САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

**Кафедра математической теории игр и статистических решений**

Нелюбин Лев Николаевич

**Магистерская диссертация**

**Иерархические коалиционные игры**

Направление 01.04.01  
«Прикладная математика и информатика»  
Магистерская программа  
«Исследование операций и системный анализ»

Научный руководитель, доктор физ.-мат. наук, профессор кафедры математической теории игр и статистических решений

Петросян Л.О.

Рецензент, аналитик данных, Сервисный центр Citi в Польше

Зимнухова А.В.

Санкт-Петербург

2022

**Оглавление**

[**Введение** 3](#_Toc103866275)

[**1.** **Иерархическая игра** 5](#_Toc103866276)

[**1.1.** **Бескоалиционная игра** 5](#_Toc103866277)

[**Пример 1.1** 8](#_Toc103866278)

[**1.2.** **Игра с коалициями** 14](#_Toc103866279)

[**1.3.** **Кооперативный вариант** 16](#_Toc103866280)

[**2.** **Нахождение решений при кооперации** 24](#_Toc103866281)

[**2.1.** **Пропорциональное решение** 24](#_Toc103866282)

[**Пример 2.1** 26](#_Toc103866283)

[**2.2.** **Двухуровневый принцип оптимальности** 35](#_Toc103866284)

[**Пример 2.2** 36](#_Toc103866285)

[**Выводы** 48](#_Toc103866286)

[**Список литературы** 49](#_Toc103866287)

[**Приложение** 50](#_Toc103866288)

# **Введение**

Теория игр ­­­­— раздел прикладной математики, предоставляющий инструменты для анализа ситуаций, в которых стороны, называемые игроками, принимают взаимозависимые решения. Эта взаимозависимость заставляет каждого игрока учитывать возможные решения или стратегии другого игрока при формулировании стратегии. Решение игры описывает оптимальные решения игроков, у которых могут быть схожие, противоположные или смешанные интересы, а также результаты, которые могут возникнуть в результате этих решений. Теория игр изучает три основных этапа процесса взаимодействия: выбор стратегии, формирование коалиций и торг внутри коалиций.

Что касается этики, теория игр полезна в качестве арбитражной техники для решения проблем, связанных с переговорами, и, в рамках распределительной справедливости, для распределения выгод от сотрудничества. И наоборот, его можно использовать для разработки правил (например, присвоения весов при голосовании в парламенте, члены которого представляют избирательные округа разного размера), чтобы нормальный ход игры приводил к справедливому исходу.

Иерархические игры моделируют конфликтно-управляемые системы с иерархической структурой. Такая структура определяется последовательностью уровней управления, следующих друг за другом в порядке определенного приоритета. В математической постановке иерархические игры классифицируются по числу уровней и характеру вертикальных связей.

Коалиционной игрой называется игра с непротивоположными интересами, в которой игроки могут обсуждать перед игрой свои стратегии, договариваться о совместных действиях, заключать союзы (коалиции) для объединения ресурсов.

В данной работе построена математическая модель иерархической игры с коалициями из игроков. И такой игре поставлена проблема распределения выигрыша между игроками в коалициях. Обоснованием актуальности таких игр может служить то, что в современной социальной и экономической жизни возникают и взаимодействую структуры, в которых имеется координирующий центр (верхний уровень иерархии) и группы – коалиции (нижний уровень иерархии), которые помимо собственных интересов обязаны выполнять и решения центра.

Цель работы: построение математической модели иерархической игры с коалициями, нахождение равновесия по Нэшу, нахождение функций выигрышей игроков, введение способов решения проблемы распределения выигрышей между игроками в коалициях, рассмотрение способов решения проблемы на примерах.

# **Иерархическая игра**

## **Бескоалиционная игра**

Важнейшим подклассом неантагонистических многошаговых игр являются иерархические игры. Иерархические игры моделируют конфликтно-управляемые системы с иерархической структурой. Такая структура определяется последовательностью уровней управления, следующих друг за другом в порядке определенного приоритета. В математической постановке иерархические игры классифицируются по числу уровней и характеру вертикальных связей. Простейшей из них является двухуровневая система, схема которой изображена на рис. 1.1.

.

.

.

Рис. 1.1. Древовидная структура управления

Двухуровневая конфликтно-управляемая система функционирует следующим образом. Управляющий центр , находящийся в первом уровне иерархии, выбирает вектор из заданного множества управлений , где – управляющее воздействие центра на подчиненные ему подразделения , находящиеся на втором уровне иерархии. В свою очередь, выбирают управления , где – множество управлений подразделения , предопределенное управлением центра . Цель центра заключается в максимизации по функционала , а подразделения , обладая собственными целями, стремятся максимизировать по функционалы .

Формализуем эту задачу как бескоалиционную игру -го лица (административного центра и производственных подразделений ) в нормальной форме.

Пусть игрок выбирает вектор , где

— множество стратегий игрока в игре . Вектор будем интерпретировать как набор ресурсов наименований, выделяемых центром для -го производственного подразделения.

Пусть в нашей задаче каждый из игроков , зная выбор , выбирает вектор , где

(1.1.1)

Вектор , интерпретируется как производственная программа -го производственного подразделения по различным видам продукции;  
 – производственная или технологическая матрица -го производственного подразделения ();   
 – вектор наличных ресурсов -го производственного подразделения ().

Под стратегиями игрока в игре будем понимать множество функций . Множество таких функций будем обозначать через .

Определим функции выигрышей игроков в игре . Для игрока функция выигрыша имеет вид

где , – фиксированный вектор, ; а – скалярное произведение векторов и . Функцию выигрыша для игроков полагаем равной

,

где – фиксированный вектор, ; а – скалярное произведение векторов и .

Таким образом игра имеет вид .

Построим ситуацию равновесия по Нэшу в игре . Пусть – решение задачи параметрического линейного программирования (параметром является вектор )

, (1.1.2)

и пусть – решение задачи

. (1.1.3)

Для простоты предполагаем, что максимумы в (1.1.2) и (1.1.3) достигаются. Заметим, что (1.1.3) – задача нелинейного программирования с существенно разрывной целевой функцией (максимизация ведется по , и , вообще говоря, – разрывные функции параметра ). Покажем, что точка является ситуацией равновесия в игре . Действительно,

*.*

Далее, при всех справедливо неравенство

для любой . Таким образом, никому из игроков невыгодно в одностороннем порядке отклоняться от ситуации , то есть она является равновесной.

### **Пример 1.1**

Разберем пример иерархической бескоалиционной игры, определенной в п. 1.1.

Пусть имеется двухуровневая система иерархической игры, заданная следующим образом:

Рис. 1.1.1. Пример игры с 1 центром и 4 подчиненными подразделениями

Игра состоит из пяти игроков , где на первом уровне иерархии находится управляющий центр ; на втором уровне иерархии определены игроки, являющиеся подчиненные подразделения игрока . Игра рассматривается как бескоалиционная игра.

Центр выбирает число для каждого игрока, . игрок выбирает число в качестве своей стратегии.

Определяем , где

число можем интерпретировать как ресурс, выдаваемый игроком для -го производственного подразделения .

Определяем , где

где число можем интерпретировать как производственная программа для -го производственного подразделения . Число – производственное число -го производственного подразделения и они нам заданы как: . Число – наличные ресурсы -го производственного подразделения, которые также заданы: .

Задача рассматривает нахождение и .

Определим функции выигрышей. Функция выигрыша для игрока будет:

где . Следовательно, нашу функцию мы можем переписать следующим образом:

Функции выигрыша для игроков равны:

где задан как: . Следовательно, наши функции для мы можем переписать следующим образом:

Построим ситуацию равновесия по Нэшу в нашей игре . Пусть – решение задачи параметрического линейного программирования (параметром является число )

.

Раз максимизация идет по , то мы можем из ограничения  
 выразить :

Далее переносим в правую часть, выражая :

Из этого можем находим :

И пусть – решение задачи

.

Найдем и из последнего выражения на максимизацию по

Подставляем значения в правую часть:

Из определения мы имеем ограничение , следовательно . Максимум из будет достигаться при большем коэффициенте . Из этого следует, что имеет больший коэффициент и ему нужно присвоить как можно больше ресурсов, то есть все ресурсы: . Исходя из того, мы можем решить уравнение

Мы нашли точку которая является ситуацией равновесия в игре . Докажем. Для этого сравним функции выигрыша с разными точками

и

Действительно,

так как отнимая у игрока ресурсы мы получаем выигрыш .

Далее, рассмотрим функции выигрышей для

и сравним с функцией выигрыша, если -ый игрок отклонится от стратегии , то есть не будет максимизирована:

Из этого следует, что

Можем сделать вывод, что

Таким образом, никому из игроков невыгодно в одностороннем порядке отклоняться от ситуации , то есть она является равновесной.

## **Игра с коалициями**

Рис. 1.2.1. Пример игры с коалициями на втором уровне

Опираясь на п 1.1. рассмотрим новую иерархическую игру, состоящую из игроков и коалиций . Допустим

Игрок выбирает вектор такой, что , где

Игрок выбирает вектор , где

Следовательно, неравенство будет выглядеть следующим образом.

При 1:

;

.

При n:

;

.

Из этого следует, что стратегии коалиции будут , понимаем ,

Функции выигрышей игроков и мы определяли в п 1.1. Определим тогда функцию выигрыша для коалиции .

, где .

Построим ситуацию равновесия по Нэшу. Нам известно равенство решения задачи параметрического линейного программирования (1.1.2), тогда для коалиций:

*.* (1.2.1)

И известно, что – решение задачи (1.1.3).

Предполагаем, что максимумы достигаются для (1.1.2), (1.1.3) и (1.2.1). В п. 1.1 мы уже показывали, что точка является ситуацией равновесия равновесие по Нэшу для и игроков. Докажем, что и для коалиций данная точка приводит к ситуации равновесия. Действительно,

для любой . Таким образом никому из игроков невыгодно в одностороннем порядке отклоняться от ситуации , то есть она является равновесной.

## **Кооперативный вариант**

В этом параграфе рассматривается кооперативный вариант ряда простейших иерархических игр. Строятся характеристические функции и исследуются условия существования непустого -ядра.

Исходя из содержательного смысла по Нэшу, для каждой коалиции  
 определим ее гарантированный доход следующим образом:

(1.2.1)

где – решение задачи параметрического линейного программирования (1.1.2). Первое из (1.2.1) имеет место, поскольку коалиция может добиться получения нулевого выигрыша игроком , выбирая все ; второе справедливо, так как игрок всегда может гарантировать для выигрыш не более чем , направляя каждому нулевой ресурс; третье равенство (1.2.1) имеет место, поскольку коалиция , содержащая в своем составе , всегда может обеспечить распределение всего ресурса только между своими членами.

Пусть – произвольная коалиция, содержащая . Обозначим через вектор, доставляющий максимум в задаче нелинейного программирования

(для выполнено условие ). Тогда для любой коалиции справедливо следующее неравенство:

Пусть и . Тогда . Принимая во внимание условия , имеем

где – выигрыш центра от «нефинансируемых» предприятий. В случаях или неравенство   
 очевидно.

Таким образом, функция , определяемая (1.2.1), супераддитивна и можно рассмотреть кооперативную игру в форме характеристической функции.

Рассмотрим -мерный вектор

, (1.2.2)

где . Вектор является дележом, поскольку выполнены следующие соотношения:

1. ;
2. ,

.

Напомним условие принадлежности дележа -ядру.

**Определение.** Множество недоминируемых дележей кооперативной игры называется ее -ядром.

Имеет место следующая теорема, которая характеризует -ядро.

**Теорема 1.** Для того, чтобы дележ принадлежал -ядру, необходимо и достаточно выполнение для всех неравенств

(1.2.3)

Доказательство. Для несущественных игр теорема очевидна, и в силу теоремы, что каждая существенная кооперативная игра эквивалентна некоторой игре в – редуцированной форме, где пусть

тогда , достаточно провести ее доказательство для игр в – редуцированной форме.

Докажем достаточность утверждения теоремы. Пусть для дележа выполнено условие (1.2.3). Покажем, что дележ принадлежит -ядру. Пусть это не так. Тогда найдется такой дележ , что , то есть и . Однако это противоречит (1.2.3).

Покажем необходимость условия (1.2.3). Для любого дележа , не удовлетворяющего (1.2.3), существует такая коалиция , что . Пусть – такая коалиция, тогда построим вектор по следующему правилу:

где – число элементов множества . Легко видеть, что и . Отсюда следует, что не принадлежит -ядру.

Из теоремы 1 следует, что -ядро является замкнутым, выпуклым подмножеством множества всех дележей (-ядро может быть пустым множеством).

Согласно теореме 1 необходимым и достаточным условием принадлежности дележа -ядру является выполнение неравенства

(1.2.4)

для всех коалиций .

Выведем условие, при котором дележ принадлежит -ядру. Если  
 или , то условие (1.2.4) выполнено, поскольку

Если , то условие (1.2.4) можно записать в виде

Следовательно, дележ (1.2.2) принадлежит -ядру, если для всех , выполнено неравенство

Заметим, что в данном случае мы определили характеристическую функцию игры, используя выигрыш в ситуации равновесия по Нэшу, и величина , вообще говоря, меньше максимального суммарного выигрыша всех игроков, равного

Характеристическую функцию игры можно построить и обычным способом, а именно: для каждой коалиции определить ее как значение антагонистической игры между этой коалицией и коалицией остальных игроков . Построим теперь характеристическую функцию именно таким образом. При этом несколько обобщим предыдущую задачу, введя в рассмотрение произвольные функции выигрышей участников игры. Как и ранее, будем предполагать, что центр распределяет ресурсы между подразделениями , которые используют эти ресурсы для производства продукции. Выигрыши управляющего центра и «производственных» подразделений зависят от продукции, производимой . Вектор ресурсов, имеющийся в распоряжении центра , обозначим через . Центр (игрок) выбирает систему векторов из множества

Здесь интерпретируется как вектор ресурса, выделяемый центром производственному подразделению . Возможности предприятия (игрока) определяются ресурсом , получаемым от , то есть предприятие выбирает свою производственную программу из множества при всех содержат нулевой вектор и монотонно возрастают по включению, то есть из  
 следует . Кроме того, выполнено условие (невозможность производства при отсутствии ресурсов).

Пусть . Выигрыш игрока определяется с помощью неотрицательной функции , а выигрыш игроков полагаем равными (выигрыш игрока зависит лишь от производственной программы). Для простоты будем считать, что выигрыш центра удовлетворяет условию

где слагаемое интерпретируется как выигрыш игрока , получаемый от игрока . Предположим также, что для всех и  
.

Представим иерархическую игру в виде бескоалиционной игры лица в нормальной форме, где стратегиями игрока будут векторы , а стратегиями игроков – функции из соответствующих множеств. Построим характеристическую функцию этой игры. Для каждого подмножества игроков будет равно значению антагонистической игры между коалициями и , в которой выигрыш коалиции определяется как сумма выигрышей, принадлежащих множеству игроков. Пусть . Тогда

Заметим, что для всех , поскольку игрок всегда может распределить весь ресурс среди членов коалиции , в которую он входит, лишив, таким образом, коалицию ресурсов (то есть всегда может положить для , что приводит к для всех ). Рассуждая аналогично, имеем , поскольку игроки всегда могут сделать выигрыш центра равным нулю, полагая для (не производя продукции). В том случае, когда коалиция содержит центр , очевидно, что будет распределять весь ресурс среди членов коалиции. Это соображение приводит к следующей формуле:

для .

Можно показать, что при таком определении характеристической функции, -ядро множества дележей

всегда непустое.

# **Нахождение решений при кооперации**

В данной главе рассмотрим некоторые варианты решений задач кооперативных случаев, предполагаем, что игроки объединяются в одну коалицию с целью максимизации суммарного выигрыша. Тогда как и принято в кооперативных играх возникает вопрос дележа суммарного выигрыша . Дележ долен удовлетворять двум условиям:

1. Величина должна распределяться между игроками без остатка
2. Каждый из игроков должен получить по крайней мере столько, сколько он получил бы, действуя индивидуально.

## **Пропорциональное решение**

Учитывая сложность задачи, предложим самый простой вариант решения, а именно так называемое пропорциональное решение, предлагая распределить величину между игроками пропорционально тому выигрышу, который игрок может обеспечить себе в одиночку. С этой целью введем величины каждый из которых будет интерпретироваться именно таким образом.

Сперва рассмотрим вопрос как определить величинудля -го производственного подразделения, то есть величина должна показать тот максимальный выигрыш, который игрок может получить, действуя индивидуально при наихудшем сценарии реализации событий. Этот наихудший сценарий происходит именно тогда, когда игрок не получает ресурсов от игрока , то есть, когда вектор В этом случае для нахождения максимального возможного выигрыша в таких условиях необходимо решить следующую задачу максимизации:

(2.1.1)

где – фиксированный вектор, ; а – скалярное произведение векторов и , при ограничениях , где

Вектор , интерпретируется как производственная программа -го производственного подразделения по различным видам продукции;  
 – производственная или технологическая матрица -го производственного подразделения;   
 – вектор наличных ресурсов -го производственного подразделения.

Для нахождения гарантированного выигрыша игрока поступим по-другому. Будем считать, что игроки действуют против игрока необязательно полностью прекращая производство, а лишь действуя так, как если бы они не получали ресурса от , то есть в этом случае

где – есть решение задачи (2.1.1).

Определим пропорциональное решение.

Очевидно, что сумма .

### **Пример 2.1**

Рис. 2.1.1. Пример иерархической игры с пропорциональным решением

Рассмотрим иерархическую игру, состоящую из игроков . Продемонстрируем на данном примере распределение выигрыша пропорциональным решением.

Управляющий центр , находящийся в первом уровне иерархии, выбирает вектор из заданного множества управлений , где – управляющее воздействие центра на подчиненные ему подразделения , находящиеся на втором уровне иерархии. В свою очередь, выбирают управления , где – множество управлений подразделения , предопределенное управлением центра . Цель центра заключается в максимизации по функционала , а подразделения , обладая собственными целями, стремятся максимизировать по функционалы .

Пусть игрок выбирает вектор , где

— множество стратегий игрока в игре . Вектор будем интерпретировать как набор ресурсов наименований, выделяемых центром для -го производственного подразделения.

Пусть в нашей задаче каждый из игроков , зная выбор , выбирает вектор , где

Вектор , интерпретируется как производственная программа -го производственного подразделения по различным видам продукции;  
:  
 – производственная или технологическая матрица -го производственного подразделения (); – вектор наличных ресурсов -го производственного подразделения (). Определим как:

А вектора наличных ресурсов будут равны:

Под стратегиями игрока в игре будем понимать множество функций . Множество таких функций будем обозначать через .

Определим функции выигрышей игроков в игре . Для игрока функция выигрыша имеет вид

где , – фиксированный вектор,:  
; а – скалярное произведение векторов и . Функцию выигрыша для игроков полагаем равной

*,*

где – фиксированный вектор, :  
; ; ; а – скалярное произведение векторов и .

Таким образом игра имеет вид .

Находим гарантированные выигрыши игроков, то есть игроки не получают дополнительные ресурсы от игрока (). Выигрыши будут исходить от наличных ресурсов каждого игрока .

Для игрока

Для игрока :

Для игрока :

Перед нами представлены задачи линейного программирования (ЗЛП). Решим данные ЗЛП симплекс-методом. В приложении код 1 отвечает за решение симплекс-методом ЗЛП, где максимизация идет по . Для начала распишем наглядно решение таких задач, а далее будем писать только ответ.

Решение симплекс-методом ЗЛП для игрока , с использованием симплексной таблицы:

Для построение первого опорного плана систему неравенств приведем к систему уравнений путем введения дополнительных переменных (переход к канонической форме). В первом неравенстве вводим базисную переменную , а во втором неравенстве вводим базисную переменную .

Матрица коэффициентов этой системы уравнений имеет вид:

Базисные коэффициенты — это переменные, которые входят только в одно уравнение системы ограничений и притом с единичным коэффициентом. Решим систему уравнений относительно базисных переменных: . Полагая, что свободные переменные равны , получим первый опорный план:

Базисное решение называется допустимым, если оно неотрицательно.

Таблица 1 Исходная симплекс-таблица

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *Базис* |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

Переходим к основному алгоритму симплекс-метода.

Итерация 0.

1. Проверка критерия оптимальности

Текущий опорный план неоптимален, так как в индексной строке находятся отрицательные коэффициенты.

1. Определение новой базисной переменной

В качестве ведущего выберем столбец, соответствующий переменной , так как это наибольший коэффициент по модулю.

1. Определение новой свободной переменной

Вычислим значения по строкам как частное от деления: , и выберем из них наименьшее: . Следовательно, вторая строка является ведущей.

Разрешающий элемент равен и находится на пересечении ведущего столбца и ведущей строки.

1. Пересчет симплекс-таблицы

Формируем следующую часть симплексной таблицы. Вместо в план 1 войдет переменная *.* Строка, соответствующая переменной в плане 1, получена в результате деления всех элементов строки плана 0 на разрешающей элемент . На месте разрешающего элемента получаем . В остальных клетках столбца записываем нули. Таким образом, в новом плане 1 заполнены строка и столбец . Все остальные элементы нового плана 1, включая элементы индексной строки, определяются по правилу прямоугольника (метод Жордановских преобразований). Для этого выбираем из старого плана четыре числа, которые расположены в вершинах прямоугольника и всегда включают разрешающий элемент .

где – элемент старого плана, – разрешающий элемент (равный ), и – элемент старого плана, образующие прямоугольник с элементами и . Произведя перерасчет каждого элемента в таблице, получаем новую симплекс-таблицу:

Таблица 2 Конечная симплекс-таблица

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *Базис* |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

Оптимальный план можно записать так:

*,*

*.*

Таким же образом высчитывается и для остальных игроков:

Выигрыши будут:

Мы получили гарантированные выигрыши для каждого игрока при условии того, что игрок направляет для каждого игрока нулевой ресурс, то есть :

Далее найдем гарантированный выигрыш для игрока . Выигрыш для данного игрока будет равен сумме скалярного произведения вектора и найденных нами для гарантированных выигрышей:

.

Найдем такое, что

Следовательно

Определим кому и сколько выделить (ресурсов). Решим эту задачу с помощью код 2 из приложения. Получаем, что ресурсы распределяем следующим образом:

Тогда наша система ограничений будем выглядеть следующим образом:

Найдем значение функции выигрыша с помощью рассмотренного ранее симплекс-метода. Получим

а функция равняется

Составим формулу пропорционального решения для нахождения распределения выигрышей -ых игроков:

Проверим вычисления пропорционального решения через условие:

если данное условие выполняется, то вычисления верны.

условие выполняется.

## **Двухуровневый принцип оптимальности**

Ранее в п. 1.2 мы рассматривали иерархическую игру, в которой центр распределял ресурс между игроками, которые в свою очередь могли входить в коалиции и образовывали некоторое коалиционное разбиение на уровне предприятий . Предполагали в п. 1.2, что на нижнем уровне иерархии образовалась коалиционное разбиение состоящая из следующих коалиций:

Смысл двухуровневого принципа оптимальности заключается в следующем. Предполагаем в начале, что коалиции действуют как индивидуальные игроки и вместе с игроком максимизируют суммарный выигрыш, а именно суммарный выигрыш игроков обозначим через . Этот максимальный суммарный выигрыш делится между игроками согласно вектору Шепли, то есть пропорциональным величинам

Для вычисления вектора Шепли необходимо построить характеристическую функцию игры, то есть определить значение характеристической функции для любого подмножества , здесь коалиция состоит из игроков коалиций и . Здесь нужно рассмотреть два случая:

В этом случае представляет собой максимальный суммарный выигрыш всех игроков, входящих в коалицию.

Тогда максимальный гарантированный выигрыш коалиции будет равен максимальному суммарному выигрышу которые игроки смогут себе обеспечить, не используя ресурсы от .

Вектор Шепли определяется формулой:

На втором этапе выигрыши, полученные в результате решенной задачи на первом этапе, делятся между членами коалиций пропорционально их гарантированному выигрышу. При этом игроку предписывается в качестве выигрыша его компонент вектора Шепли, вычисленный на первом этапе, а для игрока его выигрыш определяется как

то есть как пропорциональный дележ (пропорциональное решение) соответствующая компоненту вектора Шепли полученный на первом уровне между игроками.

### **Пример 2.2**

Рассмотрим иерархическую коалиционную игру.

Рис. 2.2.1. Пример иерархической игры для двухуровневого принципа оптимальности

Игра состоит из игроков . В игре имеются коалиции и , где и

Рассмотрим всевозможные коалиции для вектора Шепли:

Вычислим характеристическую функцию для представленных коалиций.

выбирает , выбирает .

Определяем , где

.

Определяем , где

где :

Стратегии коалиции будет .

; ; ; ;  
.

А наличные ресурсы заданы как:

; ; ; ;  
 .

Находим гарантированные выигрыши игроков ().

– функция выигрыша .

Фиксированный вектор определен следующим образом:

; ; ; ; .

Гарантированный выигрыш для представлен в виде:

Найдем решение максимизации функции с помощью симплекс-метода. Тогда

; .

Следовательно,

*.*

Гарантированный выигрыш для

Найдем решение максимизации функции с помощью симплекс-метода. Тогда

; ; ; ; .

.

Гарантированный выигрыш для :

; ; ; ; .

; .

Следовательно,

*.*

Теперь рассмотрим второй вариант разбиения на коалиции, где – коалиция, а рассматривается как отдельный игрок.

Нам известно , найдем . Так как игрок выбирает и находится в коалиции с игроком , следовательно, игрок должен отдать все ресурсы () игроку , для максимизации функции выигрыша . Определяем функцию выигрыша для коалиции *.*

Получаем

Из кода 1 получаем решение:

*; ,*

тогда

а выигрыш функции равен:

Рассмотрим третий вариант разбиения на коалиции, где – коалиция, а рассматривается как отдельный игрок.

Повторяя алгоритм решения, но подстраивая под новое условие коалиции, получаем:

Получаем производственный вектор:

выигрыш функции будет:

Четвертый вариант: в коалицию собираются игроки , а – отдельный игрок. Это значит, что коалиция не получает ресурсов от игрока , следовательно, выигрыш коалиции будет равен сумме гарантированных выигрышей игроков и .

Пятый вариант: когда игра состоит из одной большой коалиции .

Функция выигрыша для коалиции из всех игроков будет равна функции выигрыша .

Решив задачу с помощью кода 2 из приложения, получаем вывод, что распределить ресурсы выгоднее всего отдав все ресурсы коалиции , то есть:

; .

Тогда получаем вектора :

; ; ; ;  
.

Функция выигрыша в таком случае равняется:

Вектор Шепли:

где – количество игроков (будем считать равное , так как за игроков взяли коалиции), – количество игроков в коалиции, – коалиция, – игрок, для которого рассчитывается вектор Шепли.

Проверка вектора Шепли:

– условие равенства выполняется.

Найдем пропорциональный выигрыш на основе вектора Шепли для каждого игрока. Для этого используем формулу:

где – гарантированный выигрыш игрока.

Для игрока :

Получаем:

Для игрока :

Для игрока :

Для игрока :

Для игрока :

Тогда

;

;

;

;

.

Проверим, что сумма выигрышей, найденные через пропорциональное решение, каждого игрока в коалиции равняется выигрышу этой коалиции:

Действительно, данное равенство выполняется.

Мы получили распределение выигрыша между игрока, которые находятся в коалициях на втором уровне с помощью двухэтапного принципа оптимальности.

# **Выводы**

Таким образом в работе исследуется классическая иерархическая игра с управляющим центром на верхнем уровне иерархии и подчиненными ему подразделениями . Обобщен известный результат, касающийся конструкции равновесия по Нэшу в этой игре, в случаи, когда подчиненные подразделения могут вступать в коалиции, образуя на нижнем уровне иерархии некоторое коалиционное разбиение.

Рассмотрен кооперативный вариант игры и в качестве принципа оптимальности предложено пропорциональное решение. Этот результат проиллюстрирован на примере с 1 центром и подчиненными подразделениями .

При рассмотрении кооперативной игры, когда подчиненные подразделения объединяются в коалиции для решения используем двухуровневый принцип оптимальности, что представляет собой новый подход для иллюстрации указанной задачи. На первом уровне рассмотрена кооперация между игроками коалициями и выигрыш, полученный ими, распределяется в соответствии с вектором Шепли. На втором уровне выигрыши, полученные как компонента вектора Шепли каждой коалицией распределяются между ее участниками используя пропорциональное решение. Данная схема также проиллюстрирована на примере, в котором участвуют 5 подчиненных подразделений объединенные в две коалиции и .

# **Список литературы**

1. Fudenberg D., Tirole J. Game theory. — Mass: MIT Press, 1991.
2. Leitmann G. Cooperative and Non-Cooperative Many Players Differential Games, 1974.
3. Tatiana Gvozdeva, Ali Hameed, Arkadii Slinko Hierarchical Simple Games: Weightedness and Structural Characterization, 2011. — 14 с.
4. Аumann R. J. Acceptable points in general cooperative n-person games // Contributions to the Theory of Games IV ed. by Luce R.D., Tucker A.W, 1959. — P. 287-324.
5. Бондарева О. Н. Некоторые применения методов линейного программирования к теории кооперативных игр // Проблемы кибернетики, 1963. — 119–139 с.
6. Вайсборд Э.М. О коалиционных дифференциальных играх // Дифференциальные уравнения, 1974. — 613–623 с.
7. Васин А. А., Морозов В. В. Теория игр и модели математической экономики, 2005. — 272 с.
8. Дюбин Г. Н., Суздаль В. Г. Введение в прикладную теорию игр, 1981. — 311 с.
9. Захаров А.В. Теория игр в общественных науках. [Текст] – Изд. Дом Высшей школы экономики, 2015. — 304 с.
10. Зенкевич Н. А., Козловская Н. В. Устойчивый вектор Шепли в задаче экологического производства // Математическая теория игр и ее приложения, 2010.
11. Карлин С. Математические методы в теории игр, программировании и экономике, 1964. — 840 с.
12. Клейменов А. Ф. К кооперативной теории бескоалиционных позиционных дифференциальных игр, 1990. — 32–35 с.
13. Костромин А.В, Мухаметгалеев Д.М. Теория игр, 2013. — 87 с.
14. Л. А. Петросян, Н. А. Зенкевич, Е. В. Шевкопляс Теория игр, 2012. — 432 с.
15. Мак-Кинси Дж. К. Введение в теорию игр, 1960. — 371 с.
16. Максимушкина Е.В., Тараканов А.Ф. Коалиционная дифференциальная игра при неопределенности и устойчивость коалиционной структуры. 2004, — 77–83 с.
17. Оуэн Г. Теория игр, 1971. — 230 с.
18. Петросян Л. А., Зенкевич Н. А., Семина Е. А. Теория игр, 1998.
19. Петросян Л. А., Кузютин Д. В. Игры в развернутой форме: оптимальность и устойчивость, 2000.
20. Печерский С. Л., Яновская Е. Б. Кооперативные игры: решения и аксиомы, 2004. — 459 с.
21. Розенмюллер И. Кооперативные игры и рынки, 1974. — 115 с.
22. Субботин А. И., Ченцов А. Г. Оптимизация гарантии в задачах управления, 1981. — 288 с.
23. Угольницкий Г.А. Иерархическое управление устойчивым развитием. 2010.
24. Фон Нейман, Дж. и О. Моргенштейн. Теория игр и экономическое поведение, 1970. — 625 с.

# **Приложение**

Код 1.

**from** scipy**.**optimize **import** linprog

**import** numpy **as** np

**def** import\_data**(**file\_name**=**"input.txt"**):**

file **=** **open(**file\_name**,** "r"**)**

lines **=** file**.**readlines**()**

file**.**close**()**

matrices **=** **[]**

right\_vectors **=** **[]**

top\_vectors **=** **[]**

**for** line **in** lines**:**

matrix\_s**,** right\_s**,** top\_s **=** line**.**split**(**"/"**)**

matrices**.**append**(**string\_to\_matrix**(**matrix\_s**).**transpose**())**

right\_vectors**.**append**(**string\_to\_vector**(**right\_s**))**

top\_vectors**.**append**(**string\_to\_vector**(**top\_s**))**

**return** matrices**,** right\_vectors**,** top\_vectors

**def** string\_to\_vector**(**string\_line**):**

values\_s **=** string\_line**.**split**(**","**)**

vector **=** np**.**array**(list(map(int,** values\_s**)))**

**return** vector

**def** string\_to\_matrix**(**string\_line**):**

rows **=** string\_line**.**split**(**";"**)**

num\_cols **=** **len(**rows**[**0**].**split**(**","**))**

matrix **=** np**.**zeros**(**shape**=(len(**rows**),** num\_cols**))**

**for** i**,** row **in** **enumerate(**rows**):**

matrix**[**i**]** **=** string\_to\_vector**(**row**)**

**return** matrix

def simplex\_method(matrix, right\_vector, top\_vector):

opt = linprog(c=-top\_vector, A\_ub=matrix, b\_ub=right\_vector,

method="simplex")

print(opt)

if \_\_name\_\_ == "\_\_main\_\_":

m, r, t = import\_data()

for i in range(len(m)):

print(m[i])

print(r[i])

print(t[i])

simplex\_method(m[i], r[i], t[i])

print("-" \* 40)

Код 2.

**import** numpy **as** np

**from** scipy**.**optimize **import** linprog

**from** prettytable**.**prettytable **import** PrettyTable

**def** import\_data**(**file\_name**=**"input.txt"**):**

file **=** **open(**file\_name**,** "r"**)**

lines **=** file**.**readlines**()**

file**.**close**()**

matrices **=** **[]**

right\_vectors **=** **[]**

top\_vectors **=** **[]**

temp **=** string\_to\_vector**(**lines**[**0**])**

number\_of\_lines **=** temp**[**0**]**

b **=** temp**[**1**:]**

**for** line\_num **in** **range(**1**,** number\_of\_lines**+**1**):**

line **=** lines**[**line\_num**]**

matrix\_s**,** right\_s**,** top\_s **=** line**.**split**(**"/"**)**

matrices**.**append**(**string\_to\_matrix**(**matrix\_s**).**transpose**())**

right\_vectors**.**append**(**string\_to\_vector**(**right\_s**))**

top\_vectors**.**append**(**string\_to\_vector**(**top\_s**))**

**return** matrices**,** right\_vectors**,** top\_vectors**,** b

**def** string\_to\_vector**(**string\_line**):**

values\_s **=** string\_line**.**split**(**","**)**

vector **=** np**.**array**(list(map(int,** values\_s**)))**

**return** vector

**def** string\_to\_matrix**(**string\_line**):**

rows **=** string\_line**.**split**(**";"**)**

num\_cols **=** **len(**rows**[**0**].**split**(**","**))**

matrix **=** np**.**zeros**(**shape**=(len(**rows**),** num\_cols**))**

**for** i**,** row **in** **enumerate(**rows**):**

matrix**[**i**]** **=** string\_to\_vector**(**row**)**

**return** matrix

**def** simplex\_method**(**matrix**,** right\_vector**,** top\_vector**):**

opt **=** linprog**(**c**=-**top\_vector**,** A\_ub**=**matrix**,** b\_ub**=**right\_vector**,**

method**=**"simplex"**)**

**return** opt

**def** print\_x\_fun\_by\_opts**(**opts**):**

temp\_table **=** PrettyTable**()**

temp\_table**.**field\_names **=** **[**"Vector vi"**,** "Fun value"**]**

**for** i **in** opts**:**

temp\_table**.**add\_row**([**i**.**x**,** **-**i**.**fun**])**

**print(**temp\_table**)**

**def** main**():**

matrix**,** right**,** top**,** b **=** import\_data**(**"input4.txt"**)**

**print(**"Input data:"**)**

**for** i **in** **range(len(**top**)):**

**print(**matrix**[**i**])**

**print(**right**[**i**])**

**print(**top**[**i**])**

**print(**"-" **\*** 40**)**

**print(**"u ="**,** b**)**

opts **=** **[]**

opts\_with\_b **=** **[]**

**for** i **in** **range(len(**top**)):**

opts**.**append**(**simplex\_method**(**matrix**[**i**],** right**[**i**],** top**[**i**]))**

opts\_with\_b**.**append**(**simplex\_method**(**matrix**[**i**],** right**[**i**]+**b**,** top**[**i**]))**

**print(**"Just simplex method:"**)**

print\_x\_fun\_by\_opts**(**opts**)**

sum\_fun\_value **=** 0

**for** i **in** opts**:**

sum\_fun\_value **=** sum\_fun\_value **-** i**.**fun

**print(**"Simplex method with add u:"**)**

print\_x\_fun\_by\_opts**(**opts\_with\_b**)**

**print(**"Sums of fun values"**)**

temp\_to\_fund\_max **=** **[]**

**for** i**,** temp **in** **enumerate(**opts\_with\_b**):**

temp\_to\_fund\_max**.**append**(-**temp**.**fun **+** sum\_fun\_value **+** opts**[**i**].**fun**)**

**print(**"\tWhen use"**,** i**+**1**,** "equal ="**,**temp\_to\_fund\_max**[-**1**])**

max\_equal **=** temp\_to\_fund\_max**.**index**(max(**temp\_to\_fund\_max**))**

**print(**"Max fun value when use"**,** max\_equal**+**1**,** "equal"**)**

new\_opts **=** opts**.**copy**()**

new\_opts**[**max\_equal**]** **=** opts\_with\_b**[**max\_equal**]**

print\_x\_fun\_by\_opts**(**new\_opts**)**

**if** \_\_name\_\_**==**"\_\_main\_\_"**:**

main**()**