) = \left(\frac{8\alpha+5}{7}; \frac{8\alpha+5}{7}; \frac{8\alpha+5}{7}; \frac{10-12\alpha}{7}; \frac{10-12\alpha}{7}\right)$;

4) для $\alpha \in \left[\frac{23}{36}; \infty\right)$ $\nu^\alpha(v) = \left(\frac{13}{9}; \frac{13}{9}; \frac{13}{9}; \frac{3}{9}; \frac{3}{9}\right)$.

Геометрически множество α-N-ядер данной игры представляет собой ломаную, состоящую из двух последовательно соединенных отрезков. Интересным здесь является то, что все значения α, меняющие решение, принадлежат промежутку $[0,1]$, т.е. в данном примере $[0,1]$-N-ядро совпадает с множеством α-N-ядер.

Заметим, что в данной игре игроки 1, 2 и 3 симметричны относительно друг друга, также как игроки 4 и 5, что отражается в решении: симметричные игроки получают одинаковые выигрыши.

Проверив решения на коалиционную устойчивость, получаем точки пересечения C-ядра и множества α-N-ядер: $\mathcal{N}(v) \in C(v)$ при

$\alpha \in \left[\frac{1}{2}; \infty\right)$. *Хотя множество* α-N-*ядер не лежит целиком в* C-*ядре, доказанная выше теорема является справедливой, и решение* $\mathcal{N}(v)$ *можно считать частично коалиционно устойчивым.*

1.6. Некоторые классы игр с коалиционно устойчивыми решениями

Перейдем теперь к рассмотрению различных классов кооперативных ТП-игр с целью нахождения необходимого и достаточного условий одноточечности множества α-N-ядер. Это поможет выявить классы игр с решением, не зависящим от выбранных значений α, т.е. для которых не будет важно соотношение конструктивной и блокирующей сил коалиций.

Введем понятия существенной и несущественной игры, некоторым образом связанные с супераддитивностью характеристической функции.

Характеристическая функция называется *супераддитивной*, если для любых коалиций $S, T : S \cap T = \emptyset$ выполняется неравенство

$$v(S \cup T) \geqslant v(S) + v(T). \tag{8}$$

Супераддитивность свидетельствует о том, что при объединении игроков в коалицию ее выигрыш будет не меньше суммы выигрышей самостоятельно действующих игроков, т.е. их объединение является целесообразным с точки зрения увеличения выигрыша.

Наиболее слабой формой супераддитивности характеристической функции является ее аддитивность, при которой неравенство (8) обращается в равенство:

$$v(S \cup T) = v(S) + v(T), \quad \forall S, T : S \cap T = \emptyset.$$

Аддитивность характеристической функции показывает незаинтересованность игроков в образовании коалиций.

Определение 17. *Кооперативная игра с аддитивной характеристической функцией называется несущественной.*

Нетрудно видеть, что несущественная игра будет являться выпуклой. Приведем некоторые результаты, касающиеся несущественных игр.

Теорема 11. *Для того, чтобы кооперативная игра была несущественной, необходимо и достаточно выполнение следующего равенства:*

$$\sum_{i \in N} v(\{i\}) = v(N).$$

Теорема 12. *В несущественной игре существует только один дележ* $x = (v(\{1\}), v(\{2\}), \ldots, v(\{n\}))$.

Теорема 12 позволяет понять, что в несущественной игре участники не могут получить больше своего первоначального выигрыша из-за бесполезности образования всяких коалиций, так как включение любого игрока в коалицию не дает никаких дополнительных благ, которые можно было бы разделить между игроками.

В несущественной игре C-ядро и множество α-N-ядер будут совпадать и представлять собой единственную точку — дележ x, определяемый согласно теореме 12.

Далее будем рассматривать только существенные игры.

Определение 18. *Кооперативную игру* (N, v) *будем называть существенной в том случае, если*

$$\sum_{i \in N} v(\{i\}) < v(N).$$

В существенной игре с более чем одним игроком множество дележей бесконечно и может быть представлено в виде

$$x = (v(\{1\}) + a_1,\ v(\{2\}) + a_2, \ldots, v(\{n\}) + a_n),$$

где $a_i \geqslant 0, \sum\limits_{i \in N} a_i = v(N) - \sum\limits_{i \in N} v(\{i\})$.

Перейдем к рассмотрению класса игр с постоянной суммой.

Определение 19. *Кооперативная игра* (N, v) *называется игрой с постоянной суммой, если для любой ее коалиции* $S \subseteq N$ *имеет место равенство*

$$v(S) = v(N) - v(N \backslash S) = v^*(S).$$

Будем искать множество α-N-ядер игры с учетом замечания 1, т.е. через пред-N-ядро игры (N, v^α), где $v^\alpha(S) = \alpha v(S) + (1-\alpha)v^*(S)$. Зная, что пред-$N$-ядро игры (N, v) всегда существует и представляет собой единственную точку, можем прейти к равенству $\nu^\alpha(v) = \nu(v^\alpha) = \nu(v)$, или к другой его записи: $v^\alpha(S) = \alpha v(S) + (1-\alpha)v^*(S) = v(S)$. Отсюда получаем, что множество α-N-ядер будет представлять собой единственную точку, если выполняется условие $v(S) = v^*(S)$. Значит можно утверждать, что в классе кооперативных игр с постоянной суммой $\mathcal{N}(v)$ совпадает с пред-N-ядром и является одной точкой. Однако в существенной игре с постоянной суммой C-ядро оказывается пустым. Можем сделать вывод, что в классе существенных игр с постоянной суммой множество α-N-ядер невозможно проверить на коалиционную устойчивость.

Рассмотрим теперь класс существенных симметричных игр, удовлетворяющих условиям выпуклости (3).

Симметричной будем называть игру (N, v), для которой верно следующее:

$$v(S) = v(T), \quad \forall S, T \subseteq N : |S| = |T|.$$

В симметричной игре все участники имеют одинаковые возможности и получают равные доли выигрыша коалиции. Однако условия существенности и выпуклости игры побуждает их объединяться в максимальную коалицию, самым правильным и логичным решением будет присуждение каждому из n игроков доли, равной $\frac{v(N)}{n}$, и в таком случае единственно возможным дележом будет вектор $x = \left(\frac{v(N)}{n}, \frac{v(N)}{n}, \ldots, \frac{v(N)}{n}\right)$. Поскольку в данной работе симметричные игры рассматриваются при условии их выпуклости, то C-ядро таких игр непусто. Все множество α-N-ядер будет совпадать с пред-N-ядром и находиться в центре C-ядра, т.е. будет выполняться $\mathcal{N}(v) = \nu(v) = \left(\frac{v(N)}{n}, \frac{v(N)}{n}, \ldots, \frac{v(N)}{n}\right) \in C(v)$. Таким образом, решение на данном классе игр будет коалиционно устойчивым.

Глава 2. Программная реализация

В данной главе приводятся описания алгоритма вычисления пред-N-ядра и α-N-ядер, а также основная информация о необходимых для этого функциях и программах, написанных в среде MATLAB. Все листинги программ приведены в приложении.

2.1. Описание вспомогательных функций

В данном разделе будут описаны функции, необходимые для дальнейшей работы, в том числе для одной из важнейших функций — нахождения пред-N-ядра произвольной кооперативной ТП-игры n лиц, заданной с помощью характеристической функции v.

В игре (N, v) может быть сформировано 2^n коалиций. Пустую коалицию, выигрыш которой $v(\varnothing) = 0$, учитывать не будем. Тогда пронумеруем оставшиеся $(2^n - 1)$ коалицию стандартным образом: по возрастанию количества участников и их номеров, т.е. от одноэлементных коалиций до максимальной. Таким образом, последовательность коалиций для игры двух лиц выглядит следующим образом: $\{1\}, \{2\}, \{1, 2\}$. Для игры трех лиц: $\{1\}, \{2\}, \{3\}, \{1, 2\}, \{1, 3\}, \{2, 3\}, \{1, 2, 3\}$. И так далее для игр с большим количеством участников.

NumPlay(v)

Входные параметры: v — характеристическая функция, заданная в виде вектор-строки со стандартной нумерацией коалиций.

Выходные параметры: n — число игроков.

Описание: функция определяет количество игроков в заданной характеристической функцией игре, что позволяет не вводить n самостоятельно. Если характеристическая функция v не задает значения выигрышей всех коалиций, выводится сообщение об ошибке ввода входного параметра.

V2(v)

Входные параметры: v — характеристическая функция, заданная в виде вектор-строки со стандартной нумерацией коалиций.

Выходные параметры: v2 — характеристическая функция двойственной игры со стандартной нумерацией коалиций.

Описание: функция позволяет найти характеристическую функцию двойственной игры (N, v^*) к данной игре (N, v), где

$$v^*(S) = v(N) - v(N \backslash S).$$

Valpha(v, alpha)

Входные параметры: v — характеристическая функция, заданная в виде вектор-строки со стандартной нумерацией коалиций;
alpha — значение α.

Выходные параметры: v_alpha — характеристическая функция игры (N, v^α) со стандартной нумерацией коалиций

Описание: функция строит характеристическую функцию игры (N, v^α), где v^α определяется по правилу: $v^\alpha(S) = \alpha v(S) + (1 - \alpha)v^*(S)$.

Coalition(n)

Входные параметры: n — число игроков.

Выходные параметры: Coal — матрица размерности $[2^n \times n]$.

Описание: по числу игроков n строится матрица всех возможных коалиций, упорядоченных по возрастанию числа игроков. Коалициям соответствуют строки матрицы, составленные из нулей и единиц следующим образом: если игрок i участвует в коалиции, то i-тая компонента вектор-строки равна 1. Например, коалиции $\{1, 2, 4\}$ в игре пяти лиц будет соответствовать строка вида $[1\,1\,0\,1\,0]$.

Norm2Redoo(v)

Входные параметры: v — характеристическая функция, заданная в виде вектор-строки со стандартной нумерацией коалиций

Выходные параметры: v0 — характеристическая функция 0-редуцированной игры;
c — вектор-строка размерности n: $c_i = -v(\{i\})$.

Описание: функция осуществляет переход игры (N, v) к 0-редуцированной игре $(N, v0)$, в которой выигрыши одноэлементных коалиций ста-

новятся равными 0. Выигрыши остальных коалиций пересчитываются:

$$v0(S) = v(S) + \sum_{i \in S} c_i.$$

Coal2Name(Coal)

Входные параметры: Coal — коалиция, заданная в виде вектор-строки из 0 и 1, где 1 стоят на позиции состоящих в коалиции игроков.

Выходные параметры: Name — текстовая строка из цифр (символов), соответствующих номерам (именам) участников коалиции.

Описание: функция анализирует параметр Coal на состав игроков. В случае $n < 10$ игрокам присваиваются номера от 1 до n. Если $n \in [10, 26]$, то игроки получают имена первых n заглавных букв английского алфавита.

NumCoalV1(Coal)

Входные параметры: Coal — коалиция, заданная в виде вектор-строки из 0 и 1, где 1 стоят на позиции состоящих в коалиции игроков.

Выходные параметры: Num — число, соответствующее номеру коалиции в стандартной нумерации.

Описание: функция выполняет поиск порядкового номера коалиции среди всех возможных непустых коалиций игры, при условии использования стандартной нумерации.

Следующие функции позволяют преобразовать характеристическую функцию для сокращения количества итераций в алгоритме поиска пред-N-ядра. Для этого необходим переход к модифицированной нумерации коалиций. Будем также рассматривать игру (N, v) с $(2^n - 1)$ непустыми коалициями. Пусть игра строится динамическим путем добавления игрока на каждом шаге. Начиная с игры с одним игроком, единственно возможная коалиция $\{1\}$ будет иметь номер 1. Добавляя второго игрока и новые образовавшиеся коалиции, получим следующую последовательность коалиций: $\{1\}, \{2\}, \{1, 2\}$. Далее для игры трех лиц последовательность будет иметь вид: $\{1\}, \{2\}, \{1, 2\}, \{3\}, \{1, 3\}, \{2, 3\}, \{1, 2, 3\}$. И так далее.

NumCoalV2(Coal)

Входные параметры: Coal — коалиция, заданная в виде вектор-строки из 0 и 1, где 1 стоят на позиции состоящих в коалиции игроков.

Выходные параметры: Num2 — число, соответствующее номеру коалиции в модифицированной нумерации.

Описание: функция позволяет определить номер коалиции Coal среди непустых коалициий игры в модифицированной нумерации.

V2Mod(v)

Входные параметры: v — характеристическая функция, заданная в виде вектор-строки со стандартной нумерацией коалиций.

Выходные параметры: v_mod — вектор-строка значений, соответствующая характеристической функции игры с модифицированной нумерацией.

Описание: с использованием функции NumCoalV2 данная функция осуществляет переход от стандартной нумерации к модифицированной.

2.2. Проверка игры на выпуклость

Поскольку в данной работе рассматриваются выпуклые игры, первоочередным этапом в анализе коалиционной устойчивости решений является проверка заданной кооперативной игры (N, v) на выпуклость. Согласно определению 8 главы 1 игра (N, v) — выпуклая, если для любых коалиций $S, T \subset N$ выполняется

$$v(S) + v(T) \leqslant v(S \cup T) + v(S \cap T).$$

Для проверки данного условия была написана функция Convexity(v), входным параметром которой является характеристическая функция игры (N, v) со стандартной нумерацией коалиций. Во время работы программы строится матрица всех возможных коалиций игры. Далее перебираются все возможные пары коалиций S и T, находятся их пересечения и объединения и проводится подстановка полученных значений в условие выпуклости. Если на какой-нибудь паре коалиций неравенство не выполняется, производится процедура распознавания участников этой

пары коалиций с помощью функции Coal2Name и их имена записываются в матрицу D.

Выходным параметром функции является result. Если для всех пар коалиций условие выполняется, игра признается выпуклой и result принимает значение 1. Если же на некоторых парах неравенство нарушается, программой выводится матрица D (в каждой строке есть два набора символов, соответствующих именам участников коалиций) и значение 0, означающее невыпуклость заданной игры.

Для работы функции Convexity(v) необходимы следующие вспомогательные: NumPlay(), Coalition(), Coal2Name().

2.3. Нахождение вершин C-ядра

Для более четкого представления о геометрической структуре C-ядра имеет смысл найти его вершины. Так как на классе выпуклых игр C-ядро всегда непусто, то будет существовать хотя бы одна вершина. По формулам (4) можем однозначно определить через характеристическую функцию координаты $(\pi_{i_1}, \ldots, \pi_{i_n})$ всех вершин $\pi(p)$, где $p = (i_1, \ldots, i_n)$ — перестановка множества игроков $\{1, 2, \ldots, n\}$:

$$\pi_{i_k} = v(i_1, \ldots, i_k) - v(i_1, \ldots, i_{k-1}),\ k = \overline{1, n}.$$

Для автоматизированного нахождения вершин C-ядра игры (N, v) была создана функция CoreVertices(v), на вход которой подается характеристическая функция v. Далее определяется число участников игры и перебираются все их возможные перестановки. По каждой из них вычисляется вершина C-ядра по указанной выше формуле и добавляются в матрицу вершин. Так как некоторые вершины могут совпадать, определяются уникальные вершины, а повторяющиеся удаляются. Матрица вершин Core является выходным параметром данной функции и представляет собой таблицу размера $[m \times n]$, где m — количество вершин C-ядра, а n — число игроков. Каждой вершине соответствует строка матрицы Core.

Для работы функции CoreVertices(v) необходимы вспомогательные: NumPlay(), Coalition(), NumCoalV1().

2.4. Алгоритм поиска пред-N-ядра

В работе [10] Х. Майнхард предлагает способ нахождения пред-N-ядра, подходящий для выпуклых игр. Опишем кратко предложенный итеративный метод.

Пусть дана игра (N, v). Зададим начальное приближение — дележ x_0. Далее вычислим эксцессы каждой коалиции и запишем в виде вектора $e(x_0, v, S)$. По имеющемуся вектору эксцессов определяем величины $S_{ij}(x_0)$ для всех пар $i, j \in N,\ i \neq j$ по правилу:

$$S_{ij}(x_0) = \arg\max_{S \in \mathcal{G}_{ij}} e(x_0, v, S),\ \text{где}\ \mathcal{G}_{ij} = \{S |\ i \in S, j \notin S\}.$$

Если максимум эксцессу доставляет несколько коалиций S, то выбирается лексикографически минимальная из них, например, из коалиций $\{1,2\}$ и $\{1,3\}$ выбираем $\{1,2\}$. Из всех полученных значений составляем вектор $\mathcal{S} = \{S_{ij}\}$ в следующем порядке индексов:

$$\{[1,2],[1,3],\ldots,[1,n],[2,3],\ldots,[n{-}1,n],[2,1],[3,1],[3,2],[4,1],\ldots,[n,n{-}1]\}.$$

Строим матрицу $E = [-E_{1,2},\ldots,-E_{n-1,n},E_0]$, в которой величины $E_{i,j}$ определяются для всех $i, j \in N,\ i < j$ следующим образом:

$$E_0 = 1_N;\ E_{i,j} = 1_{S_{ij}} - 1_{S_{ji}},$$

где $1_S : N \to \{0;1\} : 1_S(k) = \begin{cases} 1, k \in S, \\ 0, \text{иначе}. \end{cases}$

Составляем вектор $a = [-a_{12},\ldots,-a_{n-1,n},a_0]$, где $a_0 = v(N)$, $a_{ij} = v(S_{ij}) - v(S_{ji}),\ \forall i, j \in N, i < j$.

Найдем матрицу $Q = EE^T$ и вектор $A = Ea$. Тогда решив систему уравнений $Qx + A = 0$, или $E^T x + a = 0$, получим новый дележ x_1, для которого вектор эксцессов будет лексикографически меньше (или равен) эксцессов дележа x_0. Задавая новое начальное приближение равным найденному дележу, продолжаем итерации до нахождения единственного вектора x, которому соответствует лексикографический минимум вектора эксцессов. Итерационный процесс обычно сходится не более чем за $(n+1)$ шаг.

По описанному алгоритму была написана программа PreNucl(v), позволяющая получить пред-N-ядро заданной игры (N, v). Входным значением является характеристическая функция v, которая в программе изменяет нумерацию коалиций на модифицированную с помощью вспомогательной функции V2Mod(), что позволяет сократить количество итераций. Результатом работы PreNucl(v) является единственный вектор, соответствующий пред-N-ядру игры (N, v).

2.5. Нахождение α-N-ядер

Для нахождения множества α-N-ядер игры (N, v) необходимо уметь находить не только пред-N-ядро, но и α-N-ядра с фиксированным значением α. Согласно замечанию 1 главы 1, α-N-ядро игры (N, v) может быть найдено как пред-N-ядро игры (N, v^α), где $v^\alpha(S) = \alpha v(S) + (1 - \alpha)v^*(S)$. Используя данный результат, функция AlphaNucl(v,alpha) находит α-N-ядро по входным параметрам v и alpha, соответствующим характеристической функции игры (N, v) и значению α.

Для работы функции AlphaNucl(v,alpha) необходимы вспомогательные: Valpha(), V2Mod() и PreNucl().

2.6. Проверка решений на коалиционную устойчивость

В изучении свойства коалиционной устойчивости различных концепций решения крайне важно иметь функцию для проверки дележа (выбранного решения, или же любого возможного) на принадлежность C-ядру. Для этой цели был реализован в виде функции IsInCore(Imp,v) результат теоремы Бондаревой—Шепли, согласно которой вектор x принадлежит C-ядру, если для любых коалиций $S \subset N$ верно следующее неравенство:

$$x(S) \geqslant v(S).$$

Входными переменными функции являются Imp (вектор-строка размерности n, соответствующий проверяемому дележу) и v (характеристическая функция заданной игры). Программа строит вектор $x(S)$, состоящий из компонент Imp, и сравнивает его с характеристической функцией v. Выходной параметр result показывает выполнение всех неравенств, и равен 1, если дележ Imp принадлежит C-ядру игры, и 0 в противном

случае.

В работе функции используются следующие вспомогательные: NumPlay(), Coalition().

Имея способ нахождения α-N-ядер для любых заданных α с помощью функции AlphaNucl(), появляется возможность построить последовательность значений α и исследовать поведение и коалиционную устойчивость множества α-N-ядер на определенном участке α. Для этой цели была создана функция TableAlphaPreNuclSet(v,alpha0,alpha1,d,flag). Получая на вход характеристическую функцию, левую и правую границу значений α, шаг изменения α и константу flag $\in \{0;1;2;3\}$, программа составляет таблицу, столбцами которой являются значение α, выигрыши игроков согласно полученному α-N-ядру, и результат его проверки на коалиционную устойчивость. Константа flag позволяет выбрать, в каком виде необходимо получить таблицу: если flag = 0, то таблица выводится в командном окне, если flag = 1, 2 или 3, то итоговая таблица записывается в txt-, csv- или xlsx-файл, соответственно.

В работе функции TableAlphaPreNuclSet() необходимы все вспомогательные, а также Convexity(), PreNucl(), AlphaNucl() и IsInCore().

Пример 6. *Рассмотрим игру пяти лиц из примера 5 и проверим множество α-N-ядер данной игры на коалиционную устойчивость с помощью функции TableAlphaPreNuclSet().*

Характеристическая функция задана в следующем виде:

$$v(S) = \begin{cases} 1, & S \in \big\{\{2,3,4,5\},\{1,3,4,5\},\{1,2,4,5\}\big\}, \\ 4, & S \in \big\{\{1,2,3\},\{1,2,3,5\},\{1,2,3,4\}\big\}, \\ 5, & S = N, \\ 0, & \textit{иначе}. \end{cases}$$

В виде вектора значений со стандартной нумерацией коалиций характеристическая функция будет выглядеть так:

$v = [0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 4\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 4\ 4\ 1\ 1\ 1\ 5]$.

Рассмотрим α-N-ядра данной игры при $\alpha \in [-0{,}1; 0{,}7]$ (alpha0 = –0.1, alpha1 = 0.7) с шагом d = 0.025. Выведем итоговую таблицу в xlsx-файл (flag = 3). Результат работы функции представлен на рис. 1.

	A	B	C	D	E	F	G	H
1	v =	[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4 0 0 0 0 0 0 0 0 0 4 4 1 1 1 5]						
2	No	alpha	N_1	N_2	N_3	N_4	N_5	IsInCore
3	1	-0,1	1	1	1	1	1	0
4	2	-0,075	1	1	1	1	1	0
5	3	-0,05	1	1	1	1	1	0
6	4	-0,025	1	1	1	1	1	0
7	5	0	1	1	1	1	1	0
8	6	0,025	1,01667	1,01667	1,01667	0,975	0,975	0
9	7	0,05	1,03333	1,03333	1,03333	0,95	0,95	0
10	8	0,075	1,05	1,05	1,05	0,925	0,925	0
11	9	0,1	1,06667	1,06667	1,06667	0,9	0,9	0
12	10	0,125	1,08333	1,08333	1,08333	0,875	0,875	0
13	11	0,15	1,1	1,1	1,1	0,85	0,85	0
14	12	0,175	1,11667	1,11667	1,11667	0,825	0,825	0
15	13	0,2	1,13333	1,13333	1,13333	0,8	0,8	0
16	14	0,225	1,15	1,15	1,15	0,775	0,775	0
17	15	0,25	1,16667	1,16667	1,16667	0,75	0,75	0
18	16	0,275	1,18333	1,18333	1,18333	0,725	0,725	0
19	17	0,3	1,2	1,2	1,2	0,7	0,7	0
20	18	0,325	1,21667	1,21667	1,21667	0,675	0,675	0
21	19	0,35	1,23333	1,23333	1,23333	0,65	0,65	0
22	20	0,375	1,25	1,25	1,25	0,625	0,625	0
23	21	0,4	1,26667	1,26667	1,26667	0,6	0,6	0
24	22	0,425	1,28333	1,28333	1,28333	0,575	0,575	0
25	23	0,45	1,3	1,3	1,3	0,55	0,55	0
26	24	0,475	1,31667	1,31667	1,31667	0,525	0,525	0
27	25	0,5	1,33333	1,33333	1,33333	0,5	0,5	1
28	26	0,525	1,35	1,35	1,35	0,475	0,475	1
29	27	0,55	1,36667	1,36667	1,36667	0,45	0,45	1
30	28	0,575	1,38333	1,38333	1,38333	0,425	0,425	1
31	29	0,6	1,4	1,4	1,4	0,4	0,4	1
32	30	0,625	1,42857	1,42857	1,42857	0,35714	0,35714	1
33	31	0,65	1,44444	1,44444	1,44444	0,33333	0,33333	1
34	32	0,675	1,44444	1,44444	1,44444	0,33333	0,33333	1
35	33	0,7	1,44444	1,44444	1,44444	0,33333	0,33333	1
36								

Рис. 1. Результат работы функции TableAlphaPreNuclSet()

В результате работы функции значения α записываются в столбец B, в столбцы C-G — соответствующие ему значения α-N-ядер для каждого из игроков. Результат проверки каждого из α-N-ядер на коалиционную устойчивость приведены в столбце H (0 — решение не принадлежит C-ядру, 1 — иначе). По итоговой таблице

можно сделать вывод, что при $\alpha \leqslant 0 \quad \nu^\alpha(v) = \big(1;1;1;1;1\big)$, *а при* $\alpha \geqslant 0{,}65 \quad \nu^\alpha(v) = \big(1{,}4444;1{,}4444;1{,}4444;0{,}3333;0{,}3333\big)$, *что соответствует полученному в примере 5* $\nu^\alpha(v) = \Big(\frac{13}{9};\frac{13}{9};\frac{13}{9};\frac{3}{9};\frac{3}{9}\Big)$. *Также можно убедиться, что при* $\alpha \geqslant 0{,}5$ α-N-*ядра коалиционно устойчивы. Таким образом, работа функции TableAlphaPreNuclSet() подтверждает выводы примера 5.*

2.7. Множество α-N-ядер для игры трех лиц

Рассмотрим произвольную игру трех лиц, заданную характеристической функцией v. Здесь и далее будем обозначать $v(\{i\}) = v(i),\ i = \overline{1,3}$. Приведем игру к 0-редуцированному виду ($v(1) = v(2) = v(3) = 0$) и переставим игроков таким образом, чтобы выполнялось неравенство $v(12) < v(13) < v(23)$. Введем следующее обозначение:

$$\Delta v(N) = v(N) - v(12) - v(13) - v(23).$$

Тогда для данной игры справедлива следующая теорема, доказанная Н. Смирновой и С. Тарашниной.

Теорема 13. *Пусть* (N, v) *- кооперативная ТП-игра трех лиц с* $v(i) = 0$, $i = \overline{1,3}$, *и* $v(12) < v(13) < v(23)$. *Тогда множество* α-N-*ядер игры* (N, v) *строится следующим образом:*

$$\mathcal{N}(v) = N_1^\alpha \cup N_2^\alpha \cup N_3^\alpha \cup N_4^\alpha \cup N_5^\alpha,$$

где $N_1^\alpha, N_2^\alpha, N_3^\alpha, N_4^\alpha, N_5^\alpha$ *— множества, описанные ниже.*

1. *для* $\alpha \leqslant \frac{1}{2} - \frac{|\Delta v(N)|}{2\big(v(23)-v(13)\big)}$

   $$N_1^\alpha = \Big\{\Big(\frac{v(12)+v(13)}{2}, \frac{v(23)+v(12)}{2} + \frac{\Delta v(N)}{2}, \frac{v(23)+v(13)}{2} + \frac{\Delta v(N)}{2}\Big)\Big\};$$

2. *для* $\frac{1}{2} - \frac{|\Delta v(N)|}{2\big(v(23)-v(13)\big)} \leqslant \alpha \leqslant \frac{1}{2} - \frac{|\Delta v(N)|}{2\big(v(23)+v(13)-2v(12)\big)}$

   $N_2^\alpha = \big\{\big(\nu_1^\alpha(v), \nu_2^\alpha(v), \nu_3^\alpha(v)\big)\big\}$, *где*

   $$\nu_1^\alpha(v) = \tfrac{v(13)+v(12)}{2} + \tfrac{\Delta v(N)}{4} + sgn\big(\Delta v(N)\big)\tfrac{(2\alpha-1)\big(v(23)-v(13)\big)}{4},$$
   $$\nu_2^\alpha(v) = \tfrac{v(23)+v(12)}{2} + \tfrac{\Delta v(N)}{4} + sgn\big(\Delta v(N)\big)\tfrac{(1-2\alpha)\big(v(23)-v(13)\big)}{4},$$
   $$\nu_3^\alpha(v) = \tfrac{v(13)+v(23)}{2} + \tfrac{\Delta v(N)}{2};$$

3. для $\frac{1}{2} - \frac{|\Delta v(N)|}{2\Big(v(23)+v(13)-2v(12)\Big)} \leqslant \alpha \leqslant \frac{1}{2} + \frac{|\Delta v(N)|}{2\Big(2v(23)-v(13)-v(12)\Big)}$

$N_3^\alpha = \Big\{\big(\nu_1^\alpha(v), \nu_2^\alpha(v), \nu_3^\alpha(v)\big)\Big\}$, *где*

$$\nu_1^\alpha(v) = \frac{v(13)+v(12)}{2} + \frac{\Delta v(N)}{3} + sgn\big(\Delta v(N)\big)\frac{(2\alpha-1)\Big(2v(23)-v(13)-v(12)\Big)}{6},$$

$$\nu_2^\alpha(v) = \frac{v(23)+v(12)}{2} + \frac{\Delta v(N)}{3} + sgn\big(\Delta v(N)\big)\frac{(2\alpha-1)\Big(2v(13)-v(23)-v(12)\Big)}{6},$$

$$\nu_3^\alpha(v) = \frac{v(13)+v(23)}{2} + \frac{\Delta v(N)}{3} + sgn\big(\Delta v(N)\big)\frac{(2\alpha-1)\Big(2v(12)-v(23)-v(13)\Big)}{6};$$

4. для $\frac{1}{2} + \frac{|\Delta v(N)|}{2\Big(2v(23)-v(13)-v(12)\Big)} \leqslant \alpha \leqslant \frac{1}{2} + \frac{|\Delta v(N)|}{2\Big(v(13)-v(12)\Big)}$

$N_4^\alpha = \Big\{\big(\nu_1^\alpha(v), \nu_2^\alpha(v), \nu_3^\alpha(v)\big)\Big\}$, *где*

$$\nu_1^\alpha(v) = \frac{v(13)+v(12)}{2} + \frac{\Delta v(N)}{2},$$

$$\nu_2^\alpha(v) = \frac{v(23)+v(12)}{2} + \frac{\Delta v(N)}{4} + sgn\big(\Delta v(N)\big)\frac{(2\alpha-1)\Big(v(13)-v(12)\Big)}{4},$$

$$\nu_3^\alpha(v) = \frac{v(13)+v(23)}{2} + \frac{\Delta v(N)}{3} + sgn\big(\Delta v(N)\big)\frac{(1-2\alpha)\Big(v(13)-v(12)\Big)}{4};$$

5. для $\alpha \geqslant \frac{1}{2} + \frac{|\Delta v(N)|}{2\Big(v(13)-v(12)\Big)}$

$$N_5^\alpha = \Big\{\Big(\frac{v(13)+v(12)}{2} + \frac{\Delta v(N)}{2}, \frac{v(23)+v(12)}{2} + \frac{\Delta v(N)}{2}, \frac{v(13)+v(23)}{2} + \frac{\Delta v(N)}{2},\Big)\Big\}.$$

Данная теорема позволяет находить множество α-N-ядер 0-редуцированной игры с упорядоченными строго по возрастанию значениями двухэлементных коалиций. Однако могут возникнуть следующие сложности, связанные с нарушением строгой упорядоченности $v(12)$, $v(13)$, $v(23)$:

1) некоторые из подмножеств N_i^α множества α-N-ядер обращаются в точку и потому часть интервалов для α необходимо объединять с другими;

2) невозможно определить некоторые границы интервалов для значения α, так как знаменатели соответствующих α будут равны 0.

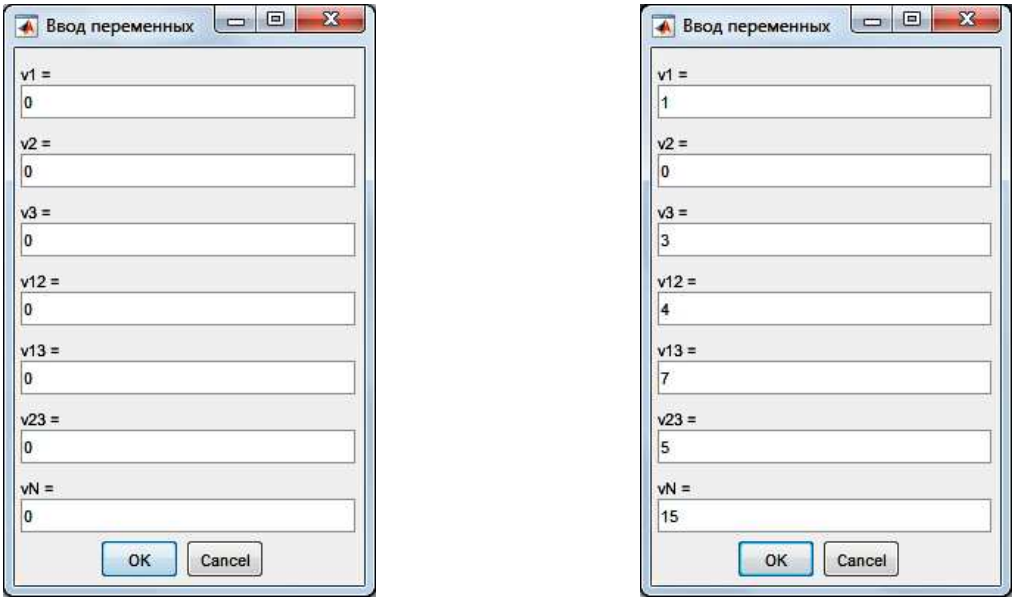
Также определенные сложности доставляет необходимость приведения игры к требуемому виду, в частности, перестановка игроков до и после непосредственного нахождения множества α-N-ядер.

Функция ThreePlayersGame(v), разработанная для нахождения множества α-N-ядер произвольной игры трех лиц (N, v), не требует от пользователя дополнительных манипуляций с характеристической функцией и последующего анализа полученной функции выигрышей и применимости указанной выше теоремы для новой игры. Входным параметром является характеристическая функция v со стандартной нумерацией коалиций. В процессе работы программа обрабатывает полученный вектор значений: приводит игру в 0-редуцированную форму и упорядочивает двухэлементные коалиции по возрастанию. Далее вычисляются граничные значения α и компоненты множества α-N-ядер для каждого игрока. Проводится анализ полученных α и исключаются некоторые из интервалов. На выходе функции ThreePlayersGame(v) — текстовое сообщение, содержащее в себе информацию о разбиении числовой прямой значений α на интервалы и соответствующие им вектора решений, в общем случае зависящие от α.

Пользовательский интерфейс программы

Главное окно программы имеет вид, представленный на рисунке 2(а).

По умолчанию во всех полях стоит значение 0. Для задания значений характеристической функции необходимо ввести в поля необходимые значения, а затем нажать кнопку "OK"(см. рис. 2(б)).



а) б)

Рис. 2. Вид окна ввода переменных

В появившемся окне будет отображен результат — вся информация о множестве α-N-ядер заданной игры трех лиц (см. рис. 3).

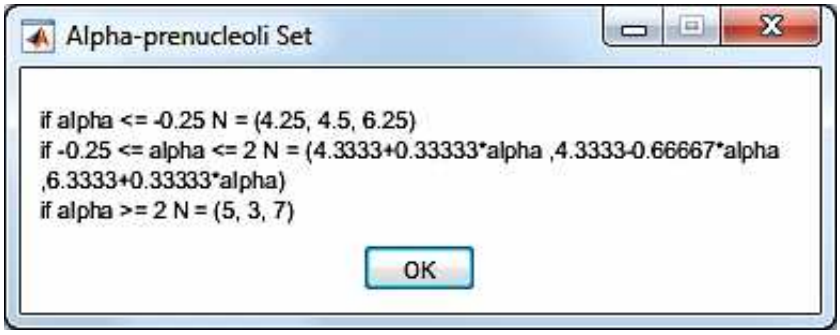


Рис. 3. Вид окна программы после расчета

Заключение

В данной работе исследовано свойство коалиционной устойчивости множества α-N-ядер в классе выпуклых кооперативных ТП-игр с конечным числом игроков. В ходе исследования изучены основные концепции решения кооперативных игр и их взаимное расположение в пространстве для класса выпуклых игр. На основе изученной информации были сформулированы и доказаны теоремы о пересечении множества α-N-ядер с C-ядром не менее чем в одной точке. Результаты теорем проиллюстрированы на нескольких примерах.

В рамках данного исследования были разработаны программы на языке программирования MATLAB, позволяющие проверять свойства выпуклости кооперативной игры и коалиционной устойчивости ее решений, находить α-N-ядра игры для любого заданного значения $\alpha \in R^1$, анализировать коалиционную устойчивость α-N-ядер с заданным интервалом и шагом для α, создавать сводную таблицу результатов проверки. Также был программно реализован результат теоремы, определяющий множество α-N-ядер произвольной кооперативной игры трех лиц.

При дальнейшем исследовании коалиционной устойчивости решений кооперативной игры планируется расширить класс рассматриваемых игр и доработать программу до нахождения всего множества α-N-ядер игры (N, v) для всех $\alpha \in R^1$ с учетом последовательной смены сбалансированных наборов на разных интервалах значения α.

Список литературы

1. Бритвин С. В., Тарашнина С И. Алгоритмы нахождения пред-N-ядра и SM-ядра в кооперативных ТП-играх // Математическая Теория Игр и ее Приложения. 2013. Т. 5. № 4. С. 14–32.

2. Мулен Э. Кооперативное принятие решений: Аксиомы и модели. М.: Мир, 1991. 464 с.

3. Печерский С. Л., Яновская Е. Б. Кооперативные игры: решения и аксиомы. СПб.: Изд. Европейского Университета в Санкт-Петербурге, 2004. 460 с.

4. Розенмюллер И. Кооперативные игры и рынки. М.: Мир, 1974. 115 с.

5. Смирнова Н. В., Тарашнина С. И. О свойствах решений кооперативных игр с трансферабельными полезностями // Известия высших учебных заведений. Математика. 2016. № 6. С. 73-85.

6. Смирнова Н. В., Тарашнина С. И. Геометрические свойства $[0, 1]$-N-ядра в кооперативных ТП-играх // Математическая Теория Игр и ее Приложения. 2012. Т. 4. № 1. С. 55-73.

7. Смирнова Н. В., Тарашнина С. И. Об одном обобщении N-ядра в кооперативных играх // Дискретный анализ и исследование операций. 2011. Т. 18. № 4. С. 77–93.

8. Kohlberg E. On the nucleolus of a characteristic function game // SIAM Journal on Applied Mathematics. 1971. V. 20. P. 62–66.

9. Maschler M., Peleg B., and Shapley L. S. Geometric properties of the kernel, nucleolus and related solution concepts // Mathematics of operations research. 1979. № 4. P. 303–338.

10. Meinhardt H. I. The Pre-Kernel as a Tractable Solution for Cooperative Games. Berlin: Springer, 2014. 242 p.

11. Scarf H. The core of an N person game // Econometrica. 1967. № 35. P. 50–69.

12. Schmeidler D. The nucleolus of a characteristic function game // SIAM Journal on Applied Mathematics. 1969. № 17. P. 1163–1170.

13. Shapley L. S. A value for n-person games // Contributions to the Theory of Games, II. Princeton University Press. 1953. P. 307–317.

14. Shapley L. S. Cores of Convex Games // International Journal of Game Theory. 1971. № 1. P, 11-26.

15. Tarashnina S. The simplified modified nucleolus of a cooperative TU-game // Operations Research and Decision Theory. 2011. V. 19. № 1. P. 150–166.

Приложение

Листинг 1. NumPlay(v).

```
function n = NumPlay(v)
m = size(v,2);
n = log2(m+1);
if round(n) ~= n
   error('"v"-dimension must agree (2^n-1) for some integer "n"')
end
```

Листинг 2. V2(v).

```
function v2 = V2(v)
m = size(v,2);
v2 = ones(1,m)*v(m) - [v(m-1:-1:1) 0];
```

Листинг 3. Valpha(v,alpha).

```
function v_alpha = Valpha(v, alpha)
v2 = V2(v);
v_alpha = alpha*v+(1-alpha)*v2;
```

Листинг 4. Coalition(v).

```
function Coal = Coalition(n)
Coal = zeros(1,n);
for i=1:(n-1)
   co = nchoosek(1:n,i);
   for j=1:length(co)
     k = zeros(1,n);
     for l=1:size(co,2)
        k(co(j,l)) = 1;
     end
     Coal = [Coal; k];
   end
end
Coal = [Coal; ones(1,n)];
```

Листинг 5. Norm2Redoo(v).

```
function [v0, c] = Norm2Redoo(v)
n = NumPlay(v);
m = 2^n-1;
coal = Coalition(n);
c = -v(1:n);
v0 = v + (coal(2:(m+1),:)*c')';
```

Листинг 6. Coal2Name(Coal).

```
function Name = Coal2Name(Coal)
n = length(Coal);
if n<10
   alf = '123456789';
elseif (n>=10) & (n<=26)
```

```
   alf = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ';
else
   error('Too many players to rename with one symbol. Specify
the names of the players by hand')
end
if Coal == zeros(1,n)
   name = '0';
else
   name = alf(find(Coal == 1));
end
Name = [blanks(n-length(name)+1) name];
```

Листинг 7. NumCoalV1(Coal).

```
function Num = NumCoalV1(Coal)
n = length(Coal);
coal = Coalition(n);
t = coal(2:length(coal),:);
y = repmat(Coal, size(t,1), 1);
c = (t == y);
u = all(c, 2);
Num = find(u);
```

Листинг 8. NumCoalV2(Coal).

```
function Num2 = NumCoalV2(Coal)
m = size(Coal,2);
Num2 = 0;
for i = 1:m
   Num2 = Num2 + Coal(i)*2^(i-1);
end
```

Листинг 9. V2Mod(v).

```
function v_mod = V2Mod(v)
n = NumPlay(v);
coal = Coalition(n);
for i = 2:2^n
   s = coal(i,:);
   nom = NumCoalV2(s);
   v_mod(nom) = v(i-1);
end
```

Листинг 10. Convexity(v).

```
function result = Convexity(v)
n = NumPlay(v);
vv = [0 v];
dim = 2^n;
coal = Coalition(n);
D = blanks(2*(n+1));
D0 = D;
for i = 1:(dim-1)
   for j = i:(dim-1)
     union = or(coal(i,:),coal(j,:));
     intersection = and(coal(i,:),coal(j,:));
     for k = 1:dim
```

```
      if coal(k,:) == union
        no_uni = k;
      end
      if  coal(k,:) == intersection
        no_int = k;
      end
    end
    if vv(i)+vv(j) > vv(no_uni)+vv(no_int)
      D1 = Coal2Name(coal(i,:));
      D2 = Coal2Name(coal(j,:));
      D = [D; D1 D2];
    end
  end
end
if D == D0
  result = 1;
else
  result = 0;
  D(1,:) = []
end
```

Листинг 11. CoreVertices(v).

```
function Core = CoreVertices(v)
vv = [0 v];
n = NumPlay(v);
players = 1:n;
k = perms(players(end:-1:1));
coal = Coalition(n);
Core0 = [];
Core = [];
for i = 1:size(k,1)
  v1 = 0;
  for j = 1:n
    v2 = v1;
    vec = k(i,1:j);
    vec2 = zeros(1,n);
    for l = 1:length(vec)
      vec2(vec(l)) = 1;
    end
    nom = NumCoalV1(vec2)+1;
    v1 = vv(nom);
    core(k(i,j)) = v1-v2;
  end;
  Core0 = [Core0; core];
end;
Core0 = unique(Core0,'rows');
for i = 1:size(Core0,1)
  d = 0;
  sum = zeros(1,2^n);
  for j = 1:(length(coal)-1)
    s = 0;
    for k = 1:n
      if coal(j,k) == 1
        s = s + Core0(i,k);
      end
    end
    if s >= vv(j)
```

```
       d = d+1;
     end
   end
   if d == (length(coal)-1)
     Core = [Core; Core0(i,:)];
   end
end
```

Листинг 12. PreNucl(v).

```
function x1 = PreNucl(v)
vv = [0 v];
n = NumPlay(v);
Coal = Coalition(n);
x1 = v(1:n);
x0 = x1 + ones(1,n);
while x0 ~= x1
   x0 = x1;
   e_x0 = vv' - Coal*x0';
   for i = 1:n
     for j = 1:n
        if i ~= j
          G = [];
          e_x00 = [];
          for k = 2:(2^n-1)
             if (Coal(k,i)==1) && (Coal(k,j)==0)
               G = [G; Coal(k,:)];
               e_x00 = [e_x00; e_x0(k)];
             end
          end
          max = min(e_x00);
          s = [];
          for k = size(G,1):-1:1
             if e_x00(k) >= max
               max = e_x00(k);
               s = G(k,:);
             end
          end
          S(i,(n*(j-1)+1):(n*(j-1)+n)) = s;
        end
     end
   end
   E = [];
   A = [];
   for i = 1:(n-1)
     for j = (i+1):n
        e = S(i,(n*(j-1)+1):(n*j))-S(j,(n*(i-1)+1):(n*i));
        E = [E -e'];
        a = v(NumCoalV1(S(i,(n*(j-1)+1):(n*j))))-
v(NumCoalV1(S(j,(n*(i-1)+1):(n*i))));
        A = [A; -a];
     end
   end
   E = [E ones(n,1)];
   A = [A; vv(end)];
   x1 = (E'\ A)';
end
```

Листинг 13. AlphaNucl(v,alpha).

```
function Ans = AlphaNucl(v,alpha)
Ans = PreNucl(Valpha(v,alpha));
```

Листинг 14. IsInCore(Imp,v).

```
function result = IsInCore(Imp, v)
m = length(v);
n = NumPlay(v);
coal = Coalition(n);
vec_imp = [];
for i = 2:(length(coal)-1)
   vec_imp(i-1) = sum(coal(i,:)*Imp');
end
if (vec_imp >= v(1:(m-1))) & (roundn(sum(Imp)-v(m),-8) == 0)
   result = 1;
else
   result = 0;
end
```

Листинг 15. TableAlphaPreNuclSet(v,alpha0,alpha1,d,flag).

```
function TableAlphaPreNuclSet(v,alpha0,alpha1,d,flag)
n = NumPlay(v);
col = abs(alpha1-alpha0)/d;
Table = [];
for i = 1:(col+1)
   alpha = alpha0 + (i-1)*d;
   Imp = AlphaNucl(v,alpha);
   InCore = IsInCore(Imp,v);
   Table = [Table; i alpha Imp InCore];
end

if flag == 0
   Table
elseif flag == 1
% save Table in file matrica.txt
   fileID = fopen('matrica.txt','w');
   formatSpec = '%5.0f %10.4f ';
   Head = ['   No      alpha  '];
   for i = 1:n
     formatSpec = strcat(formatSpec, '%9.4f ');
     Head = [Head strcat('   N_',num2str(i)),'   '];
   end;
   formatSpec = strcat(formatSpec, '%5.0f \n');
   Head = [Head 'IsInCore'];
   Head1 = [blanks(10),'v = [', num2str(v),']']
   fprintf(fileID, '%s \n', Head1);
   fprintf(fileID, '%s \n', Head);
   fprintf(fileID, formatSpec, Table');
   fclose(fileID);
elseif flag == 2
% save Table in file matrica.csv
   filename = 'matrica.csv';
   if exist(filename, 'file')==2
     delete(filename);
```

```
   end
   Head1 = 'v = ', ['[' num2str(v) ']'];
   Head2 = cell(n+3);
   Head2 = 'No','alpha';
   for i = 1:n
     Head2(2+i) = strcat('N_',num2str(i));
   end;
   Head2(n+3) = 'IsInCore';
   xlswrite(filename,Head1,1,'A1')
   xlswrite(filename,Head2,1,'A2')
   xlswrite(filename,Table,1,'A3')
elseif flag == 3
% save Table in file matrica.xlsx
   filename = 'matrica.xlsx';
   if exist(filename, 'file')==2
     delete(filename);
   end
   Head1 = 'v = ', ['[' num2str(v) ']'];
   Head2 = cell(n+3);
   Head2 = 'No','alpha';
   for i = 1:n
     Head2(2+i) = strcat('N_',num2str(i));
   end;
   Head2(n+3) = 'IsInCore';
   xlswrite(filename,Head1,1,'A1')
   xlswrite(filename,Head2,1,'A2')
   xlswrite(filename,Table,1,'A3')
end
```

Листинг 16. ThreePlayersGame(v).

```
function ThreePlayersGame_e
clear all; close all;
variable_name = {'v1 = ', 'v2 = ', 'v3 = ', 'v12 = ', 'v13 = ',
'v23 = ', 'vN = '};
peremennie_initial = {'0','0','0','0','0','0','0'};

Peremennie = inputdlg(variable_name,'Ввод переменных',1,
peremennie_initial,'on');
for i = 1:length(Peremennie)
   v(i) = str2num(Peremennie{i});
end
[v0,c] = Norm2Redoo(v);

cArray = num2cell(v0(4:6));
[v_12 v_13 v_23] = cArray{:};
cArr = num2cell(sort(v0(4:6)));
[v12 v13 v23] = cArr{:};

delta_vN = v0(7) - sum(v0(4:6));
sgn = sign(delta_vN);

alpha1 = 0.5 - abs(delta_vN)/(2*v23-2*v13);
alpha2 = 0.5 - abs(delta_vN)/(2*v23+2*v13-4*v12);
alpha3 = 0.5 + abs(delta_vN)/(4*v23-2*v13-2*v12);
alpha4 = 0.5 + abs(delta_vN)/(2*v13-2*v12);

alpha = [ alpha1;
```

```
          alpha2;
          alpha3;
          alpha4 ];

N1_x1 = (v12+v13)/2;
N1_x2 = (v23+v12+delta_vN)/2;
N1_x3 = (v23+v13+delta_vN)/2;

N2_x1 = (2*v13+2*v12+delta_vN)/4 - sgn*(v23-v13)/4;
N2_x2 = (2*v23+2*v12+delta_vN)/4 + sgn*(v23-v13)/4;
N2_x3 = (v13+v23+delta_vN)/2;

N2_a1 = sgn*(v23-v13)/2;
N2_a2 = -sgn*(v23-v13)/2;
N2_a3 = 0;

N3_x1=((3*v13+3*v12+2*delta_vN)-sgn*(2*v23-v13-v12))/6;
N3_x2=((3*v23+3*v12+2*delta_vN)-sgn*(2*v13-v23-v12))/6;
N3_x3=((3*v23+3*v13+2*delta_vN)-sgn*(2*v12-v23-v13))/6;

N3_a1 = sgn*(2*v23-v13-v12)/3;
N3_a2 = sgn*(2*v13-v23-v12)/3;
N3_a3 = sgn*(2*v12-v23-v13)/3;

N4_x1 = (v13+v12+delta_vN)/2;
N4_x2 = (2*v23+2*v12+delta_vN)/4 - sgn*(v13-v12)/4;
N4_x3 = (2*v23+2*v13+delta_vN)/4 + sgn*(v13-v12)/4;

N4_a1 = 0;
N4_a2 = sgn*(v13-v12)/2;
N4_a3 = -sgn*(v13-v12)/2;

N5_x1 = (v13+v12+delta_vN)/2;
N5_x2 = (v23+v12+delta_vN)/2;
N5_x3 = (v13+v23)/2;

N = [ N1_x1 N1_x2 N1_x3;
      N2_x1 N2_x2 N2_x3;
      N2_a1 N2_a2 N2_a3;
      N3_x1 N3_x2 N3_x3;
      N3_a1 N3_a2 N3_a3;
      N4_x1 N4_x2 N4_x3;
      N4_a1 N4_a2 N4_a3;
      N5_x1 N5_x2 N5_x3];

if v12 == v_12
   if v13 == v_13
     N_ = N;
   else
     N_ = N(:,[ 2 1 3 ]);
   end
elseif v12 == v_13
   if v13 == v_12
     N_ = N(:,[ 1 3 2 ]);
   else
     N_ = N(:,[ 2 3 1 ]);
   end
```

```
else
   if v13 == v_12
     N_ = N(:,[ 3 1 2 ]);
   else
     N_ = N(:,[ 3 2 1 ]);
   end
end

N_x = N_([1 2 4 6 8],:) - c;
N_a = [zeros(1,3); N_([3 5 7],:); zeros(1,3)];

ind_inf1 = find(alpha == -Inf)';
ind_inf2 = find(alpha == Inf)';
ind_Inf_a = [ind_inf1 ind_inf2];

if length(ind_Inf_a) ~= 4
   % removing solutions with alpha = -Inf
   alpha(ind_inf1) = [];
   N_x(ind_inf1,:) = [];
   N_a(ind_inf1,:) = [];

   % removing solutions with alpha = +Inf
   alpha(ind_inf2) = [];
   N_x(ind_inf2+1,:) = [];
   N_a(ind_inf2+1,:) = [];

   % removing solutions with alpha = NaN (when 0/0)
   ind_NaN1 = find(isnan(alpha(1:fix(length(alpha)/2))) == 1)';
   ind_NaN2 = find([zeros(1,fix(length(alpha)/2))
 isnan(alpha((fix(length(alpha)/2)+1):length(alpha)))'] == 1);
   ind_NaN_a = [ind_NaN1 ind_NaN2];
   if length(ind_NaN_a) ~= length(alpha)
     alpha(ind_NaN1) = [];
     N_x(ind_NaN1,:) = [];
     N_a(ind_NaN1,:) = [];
     alpha(ind_NaN2) = [];
     N_x(ind_NaN2+1,:) = [];
     N_a(ind_NaN2+1,:) = [];
   else
     alpha = alpha(1);
   end

   % removing solutions with equal alpha
   if length(alpha) > 1
     ind_alpha = [];
     for i = 2:length(alpha)
        if (alpha(i-1) == alpha(i))
          ind_alpha = [ind_alpha i];
        end
     end
     alpha(ind_alpha) = [];
     N_x(ind_alpha,:) = [];
     N_a(ind_alpha,:) = [];
   end

   if (length(alpha) == 1)
     if (N_x(1,:) == N_x(end,:))
```

```
       RES = strcat('for any alpha N = (',num2str(N_x(1,1)),',',
32, num2str(N_x(1,2)),',',32,num2str(N_x(1,3)),')');
     else
        warning('length(alpha)==1 and (N_x(1,:)!=N_x(end,:))');
     end
   else
     res1 = strcat('if alpha <=',32,num2str(alpha(1)),' N = (',
num2str(N_x(1,1)), ',', 32, num2str(N_x(1,2)), ',', 32,
num2str(N_x(1,3)), ')');
     simbols = length(res1);
     for j = 2:length(alpha)
        res = strcat('if',32,num2str(alpha(j-1)),' <= alpha <=',32,
num2str(alpha(j)),' N = (');
        for k = 1:3
          if N_a(j,k) == 0
             if k<3
               okon = strcat(' ,',32);
             else
               okon = strcat(')');
             end
          else
             if sign(N_a(j,k))==1
               znak = '+';
             else
               znak = '-';
             end
             if k<3
               okon = strcat(znak,num2str(abs(N_a(j,k))),
'*alpha , ',32);
             else
               okon = strcat(znak,num2str(abs(N_a(j,k))),'*alpha)');
             end
          end
          res = strcat(res, num2str(N_x(j,k)), okon);
        end
        eval(['res', num2str(j),' = res;']);
        eval(['simbols = [simbols length(res', num2str(j), ')];']);
     end
     res = strcat('if alpha >=', 32, num2str(alpha(end)), ' N = (',
num2str(N_x(end,1)), ',', 32, num2str(N_x(end,2)), ',', 32,
num2str(N_x(end,3)),')');
     eval(['res', num2str(j+1),' = res;']);
     eval(['simbols = [simbols length(res',num2str(j+1),')];']);
     simba = max(simbols);
     RES = [];
     for ll = 1:(j+1)
        eval(['RES = [RES; [res', num2str(ll),
' blanks(simba+1-simbols(ll))]];']);
     end
   end
else
   RES = strcat('for any alpha N =(',num2str(N_x(3,1)), ',', 32,
num2str(N_x(3,2)), ',', 32, num2str(N_x(3,3)), ')');
end

msgbox({RES},'Alpha-prenucleoli Set');
end
```