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

Варламова Анна Николаевна

Выпускная квалификационная работа бакалавра

Методы нахождения кооперативных решений в игре "Аэропорт"

Направление 010400

Прикладная математика, фундаментальная информатика и основы программирования

Научный руководитель,
кандидат физ.-мат. наук,
ассистент
Панкратова Я. Б.

Санкт-Петербург

2017

Содержание

Введение

В данной работе рассматривается задача, в которой конкурирующие авиакомпании объединяются для обслуживания взлетно-посадочной полосы. Существуют разные самолеты, для которых нужны разные длины полос. Соответственно, нет никакого смысла авиакомпаниям оплачивать обслуживание той части полосы, которой они не пользуются. Также есть множество других факторов, которые стоит учесть: количество посадочных мест (чем больше мест, тем больше самолет, следовательно больше вес самолета, очевидно, что тогда он сильнее влияет на износ взлетно-посадочной полосы, поэтому денежный вклад в обслуживание должен быть больше), количество полетов. В работе эти факторы учитываются в виде двух значений: затраты и доход.

Такая проблема в Теории игр называется Игрой Аэропорт с прибылью [1]. Игроки – это авиакомпании, которые объединяются, для минимизации своих затрат. В формулировке проблемы Аэропорта, которую изначально предложили Littlechild S. C. и Owen G. [2], не учитывается доход авиакомпаний, важный в учете распределения средств на обслуживание, поскольку ни одна из авиакомпаний не согласится заплатить больше своего дохода. В данной работе это учитывается.

Проблемы с распределением затрат возникают во многих реальных жизненных ситуациях, когда люди со своими собственными целями, решают работать вместе. Например, построение асфальтированной дороги в частном секторе, проведение водопровода на нескольких жильцов. В этих ситуациях возникает проблема, как разделить между участниками совместные затраты, которые являются результатом сотрудничества. Существует много видов предложений по решению проблемы распреде-

ления затрат.

В качестве решения в данной работе выбрано N-ядро. Такой выбор был связан, во-первых, с тем, что это решение определяется единственным образом. Во-вторых, принадлежит C-ядру, и если оно существует, то есть N-ядро является недоминируемым дележом.

В работе рассматривается два способа построения N-ядра: с использованием теоремы Колберга [3] и с помощью алгоритма, предложенного в статье [1], который разработан специально для Игры Аэропорт с прибылью.

Постановка задачи

Пусть взлетно-посадочную полосу используют n авиакомпаний, то есть дано дискретное множество игроков. Кроме того, заданы расходы каждой авиакомпании и прибыль. Игроки объединяются для минимизации издержек на ремонт и содержание полосы. Нужно распределить затраты на обслуживание взлетно-посадочной полосы, исходя из этих данных.

Целью работы является нахождение решения игры Аэропорт с прибылью. Для достижения поставленной цели нужно выполнить ряд задач:

1. Изучить различные алгоритмы построения N-ядра;

2. Построить N-ядро в игре Аэропорт с прибылью двумя способами;

3. Применить решение к реальным данным;

4. Обеспечить программную реализацию, способную строить N-ядро для n игроков.

Обзор литературы

При написании данной работы были использованы научная, а также учебно-методическая литература, публикации в различных научных изданиях.

Основные понятия и определения для нахождения N-ядра на основании теоремы Колберга были изучены при помощи монографии [3]. Также об этом было прочитано в учебнике [4] и статье [5].

Главным источником для изучения альтернативного алгоритма построения N-ядра в игре Аэропорт с прибылью стала публикация в научном издании [1] и для постановки задачи [6]. Авторы Branzei R., Inarra E., Tijs S. и Zarzuelo J.[1] вывели интересный для изучения алгоритм с учетом прибыли игроков. В статье рассматриваются и доказываются утверждения и теоремы, на основании которых построен алгоритм.

Поскольку игра Аэропорт впервые была предложена Littlechild S. C. и Thompson G. F. [7], а чуть позже Litllechild S. C. и Owen G. рассмотрели простой алгоритм построения решения этой игры в виде вектора Шепли, была рассмотрена их статья [2], и в качестве примера взяты те же данные годовых затрат авиакомпаний на обслуживание взлетно-посадочной полосы аэропорта Бермингема за 1968-1969 гг.

Помимо этого некоторые определения и формулировки были изучены в работе [8].

Реальные данные для примера были взяты из годовых отчетов авиакомпаний за 2015 г. [10]-[13].

Глава 1. Нахождение N-ядра с использованием теоремы Колберга

1.1. Основные понятия и определения

Рассмотрим метод нахождения N-ядра, основанный на теореме Колберга. Прежде чем приступить к алгоритму, введем основные определения и понятия.

Определение 1.1. [1] Набор $(N, \preceq, C, r)$ называется *игрой Аэропорт с прибылью*, если

N – конечное непустое множество;

$\preceq$ – отношение порядка на множестве N;

$C : N \to \mathbf{R}_+$ неубывающая функция, то есть $(i \leq j \Rightarrow C(i) \leq C(j))$;

$r \in \mathbf{R}_+^N$.

Определение 1.2. [1] Множество N – это множество игроков. Любое непустое подмножество S множества N называется *коалицией.*

Определение 1.3. [3] *Кооперативной игрой с трансферабельными полезностями (ТП-игрой)* или *игрой в форме характеристической функции* называется пара (N, v), где N – конечное множество игроков, $v : 2^N \to \mathbf{R}$ – характеристическая функция, такая что $v(\emptyset) = 0$.

Характеристическая функция Игры Аэропорт с прибылью задается формулой [1]:

$$v^{(N,C,r)}(S) = \max\{r(R) - C(R) : R \subseteq S\} \text{ для любого } S \subseteq N, \quad (1)$$

где $r(S)$ – общая прибыль членов коалиции и находится по формуле: $r(S) = \sum\limits_{i \in S} r_i$.

Рассмотрим ТП-игру. Полагая, что игроки сформировали максималь-

ную коалицию N, рассмотрим задачу распределения величины $v(N)$ между всеми игроками игры (N, v).

Определим множество допустимых векторов выигрышей в игре (N, v) [5]:

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

Определение 1.4. [3] *Множеством эффективно-рациональных векторов выигрышей* в игре (N, v) называется множество

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

Определение 1.5. [5] *Множеством дележей* $X(N, v)$ в игре (N, v) называется множество эффективно-рациональных выигрышей, удовлетворяющих условию индивидуальной рациональности, если для векторов $x \in X^0(N, v)$, выполняется $x_i \geq v(\{i\})$ для всех $i \in N$.

Определение 1.6. [3] *Эксцессом* любой коалиции S для любого x называется функция

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

То есть эксцесс – это мера неудовлетворенности игроков распределением выигрыша.

Определение 1.7. [3] Набор коалиций $T \subset N$ называется *сбалансированным*, если существуют такие $\lambda_S > 0, S \subset T$, что для любого $i \in N$ выполняется $\sum\limits_{\substack{T \subset S \\ i \in S}} \lambda_S = 1$.

Определение 1.8. [14] Сбалансированный набор коалиций называется *минимальным*, если не существует его подмножества, являющегося сбалансированным.

Минимальные сбалансированные наборы для $n = 3$:

$T_1 = \{\{1\}, \{2\}, \{3\}\}$;

$T_2 = \{\{1,2\},\{1,3\},\{2,3\}\}$;

$T_3 = \{\{1\},\{2,3\}\}$;

$T_4 = \{\{2\},\{1,3\}\}$

$T_5 = \{\{3\},\{1,2\}\}$

Определение 1.9. [3] N-ядром относительно множества $X \subset X^0$ называется множество векторов $x \in X$ таких, что:

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

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

Обозначим через $T_\gamma(x)$ следующий набор коалиций [3]:

$$T_\gamma(x) = \{(S \subsetneqq N)|e(S,x) \geq \gamma\},$$

где $x \in X^0(N,v)$ и $\gamma \in \mathbf{R}$.

Тогда справедлива теорема.

Теорема 1.1. [9] (Теорема Колберга 1971) Для того, чтобы вектор выигрышей $x \in X$ ТП-игры (N,ν) являлся ее N-ядром, необходимо и достаточно, чтобы наборы $T_\gamma(x)$ были пусты или сбалансированны для любого $\gamma \in \mathbf{R}$.

1.2. Метод построения N-ядра, основанный на теореме Колберга

Метод построения N-ядра в игре, заданной в форме характеристической функции, основанный на Теореме 1.1. предлагает следующий алгоритм действий.

Сначала выписывают характеристическую функцию для всех подмножеств множества N, которая рассчитывается по формуле (1).

Далее рассчитываются компоненты вектора эксцесса для каждой коалиции.

После этого нужно рассматривать сбалансированные наборы T_γ. Приравниваем эксцессы тех коалиций, которые присутствуют в наборе и составляем систему линейных алгебраических уравнений. При составлении системы строки могут сокращаться, поэтому следует помнить, что x – это дележ, который обладает свойством:

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

Его можно добавить, если система окажется неполной.

Решая систему уравнений, получаем вектор $x = (x_1, x_2, \ldots, x_n)$, следует проверить, удовлетворяет ли найденное решение системе неравенств, составленной по Теореме 1.1. То есть, нужно убедиться, что эксцессы данного набора не меньше эксцессов остальных.

Если это выполняется, значит найденный вектор x и есть N-ядро игры (N, v).

1.3. Численный пример для $N = 1, 2, 3$

Рассмотрим игру Аэропорт с прибылью, используя данные из Аэропорта Бермингема за 1968-1969 гг. (Таблица 1.).

Таблица 1: Данные из Аэропорта Бермингема за 1968-1969 гг.

Авиакомпания	**Прибыль,** C_i	**Затраты,** r_i
1	65899	97436
2	76725	102496
3	95200	98142

Выпишем характеристическую функцию для всех $S \subseteq N$:

$$v(\{1\}) = \max\{r(\{1\}) - C(\{1\})\} = 97436 - 65899 = 31537;$$

$$v(\{2\}) = \max\{r(\{2\}) - C(\{2\})\} = 102496 - 76725 = 25771;$$

$$v(\{3\}) = \max\{r(\{3\}) - C(\{3\})\} = 98142 - 95200 = 2942;$$

$$v(\{1,2\}) = \max\{r(\{1\}) - C(\{1\}), r(\{2\}) - C(\{2\}),$$
$$r(\{1,2\}) - C(\{1,2\})\} = \max\{31537, 25771, 123207\} = 123207;$$

$$v(\{1,3\}) = \max\{r(\{1\}) - C(\{1\}), r(\{3\}) - C(\{3\}),$$
$$r(\{1,3\}) - C(\{1,3\})\} = \max\{31537, 2942, 100378\} = 100378;$$

$$v(\{2,3\}) = \max\{r(\{2\}) - C(\{2\}), r(\{3\}) - C(\{3\}),$$
$$r(\{2,3\}) - C(\{2,3\})\} = \max\{25771, 2942, 105438\} = 105438;$$

$$v(\{1,2,3\}) = \max\{r(\{1\}) - C(\{1\}), r(\{2\}) - C(\{2\}),$$
$$r(\{3\}) - C(\{3\}), r(\{1,2\}) - C(\{1,2\}), r(\{1,3\}) - C(\{1,3\}),$$
$$r(\{2,3\}) - C(\{2,3\}), r(\{1,2,3\}) - C(\{1,2,3\})\} =$$
$$\max\{31537, 25771, 2942, 123207, 100378, 105438, 202874\} = 202874.$$

Обозначим через $x = (x_1, x_2, x_3)$ дележ. Посчитаем компоненты вектора эксцесса для каждой коалиции, используя Определение 1.6.

$e(\{1\}, x) = 31537 - x_1$

$e(\{2\}, x) = 25771 - x_2$

$e(\{3\}, x) = 2942 - x_3$

$e(\{1,2\}, x) = 123207 - x_1 - x_2$

$e(\{1,3\}, x) = 100378 - x_1 - x_3$

$e(\{2,3\}, x) = 105438 - x_2 - x_3$

Так как $x = (x_1, x_2, x_3)$ является дележом, то должна выполняться система:

$$\begin{cases} x_1 + x_2 + x_3 = v(N) \\ x_1, x_2, x_3 \geq 0 \end{cases}$$

- Рассмотрим сбалансированный набор $T_1 = \{\{1\}, \{2\}, \{3\}\}$

  Приравниваем эксцессы, составляем и решаем систему линейных алгебраических уравнений:

$$31537 - x_1 = 25771 - x_2 = 2942 - x_3$$

$$\begin{cases} 31537 - x_1 = 25771 - x_2 \\ 31537 - x_1 = 2942 - x_3 \\ x_1 + x_2 + x_3 = 202874 \end{cases} \quad (2)$$

Решением системы уравнений (2) является вектор:

$$x = \left(\frac{237235}{3}; \frac{219937}{3}; \frac{151450}{3}\right),$$

проверим, удовлетворяет ли он системе неравенств:

$$\begin{cases} 31537 - x_1 \geq 123207 - x_1 - x_2 \\ 25771 - x_2 \geq 105438 - x_2 - x_3 \\ 2942 - x_3 \geq 100378 - x_1 - x_3 \end{cases}$$

Используя элементарные преобразования, получаем:

$$\begin{cases} x_1 \geq 97436 \\ x_2 \geq 91670 \\ x_3 \geq 79667 \end{cases} \tag{3}$$

Найденные x_1, x_2, x_3 не удовлетворяют системе неравенств (3). Следовательно, решение на этом наборе не подходит и не является N-ядром.

- Рассмотрим сбалансированный набор $T_2 = \{\{1,2\},\{1,3\},\{2,3\}\}$

  Приравниваем эксцессы, составляем и решаем систему линейных алгебраических уравнений:

$$123207 - x_1 - x_2 = 100378 - x_1 - x_3 = 105438 - x_2 - x_3$$

$$\begin{cases} 123207 - x_1 - x_2 = 100378 - x_1 - x_3 \\ 123207 - x_1 - x_2 = 105438 - x_2 - x_3 \\ x_1 + x_2 + x_3 = 202874 \end{cases}$$

  Решением системы уравнений является вектор

$$x = (71861; 76921; 54092),$$

проверим, удовлетворяет ли это решение системе неравенств. Записываем систему, согласно Теореме 1.1.

$$\begin{cases} 123207 - x_1 - x_2 \geq 31537 - x_1 \\ 105438 - x_2 - x_3 \geq 25771 - x_2 \\ 100378 - x_1 - x_3 \geq 2942 - x_3 \end{cases}$$

После преобразований получаем:

$$\begin{cases} x_1 \leq 97436 \\ x_2 \leq 91670 \\ x_3 \leq 79667 \end{cases}$$

Действительно, эксцессы данного набора не меньше эксцессов остальных, поэтому нет смысла рассматривать остальные наборы. Получено N-ядро данной игры:

$$v = (71861; 76921; 54092).$$

Глава 2. Альтернативный алгоритм нахождения N-ядра

2.1. Основные утверждения и описание алгоритма

В данном параграфе введем основные утверждения, необходимые для описания алгоритма.

Рассмотрим Игру Аэропорт с прибылью (Определение 1.1.). Каждый игрок $i \in N$ несет затраты $C(i) \geq 0$, когда действует в одиночку. И $i \preceq j$, соответственно, $C(i) \leq C(j)$. Игроки объединяются в максимальную коалицию с целью снижения затрат.

Определение 2.1. [1] Пусть $x \in \mathbf{R}^N$ и $S \subseteq N$, $S \neq \emptyset$. Обозначим через $x_S \in \mathbf{R}^S$ – дележ игроков из коалиции S, и $x(S) = \sum\limits_{i \in S} x_i$. Тогда $r(S)$ – *общий выигрыш* членов коалиции S и $r(S) = \sum\limits_{i \in S} r_i$.

Так как на самом деле $C \in \mathbf{R}^N_+$, обозначим

$$C(S) = \max\{C(i) : i \in S\}.[1]$$

Таким образом, $C(S)$ – это затраты на потребности коалиции S. Будем считать, что $C(\emptyset) = 0$.

Определение 2.2. [1] Каждую тройку (N, C, r) сопоставим с ТП-игрой $(N, \nu^{(N,C,r)})$, которую назовем *Игра Аэропорт*, где

$$v^{(N,C,r)}(S) = \max\{r(R) - C(R) : R \subseteq S\},$$

для любого $S \subseteq N$. $v^{(N,C,r)}(S)$ – это *выигрыш*, который получила коалиция S.

Определение 2.3. [4] Игра (N, v) называется *выпуклой*, если

$$v(Q \cup \{i\}) - v(Q) \leq v(T \cup \{i\}) - v(T), \text{ для любого } Q \subseteq T \text{ и } i \notin T.[4]$$

ТП-игра является *выпуклой*, если предельный вклад каждого игрока увеличивается с увеличением размера коалиции, к которой он присоединяется. То есть, если для любой коалиции S и T выполняется

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

Утверждение 2.1. [1] Игра Аэропорт (N, C, r) или (N, v) является *выпуклой*.

Определение 2.4. [1] Пусть $(N, \preceq, C, r)$ – игра Аэропорт, S - подмножество множества N и $x \in \mathbf{R}_+^N$. *Редуцированной игрой Аэропорт* игры $(N, \preceq, C, r)$ относительно S и x называется игра $(S, \preceq_S, C^{S,x}, b_S)$, где

$$C^{S,x}(i) = (\min \{C(Q \cup i) - (r(Q) - x(Q)) : Q \subseteq N \setminus S\})_+,$$

для любого $i \in S$.

$(a_+ = \max \{a, 0\})$

Далее будем записывать как C^x и (S, C^x, r).

Предположим, что авиакомпании, которые обслуживают взлетно-посадочную полосу (далее игроки), согласны, что игрок i должен получить излишек x_i, то есть его вклад в финансирование будет равен $r_i - x_i$. Затем i-ый игрок покидает остальных игроков, которые продолжают вести переговоры. Предположим, что i-ый игрок указывает точную часть взлетно-посадочный полосы, за которую он будет платить. Разумно полагать, что i-ый игрок будет платить за фрагмент между начальной и последней точкой полосы, который он использует.

Если i-ый игрок оплачивает часть участка взлетно-посадочной полосы, остальные игроки столкнутся с редуцированной игрой Аэропорт, где функция затрат будет C^x, а доходы останутся прежними.

Определение 2.5. [1] Пусть (N, v) – ТП-игра, $S \subset N$ – коалиция, $x \in$

$\mathbf{R}_+^N$. *Редуцированной игрой* (N, v) относительно S и x называется игра $(S, v^{S,x})$, которая задается системой:

$$v^{S,x}(T) = \begin{cases} v(N) - x(N \setminus S), \text{ если } T = S \\ \max\{v(T \cup Q) - x(Q) : Q \subseteq N \setminus S\}, \text{ если } T \neq \emptyset, S \\ 0, \text{ если } T = \emptyset \end{cases}$$

Утверждение 2.2. [1] Если $S \subseteq N$ и x – решение игры $v^{(N,C,r)}$. Тогда

$$(v^{(N,C,r)})^{S,x} = v^{(S,C^x,r)}.$$

То есть редуцированная игра Аэропорт с прибылью соответствует редуцированной игре Аэропорт.

Далее будем считать $(N, \preceq, C, r)$ фиксированной игрой Аэропорт, где $|N| \geq 2$, $N = \{1, 2, ..., n\}$, $\preceq$ – обычный порядок. Обозначим через v – игру Аэропорт с прибылью, и $\nu - N$-ядро.

Следующие величины являются основным инструментом для алгоритма [1]:

$$\alpha_i = \frac{b(\{i+1, \ldots, n\}) - (C(N) - C(i)}{n - i + 1}, i = 1, \ldots, n-1,$$

$$\beta_i = \frac{C(i)}{i+1}, i = 1, \ldots, n-1,$$

$$\gamma_i = \frac{b_i}{2}, i = 1, ..., n-1,$$

$$\delta = \frac{b(N) - C(N)}{|N|}.$$

$$\lambda = \min\left(\{\alpha_i : i \neq n\} \cup \{\beta_i : i \neq n\} \cup \{\gamma_i : i \neq n\} \cup \{\delta\}\right).$$

Определение 2.6. [5] В игре (N, v) игрок $i \in N$ называется *"болваном"*, если

$$\Delta_i = v(S) - v(S/i) = 0, \text{ для любого } S \subseteq N : i \in S.$$

Рассмотрим два случая: $\lambda > 0$ и $\lambda \leq 0$. Эти случаи соответствуют

ситуациям, когда в v нет игроков "болванов" ($\lambda > 0$), и когда они есть ($\lambda \leq 0$).

I. $\lambda > 0$.

Замечание 2.1. [1]

1. Если $0 < \lambda \leq \alpha_i$, то $r(i, \ldots, n) > C(n) - C(i-1)$, для любого $i \in N$ (в частности $v(N) = r(N) - C(N)$).

2. Если $0 < \lambda \leq \beta_i$, тогда $C(i) > 0$, для любого $i \in N$.

Далее рассмотрим структуру набора коалиций с максимальным эксцессом в N-ядре для случая $\lambda > 0$.

Также обозначим

$$D_1(x) = \{S : S \subseteq N : e(S, x) \geq e(T, x), \text{ для всех } T \subseteq N, S \neq N, \emptyset\}.$$

т.е. семейство коалиций множества N с максимальным эксцессом для x.

Определение 2.7. [1] Если $S_1, \ldots, S_k$ – это разбиение множества N. Тогда множество$\{N \setminus S_1, \ldots, N \setminus S_k\}$, образованное с помощью дополнений, называется *антиразбиением.*

Теорема 2.1. [1] (Arin and Inarra 1998) Для каждой выпуклой игры на множестве N, семейство коалиций с максимальными эксцессами в N-ядре содержат разбиения или антиразбиения множества N.

Наша цель определить разбиения или антиразбиения в $D_1(\nu)$. Для этого рассмотрим следующие леммы.

Пусть (N, v) – ТП-игра, $x \in \mathbf{R}^N_+$ и B это семейство коалиций из N. Обозначим

$$e(B, x) = \frac{\sum\limits_{S \in B} e(S, x)}{|B|},$$

Замечание 2.1 (Arin and Inarra 1998) Если B – разбиение или анти-

разбиение множества N и $x(N) = v(N)$, тогда $e(B, x)$ не зависит от x. [1]

Если P – это разбиение, $e(P, x) = \dfrac{\sum\limits_{S \in P} v(S) - v(N)}{|P|}$,

Если A – это антиразбиение, $e(A, x) = \dfrac{\sum\limits_{S \in A} v(S) - (|A| - 1)v(N)}{|A|}$.

Лемма 2.1. [1] Если $P = S_1, \ldots, S_k$ – это разбиение множества N, тогда $v(N) > \sum\limits_{S_i \in P} v(S_i)$.

Лемма 2.2. [1] Если $A = S_1, \ldots, S_k$ – это антиразбиение множества N, тогда $(k-1)v(N) > \sum\limits_{j=1}^{k} v(S_j)$.

Лемма 2.3. [1] Для любого множества $S \in D_1(\nu)$, существует $e(S, v) < 0$.

Лемма 2.4. [1] Для всех $i \in N, 0 < v_i < r_i$.

Лемма 2.5. [1]

1. Если $S \in D_1(\nu)$ и $|S| > 1$, тогда $v(S) > 0$.

2. Если $S \in D_1(\nu)$ и $|S| > 1$, тогда $v(S) = r(S) - C(S)$.

3. Если $j \in D_1(\nu)$, и $j \neq 1$, тогда $v(j) = 0$.

Определение 2.8. [1] Коалиция называется *полной*, если выполняется одно из следующих условий:

1. $|S| = 1$.

2. $|S| = n - 1$.

3. Существует $i_0 \neq n$, такое что $S = 1, 2, \ldots, i_0$.

Лемма 2.6. [1] $D_1(\nu)$ содержит только полные коалиции.

Следующее утверждение помогает идентифицировать разбиения или антиразбиения содержатся в $D_1(\nu)$. Согласно Теореме 2.1, случаи 1, 3 и 4 соответствуют разбиению, а 2 и 3 антиразбиению. (Случай 3 относится одновременно к разбиению и антиразбиению.)

Утверждение 2.3. [1] Как минимум одно из следующих высказываний истинно:

1. Существует $i_0 \neq n$, такое что

(1.1) $\{\{1,\ldots,i_0\},\{i_0+1\},\ldots,\{n\}\} \subseteq D_1(\nu)$ и

(1.2) $\nu(\{1,\ldots,i_0\}) \neq 0$, и $\nu(\{i_0+1\}) = \ldots = \nu(\{n\}) = 0$.

2. Существует $i_0 \neq n$, такое что

(2.1) $\{\{1,\ldots,i_0\}, N \setminus \{1\},\ldots, N \setminus \{i_0\}\} \subseteq D_1(\nu)$ и

(2.2) $\nu(\{1,\ldots,i_0\}) \neq 0$.

3. Существует $i_0 \neq n$, такое что

(3.1) $\{\{i_0\}, N \setminus \{i_0\}\} \subseteq D_1(\nu)$ и

(3.2) $\nu(\{i_0\}) = 0$.

4.

(4.1) $\{\{1\},\{2\},\ldots,\{n\}\} \subseteq D_1(\nu)$ и

(4.2) $\nu(\{1\}) = \ldots = \nu(\{n\}) = 0$.

Следующее утверждение позволяет легко рассчитать N-ядро для некоторых игроков в игре Аэропорт с прибылью.

Утверждение 2.4. [1] Как минимум одно из следующих высказываний истинно:

1. Пусть $P = \{\{1,\ldots,i_0\},\{i_0+1\},\ldots,\{n\}\}, i_0 \neq n$, такое что $\nu(\{1,\ldots,i_0\}) \neq 0$, и $\nu(\{i_0+1\}) = \ldots = \nu(\{n\}) = 0$. Тогда

(1.1) $e(P,v) \geq -\alpha_{i_0}$;

(1.2) Если $P \subseteq D_1(\nu)$, тогда $e(P,v) = -\alpha_{i_0}$.

2. Пусть $A = \{\{1,\ldots,i_0\}, N \setminus \{1\},\ldots, N \setminus \{i_0\}\}$ и $\nu(\{1,\ldots,i_0\}) \neq 0$. Тогда

(2.1) $e(A,v) \geq -\beta_{i_0}$;

(2.2) Если $A \subseteq D_1(\nu)$, тогда $e(A,v) = -\beta_{i_0}$.

3. Пусть $P = \{\{i_0\}, N \setminus \{i_0\}\}$ и $\nu(\{i_0\}) = 0$. Тогда

(3.1) $e(P, v) \geq -\gamma_{i_0}$;

(3.2) Если $P \subseteq D_1(\nu),$ тогда $e(P, v) = -\gamma_{i_0}$.

4. Пусть $P = \{\{1\}, \{2\}, \ldots, \{n\}\}$ и $\nu(\{1\}) = \ldots = \nu(\{n\}) = 0$. Тогда

(4.1) $e(P, v) \geq -\delta$

(4.2) Если $P \subseteq D_1(\nu),$ тогда $e(P, v) = -\delta$.

Утверждение 2.5. [1]

1. Если $\lambda = \alpha_{i_0}$, то $v_i = \lambda$, для любого $i > i_0$.

2. Если $\lambda = \beta_{i_0}$, то $v_i = r_i - \lambda$, для любого $i \leq i_0$.

3. Если $\lambda = \gamma_{i_0}$, то $v_{i_0} = \lambda$.

4. Если $\lambda = \delta$, то $v_i = \lambda$, для любого $i \in N$.

II. $\lambda \leq 0$.

Заметим, что если $\lambda \leq 0$, то $\lambda \neq \gamma_i$, для всех $i = 1, \ldots, n$.

Утверждение 2.6. [1]

1. Если $\lambda = \alpha_{i_0} \leq 0$, для некоторого $i_0 \in N$, то $v_i = 0$, для любого $i > i_0$.

2. Если $\lambda = \beta_{i_0}$, для некоторого $i_0 \in N$, то $v_i = r_i$, для любого $i \leq i_0$.

3. Если $\lambda = \delta \leq 0$, то $v_i = 0$, для любого $i \in N$.

2.2. Алгоритм построения N-ядра [1]

Утверждение 2.7. [1]

Если $\lambda = \alpha_{i_0}$, тогда $v_{i_0} = \lambda_+$ для любого $i > i_0$.

Если $\lambda = \beta_{i_0}$, тогда $v_{i_0} = r_i - \lambda_+$ для любого $i \leq i_0$.

Если $\lambda = \gamma_{i_0}$, тогда $v_{i_0} = \lambda_+$ для любого $i \in N$.

Если $\lambda = \delta$, тогда $v_{i_0} = \lambda_+$ для любого $i \in N$.

Утверждение 2.2 и Утверждение 2.7 позволили разработать алгоритм

вычисления N-ядра Игры Аэропорт с прибылью.

Построим конечную последовательность игр Аэропорт (N_m, C_m, r_m), где $m = 1, \ldots, M$, где $(N_1, C_1, r) = (N, C, r)$. На каждом шаге m вычисляем N-ядро, x для подмножества игроков $Z_m \subseteq N_m$ в соответствии с Утверждением 2.7. Когда $Z_m = N$, алгоритм останавливается. В противном случае, $N_{m+1} = N_m \backslash Z_m$ и $C_{m+1} = C_m^x$ и рассмотрим редуцированную игру $(N_{m+1}, C_{m+1}, b_{N_{m+1}})$ Точнее, на каждом шаге m вычисляем следующие значения для каждого $i \in N_m \setminus \{l_m\}$, где $l_m = \max\{i : i \in N_m\}$:

$$\alpha_i^m = \frac{r(\{k \in N_m : k > i\}) - (C_m(N_m) - C_m(i)}{\mid \{k \in N_m : k > i\} \mid +1},$$

$$\beta_i^m = \frac{C_m(i)}{\mid \{k \in N_m : k \leq i\} \mid +1},$$

$$\gamma_i^m = \frac{r_i}{2},$$

$$\delta = \frac{r(N_m) - C_m(N_m)}{|N_m|}.$$

Далее, рассмотрим минимальное значение λ^m, т.е.

$$\lambda^m = \min\{\{\alpha_i^m : i \in N_m \setminus \{l_m\}\} \cup \{\beta_i^m : i \in N_m \setminus \{l_m\}\} \cup \{\gamma_i^m : i \in N_m\{l_m\}\} \cup \{\delta^m\}\}.$$

Определим x_i^m в соответствии с [1]:

(1) Если $\lambda^m = \alpha_{k_m}^m$, тогда $x_i^m = \lambda_+^m$, для любого $i \in N_m, i > k_m$.

(2) Если $\lambda^m = \beta_{k_m}^m$, тогда $x_i^m = r_i - \lambda_+^m$, для любого $i \in N_m, i \leq k_m$.

(3) Если $\lambda^m = \gamma_{k_m}^m$, тогда $x_i^m = \lambda_+^m$.

(4) Если $\lambda^m = \delta^m$, тогда $x_i^m = \lambda_+^m$, для любого $i \in N_m$.

Пусть Z_m - множество игроков, для которых x_i^m определяется в соответствии с условиями (1)-(4). Тогда справедлива следующая теорема.

Теорема 2.2. N-ядро игры v определяется таким соотношением:

$$\nu_i = x_i^m, i \in Z_m, m = 1, \ldots, M. \text{ [1]}$$

Замечание 2.2. Описанная выше процедура имеет не более n шагов. Для каждого шага m, учитывая особый характер множеств $N \setminus N_m$, редуцированная функция стоимости для любого $i \in N_m$:

$$C^{N_m,x}(i) = (\min\{C(\{i\}), C(N \setminus N_m \cup \{i\}) - (b(N \setminus N_m) - x(N \setminus N_M))\})_+.$$

Таким образом, на каждом шаге вычисляются $O(n)$, подразумевая, что этот алгоритм может быть выполнен за время $O(n^2)$. [1]

2.3. Численный пример для $N = \{1, 2, 3\}$

Решим игру Аэропорт с прибылью, снова используя данные из Аэропорта Бермингема за 1968-1969 гг. (Таблица 1.).

Шаг 1.

$m = 1$

$(N_1, C_1, r) = (N, C, r)$

$N_1 = \{1, 2, 3\}$

$C_1 = (65899, 76725, 95200)$

$r = (97436, 102496, 98142)$

$i \in N_1 \setminus \{l_1\}$, где $l_1 = \max\{i : i \in N_1\}$

Следовательно $i \in \{1, 2\}$

- Рассмотрим $i = 1$

$\alpha_1^1 = \dfrac{r(\{2,3\}) - (C_1(\{1,2,3\}) - C_1(1))}{|\{2,3\}| + 1} = \dfrac{200638 - (95200 - 65899)}{2+1} = \dfrac{171337}{3} = 57112\dfrac{1}{3};$

$\beta_1^1 = \dfrac{C_1(1)}{|\{1\}| + 1} = \dfrac{65899}{1+1} = 32949,5;$

$\gamma_1^1 = \dfrac{r_1}{2} = \dfrac{97436}{2} = 48718;$

$\delta^1 = \dfrac{b(\{1,2,3\}) - C(\{1,2,3\})}{|\{1,2,3\}|} = \dfrac{298074 - 95200}{3} = \dfrac{202874}{3} = 67624\dfrac{2}{3};$

- Рассмотрим $i = 2$:

$\alpha_2^1 = \dfrac{b(\{3\}) - (C_1(\{1,2,3\}) - C_1(2))}{|\{3\}| + 1} = \dfrac{98142 - (95200 - 76725)}{1+1} = \dfrac{79667}{2} = 39833,5;$

$\beta_2^1 = \dfrac{C_1(2)}{|\{1,2\}|+1} = \dfrac{76725}{2+1} = 25575;$

$\gamma_2^1 = \dfrac{b_2}{2} = \dfrac{102496}{2} = 51248;$

$\lambda^1 = \min\{\{\alpha_1^1, \alpha_2^1\} \cup \{\beta_1^1, \beta_2^1\} \cup \{\gamma_1^1, \gamma_2^1\} \cup \{\delta^1\}\} = \beta_2^1 = 25575.$

Тогда $k_1 = 2$ и можно найти первые две компоненты вектора x

$$x_i^1 = b_i - \lambda_+^1, \text{ для любого } i \in N_1, i \le k_1$$

$$x_1^1 = r_1 - \beta_2^1 = 97436 - 25575 = 71861$$

$$x_2^1 = r_2 - \beta_2^1 = 102496 - 25575 = 76921$$

$Z_1 = \{1,2\}$.

Шаг 2.

$m = 2$

$N_2 = N_1 \setminus Z_1 = \{3\}$

$C_2 = C_1^x$

$C_2 = (\min\{C(Q \cup \{i\}) - (r(Q) - x(Q)) : Q \subseteq N_1 \setminus N_2\})_+,$ для любого $i \in N_2$

$C_2 = (\min\{C(\{1,3\}) - (r(1) - x(1)), C(\{2,3\}) - (r(2) - x(2)), C(\{1,2,3\}) - (r(\{1,2\}) - x(\{1,2\})\})_+ = (\min\{69625, 69625, 44050\})_+ = 44050$

$N_2 = \{3\}$

$i \in N_2 \setminus l_2,$ где $l_2 = \max\{i \in N_2\}$, следовательно $i \in \emptyset$.

Поэтому можно посчитать только

$\delta^2 = \dfrac{r(3) - C_2^x}{|\{3\}|} = \dfrac{98142 - 44050}{1} = 54092$

$\lambda^2 = \delta^2,$ значит $x_3^2 = \delta^2,$ для любого $i \in N_2$

Получаем последнюю компоненту вектора решения:

$$x_3^2 = 54092$$

Вследствие того, что $Z_2 = \{1, 2, 3\} = N$, алгоритм останавливается.

Получили x_1, x_2, x_3, согласно Теореме 2.2., можно записать N-ядро:

$$\nu = (71861, 76921, 54092).$$

2.4. Программная реализация алгоритма

Программная реализация изученного алгоритма была проведена на языке программирования R.

На вход программы подаются следующие значения:

$C = (C_1, C_2, ..., C_n)$ – вектор затрат;

$r = (r_1, r_2, ..., r_n)$ – вектор прибыли.

Для удобства обращения к программе и изменения данных была создана функция, в которую помещен весь алгоритм.

```
find.nucleolus = function(cost, r)
```

Весь алгоритм был помещен в цикл, который выполняется пока количество игроков больше 1, потому что если на последнем шаге остался один игрок, то решение для него определяется единственным образом и равно δ. Сначала убираем из рассмотрения последнего игрока, для того, чтобы определить множество значений i, так как $i \in N_m \setminus \{l_m\}$, где $l_m = \max\{i : i \in N_m\}$.

```
tmp.players = setdiff(players, players[which.max(players)])
```

Далее находятся $\alpha, \beta, \gamma, \delta$ по формулам, и их минимальное значение – λ. Затем выбираем один из четырех путей, чтобы получить решение, исходя из Утверждения 2.7. Получаем решение и обновляем множество

Z, получаем все подмножества множества Z для того, чтобы пересчитать функцию затрат. Цикл останавливается, когда множество игроков становится пустым или его мощность равна 1.

2.5. Пример для реальных данных 2015 года

Рассмотрим N-ядро игры Аэропорт с прибылью с данными российских авиакомпаний [10]-[13].

Авиакомпания	**Прибыль за 2016 г., млн,** C_i	**Затраты за 2015 г., млн,** r_i
Ural Airlines	285,3	72,08
ЮТэйр	326,9	184,2
S7 Airlines	923	612,4
Аэрофлот	19802	9443

Подаем программе на вход два вектора:

$$C = (285.3,\ 326.9,\ 923,\ 19802)$$

$$r = (72.08,\ 184.2,\ 612.4,\ 9443)$$

Получаем N-ядро:

$$\nu = (36.04;\ 92.1;\ 306.2;\ 10793.34)$$

Значит, вклад в обслуживание взлетно-посадочной полосы авиакомпании Ural Airlines мог бы составить: $r_1 - \nu_1 = 72.08 - 36.04 = 36.04$ вместо 72.08. То есть, если все авиакомпании объединятся, затраты этого игрока уменьшатся в 2 раза.

Такая же ситуация у авиакомпаний ЮТэйр: $r_2 - \nu_2 = 184.2 - 92.1 = 92.1$ и S7 Airlines: $r_3 - \nu_3 = 612.4 - 306.2 = 306.2$.

Поскольку у авиакомпании Аэрофлот прибыль во много раз больше, значит и вклад соответствующий: $r_4 - \nu_4 = 19802 - 10793.34 = 9008.66$. Но затраты в любом случае гораздо меньше, чем если бы она действовала в одиночку.

Заключение

В ходе данной работы были выполнены все поставленные задачи. Изучены два алгоритма построения N-ядра, получено N-ядро в игре Аэропорт с прибылью двумя способами: на основании Теоремы Колберга и с использованием алгоритма, предложенного авторами Branzei R., Inarra E., Tijs S., Zarzuelo J. M. [1], разработанным специально для Игры Аэропорт с прибылью. Были найдены реальные данные: прибыль и расходы российских авиакомпаний за 2015 год [10]-[13], и посчитано решение по этим данным. Предложенное решение позволило уменьшить затраты всех авиакомпаний.

Решения, полученные разными способами, оказались одинаковыми. При использовании теоремы Колберга необходимо рассмотреть много сбалансированных наборов, особенно для большого количества игроков, а так же решать системы линейных уравнений. При $n > 3$ целесообразнее использовать второй алгоритм.

Результатом проведенного исследования является программная реализация альтернативного алгоритма.

Список литературы

1. Branzei R., Inarra E., Tijs S., Zarzuelo J. M. A Simple Algorithm for the Nucleolus of Airport Profit Games // Int J Game Theory, Springer-Verlag, 2006, с.259-272.

2. Littlechild S. C., Owen G. A Simple Expression for the Shapley Value in A Special Case // Managment Science, Vol.20, No.3, Theory Series.(Nov.,1973), с.370-372.

3. Печерский С. Л., Яновская Е. Б. Кооперативные игры: решения и аксиомы. М.: Европейский университет в Санкт-Петербурге, 2004. с.32-47.

4. Петросян Л. А., Зенкевич Н. А., Шевкопляс Е. В. Теория игр: учебник СПб.: БХВ-Петербург, 2012. 159 с.

5. Тарашнина С. И., Смирнова Н. В. Геометрические свойства $[0-1]-N$-ядра в кооперативных играх // Математическая Теория Игр и ее Приложения, 2012, т.4, в.1, с.55-58.

6. Hou D. Models, Theory and Applications in Cooperative Game Theory. Enschede, 2013. 109 с.

7. Littlechild S. C., Thompson G. F. Aircraft Landing Fees: A Game Theory Approach // The Bell Journal of Economics, Vol. 8, No. 1 (Spring, 1977), с. 186-204.

8. Farrokhi M. Coalition Formation in the Airport Problem // Institute of Mathematical Economics, Bielefeld University. March, 2009, с.6-7.

9. Kohlberg E. On the nucleolus of a characteristic function game // SIAM Journal on Applied Mathematics. 1971. V. 20. P. 62Ц66.

10. Консолидированная финансовая отчетность в соответствии с международными стандартами финансовой отчетности за 2016 год. http://ir.aeroflot.ru/fileadmin/

11. Годовой отчет ПАО Авиакомпания "ЮТэйр" за 2015 год. https://www.utair.ru/upload/annual

12. Годовой отчет по результатам работы за 2015 год. https://www.s7.ru/files/ru/investor/

13. Годовой отчет открытого акционерного общества Авиакомпания "Уральские авиалинии" за 2016 год. http://www.uralairlines.ru/content/files/Aktsioneram/

14. Сбалансированные игры. http://xity.narod.ru/game/balance.pdf

Приложение

Программная реализация альтернативного алгоритма

```
get.all.subsets = function(set){
  n = length(set)
  if (n == 1){
    return (list(set))
  } else {
    ans = c()
    for (i in 1:n){
      ans = c(ans, combn(set, i, simplify = F))
    }
    return (ans)
  }
}
find.nucleolus = function(cost, r){
  get.r = function(set){
    sum(r[set])
  }
  get.c = function(set){
    max(cost[set])
  }
  get.x = function(set){
    sum(x[set])
  }
```

```
n.players = length(cost)
players = 1:n.players #Мн-во игроков
x = rep(0, n.players) #Мн-во решений

#Прокручиваем данный алгоритм,
#пока число игроков > 1
while (n.players > 1){
  #Отбрасываем последнего игрока
  tmp.players = setdiff(players,
            players[which.max(players)])

  #Считаем альфа, бетта, гамма, дельта
  alpha = sapply(tmp.players, function(i)
(get.r(players[players > i]) - get.c(players) + get.c(i)) /
       (length(players[players > i]) + 1) )
  betta = sapply(tmp.players, function(i) get.c(i) /
            (length(players[players <= i]) + 1))
  gamma = sapply(tmp.players, function(i) r[i]/2)
  delta = (get.r(players) - get.c(players))
                          / length(players)

  #Выбираем лямбда
  lambda = min(c(alpha, betta, gamma, delta))

  #Выбираем один из четырех путей
  if (lambda %in% alpha){
```

```
  k = tmp.players[which(alpha == lambda)]
  Z = players[players > k]
  x[Z] = max(lambda, 0)
}

if (lambda %in% betta){
  k = tmp.players[which(betta == lambda)]
  Z = players[players <= k]
  x[Z] = r[Z] - max(lambda, 0)
}

if (lambda %in% gamma){
  k = tmp.players[which(gamma == lambda)]
  Z = k
  x[Z] = max(lambda, 0)
}

if (lambda == delta){
  Z = players
  x[players] = max(lambda, 0)
}

#Обновляем мн-во игроков
players = setdiff(players, Z)
n.players = length(players)
```

```
    #Получаем все подмножества Z
    Q = get.all.subsets(Z)

    #Обновляем мн-во затрат
    cost[players] = sapply(players, function(i)
    min(sapply(Q, function(q) get.c(union(i, q))
                        - get.r(q) + get.x(q))) )
  }

  #Если остался один последний игрок,
  то для него решение - дельта
  if (n.players == 1){
    x[players] = (get.r(players) - get.c(players))
                         / length(players)
  }

  return (x)
}
```

Для того, чтобы обратиться к программе, необходимо в консоли написать три команды:

```
cost <- c(<значения вектора затрат>)
r <- c(<значения вектора прибыли>)

find.nucleolus(cost, r)
```