

**Московский Государственный Технологический Университет
“СТАНКИН”**

**Факультет
Информационных
Технологий**

**Кафедра
Прикладной
Математики**

Бондаренко А.А.

**ИТЕРАЦИОННЫЕ МЕТОДЫ
РЕШЕНИЯ МАТРИЧНЫХ ИГР**

Выпускная работа на соискание академической степени
бакалавра техники и технологии
по направлению 552800 “Информатика и вычислительная техника”

Заведующий кафедрой Прикладная Математика

д.ф.-м.н., профессор Уварова Л.А.

Научный руководитель

к.ф.-м.н., доцент Елисеева Ю.В.

Москва – 2007

**Московский Государственный Технологический Университет
“СТАНКИН”**

**Факультет
Информационных
Технологий**

**Кафедра
Прикладной
Математики**

Утверждаю

Зав. кафедрой Уварова Л.А.

**Задание
на выполнение выпускной работы**

студенту гр. И-8-1 Бондаренко А.А.

Направление подготовки 558200 «Информатика и вычислительная техника»

Тема работы

Итерационные методы решения матричных игр

Задание

1. Изучить научную литературу, посвященную современным методам решения матричных игр.
2. Сделать программную реализацию для изученных итерационных методов решения матричных игр и провести их сравнительный анализ.
3. Разработать методы получения точного решения матричных игр большой размерности.

Заведующий кафедрой

д.ф.-м.н., профессор Уварова Л.А.

Научный руководитель

к.ф.-м.н., доцент Елисеева Ю.В.

Студент

Бондаренко А.А.

Москва - 2007

Отзыв на работу Бондаренко А.А. «Итерационные методы решения матричных игр»

Работа посвящена изучению и сравнительному анализу итерационных методов решения матричных игр. В процессе работы над дипломом автор изучил дополнительно необходимые разделы теории игр, линейного программирования и численных методов. Подробно проанализированы и программно реализованы пять итерационных методов: Брауна-Робинсон, монотонный итерационный метод, метод Борисовой-Магарик, симметризации и Амвросенко. Часть работы использована при проведении лабораторных работ по курсу «Теория игр и исследование операций», подготовлены соответствующие методические указания.

Автором предложен алгоритм решения матричных игр большой размерности, основанный на усечении доминируемых стратегий, который программно реализован в работе.

Считаю, что работа заслуживает оценки отлично, а ее автор заслуживает присуждения ему степени бакалавра техники и технологии по направлению 552800 «Информатика и вычислительная техника».

Научный руководитель

к.ф.-м.н. Елисеева Ю.В.

ОГЛАВЛЕНИЕ

АННОТАЦИЯ	4
ВВЕДЕНИЕ	5
ГЛАВА I. Первоначальные сведения о матричных играх	6
1.1 ОПРЕДЕЛЕНИЕ АНТАГОНИСТИЧЕСКОЙ ИГРЫ В НОРМАЛЬНОЙ ФОРМЕ.....	6
1.2 МАКСИМИННЫЕ И МИНИМАКСНЫЕ СТРАТЕГИИ	7
1.3 СИТУАЦИИ РАВНОВЕСИЯ	8
1.4 СМЕШАННОЕ РАСШИРЕНИЕ ИГРЫ.....	10
1.5 ДОМИНИРОВАНИЕ СТРАТЕГИЙ	13
ГЛАВА II. Итеративные методы решения матричных игр.	16
2.1 ТЕОРЕТИЧЕСКИЕ ОСНОВЫ ИТЕРАТИВНОГО МЕТОДА БРАУНА-РОБИНСОН РЕШЕНИЯ МАТРИЧНЫХ ИГР	16
2.2 МОДИФИКАЦИЯ АМВРОСЕНКО МЕТОДА БРАУНА-РОБИНСОН	22
2.3 МОДИФИКАЦИИ БОРИСОВОЙ, МАГАРИК МЕТОДА БРАУНА-РОБИНСОН	26
2.4 МОНОТОННЫЙ ИТЕРАТИВНЫЙ АЛГОРИТМ РЕШЕНИЯ МАТРИЧНЫХ ИГР... ..	32
2.5 РЕШЕНИЕ СИММЕТРИЧНЫХ ИГР.....	34
2.6 СРАВНЕНИЕ ИТЕРАЦИОННЫХ МЕТОДОВ	38
ГЛАВА III Определение точного решения матричной игры.....	40
ОТСЕЧЕНИЕ СТРОГОГО ДОМИНИРУЕМЫХ СТРАТЕГИЙ МЕТОДОМ БРАУНА- РОБИНСОН	40
ОПРЕДЕЛЕНИЕ ТОЧНОГО РЕШЕНИЯ МАТРИЧНОЙ ИГРЫ МОДИФИЦИРОВАННЫМ МЕТОДОМ БРАУНА-РОБИНСОН	42
ВЫВОДЫ.....	43
СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ	44

АННОТАЦИЯ

В дипломной работе рассмотрены и программно реализованы следующие итерационные методы решения матричных игр: Брауна-Робинсон, модификация Амвросенко метода Брауна-Робинсон, модификация Борисовой, Магарик метода Брауна-Робинсон, монотонный метод, метод решения симметричных игр. Приведена их сравнительная характеристика. Для нахождения точных решений задач матричных игр большой размерности предложен алгоритм, сокращающий время вычислений. Алгоритм основан на решении игры с усеченной матрицей, путем сведения её к двойственным задачам линейного программирования, последующем восстановлении некоторых оптимальных стратегий исходной матричной игры из решений игры с усеченной матрицей. Приведены теоретическая основа разработанного метода, сравнительные расчеты.

ВВЕДЕНИЕ

Под теорией игр часто понимают теорию математических моделей принятия оптимальных решений в условиях конфликта. Таким образом, моделями теории игр можно в принципе содержательно описывать весьма разнообразные явления: экономические, правовые и классовые конфликты, взаимодействие человека с природой, биологическую борьбу за существование и т. д. Все такие модели в теории игр принято называть играми.

Одним из примеров таких моделей является, задача организации стохастически оптимального функционирования большемасштабных распределенных вычислительных систем (см. [1,2]). Применяя теоретико-игровой подход к решению задачи, строится матричная игра, моделирующая «диспетчеризацию». Решая которую получают смешанную стратегию выбора количества ЭМ.

Матричные игры без седловой точки решают, сводя их к двойственным вспомогательным задачам линейного программирования. Такие задачи решают симплекс методом, или различными его модификациями. Однако решение задач большой размерности до сих пор является актуальной проблемой. Существуют и другие методы решения, матричных игр. Так называемые итерационные методы. Они дают решение, приближенное к оптимальному с некоторой наперед заданной погрешностью. При увеличении размерности матрицы, и точности вычислений, существенно возрастает время, и число итераций необходимых для получения приближенного решения с заданной точностью. Поэтому актуальным направлением остается решение матричных игр большой размерности.

ГЛАВА I. Первоначальные сведения о матричных играх

Математическое описание игры сводится к перечислению всех действующих в ней игроков, указанию для каждого игрока всех его стратегий, а также численного выигрыша, который он получит после того, как игроки выберут свои стратегии. В результате игра становится формальным объектом, который поддается математическому анализу. Приведем основные понятия теории матричных игр, которая более содержательно описана в [3,4,5,6,7].

1.1 ОПРЕДЕЛЕНИЕ АНТАГОНИСТИЧЕСКОЙ ИГРЫ В НОРМАЛЬНОЙ ФОРМЕ

Определение. Система

$$\Gamma = (X, Y, K),$$

где X и Y — непустые множества, и функция $K : X \times Y \rightarrow \mathbb{R}^1$ называется антагонистической игрой в нормальной форме.

Элементы $x \in X$ и $y \in Y$ называются стратегиями игроков 1 и 2 соответственно в игре Γ , элементы декартового произведения $X \times Y$ (т.е. пары стратегий (x, y) , где $x \in X$ и $y \in Y$) — ситуациями, а функция K — функцией выигрыша игрока 1. Выигрыш игрока 2 в ситуации (x, y) полагается равным $[-K(x, y)]$, поэтому функция K также называется функцией выигрыша самой игры Γ , а игра Γ — игрой с нулевой суммой. Таким образом, используя принятую терминологию, для задания игры Γ необходимо определить множества стратегий X, Y игроков 1 и 2, а также функцию выигрыша K , заданную на множестве всех ситуаций $X \times Y$.

Игра Γ интерпретируется следующим образом. Игроки одновременно и независимо выбирают стратегии $x \in X, y \in Y$. После этого игрок 1 получает выигрыш, равный $K(x, y)$, а игрок 2 — $[-K(x, y)]$.

Определение. Игра $\Gamma' = (X', Y', K')$ называется подыгрой игры $\Gamma = (X, Y, K)$, если $X' \subset X$, $Y' \subset Y$, а функция $K' : X' \times Y' \rightarrow \mathbb{R}^1$ является сужением функции K на $X' \times Y'$.

Определение. Антагонистические игры, в которых оба игрока имеют конечные множества стратегий, называются матричными.

Пусть игрок 1 в матричной игре (1.1) имеет всего m стратегий. Упорядочим множество X стратегий первого игрока, т. е. установим взаимно однозначное соответствие между множествами $M = \{1, 2, \dots, m\}$ и X . Аналогично, если игрок 2 имеет n стратегий, то можно установить взаимно однозначное соответствие между множествами $N = \{1, 2, \dots, n\}$ и Y . Тогда игра Γ полностью определяется заданием матрицы $A = \{\alpha_{ij}\}$, где $\alpha_{ij} = K(x_i, y_j)$, $(i, j) \in M \times N$, $(x_i, y_j) \in X \times Y$, $i \in M$, $j \in N$ (отсюда и название игры — матричная). При этом игра Γ реализуется следующим образом. Игрок 1 выбирает строку $i \in M$, а игрок 2 (одновременно с ним) — столбец $j \in N$. После этого игрок 1 получает выигрыш α_{ij} , а второй — $(-\alpha_{ij})$. Если выигрыш равен отрицательному числу, то речь идет о фактическом проигрыше игрока.

1.2 МАКСИМИННЫЕ И МИНИМАКСНЫЕ СТРАТЕГИИ

Рассмотрим антагонистическую игру $\Gamma = (X, Y, K)$. Здесь каждый из игроков выбором стратегии стремится максимизировать свой выигрыш. Но для игрока 1 он определяется функцией $K(x, y)$, а для второго $K(x, y)$, т. е. цели игроков прямо противоположны. При этом заметим, что выигрыш игрока 1(2) определен на ситуациях $(x, y) \in X \times Y$, складывающихся в процессе игры. Но каждая ситуация, следовательно, и выигрыш игрока зависят не только от его выбора, но и от того, какая стратегия будет выбрана противником. Поэтому, стремясь получить, возможно, больший выигрыш, каждый игрок должен учитывать поведение противника.

В теории игр предполагается, что оба игрока действуют разумно, т. е. стремятся к получению максимального выигрыша, считая, что соперник действует наилучшим (для себя) образом. Что может себе гарантировать игрок 1? Пусть игрок 1 выбрал стратегию x . Тогда в худшем случае он выиграет $\min_y K(x, y)$. Поэтому игрок 1 всегда может гарантировать себе выигрыш

$\max_x \min_y K(x, y)$. Если отказаться от предположения достижимости экстремума, то игрок 1

может всегда получить выигрыш, сколь угодно близкий к величине

$$\underline{v} = \sup_{x \in X} \inf_{y \in Y} K(x, y), \quad (1.2.1)$$

которую будем называть нижним значением игры. Если же внешний экстремум в (1.2.1) достигается, то величина \underline{v} называется также максимином, принцип построения стратегии x , основанный на максимизации минимального выигрыша, — принципом максимина, а

выбираемая в соответствии с этим принципом стратегия x — максиминной стратегией игрока 1.

Для игрока 2 можно провести аналогичные рассуждения. Пусть он выбрал стратегию y . Тогда в худшем случае он проиграет $\max_x K(x, y)$. Поэтому второй игрок всегда может себе гарантировать проигрыш — $\min_y \max_x K(x, y)$. Число

$$\bar{v} = \inf_{y \in Y} \sup_{x \in X} K(x, y), \quad (1.2.2)$$

называется верхним значением игры Γ , а в случае достижения внешнего экстремума в (1.2.2) и минимаксом. При этом принцип построения стратегии y , основанный на минимизации максимальных потерь, называется принципом минимакса, а выбираемая в соответствии с этим принципом стратегия y — минимаксной стратегией игрока 2. Подчеркнем, что существование минимаксной (максиминной) стратегии определяется достижимостью внешнего экстремума в (1.2.1) и (1.2.2).

Пусть задана матричная $(m \times n)$ -игра Γ_A . Тогда экстремумы в (1.2.1) и (1.2.2) достигаются, а нижнее и верхнее значения игры соответственно равны

$$\underline{v} = \max_{1 \leq i \leq m} \min_{1 \leq j \leq n} \alpha_{ij}$$

$$\bar{v} = \min_{1 \leq j \leq n} \max_{1 \leq i \leq m} \alpha_{ij}$$

Минимакс и максимин для игры Γ_A могут быть найдены по следующей схеме:

$$\left[\begin{array}{cccc} \alpha_{11} & \alpha_{12} & \dots & \alpha_{1n} \\ \alpha_{21} & \alpha_{22} & \dots & \alpha_{2n} \\ \dots & \dots & \dots & \dots \\ \alpha_{m1} & \alpha_{m2} & \dots & \alpha_{mn} \end{array} \right] \left. \begin{array}{l} \min_j \alpha_{1j} \\ \min_j \alpha_{2j} \\ \dots \\ \min_j \alpha_{mj} \end{array} \right\} \max_i \min_j \alpha_{ij} = \underline{v}$$

$$\underbrace{\max_i \alpha_{i1} \quad \max_i \alpha_{i2} \quad \dots \quad \max_i \alpha_{in}}_{\min_i \max_j \alpha_{ij} = \bar{v}}$$

Для любой игры $\Gamma = (X, Y, K)$ справедливо следующее утверждение.

Лемма 1.2.1. В антагонистической игре Γ

$$\underline{v} \leq \bar{v}$$

1.3 СИТУАЦИИ РАВНОВЕСИЯ

Рассмотрим вопрос об оптимальном поведении игроков в антагонистической игре. Естественно считать оптимальной в игре $\Gamma = (X, Y, K)$ такую ситуацию $(x^*, y^*) \in X \times Y$, от которой ни одному из игроков невыгодно отклоняться. Такая ситуация (x^*, y^*) называется равновесной, а принцип оптимальности, основанный на построении равновесной ситуации,— принципом равновесия. Для антагонистических игр, как это будет показано ниже, принцип равновесия эквивалентен принципам минимакса и максимины. Конечно, для этого необходимо существование равновесия (т. е. чтобы принцип оптимальности был реализуем).

Определение. В антагонистической игре $\Gamma = (X, Y, K)$ ситуация (x^*, y^*) называется ситуацией равновесия или седловой точкой, если

$$K(x, y^*) \leq K(x^*, y^*) \leq K(x^*, y)$$

для всех $x \in X$ и $y \in Y$.

Множество всех ситуаций равновесия в игре Γ обозначим через $Z(\Gamma)$, $Z(\Gamma) \subset X \times Y$.

Для матричной игры Γ_A речь идет о седловых точках матрицы выигрышей A , т. е. таких точках (i^*, j^*) , что для всех $i \in M$ и $j \in N$ выполняются неравенства

$$\alpha_{ij^*} \leq \alpha_{i^*j^*} \leq \alpha_{i^*j}$$

В седловой точке элемент матрицы $\alpha_{i^*j^*}$ является одновременно минимумом в своей строке и максимумом в своем столбце.

Множество ситуаций равновесия в антагонистической игре Γ обладает свойствами, которые позволяют говорить об оптимальности ситуации равновесия и входящих в нее стратегий.

Теорема 1.3.1 Пусть $(x_1^*, y_1^*), (x_2^*, y_2^*)$ — две произвольные ситуации равновесия в антагонистической игре Γ . Тогда

$$K(x_1^*, y_1^*) = K(x_2^*, y_1^*), K(x_1^*, y_2^*) = K(x_2^*, y_1^*)$$

$$(x_1^*, y_2^*) \in Z(\Gamma), (x_2^*, y_1^*) \in Z(\Gamma)$$

Из теоремы 1.3.1 следует, что функция выигрыша принимает одно и то же значение во всех ситуациях равновесия. Поэтому разумно ввести следующее определение.

Определение. Пусть (x^*, y^*) — ситуация равновесия в игре Γ . Тогда число

$$v = K(x^*, y^*)$$

называется значением игры Γ .

Из второго утверждения теоремы следует, в частности, такой факт. Обозначим X^* и Y^* проекции множества $Z(\Gamma)$ на X и Y соответственно, т. е.

$$X^* = \{x^* \mid x^* \in X, \exists y^* \in Y, (x^*, y^*) \in Z(\Gamma)\}$$

$$Y^* = \{y^* \mid y^* \in Y, \exists x^* \in X, (x^*, y^*) \in Z(\Gamma)\}$$

Тогда множество $Z(\Gamma)$ можно представить в виде

$$Z(\Gamma) = X^* \times Y^*$$

Определение. Множество $X^*(Y^*)$ называется множеством оптимальных стратегий игрока 1(2) в игре Γ , а его элементы — оптимальными стратегиями игрока 1 (2).

Заметим, что любая пара оптимальных стратегий образует ситуацию равновесия, а выигрыш в ней равен значению игры.

Теперь приведем связь между принципом равновесия и принципами минимакса и максимина в антагонистической игре.

Теорема 1.3.2. Для того чтобы в игре $\Gamma = (X, Y, K)$ существовала ситуация равновесия, необходимо и достаточно, чтобы существовали минимакс и максимин

$$\min_y \sup_x K(x, y), \max_x \inf_y K(x, y)$$

и выполнялось равенство

$$\underline{v} = \max_x \inf_y K(x, y) = \min_y \sup_x K(x, y) = \bar{v}$$

Игры, в которых существуют ситуации равновесия, называются *вполне определенными*. Для матричной игры Γ_A из теоремы получаем:

Следствие. Для того чтобы матричная $(m \times n)$ -игра Γ_A была вполне определена, необходимо и достаточно выполнение равенства

$$\max_{1 \leq i \leq m} \min_{1 \leq j \leq n} \alpha_{ij} = \min_{1 \leq j \leq n} \max_{1 \leq i \leq m} \alpha_{ij}$$

1.4 СМЕШАННОЕ РАСШИРЕНИЕ ИГРЫ

Если в игре Γ_A не существует ситуации равновесия. Тогда согласно теореме 1.3.2 и лемме 1.2.1 имеем

$$\min_{1 \leq j \leq n} \max_{1 \leq i \leq m} \alpha_{ij} - \max_{1 \leq i \leq m} \min_{1 \leq j \leq n} \alpha_{ij} > 0$$

В этом случае максиминная и минимаксная стратегии не являются оптимальными. Более того, игрокам бывает невыгодно их придерживаться, так как они могут получить больший

выигрыш. Однако сообщение о выборе стратегии противнику может привести к еще большим потерям, чем в случае максиминной или минимаксной стратегии.

Оказывается, что в этом случае игрокам разумно действовать случайно, что обеспечивает наибольшую скрытность выбора стратегии. Результат выбора не может стать известным противнику, поскольку до реализации случайного механизма не известен самому игроку.

Определение. Случайная величина, значениями которой являются стратегии игрока, называется его смешанной стратегией.

Так, для матричной игры Γ_A смешанной стратегией игрока 1 является случайная величина, значениями которой являются номера строк $i \in M$, $M = \{1, 2, \dots, m\}$ матрицы A. Аналогично определяется смешанная стратегия игрока 2, значениями которой являются номера $j \in N$ столбцов матрицы A.

Учитывая только что введенное определение смешанных стратегий, прежние стратегии будем называть «чистыми». Так как случайная величина характеризуется своим распределением, то будем отождествлять в дальнейшем смешанную стратегию с вероятностным распределением на множестве чистых стратегий. Таким образом, смешанная стратегия x игрока 1 в игре есть m -мерный вектор

$$x = (\xi_1, \dots, \xi_m) \in \mathbb{R}^m, \sum_{i=1}^m \xi_i = 1, \xi_i \geq 0, i = 1, \dots, m$$

Аналогично, смешанная стратегия у игрока 2 есть n -мерный вектор

$$y = (\eta_1, \dots, \eta_n) \in \mathbb{R}^n, \sum_{j=1}^n \eta_j = 1, \eta_j \geq 0, j = 1, \dots, n$$

При этом $\xi_i \geq 0$ и $\eta_j \geq 0$ — вероятности выбора чистых стратегий $i \in M$ и $j \in N$ соответственно при использовании игроками смешанных стратегий x и y .

Обозначим через X а Y соответственно множества смешанных стратегий первого и второго игроков.

Определение. Пусть $x = (\xi_1, \dots, \xi_m) \in X$ — смешанная стратегия игрока 1. Тогда множество индексов

$$M_x = \{i \mid i \in M, \xi_i > 0\}$$

где $M = \{1, 2, \dots, m\}$, назовем спектром стратегии x .

Аналогично для смешанной стратегии $y = (\eta_1, \dots, \eta_n) \in Y$ игрока 2 спектр N_y определяется следующим образом:

$$N_y = \{j \mid j \in N, \eta_j > 0\}$$

где $N = \{1, 2, \dots, n\}$. Спектр смешанной стратегии состоит из таких чистых стратегий, которые выбираются с положительными вероятностями.

Для любой смешанной стратегии x спектр $M_x \neq \emptyset$, поскольку вектор x имеет неотрицательные компоненты, сумма которых равна 1.

Рассмотрим смешанную стратегию $u_i = (\xi_1, \dots, \xi_i, \dots, \xi_m)$, где $\xi_i = 1$, $\xi_j = 0$, $j \neq i$, $i = 1, 2, \dots, m$.

Такая стратегия предписывает выбор i -й строки матрицы A с вероятностью 1. Естественно отождествлять смешанную стратегию $x \in X$ с выбором i -й строки, т. е. с чистой стратегией $i \in M$ игрока 1. Аналогично отождествим смешанную стратегию $w_j = (\eta_1, \dots, \eta_j, \dots, \eta_n) \in Y$, где $\eta_j = 1$, $\eta_i = 0$, $i \neq j$, $j = 1, 2, \dots, m$, с чистой стратегией $j \in N$ игрока 2. Тем самым мы получили, что множество смешанных стратегий игрока есть расширение его пространства чистых стратегий.

Определение. Пара (x, y) смешанных стратегий игроков в матричной игре Γ_A называется ситуацией в смешанных стратегиях.

Определим выигрыш игрока 1 в ситуации (x, y) в смешанных стратегиях для матричной $(m \times n)$ -игры Γ_A как математическое ожидание его выигрыша при условии, что игроки используют смешанные стратегии соответственно x и y . Выбор стратегий игроками осуществляется независимо друг от друга, поэтому математическое ожидание выигрыша $K(x, y)$ в ситуации (x, y) в смешанных стратегиях $x = (\xi_1, \dots, \xi_m)$, $y = (\eta_1, \dots, \eta_n)$ равно

$$K(x, y) = \sum_{i=1}^m \sum_{j=1}^n \alpha_{ij} \xi_i \eta_j = (xA)y = x(Ay)$$

При этом функция $K(x, y)$ является непрерывной по $x \in X$ и $y \in Y$. Заметим, что выигрыши $K(i, y)$, $K(x, j)$ при применении одним из игроков чистой стратегии (i или j соответственно), а другим — смешанной стратегии (y или x) имеют вид

$$K(i, y) = K(u_i, y) = \sum_{j=1}^n \alpha_{ij} \eta_j = a_i y, \quad i = 1, \dots, m,$$

$$K(x, j) = K(x, w_j) = \sum_{i=1}^m \alpha_{ij} \xi_i = x a^j, \quad j = 1, \dots, n,$$

где a_i , a^j — i -я строка и j -й столбец соответственно $(m \times n)$ -матрицы A .

Таким образом, от матричной игры $\Gamma_A = (M, N, A)$ мы пришли к новой игре $\bar{\Gamma}_A = (X, Y, K)$, где X и Y — множества смешанных стратегий в игре Γ_A , а K — функция выигрыша в

смешанных стратегиях. Игру $\bar{\Gamma}_A$ будем называть смешанным расширением игры Γ_A . Игра Γ_A является подыгрой для $\bar{\Gamma}_A$, т. е. $\Gamma_A \subset \bar{\Gamma}_A$.

Определение. Ситуация (x^*, y^*) в игре $\bar{\Gamma}_A$ образует ситуацию равновесия, а число $v = K(x^*, y^*)$ является значением игры $\bar{\Gamma}_A$, если для всех $x \in X$ и $y \in Y$.

$$K(x, y^*) \leq K(x^*, y^*) \leq K(x^*, y)$$

Стратегии (x^*, y^*) , входящие в ситуацию равновесия, являются также оптимальными. Более того, стратегии x^* и y^* являются соответственно максиминной и минимаксной.

Теорема 1.4.1. Всякая матричная игра имеет ситуацию равновесия в смешанных стратегиях. Матричные игры решают, сводя их к двойственным вспомогательным задачам линейного программирования

$$\begin{array}{ll} xu \rightarrow \min, & yw \rightarrow \max, \\ xA \geq w, \quad x \geq 0 & Ay \leq u, \quad y \geq 0 \end{array}$$

$$\text{где } u = (1, \dots, 1) \in \mathbb{R}^m, \quad w = (1, \dots, 1) \in \mathbb{R}^n$$

Такие задачи решают симплекс методом, или различными его модификациями. Однако решение задач большой размерности до сих пор является актуальной проблемой см.[8].

1.5 ДОМИНИРОВАНИЕ СТРАТЕГИЙ

Сложность решения матричной игры возрастает с увеличением размеров матрицы A. Вместе с тем в ряде случаев анализ матрицы выигрышей позволяет сделать вывод, что некоторые чистые стратегии не входят в спектр оптимальной стратегии. Это приводит к замене первоначальной матрицы на матрицу выигрышней меньшей размерности.

Определение. Чистая стратегия $i \in M(j \in N)$ игрока 1 (2) называется существенной или активной стратегией, если существует оптимальная стратегия $x^* = (\xi_1^*, \dots, \xi_m^*)$ ($y^* = (\eta_1^*, \dots, \eta_n^*)$) этого игрока, для которой $\xi_i^* > 0$ ($\eta_j^* < 0$).

Определение. Говорят, что стратегия x' игрока 1 доминирует стратегию x'' в $(m \times n)$ -игре Γ_A , если для всех чистых стратегий $j \in \{1, \dots, n\}$ игрока 2 выполняются неравенства

$$x' a^j \geq x'' a^j \tag{1.5.1}$$

Аналогично, стратегия y' игрока 2 доминирует его стратегию y'' , если для всех чистых стратегий $i \in \{1, \dots, m\}$ игрока 1

$$a_i y' \leq a_i y'' \quad (1.5.2)$$

Если неравенства (1.5.1), (1.5.2) выполняются как строгие, то говорят о строгом доминировании. Частным случаем доминирования стратегий является их эквивалентность.

Определение. Будем называть стратегии x' и x'' игрока 1 эквивалентными в игре Γ_A , если для всех $j \in \{1, \dots, n\}$

$$x' a^j = x'' a^j,$$

и обозначать $x' \sim x''$.

Для двух эквивалентных стратегий x' и x'' выполняется (для каждого $y \in Y$) равенство

$$K(x', y) = K(x'', y).$$

Определение. Будем говорить, что стратегия $x''(y'')$ игрока 1(2) доминирует, если существует стратегия $x' \neq x''(y' \neq y'')$ этого игрока, которая доминирует $x''(y'')$. В противном случае стратегия $x''(y'')$ недоминирует.

Аналогично стратегия $x''(y'')$ игрока 1(2) называется строго доминируемой, если существует стратегия $x' \neq x''(y' \neq y'')$ этого игрока, которая строго доминирует $x''(y'')$, т. е. для всех $j = 1, \dots, n (i = 1, \dots, m)$ выполняются неравенства

$$x' a^j > x'' a^j \quad (a_i y' < a_i y'')$$

В противном случае говорят, что стратегия $x''(y'')$ игрока 1 (2) недоминируется строго.

Покажем, что игроки могут не использовать доминируемые стратегии. Этот факт устанавливает следующее утверждение.

Теорема 1.5.1. Если в игре Γ_A стратегия x' одного из игроков доминирует оптимальную стратегию x^* , то стратегия x' также оптимальна.

Итак, оптимальная стратегия может быть доминируема лишь оптимальной стратегией. С другой стороны, никакая оптимальная стратегия не является строго доминируемой.

Следствие: игроки не должны использовать строго доминируемые стратегии.

Теорема 1.5.2. Если в игре Γ_A стратегия x^* одного из игроков оптимальна, то x^* — недоминируема строго.

С другой стороны, интуитивно понятно, что если i -я строка матрицы A (j -й столбец) доминируется, то нет необходимости приписывать ей (ему) положительную вероятность. Таким образом, для нахождения оптимальных стратегий вместо игры Γ_A достаточно решить подыгру $\Gamma_{A'}$, где A' — матрица, получаемая из матрицы A вычеркиванием доминируемых строк и столбцов.

Введем понятие расширения смешанной стратегии x на i -м месте. Если $x = (\xi_1, \dots, \xi_m) \in X$ и $1 \leq i \leq m+1$, то расширением стратегии x на i -м месте будем называть вектор $\bar{x}_i = (\xi_1, \dots, \xi_{i-1}, 0, \xi_{i+1}, \dots, \xi_m) \in \mathbb{R}^{m+1}$

Теорема 1.5.3. Пусть Γ_A — $(m \times n)$ -игра. Предположим, что i -я строка матрицы A доминируема (т. е. доминируема чистая стратегия i первого игрока) и пусть $\Gamma_{A'}$ — игра с матрицей A' , получаемой из A вычеркиванием i -й строки. Тогда справедливы следующие утверждения:

1. $v_A = v_{A'}$
2. Всякая оптимальная стратегия y^* игрока 2 в игре $\Gamma_{A'}$ является оптимальной и в игре Γ_A .
3. Если x^* — произвольная оптимальная стратегия игрока 1 в игре $\Gamma_{A'}$ и \bar{x}_i^* — расширение стратегии x^* на i -м месте, то \bar{x}_i^* — оптимальная стратегия этого игрока в игре Γ_A .
4. Если i -я строка матрицы A строго доминируема, то произвольная оптимальная стратегия x^* игрока 1 в игре Γ_A может быть получена из некоторой оптимальной стратегии \bar{x}^* в игре $\Gamma_{A'}$ расширением на i -м месте.

Теорема 1.5.4. Пусть Γ_A — $(m \times n)$ -игра. Предположим, что j -й столбец матрицы A доминирует и пусть $\Gamma_{A'}$ — игра с матрицей A' , получаемой из A вычеркиванием j -го столбца. Тогда справедливы следующие утверждения:

1. $v_A = v_{A'}$
2. Всякая оптимальная стратегия x^* игрока 1 в игре $\Gamma_{A'}$ является оптимальной и в игре Γ_A .
3. Если y^* — произвольная оптимальная стратегия игрока 2 в игре $\Gamma_{A'}$ и \bar{y}_j^* — расширение стратегии y^* на j -м месте, то \bar{y}_j^* — оптимальная стратегия этого игрока в игре Γ_A .
4. Если j -й столбец матрицы A строго доминирует, то произвольная оптимальная стратегия \bar{y}^* игрока 2 в игре Γ_A может быть получена из некоторой оптимальной стратегии y^* в игре $\Gamma_{A'}$ расширением на j -м месте.

ГЛАВА II. Итеративные методы решения матричных игр.

2.1 ТЕОРЕТИЧЕСКИЕ ОСНОВЫ ИТЕРАТИВНОГО МЕТОДА БРАУНА-РОБИНСОН РЕШЕНИЯ МАТРИЧНЫХ ИГР

Идея метода — многократное фиктивное разыгрывание игры с заданной матрицей выигрыша см. [7, 9]. Одно повторение игры будем называть партией. Пусть разыгрывается игра с $(m \times n)$ -матрицей $A = \{\alpha_{ij}\}$. В 1-й партии оба игрока выбирают совершенно произвольные чистые стратегии. В k -й партии каждый игрок выбирает ту чистую стратегию, которая максимизирует его ожидаемый выигрыш против наблюдаемого эмпирического вероятностного распределения противника за $(k-1)$ партий.

Итак, предположим, что за первые k разыгрываний игрок 1 использовал i -ю стратегию ξ_i^k раз ($i=1, \dots, m$), а игрок 2 — j -ю стратегию η_j^k раз ($j=1, \dots, n$). Тогда в $(k+1)$ -й партии игрок 1 будет использовать i_{k+1} -ю стратегию, а игрок 2 — свою j_{k+1} -ю стратегию, где

$$\bar{v}^k = \max_i \sum_j \alpha_{ij} \eta_j^k = \sum_j \alpha_{i_{k+1}j} \eta_j^k, \underline{v}^k = \min_j \sum_i \alpha_{ij} \xi_i^k = \sum_i \alpha_{ij_{k+1}} \xi_i^k$$

и

$$\underline{v}^k = \min_j \sum_i \alpha_{ij} \xi_i^k = \sum_i \alpha_{ij_{k+1}} \xi_i^k$$

Пусть v — значение матричной игры Γ_A . Рассмотрим отношения

$$\bar{v}^k / k = \max_i \sum_j \alpha_{ij} \eta_j^k / k = \sum_j \alpha_{i_{k+1}j} \eta_j^k / k, \quad (2.1.1)$$

$$\underline{v}^k / k = \min_j \sum_i \alpha_{ij} \xi_i^k / k = \sum_i \alpha_{ij_{k+1}} \xi_i^k / k \quad (2.1.2)$$

Векторы $x^k = (\xi_1^k / k, \dots, \xi_m^k / k)$, и $y^k = (\eta_1^k / k, \dots, \eta_n^k / k)$ являются смешанными стратегиями игроков 1 и 2 соответственно, поэтому по определению значения игры имеем

$$\max_k \underline{v}^k / k \leq v \leq \min_k \bar{v}^k / k \quad (2.1.3)$$

Таким образом, получен некоторый итеративный процесс, позволяющий находить приближенное решение матричной игры, при этом степень близости приближения к истинному значению игры определяется длиной интервала $[\max_k \underline{v}^k / k, \min_k \bar{v}^k / k]$.

Сходимость алгоритма гарантируется следующей теоремой см. [7, 10]:

Теорема 2.1.1

$$\lim_{k \rightarrow \infty} \left(\min_k \bar{v}^k / k \right) = \lim_{k \rightarrow \infty} \left(\max_k \underline{v}^k / k \right) = v$$

Приведем ее доказательство.

Введем обозначения: $A_{i\cdot}$ означает i -ю строку A , а $A_{\cdot j}$ — j -ый столбец. Аналогично, если $V(t)$ — вектор, то $v_j(t)$ означает его j -ю компоненту. Положим

$$\max V(t) = \max_j v_j(t), \min V(t) = \min_j v_j(t).$$

В этих обозначениях можно переписать (2.1.3) следующим образом:

$$\min \sum_i A_{i,j} x_i \leq \max \sum_j A_{i,j} y_j \quad (2.1.4)$$

если только $x_i \geq 0$, $\sum_i x_i = 1$, $y_j \geq 0$, $\sum_j y_j = 1$.

Определение. Система (U, V) , состоящая из последовательности n -мерных векторов $U(0), U(1), \dots$ и последовательности m -мерных векторов $V(0), V(1), \dots$, называется *векторной системой для A* , если

$$\min U(0) = \max V(0)$$

и

$$U(t+1) = U(t) + A_{i,*}, \quad V(t+1) = V(t) + A_{*,j},$$

где i и j удовлетворяют условиям

$$v_i(t) = \max V(t), \quad u_j(t) = \min U(t)$$

Таким образом, векторная система для A может быть построена рекурсивно по данным $U(0)$ и $V(0)$. На каждом шаге построения строка, прибавляемая к U , определяется максимальной компонентой V , а столбец, прибавляемый к V , определяется минимальной компонентой U .

Альтернативное понятие векторной системы получается, если в определении 1 заменить налагаемое на j условие следующим:

$$u_j(t+1) = \min U(t+1),$$

Векторная система нового типа также может быть определена рекурсивно. Отличие от предыдущего состоит здесь в том, что последовательные U и V определяются поочередно, тогда, как при первом определении U и V могут быть определены одновременно. Далее под векторной системой понимается векторная система в любом из двух приведенных смыслов.

В частном случае, когда $U(0) = 0$ и $V(0) = 0$, мы видим, что $\frac{U(t)}{t}$ является средним взвешенным строк, а $\frac{V(t)}{t}$ средним взвешенным столбцов. Следовательно, для каждого t и t' .

$$\frac{\min U(t)}{t} \leq v \leq \frac{\max V(t)}{t}$$

Если для некоторых t и t' эти две границы равны, то мы имеем решение игры. К сожалению, это не всегда имеет место. Однако Дж. Браун высказал предположение, что при $t, t' \rightarrow \infty$ эти две границы приближаются к v . В этой части приведено доказательство этого факта для произвольной векторной системы. В численных примерах

оказывается, что векторные системы второго типа дают более быструю сходимость, чем системы первого типа. В новой формулировке:

Теорема 2.1.1. Если (U, V) — векторная система для A , то

$$\lim_{t \rightarrow \infty} \frac{\min U(t)}{t} = \lim_{t \rightarrow \infty} \frac{\max V(t)}{t} = v$$

Доказательство теоремы состоит из четырех лемм.

Лемма 1. Если (U, V) — векторная система для матрицы A , то

$$\liminf_{t \rightarrow \infty} \frac{\max V(t) - \min U(t)}{t} \geq 0$$

Доказательство. Для всякого t

$$V(t) = V(0) + t \sum_j y_j A_{j*}$$

где $y_j \geq 0$, $\sum_j y_j = 1$, и

$$U(t) = U(0) + t \sum_i x_i A_{i*},$$

где $x_i \geq 0$, $\sum_i x_i = 1$.

Следовательно,

$$\max V(t) \geq \min V(0) + t \max \sum_j y_j A_{j*} \geq \min V(0) + tv$$

и

$$\min U(t) \leq \max U(0) + t \min \sum_i x_i A_{i*} \leq \max U(0) + tv$$

Поэтому

$$\liminf_{t \rightarrow \infty} \frac{\max V(t) - \min U(t)}{t} \geq \liminf_{t \rightarrow \infty} \frac{\min V(0) - \max U(0)}{t} = 0$$

Определение. Если (U, V) — векторная система для A , то мы говорим, что i -ая строка *существенна в интервале* (t, t') , если существует такое t_1 , что $t_1 \in (t, t')$ и $v_i(t_1) = \max V(t_1)$.

Аналогично j -ый столбец называется существенным в интервале (t, t') , если существует t_2 для которого $t \leq t_2 \leq t'$ и $u_j(t_2) = \min U(t_2)$.

Лемма 2. Если (U, V) — векторная система для A , то из существенности в интервале $(s, s+t)$ всех строк и столбцов A следует

$$\max U(s+t) - \min U(s+t) \leq 2at,$$

$$\max V(s+t) - \min V(s+t) \leq 2at,$$

где $a = \max_{i,j} |a_{ij}|$.

Доказательство. Пусть j таково, что $u_j(s+t) = \max U(s+t)$. Выберем $t' \in [s, s+t]$ так, чтобы $u_j(t') = \min U(t')$. Тогда

$$u_j(s+t) \leq u_j(t') + at = \min U(t') + at$$

так как изменение i -й компоненты за t шагов не превосходит at . Но

$$\min U(s+t) \geq \min U(t') - at, \text{ так что}$$

$$\max U(s+t) - \min U(s+t) \leq 2at,$$

Аналогично

$$\max V(s+t) - \min V(s+t) \leq 2at,$$

Лемма 3.- Если все строки и столбцы A существенны в $(s, s+t)$ для некоторой системы (U, V) , то

$$\max V(s+t) - \min U(s+t) \leq 4at,$$

Доказательство. По лемме 2

$$\max V(s+t) - \min U(s+t) \leq 4at + \min V(s+t) - \max U(s+t).$$

Следовательно, достаточно показать, что

$$\min V(s+t) \leq \max U(s+t).$$

Применяя (1.2) к транспонированной с A матрице, мы имеем

$$\min \sum_j A_{\cdot j} y_j \leq \max \sum_i A_{i \cdot} x_i$$

при любых $x_i, y_j \geq 0$, $\sum_i x_i = \sum_j y_j = 1$. В частности, это неравенство будет справедливо и для

тех x_i и y_j для которых

$$U(s+t) = U(0) + (s+t) \sum_i A_{i \cdot} x_i,$$

$$V(s+t) = V(0) + (s+t) \sum_j A_{\cdot j} y_j.$$

Тогда

$$\begin{aligned} \min V(s+t) &\leq \max V(0) + (s+t) \min \sum_j A_{\cdot j} y_j \leq \\ &\leq \min U(0) + (s+t) \max \sum_i A_{i \cdot} x_i \leq \max U(s+t) \end{aligned}$$

Л е м м а 4. Для всякой матрицы A и произвольного $\varepsilon > 0$ существует такое t_0 , что для всякой векторной системы (U, V)

$$\max V(t) - \min U(t) < \varepsilon t$$

при $t \geq t_0$.

Доказательство. Утверждение верно для матриц порядка 1, так как тогда $U(t) = V(t)$ для всех t . Примем, что теорема верна для всех подматриц A , и докажем, что она верна и для A .

Выберем t^* так, чтобы для произвольной векторной системы (U', V') , соответствующей подматрице A' матрицы A , мы имели

$$\max V'(t) - \min U'(t) < \frac{1}{2} \varepsilon t$$

при $t \geq t^*$. Докажем, что если в данной векторной системе (U, V) для A некоторый столбец (или строка) несуществен в интервале $(s, s + t^*)$, то

$$\max V(s + t^*) - \min U(s + t^*) < \max V(s) - \min U(s) + \frac{1}{2} \varepsilon t^*. \quad (2.1.5)$$

Предположим, например, что k -я строка несущественна в интервале $(s, s + t^*)$. Тогда мы можем построить для A' векторную систему (U', V') (A' получается из A выбрасыванием k -й строки) следующим образом:

$$\begin{aligned} U'(t) &= U(s + t) + C, \\ V'(t) &= \Pr_k V(s + t), \\ t &= 0, 1, \dots, t^* \end{aligned}$$

где C является n -мерным вектором, все компоненты которого равны $\max V(s) - \min U(s)$, а $\Pr_k V$ — вектор, получаемый из V опусканием k -й его компоненты. Строки A' нумеруются числами $1, 2, \dots, k-1, k+1, \dots, m$. Отметим сначала, что

$$\min U'(0) = \max V'(0).$$

Далее, если

$$U(s + t + 1) = U(s + t) + A_{i*}, \quad V(s + t + 1) = V(s + t) + A_{j*}$$

то

$$U'(t + 1) = U'(t) + A'_{i*}, \quad V'(t + 1) = V'(t) + A'_{j*}$$

Таким образом,

$$v_i(s+t) = \max V(s+t)$$

тогда и только тогда, когда

$$v'_i(t) = \max V'(t),$$

а

$$u_j(s+t) = \min U(s+t)$$

тогда и только тогда, когда

$$u'_j(t) = \min U'(t)$$

при $0 \leq t \leq t^*$. Значит, U' и V' должны удовлетворять рекурсивным условиям, приведенным в определении векторной системы для $0 \leq t \leq t^*$, так как им удовлетворяют U и V .

Естественно, мы можем продолжать U' и V' бесконечно для получения векторной системы для A' .

На основании выбора t^*

$$\max V'(t^*) - \min U'(t^*) < \frac{1}{2} \varepsilon t^*.$$

Поэтому

$$\begin{aligned} \max V(s+t^*) - \min U(s+t^*) &= \\ &= \max V'(t^*) - \min U'(t^*) + \max V(s) - \min U(s) < \\ &< \max V(s) - \min U(s) + \frac{1}{2} \varepsilon t^* \end{aligned}$$

Мы можем теперь показать, что для произвольной векторной системы (U, V) для A

$$\max V(t) - \min U(t) < \varepsilon t$$

при $t > \frac{8at^*}{\varepsilon}$. Рассмотрим $t > t^*$. Пусть $0 \leq \theta \leq 1$ и q таково, что $t = (\theta + q)t^*$.

Случай 1. Пусть существует такое целое положительное число $s \leq q$, что все строки и столбцы A в интервале $((\theta + s - 1)t^*, (\theta + s)t^*)$ существенны. Беря наибольшее из этих s , имеем

$$\max V(t) - \min U(t) \leq \max V((\theta + s)t^*) - \min U(((\theta + s)t^*)) + \frac{1}{2} \varepsilon (q - s)t^* \quad (2.1.6)$$

Мы получаем это неравенство итерацией (2.1.5), так как в каждом из интервалов

$((\theta + s - 1)t^*, (\theta + s)t^*)$ для $r = s + 1, \dots, q$ некоторая строка или столбец A несущественны.

Из леммы 3 на основании выбора s имеем

$$\max V((\theta + s)t^*) - \min U(((\theta + s)t^*)) \leq 4at^* \quad (2.1.7)$$

Из (2.1.6) и (2.1.7) мы получаем

$$\max V(t) - \min U(t) \leq 4at^* + \frac{1}{2}\varepsilon(q-s)t^* < \left(4a + \frac{1}{2}\varepsilon q\right)t^*$$

Случай 2. Если такого s не существует, то в каждом интервале $((\theta+s-1)t^*, (\theta+s)t^*)$ некоторая строка или столбец несущественны. Следовательно,

$$\max V(t) - \min U(t) < \max V(\theta t^*) - \min U(\theta t^*) + \frac{1}{2}\varepsilon qt^* \leq 2a\theta t^* + \frac{1}{2}\varepsilon qt^*$$

Поэтому во всяком случае

$$\begin{aligned} \max V(t) - \min U(t) &< \left(4a + \frac{1}{2}\varepsilon q\right)t^* \leq 4at^* + \frac{1}{2}\varepsilon t < \varepsilon t \\ t &\geq \frac{8at^*}{\varepsilon}. \end{aligned}$$

Из лемм 1 и 4 мы видим, что

$$\lim_{t \rightarrow \infty} \frac{\max V(t) - \min U(t)}{t} = 0$$

но на основании (2.1.4)

$$\limsup_{t \rightarrow \infty} \frac{\min U(t)}{t} \leq v, \quad \liminf_{t \rightarrow \infty} \frac{\max V(t)}{t} \geq v$$

Следовательно,

$$\lim_{t \rightarrow \infty} \frac{\min U(t)}{t} = \lim_{t \rightarrow \infty} \frac{\max V(t)}{t}$$

что и завершает доказательство теоремы о сходимости метода Брауна.

Оценка снизу скорости сходимости метода Брауна приведена в [11]. Эта оценка гарантирует лишь весьма медленную сходимость итеративного процесса, для t -го шага процесса оценка погрешности порядка $1/\sqrt{t}$.

2.2 МОДИФИКАЦИЯ АМВРОСЕНКО МЕТОДА БРАУНА-РОБИНСОН

В этом параграфе рассматривается модификация метода Брауна-Робинсон существенно ускоряющая итеративный процесс. Матричная игра, Γ определяется матрицей вида

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix} = \{a_{ij}\}$$

Решением матричной игры Γ называется такая пара смешанных стратегий

$$x = (x_1, \dots, x_m), x_i \geq 0, \sum_i x_i = 1, y = (y_1, \dots, y_n), y_j \geq 0, \sum_j y_j = 1, \text{ и действительное число } v,$$

что

$$\sum_{i=1}^m x_i a_{ij} \geq v \quad (j = 1, \dots, n)$$

и

$$\sum_{j=1}^n y_j a_{ij} \leq v \quad (i = 1, \dots, m)$$

В этом параграфе изложены приемы (взятые из статьи [12]) позволяющие сократить число итерации для достижения заданной точности решения.

Предварительно введем некоторые обозначения.

Обозначим сумму строк матрицы на i -й итерации, получаемую при использовании метода Брауна для решения матрицы игры $A(m \times n)$.

$$\{\underline{s}^{(i)}\} = (\underline{s}_1^{(i)}, \underline{s}_2^{(i)}, \dots, \underline{s}_n^{(i)}),$$

сумму столбцов

$$\{\bar{s}^{(i)}\} = (\bar{s}_1^{(i)}, \bar{s}_2^{(i)}, \dots, \bar{s}_m^{(i)}),$$

частоты применения первым игроком своих чистых стратегий

$$\{p_k^{(i)}\} = (p_1^{(i)}, p_2^{(i)}, \dots, p_m^{(i)})$$

вторым игроком

$$\{q_k^{(i)}\} = (q_1^{(i)}, q_2^{(i)}, \dots, q_n^{(i)})$$

т. е. количество применений первым игроком k -й стратегии до i -ой итерации включительно равно $i \cdot p_k^{(i)}$, вторым игроком — $i \cdot q_k^{(i)}$. Обозначим также

$$\frac{\underline{s}_k^{(i)}}{i} = u_k^{(i)}, \quad \frac{\bar{s}_k^{(i)}}{i} = v_k^{(i)}$$

Таким образом, нижняя цена игры на i -й итерации равна минимальному элементу строки $\{u_k^{(i)}\} = (u_1^{(i)}, u_2^{(i)}, \dots, u_n^{(i)})$, а верхняя — максимальному элементу строки

$$\{v_k^{(i)}\} = (v_1^{(i)}, v_2^{(i)}, \dots, v_m^{(i)}), \underline{v}^{(i)} = \min_k \{u_k^{(i)}\}, \bar{v}^{(i)} = \max_k \{v_k^{(i)}\}.$$

Обычно итерационный процесс заканчивают на первой же итерации t , для которой

$$\bar{v}^{(t)} - \underline{v}^{(t)} \leq \varepsilon \tag{2.2.1}$$

где ε — заранее заданное число.

Рассмотрим первый, довольно тривиальный прием сокращения числа итераций, заключающийся в том, что процесс решения заканчивается на первой же итерации, для которой выполняется неравенство

$$\min_{s,k} (\bar{v}^{(s)} - \underline{v}^{(k)}) \leq \varepsilon \quad (2.2.2)$$

для $k, s = 1, 2, \dots, t$

При реализации этого приема необходимо на каждой $(i+1)$ -й итерации иметь максимальное значение $\underline{v}^{[i]}$ нижних сумм $\underline{v}^{(i)}$ за все предыдущие i итераций и соответствующий ей набор $\{p_k^{[i]}\}$ и минимальное значение, $\bar{v}^{[i]}$ и соответствующий набор $\{q_k^{[i]}\}$.

Получаемое на $(i+1)$ -й итерации значение $\underline{v}^{(i+1)}$ сравнивать с $\underline{v}^{[i]}$

$$\underline{v}^{(i+1)} > \underline{v}^{[i]} \quad (2.2.3)$$

и при выполнении неравенства (2.2.3) положить

$$\underline{v}^{[i+1]} = \underline{v}^{(i+1)}, \{p_k^{[i+1]}\} = \{p_k^{(i+1)}\}$$

а при невыполнении неравенства (2.2.3) оставить

$$\underline{v}^{[i+1]} = \underline{v}^{[i]}, \{p_k^{[i+1]}\} = \{p_k^{[i]}\}$$

Аналогично поступать с $\bar{v}^{[i]}, \{q_k^{[i]}\}, \bar{v}^{(i+1)}, \{q_k^{(i+1)}\}$ при выполнении неравенства $\bar{v}^{(i+1)} < \bar{v}^{[i]}$.

Предлагаемый прием не будет удлинять итерационный процесс, так как (2.2.1) является частным случаем (2.2.2), а будет, как правило, его сокращать.

Более существенное влияние на сокращение длительности итерационного процесса решения игровой матрицы оказывает следующий, прием.

В обычном итерационном процессе часто оба игрока одновременно применяют некоторое число l ($l > 1$) раз подряд одни и те же стратегии, т. е. l раз подряд к строке $\{\underline{s}_k^{(i)}\}$ прибавляется некоторая i' -я строка матрицы A (сохраняется номер i' максимального элемента в строках $\{\underline{s}_k^{(i)}\}, \{\underline{s}_k^{(i+1)}\}, \dots, \{\underline{s}_k^{(i+l-1)}\}$ и к строке $\{\bar{s}_k^{(i)}\}$ — некоторый j' -й столбец исходной матрицы A (сохраняется номер j' минимального элемента в строках $\{\bar{s}_k^{(i)}\}, \{\bar{s}_k^{(i+1)}\}, \dots, \{\bar{s}_k^{(i+l-1)}\}$.

Предлагается в каждой итерации определять число l одновременных повторений игроками одних и тех же ходов и прибавлять к строке $\{\underline{s}_k^{(i)}\}$ умноженную на l соответствующую i' -ую строку и к столбцу $\{\bar{s}_k^{(i)}\}$ — умноженный на l соответствующий j' -й столбец матрицы A и соответствующим образом изменять $\{p_k^{(i)}\}$ и $\{q_k^{(i)}\}$ (с учетом l -разового применения

стратегий). Тогда в каждой итерации хотя бы один из игроков будет применять стратегию, отличную от применяемой в предыдущей итерации.

Алгоритм указанного приема следующий.

Выходными данными к $(i + 1)$ -й итерации являются $\{\underline{s}_k^{(i)}\}$, $\{\bar{s}_k^{(i)}\}$, $\{\alpha_k^{(i)}\}$, $\{\beta_k^{(i)}\}$ и исходная игровая матрица $A = \{a_{ij}\}$. Совокупности $\{\alpha_k^{(i)}\}$ и $\{\beta_k^{(i)}\}$ определяют смешанные стратегии первого и второго игроков за i итераций:

$$p_k^{(i)} = \alpha_k^{(i)} / \sum_{k=1}^m \alpha_k^{(i)}, \quad q_k^{(i)} = \beta_k^{(i)} / \sum_{k=1}^n \beta_k^{(i)}$$

На $(i + l)$ -й итерации определяются

$$u_k^{(i+1)} = \underline{v}^{(i+1)} = \min_k \{u_k^{(i)}\} = \frac{1}{c_i} \min_k \{\underline{s}_k^{(i)}\}$$

$$v_{k''}^{(i+1)} = \bar{v}^{(i+1)} = \max_k \{v_k^{(i)}\} = \frac{1}{c_i} \max_k \{\bar{s}_k^{(i)}\}$$

где c_i - то сколько раз применили одну и туже стратегию первый и второй игроки.

Последовательно определяются $n_j''(i+1)$ и $n_j'(i+1)$

$$n_j''(i+1) = \begin{cases} \left\lfloor \frac{\bar{s}_{k''}^{(i)} - \underline{s}_j^{(i)}}{a_{j,k'} - a_{k'',k'}} + 1 \right\rfloor & \text{для } a_{j,k'} > a_{k'',k'} \\ \infty & \text{для } a_{j,k'} \leq a_{k'',k'} \end{cases}$$

$$n_j'(i+1) = \begin{cases} \left\lfloor \frac{\underline{s}_j^{(i)} - \underline{s}_{k'}^{(i)}}{a_{k'',k'} - a_{k'',j}} + 1 \right\rfloor & \text{для } a_{k'',k'} > a_{k'',j} \\ \infty & \text{для } a_{k'',k'} \leq a_{k'',j} \end{cases}$$

Затем выбирается

$$n(i+1) = \min [n_j'(i+1), n_j''(i+1)]$$

где

$$n'(i+1) = \min_j \{n_j'(i+1)\}, \quad n''(i+1) = \min_j \{n_j''(i+1)\}$$

Формируются

$$\begin{aligned}
\underline{s}_j^{(i+1)} &= \underline{s}_j^{(i)} + n(i+1) \cdot a_{k^*, j} \quad (j = 1, \dots, n) \\
\bar{s}_j^{(i+1)} &= \bar{s}_j^{(i)} + n(i+1) \cdot a_{j, k^*} \quad (j = 1, \dots, m) \\
\alpha_k^{(i+1)} &= \alpha_k^{(i)} \quad \text{для } k \neq k^* \\
\alpha_{k^*}^{(i+1)} &= \alpha_{k^*}^{(i)} + n(i+1) \\
\beta_k^{(i+1)} &= \beta_k^{(i)} \quad \text{для } k \neq k^* \\
\beta_{k^*}^{(i+1)} &= \beta_{k^*}^{(i)} + n(i+1)
\end{aligned}$$

Определяются нижняя и верхняя цены игры:

$$\begin{aligned}
\underline{v}^{(i+1)} &= \frac{\min_j \{\underline{s}_j^{(i+1)}\}}{\sum_{k=1}^m \alpha_k^{(i+1)}}, \quad \bar{v}^{(i+1)} = \frac{\max_j \{\bar{s}_j^{(i+1)}\}}{\sum_{k=1}^n \beta_k^{(i+1)}}
\end{aligned}$$

Затем проверяется неравенство (2.2.1), и в случае его невыполнения переходят к следующей, $(i+2)$ -й итерации.

2.3 МОДИФИКАЦИИ БОРИСОВОЙ, МАГАРИК МЕТОДА БРАУНА-РОБИНСОН

В рассматриваемых методах каждый из игроков на одной итерации может изменить только одну чистую стратегию, т. е. направлением изменения смешанной стратегии каждого игрока на одной итерации является единичный вектор.

Пусть матричная игра двух лиц с нулевой суммой описывается матрицей A :

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix}$$

строки матрицы A обозначим через A_i ($i = 1, 2, \dots, m$), а столбцы через A_j ($j = 1, 2, \dots, n$).

Напомним, чистыми стратегиями первого игрока являются выборы строк A , чистыми стратегиями 2-го — выборы столбцов матрицы A . Элемент матрицы a_{ij} является выигрышем 1-го игрока у 2-го, если 1-й выбирает i -ю чистую стратегию, а 2-й — j -ю.

Смешанные стратегии 1-го и 2-го игроков через x и y :

$$x = (x_1, \dots, x_m), \quad \sum_i x_i = 1, \quad x_i \geq 0,$$

$$y = (y_1, \dots, y_n), \quad y_j \geq 0, \quad \sum_j y_j = 1,$$

а соответствующие им нижнюю и верхнюю оценки игры через U и V :

$$U = \min_j (xA)_j = \min_j (u_1, u_2, \dots, u_n)$$

$$V = \max_i (Ay)_i = \max(v_1, v_2, \dots, v_m)^T$$

Обозначим

$$J = \{j : u_j = U\},$$

$$I = \{i : v_i = V\}.$$

Оптимальным ответом (или наилучшей чистой стратегией) 1-го игрока на стратегию u является i -я чистая стратегия ($i \in I$), а 2-го игрока, на стратегию x — j -я чистая стратегия ($j \in J$).

Итеративный метод Брауна с последовательным выбором чистых стратегий 1-м и 2-м игроками заключается в выборе игроком на каждой итерации наилучшей чистой стратегии против накопленной смешанной стратегии противника (шаги игроки производят поочередно); смешанная стратегия игрока на $(k+1)$ -й итерации составляется из его смешанной стратегии x^k (y^k) на предыдущей k -й итерации, взятой с весом $k/(k+1)$, и выбранной чистой стратегии, взятой с весом $1/(k+1)$. Таким образом, переход от k -й к $(k+1)$ -й итерации осуществляется, но формулам

$$x^{k+1} = \frac{k}{k+1} x^k + \frac{1}{k+1} e_{i_k}, \quad y^{k+1} = \frac{k}{k+1} y^k + \frac{1}{k+1} e_{j_{k+1}}$$

где e_{i_k} — единичный m -мерный вектор; $e_{j_{k+1}}$ — единичный n -мерный вектор; индексы i_k и j_{k+1} удовлетворяют условиям

$$v_{i_k} = \max_i (Ay^k)_i, \tag{2.3.1}$$

$$u_{j_{k+1}} = \min_j (x^{k+1} A)_j. \tag{2.3.2}$$

В нем изменение нижней и верхней оценок игры на каждой итерации не является, вообще говоря, монотонным и, как показывает опыт решения численных примеров, носит колебательный характер.

В модификации Амвросенко одна итерация соответствует нескольким «склеенным» итерациям метода Брауна с одновременным выбором чистых стратегий 1-м и 2-м игроками, изменение нижней и верхней оценок игры на каждой итерации носит также колебательный характер.

В рассматриваемых ниже двух модификациях алгоритма Брауна смешанная стратегия каждого игрока на $(k+1)$ -й итерации составляется так же, как и в алгоритме В, из его смешанной стратегии на предыдущей k -й итерации и выбранной чистой стратегии, по веса, с

которыми берутся эти стратегии, определяются из условий возрастания нижней оценки игры U и убывания верхней оценки игры V .

Рассмотрим подробнее вопрос о величине шага (веса) по выбранной наилучшей чистой стратегии. Пусть в итеративном процессе переход от точки x к следующей точке $x(\lambda)$ осуществляется по формуле:

$$x(\lambda) = (1 - \lambda)x + \lambda e_i, \quad 0 \leq \lambda \leq 1, \quad i \in I \quad (2.3.3)$$

Обозначим нижнюю оценку игры в точке $x(\lambda)$ через $U(\lambda)$:

$$U(\lambda) = \min_{1 \leq p \leq n} [(1 - \lambda)u_p + \lambda a_{ip}]. \quad (2.3.4)$$

Покажем, что, зная наперед только точку x , направление движения e_i , и оптимальные ответы 2-го игрока на x -стратегию 1-го игрока, можно сказать заранее, найдется ли при движении в направлении e_i точке с большим значением нижней оценки игры, чем U . Вычислим производную кусочно-линейной функции $U(\lambda)$ в точке $\lambda = 0$:

$$\frac{dU(\lambda)}{\lambda} \Big|_{\lambda=0} = \min_{j \in J} (-u_j + a_{ij}) = -U + \min_{j \in J} a_{ij}$$

Таким образом, если

$$a_{ij} > U \text{ для всех } j \in J = \{j : u_j = U\},$$

то в окрестности $\lambda = 0$ найдется такое $\lambda > 0$, что $U(\lambda) > U(0) = U$. Рассмотрим геометрический смысл функции $U(\lambda)$. Отложим значения координат вектора xA на оси

ординат, а на оси абсцисс — значения λ (рис. 1).

Жирной линией изображена кусочно-линейная функция $U(\lambda)$, заданная формулой (2.3.4) с положительной производной в нуле.

На рис. 1 через $\bar{\lambda}$ обозначена абсцисса вершины ломаной $U(\lambda)$ с максимальным значением $U(\bar{\lambda})$ на отрезке $[0, 1]$:

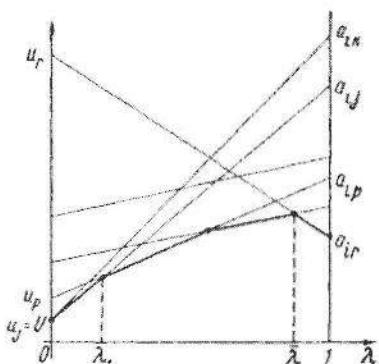


Рис.1

$$U(\bar{\lambda}) = \max_{\lambda} U(\lambda) = \max_{\lambda} \min_p [(1 - \lambda)u_p + \lambda a_{ip}] \quad (2.3.5)$$

Очевидно, что при осуществлении итеративного процесса по формуле (2.3.4) максимальное увеличение нижней оценки игры U на одной итерации произойдет при выборе $\lambda = \bar{\lambda}$.

Через λ_1 на рисунке обозначена абсцисса вершины ломаной $U(\lambda)$, ближайшей к вершине $(0, U)$:

$$\lambda_1 = \min_{\substack{p \\ a_{ip} < a_{ij}}} \frac{u_p - u_j}{u_p - u_j + a_{ij} - a_{ip}}, \text{ где } a_{ij} = \min_{k \in J} a_{ik}. \quad (2.3.6)$$

По аналогии для 2-го игрока, если переход от точки y к $y(\mu)$ осуществляется по формуле:

$$y(\mu) = (1 - \mu)y + \mu e_j, \quad 0 \leq \mu \leq 1, \quad j \in J \quad (2.3.7)$$

то условие убывания функции $V(\mu)$ в точке $\mu = 0$ имеет вид

$$a_{ij} < V \text{ для всех } i \in I = \{i : v_i = V\}.$$

Минимальное значение $V(\mu)$ на отрезке $[0, 1]$ достигается в точке $\mu = \bar{\mu}$:

$$V(\bar{\mu}) = \min_{\mu} \max_q [(1 - \mu)v_q + \mu a_{qj}].$$

Абсцисса первой ближайшей вершины ломаной $V(\mu)$ к вершине $(0, V)$ определяется следующим образом:

$$\mu_1 = \min_{\substack{q \\ a_{qj} > a_{ij}}} \frac{v_j - v_q}{v_j - v_q + a_{qj} - a_{ij}}, \text{ где } a_{ij} = \max_{l \in I} a_{lj} \quad (2.3.8)$$

Рассмотрим два итеративных алгоритма, в которых переход от предыдущих смешанных стратегий игроков к последующим осуществляете по формулам типа (2.3.3) и (2.3.7), причем движение в направлении выбранных чистых стратегий e_i и e_j осуществляется по возможности с увеличением нижней оценки игры и уменьшением верхней оценки игры. Другими словами, 1-й игрок на каждой итерации старается сделать такой шаг в направлении e_i , который бы увеличил его минимальный выигрыш U на предыдущей итерации (если это возможно), а 2-й старается сделать шаг в направлении e_j уменьшающий его максимальный проигрыш V на предыдущей итерации. В первом алгоритме (назовем его алгоритмом B1) движение в направлении выбранных чистых стратегий осуществляется до точек $\bar{\lambda}$ и $\bar{\mu}$. Во втором алгоритме (назовем его алгоритмом B2) движение осуществляется до точек λ_1 и μ_1 . В обоих алгоритмах игроки выполняют шаги поочередно.

В алгоритме B1 итеративный процесс осуществляется по формулам:

$$x^{k+1} = (1 - \lambda^k)x^k + \lambda^k e_{i_k}, \quad (2.3.9)$$

$$y^{k+1} = (1 - \mu^k)y^k + \mu^k e_{j_{k+1}}, \quad (2.3.10)$$

где индексы i_k и j_{k+1} удовлетворяют условиям (2.3.1) и (2.3.2):

(причем в случае неоднозначности выбора i_k или j_{k+1} выбирается компонента соответствующую вектора с минимальным номером), а

$$\lambda^k = \begin{cases} a) & \bar{\lambda}^k, \text{ если } \min_{j_k \in J} a_{i_k j_k} > U, J = \{j_k : u_{j_k} = U\} \\ & u \quad \bar{\lambda}^k \geq \frac{1}{s(k)+1} \\ b) & 0 \quad \text{или} \\ c) & \frac{1}{s(k)+1} \end{cases} \quad \text{в противном случае} \quad (2.3.11)$$

$$\mu^k = \begin{cases} a) & \bar{\mu}^k, \text{ если } \min_{i_k \in I} a_{i_k j_{k+1}} < V, I = \{i_k : v_{i_k} = V\} \\ & \bar{\mu}^k \geq \frac{1}{r(k)+1} \\ b) & 0 \quad \text{или} \\ c) & \frac{1}{r(k)+1} \end{cases} \quad \text{в противном случае} \quad (2.3.12)$$

В формуле (2.3.11.a) через $\bar{\lambda}^k$ обозначена абсцисса вершины ломаной $U(\lambda^k)$, определяемая формулой (2.3.5), т.е. на $(k + 1)$ -й итерации 1-й игрок делает шаг с максимальным увеличением нижней оценки игры в направлении e_{i_k} , если производная

$\left. \frac{dU(\lambda^k)}{\lambda^k} \right|_{\lambda^k=+0} > 0$ и величина шага $\bar{\lambda}^k$ не меньше $\frac{1}{s(k)+1}$. Для 2-го игрока формулы (2.3.10)

и (2.3.12.a) означают переход из y^k в направлении $e_{j_{k+1}}$ в точку y^{k+1} с минимальным

значением верхней оценки игры, если $\left. \frac{dV(\mu^k)}{\mu^k} \right|_{\mu^k=+0} < 0$ и $\bar{\mu}^k \geq \frac{1}{r(k)+1}$. Если на данной

$(k+1)$ -й итерации только один игрок может улучшить значение U -нижней оценки или значение V — верхней оценки игры, то другой оставляет свою смешанную стратегию неизменной, т. е. в итеративном процессе в этом случае участвуют формулы (2.3.11.a) и (2.3.12.6) или (2.3.12.a) и (2.3.11.6).

Шаги с весами λ^k и μ^k , вычисляемыми по формулам (2.3.11.b) и (2.3.12.b), назовем вынужденными. В алгоритме B1 выполнение вынужденных шагов означает уменьшение нижней оценки игры U или увеличение верхней оценки игры V . Эти шаги выполняются игроками поочередно (одним на игроков на $(k+1)$ -й итерации) и только в том случае, если ни один из игроков не может улучшить U или V , т. е. применить формулу (2.3.11.a) или (2.3.12.a). В итеративном процессе в этом случае участвуют формулы (2.3.11.6) и (2.3.12.b) или (2.3.11.b) и (2.3.12.6). Нарушение монотонности изменения U или V происходит только при применении формулы (2.3.11.b) или (2.3.12.b).

Величины $s(k+1)$ и $r(k+1)$ определяются следующим образом:

$$s(k+1) = \begin{cases} \frac{s(k)}{1-\lambda^k}, & \text{если } \lambda^k = \lambda_1^k, \\ s(k), & \text{если } \lambda = 0, \\ s(k)+1, & \text{если } \lambda^k = \frac{1}{s(k)+1}, \end{cases} \quad (2.3.13)$$

$$r(k+1) = \begin{cases} \frac{r(k)}{1-\mu^k}, & \text{если } \mu^k = \bar{\mu}^k, \\ r(k), & \text{если } \mu^k = 0, \\ r(k)+1, & \text{если } \mu^k = \frac{1}{r(k)+1}. \end{cases} \quad (2.3.14)$$

В случае $\lambda_k = 1$, $s(k+1)$ задается формулой: $s(k+1) = 1$, т. е. на $(k+1)$ -й итерации 1-й игрок заменяет накопленную смешанную стратегию x^k чистой стратегией e_{ik} . Аналогично для $\mu^k = 1$ задаем $r(k+1) = 1$.

Эти величины являются в некотором смысле аналогами числа «склеенных» шагов для каждого игрока в модификации Амвросенко алгоритма Брауна с последовательным выбором чистых стратегий.

В алгоритме B2 итеративный процесс осуществляется так же, как и в B1, по формулам (2.3.9), (2.3.10), (2.3.1), (2.3.2), но веса λ^k и μ^k вычисляются следующим образом:

$$\lambda^k = \begin{cases} a) \quad \lambda_1^k, \text{ если } \min_{j_k \in J} a_{i_k j_k} > U, J = \{j_k : u_{j_k} = U\} \text{ и } \lambda_1^k \geq \frac{1}{s(k)+1}, \\ b) \quad \frac{1}{s(k)+1} \text{ в противном случае,} \end{cases}$$

$$\eta^k = \begin{cases} a) \quad \mu_1^k, \text{ если } \max_{i_k \in I} a_{i_k j_{k+1}} < V, I = \{i_k : v_{i_k} = V\} \text{ и } \mu_1^k \geq \frac{1}{r(k)+1}, \\ b) \quad \frac{1}{r(k)+1} \text{ в противном случае.} \end{cases}$$

Здесь через λ_1^k обозначена абсцисса вершины ломаной $U(\lambda^k)$, ближайшей к вершине $(0, U)$ (см. рис. 1), λ_1^k определяется формулой (2.3.6). Аналогично для 2-го игрока μ_1^k определяется формулой (2.3.8).

В алгоритме В2 на $(k+1)$ -й итерации 1-й игрок делает шаг (с весом λ_1^k), увеличивающий значение нижней оценки игры U на k -й итерации, если это возможно, или выполняет вынужденный шаг с весом $\frac{1}{s(k)+1}$. Точно так же поступает 2-й игрок. Величины $s(k+1)$ и $r(k+1)$ определяются формулами:

$$s(k+1) = \begin{cases} \frac{s(k)}{1-\lambda_1^k}, \text{ если } \lambda^k = \lambda_1^k \\ s(k)+1 \text{ в противном случае,} \end{cases}$$

$$r(k+1) = \begin{cases} \frac{r(k)}{1-\mu_1^k}, \text{ если } \mu^k = \mu_1^k, \\ r(k)+1 \text{ в противном случае.} \end{cases}$$

2.4 МОНОТОННЫЙ ИТЕРАТИВНЫЙ АЛГОРИТМ РЕШЕНИЯ МАТРИЧНЫХ ИГР

В этом параграфе рассмотрим итеративный алгоритм решения матричных игр, принципиально отличающийся от алгоритма Брауна-Робинсон, его модификаций Амвросенко [12] и Борисовой и Магарик [13] и обобщения [15,16].

Особенностью алгоритма является его способность генерировать строго монотонно возрастающую последовательность оценок цены игры, что не свойственно вышеупомянутым алгоритмам.

Введем обозначения. Пусть A_i — i -я строка матрицы A , x^N — m -мерный вектор такой, что

$$x_i^N \geq 0, \sum_{i=1}^m x_i^N = 1,$$

N — номер шага, s^N — n -мерный вектор, определяющий средний накопленный выигрыш на N -м шаге. Зададим произвольные начальные условия: $s^0 = A_{i_0}$, $x^0 = (0, \dots, 0, 1, 0, \dots, 0)$ где единица занимает i_0 -ю позицию.

Определим итеративный алгоритм следующим образом: по известным векторам x^{N-1} и s^{N-1} находим векторы

$$x^N = (1 - \alpha_N) x^{N-1} + \alpha_N \tilde{x}^N \quad (2.4.1)$$

$$s^N = (1 - \alpha_N) s^{N-1} + \alpha_N \tilde{s}^N \quad (2.4.2)$$

где параметр $0 \leq \alpha \leq 1$ определяется из соотношения

$$\max_{\alpha} \min_j [(1 - \alpha) s_j^{N-1} + \alpha \tilde{s}_j^N] = \min_j [(1 - \alpha_N) s_j^{N-1} + \alpha_N \tilde{s}_j^N] \quad (2.4.3)$$

а векторы \tilde{x}^N и \tilde{s}^N вводятся далее. Для этого введем в рассмотрение множество J^{N-1} индексов $j_1^{N-1}, j_2^{N-1}, \dots, j_k^{N-1}$ таких, что

$$\min_j s_j^{N-1} = s_{j_1^{N-1}}^{N-1} = s_{j_2^{N-1}}^{N-1} = \dots = s_{j_k^{N-1}}^{N-1}, \quad k \leq n$$

Рассмотрим подыгру Γ^N игры Γ , матрица выигрышей которой имеет вид

$$A^N = \{a_{j^{N-1}}\}, \quad i = 1, \dots, m, \quad j^{N-1} \in J^{N-1}$$

Находим в игре Γ^N любую из стратегий первого игрока $\tilde{x}^N = (\lambda_1^N, \dots, \lambda_m^N)$, для которой выполняется неравенство

$$\min_j \sum_{i=1}^m \lambda_i^N A_{ji}^N \geq v$$

Заметим, что такая стратегия, например оптимальная в игре Γ^N , всегда существует, ибо в противном случае второй игрок в игре Γ мог бы проиграть меньше, чем v , что противоречит определению цены игры. После того как найдена стратегия \tilde{x}^N , находим

$$\tilde{s}^N = \sum_{i=1}^m \lambda_i^N A_{i*}^N = (\tilde{s}_1^N, \dots, \tilde{s}_n^N)$$

и решаем игру $2 \times n$, в которой у первого игрока две чистые стратегии, а у второго игрока n чистых стратегий; эта игра имеет матрицу выигрышей

$$\begin{pmatrix} s_1^{N-1} & s_2^{N-1} & \dots & s_n^{N-1} \\ \tilde{s}_1^N & \tilde{s}_2^N & \dots & \tilde{s}_n^N \end{pmatrix}$$

Решение этой игры, точнее оптимальная стратегия первого игрока, позволяет найти в соответствии с (2.4.3) весовой коэффициент α_N . Далее, применяя (2.4.1) и (2.4.2), вычисляем s^N , x^N и переходим к следующему шагу. Вычислительный процесс завершается в том случае,

когда-либо $\alpha_N = 0$, либо достигнута некоторая заданная точность оценки цены игры.

Определим

$$\underline{v}^N = \min s_j^N$$

которую назовем нижней оценкой цены игры.

Из теоремы о минимаксе следует, что $\underline{v}^N \leq v$ для любого N , причем равенство имеет место при $\alpha_N = 0$. Справедливы следующие утверждения.

Теорема 1. Для любого шага $N \geq 1$ итеративного алгоритма, определенного условиями (2.4.1) — (2.4.3), выполняется неравенство

$$v^N > \underline{v}^{N-1}.$$

Теорема утверждает, что описанный алгоритм генерирует строго монотонно возрастающую последовательность оценок цены игры $\{\underline{v}^N\}$. Если обозначить через v^* точную верхнюю грань последовательности $\{\underline{v}^N\}$, а через x^* одну из предельных точек последовательности $\{x^N\}$, то верна следующая теорема о сходимости.

Теорема 2. Для любой матричной игры Γ и любых начальных условий s^0 и x^0 справедливы равенства

$$\underline{v}^* = v, \quad x^* = x_0$$

где x_0 — одна из оптимальных стратегий.

Рассмотренный алгоритм практически трудно реализовать, если на каждом шаге придется решать подигру Γ^N размерности $m \times 3$ или большей. Поскольку априорно неизвестно, какие векторы s^{N-1} и \tilde{s}^N появятся на N -м шаге, так как можно говорить об их случайном появлении.

2.5 РЕШЕНИЕ СИММЕТРИЧНЫХ ИГР

Симметричной называется игра с кососимметрической матрицей выигрышей; по существу это означает, что в симметричной игре оба игрока играют одну и ту же роль и располагают одним и тем же множеством чистых стратегий. Ввиду своего специального характера симметричные игры представляют естественный интерес; особое значение им придают вычислительные методы, рассмотренные Дж. У. Брауном. В этом параграфе мы приведем метод симметризации произвольной игры еще (один метод симметризации приведен в [17]). Под симметризацией мы понимаем построение симметричной игры, матрица которой кососимметрическая.

Если мы симметризуем игру для вычислительных целей, то решающее значение имеет, разумеется, размер получающейся матрицы. Здесь мы рассмотрим $(m \times n)$ -матрицу игры $A = (a_{ij})$ и образуем симметричную игру, описываемую матрицей

$$S = \begin{pmatrix} 0 & A & -1 \\ -A^T & 0 & 1 \\ 1 & -1 & 0 \end{pmatrix}$$

где — A^T есть взятая со знаком минус транспозиция матрицы A , ± 1 — векторы, составленные из надлежащего числа единиц (± 1), а нули изображают матрицы, состоящие целиком из нулей, заполняющих остаток матрицы S .

Как известно, прибавлением ко всем элементам матрицы A положительной постоянной можно добиться того, чтобы v — значение игры A — было положительным; множества оптимальных стратегий при этом не меняются. Следовательно, не ограничивая общности, мы можем предположить, что $v > 0$.

Теперь, если $(x_1, \dots, x_m; y_1, \dots, y_n; \lambda)$ — оптимальная смешанная стратегия для игры S , имеем

$$-\sum_j a_{ij} y_j + \lambda \geq 0, \quad (2.5.1)$$

$$\sum_i x_i a_{ij} - \lambda \geq 0 \quad (2.5.2)$$

$$-\sum_i x_i + \sum_j y_j \geq 0. \quad (2.5.3)$$

Пусть $\sum_j y_j = 0$. Тогда в силу (2.5.3) $\sum_i x_i = 0$ и из (2.5.1) следует $\lambda = 0$, что противоречит

соотношению

$$\sum_i x_i + \sum_j y_j + \lambda = 1.$$

Следовательно, $\sum_j y_j = \mu > 0$ и $y^* = \frac{y}{\mu}$ есть смешанная стратегия для игры A .

Пусть $\lambda = 0$. Тогда неравенство (2.5.1) превращается в $\sum_j a_{ij} y_j^* \leq 0$, а это противоречит

нашему предположению о том, что значение игры A положительно. Следовательно, $\lambda > 0$.

Поэтому, умножая каждое неравенство (2.5.1) на x_i , каждое неравенство (2.5.2) на y_j и суммируя, получаем

$$\sum_i x_i \geq \frac{1}{\lambda} \sum_{i,j} x_i a_{ij} y_j \geq \sum_j y_j. \quad (2.5.4)$$

Комбинируя (2.5.3) и (2.5.4), получаем

$$\sum_i x_i = \sum_j y_j = \mu;$$

если теперь положить $x^* = x/\mu$, неравенства (2.5.1) и (2.5.2) принимают такой вид:

$$\sum_j a_{ij} y_j^* \leq \frac{\lambda}{\mu},$$

$$\sum_i x_i^* a_{ij} \geq \frac{\lambda}{\mu}.$$

Таким образом, x^* и y^* — оптимальные стратегии для игры A и $v = \frac{\lambda}{\mu}$.

Опишем итеративный алгоритм для решения симметричной матричной игры (см. [15] стр. 188-196), т. е. игры с платежной матрицей A , удовлетворяющей равенству $A = -A^T$. Выше было показано, что любая задача может быть сведена к такой игре.

Исходя из свойства симметричной матричной игры, что если стратегия x^* оптимальная для первого игрока, она же оптимальна для второго, следует что для нахождения какого-нибудь оптимального решения достаточно найти какого-нибудь стратегию оптимальную для первого игрока.

Задачу нахождения минимаксной стратегии можно записать в форме:

$$Ax \leq \sigma e, \quad e = (1, 1, \dots, 1)$$

$$\sigma \rightarrow \min, \quad x \in S^m = \left\{ x : x \geq 0, \sum_{j=1}^m x_j = 1 \right\}$$

с неизвестными $x = (x_1, x_2, \dots, x_m)$ и скаляром σ . Если обозначить вектор Ax через ξ :

$$\xi = Ax$$

то задача сводится к минимизации функции

$$\sigma(x) = \max_i \xi_i = \max_i (Ax)_i,$$

неотрицательной на симплексе S^m и обращающейся в нуль при $x = x^*$. Итеративный процесс состоит в последовательном уменьшении σ .

Пусть имеется некоторая стратегия $x \in S_n$ и $\xi = Ax$. Предположим, что ξ_i — единственная максимальная компонента вектора ξ . Тогда от стратегии x можно перейти к стратегии

$$x' = (1 - \alpha)x + \alpha e_i, \quad e_i = (0, \dots, \underset{i}{1}, \dots, 0)$$

(отметим, что e_i — оптимальный ответ на стратегию x), которая при малых α будет лучше x в том смысле, что $\sigma(x') < \sigma(x)$ при малых α . Действительно,

$$\xi' = Ax' = (1 - \alpha)\xi + \alpha A^i,$$

где через A^i обозначен i -й столбец матрицы A . Так как $a_{ii} = 0$, то i -я компонента вектора ξ' убывает с ростом α . Но ξ'_i при малых α будет оставаться (единственной по предположению) максимальной компонентой вектора ξ' , т. е.

$$\sigma' = \sigma(x') = \xi'_i = (1 - \alpha)\xi_i = (1 - \alpha)\sigma < \sigma.$$

Ясно, что α можно увеличивать до тех пор, пока ξ'_i остается максимальной компонентой, т. е. пока не наступит равенство

$$\xi'_i = \xi'_k$$

при некотором $k \neq i$ (если ни при каком $0 < \alpha < 1$ это равенство не наступит, то можно взять $\alpha = 1$, тогда $x' = e_i$ — решение игры, поскольку при этом $\sigma' = 0$). Запишем это равенство подробнее

$$(1 - \alpha)\xi_i = (1 - \alpha)\xi_k + \alpha a_{ki}. \quad (2.5.5)$$

Отсюда видно, что конкурентной компонентой будет компонента с индексом

$$k = \arg \min_{j \neq i} \frac{\xi_i - \xi_j}{a_{ji} + \xi_i - \xi_j},$$

при этом величина шага будет равна

$$\alpha = \min_{j \neq i} \frac{\xi_i - \xi_j}{a_{ji} + \xi_i - \xi_j} = \frac{\xi_i - \xi_k}{a_{ki} + \xi_i - \xi_k}. \quad (2.5.6)$$

Отметим, что из равенства (2.5.5) следует, что $a_{ki} > 0$.

Перейдя от стратегии x к стратегии x' , можно повторить описанный процесс с x' вместо x . Надо только уточнить одно обстоятельство. Мы предполагали, что ξ_i — единственная максимальная компонента вектора ξ . Теперь их у нас по крайней мере две: ξ'_i и ξ'_k . Однако в новом состоянии роль i будет играть k , а i отходит на задний план и не может составить конкуренции k , поскольку $a_{ik} = -a_{ki} < 0$.

Эти рассуждения показывают, что возможен процесс последовательных переходов по чистым стратегиям. В r -м переходе величина σ уменьшается в $(1 - \alpha_r)$ раз, следовательно, процесс будет сходиться, если

$$\sum \alpha_r = \infty.$$

Этого, к сожалению, гарантировать нельзя, так как наше первоначальное предположение о единственности максимальной компоненты может не выполняться, или будут возникать компоненты, близкие к максимальной, что приведет к слишком малым шагам. Для борьбы с таким «заеданием» можно делать нестандартный шаг: если при вычислении α по формуле (2.5.6) окажется, что

$$\alpha < \rho\sigma,$$

где ρ — малая ($\approx 0,01$) константа, то величина шага берется равной

$$\alpha = \rho\sigma.$$

При этом, конечно, σ' станет больше σ , но зато процесс выйдет из неопределенного состояния, когда не существует ни одной чистой стратегии e_i , к которой можно было бы двигаться с уменьшением σ . Следующий шаг опять приведет к уменьшению σ и процесс будет продолжаться регулярным образом.

Значительное ускорение процесса дает применение «овражного шага». Он состоит в следующем. В последовательности сменяющих друг друга чистых стратегий e_i выявляется цикл. Пусть x_1 и x_2 — значения текущего вектора x , отвечающие двум последовательным моментам появления какой-либо (любой из повторяющихся) одной и той же чистой стратегии, причем $\sigma_2 < \sigma_1$. Тогда из состояния x_2 делается переход в состояние x_α

$$x_\alpha = x_2 + \alpha(x_2 - x_1)$$

с некоторым шагом α , который определяется как максимально возможный, не выводящий x_α за пределы симплекса S^m .

2.6 СРАВНЕНИЕ ИТЕРАЦИОННЫХ МЕТОДОВ

Эта глава посвящена приближенным методам решения матричных игр.

Общим для всех этих методов является то, что все они строятся на формуле

$$x_{n+1} = \alpha_n x_n + (1 - \alpha_n) p_n, \quad (2.6.1)$$

где x_n — приближенное оптимальное решение на n -ом шаге; p_n — направление в котором делается шаг, $\alpha_n > 0$ — шаг. При этом для всех выше приведенных методов верно:

$$\lim_{n \rightarrow \infty} \alpha_n = 0, \quad \sum_{n=1}^{\infty} \alpha_n = \infty \quad (2.6.2)$$

Линейные итеративные процессы вида (2.6.1), для которых выполняется (2.6.2) называются регулярными. Более подробная теория регулярных итеративных процессов изложена в [15,16].

Алгоритмы реализованы на языке Matlab и приведены в приложении 1. Таблицы расчетов методов для различных матриц, с различной точностью приведены в приложении 2.

Из приведенных расчетов можно дать характеристику каждому методу.

Недостатки метода Брауна-Робинсон:

Долгая сходимость метода, для t -го шага процесса оценка погрешности порядка $1/t^{(m+n-2)/2}$.

Не монотонность последовательностей $\underline{y}^t / t, \bar{\underline{v}}^t / t$

Достоинства модификации Амвросенко:

Увеличена скорость сходимости, за счет специального вычисления погрешности метода.

Увеличена скорость сходимости, за счет проводимых расчетов оценивающий количество партий игр, в которых игроки применяют одни и те же стратегии.

Достоинства модификации Борисовой, Магарик:

Увеличена скорость сходимости, за счет вычисления веса шага в новом направлении

Достоинства монотонного метода:

Последовательность $\underline{y}^t / t, \bar{\underline{v}}^t / t$ сходится монотонно.

Недостатки монотонного метода:

Приходится решать много вспомогательных под игр

Достоинства метода симметризации:

Предназначен для решения игр с кососимметрической матрицы

Недостатки метода симметризации:

При возникновении компонент вектора оценки выигрыша, близких к максимальной, метод может расходиться.

При увеличении размерности матрицы, и точности вычислений, существенно росли время, и число итераций необходимых для получения приближенного решения.

ГЛАВА III Определение точного решения матричной игры

ОТСЕЧЕНИЕ СТРОГО ДОМИНИРУЕМЫХ СТРАТЕГИЙ МЕТОДОМ БРАУНА-РОБИНСОН

Теорема 3.1.1. При решении матричной игры методом Брауна-Робинсон с дополнительными условиями выбора наилучшей стратегии на каждом шаге разыгрывания фиктивной игры

$$\begin{aligned} \text{для первого игрока } i^* & \left\{ \begin{array}{l} \max_r \bar{s}_r^t = \bar{s}_{r_1}^t = \dots = \bar{s}_{r_q}^t, \{r_1, \dots, r_q\} = Q \\ \max_{r \in Q} \sum_{j=1}^n \alpha_{rj} = \sum_{j=1}^n \alpha_{i^* j} \end{array} \right. \\ \text{для второго игрока } j^* & \left\{ \begin{array}{l} \min_l \underline{s}_l^t = \underline{s}_{l_1}^t = \dots = \underline{s}_{l_p}^t, \{l_1, \dots, l_p\} = P \\ \min_{l \in P} \sum_{k=1}^m \alpha_{kl} = \sum_{k=1}^m \alpha_{j^* k} \end{array} \right. \end{aligned}$$

Относительные частоты выбора чистых строго доминируемых стратегий, а также чистых стратегий нестрого доминируемых чистыми стратегиями равна 0.

Надо отметить, что выбор наилучшей стратегии в методе Брауна-Робинсон определяется формулами: для первого игрока $\max_r \bar{s}_r^t = \bar{s}_{r_1}^t = \dots = \bar{s}_{r_q}^t$, для второго игрока

$\min_k \underline{s}_k^t = \underline{s}_{k_1}^t = \dots = \underline{s}_{k_p}^t$, при этом при $q, p > 1$, выбираются стратегии с наименьшим номером r_i, k_j (лексикографическое правило выбора).

Доказательство:

Рассмотрим, как будет действовать метод Брауна-Робинсон при решении матричной игры с матрицей $\Gamma = (\alpha_{ij})$ размером $m \times n$, если в ней встречаются доминируемые стратегии.

Для определенности будем рассматривать, как производит выбор второй игрок на основе накопленных выигрышах первого игрока. Пусть первый игрок за t разыгрываний метода

Брауна-Робинсон использовал i -ую стратегию — ξ_i^t раз, тогда величины $\underline{s}_j^t = \sum_{i=1}^m \xi_i^t \cdot \alpha_{ij}$,

определяют $\underline{s}^t = (\underline{s}_1^t, \dots, \underline{s}_n^t)$ — вектор накопленных проигрышей второго игрока.

Первый случай: в задаче встречаются чистые стратегии нестрого доминирующие другие чистые стратегии. Без ограничения общности примем пусть i -ая стратегия доминирует j -ой стратегией второго игрока, что означает $\alpha_{ki} \geq \alpha_{kj}$, $k = 1, \dots, m$. Доминируемая стратегия может

быть выбрана, если $\underline{s}_i^t < \underline{s}_j^t$ или $\underline{s}_i^t = \underline{s}_j^t, i < j$. Но при этом $\underline{s}_p^t = \sum_{k=1}^m \xi_k^t \cdot \alpha_{kp}$ и

$$\alpha_{ki} \geq \alpha_{kj}, k = 1, \dots, m, \text{ а значит } \underline{s}_i^t = \sum_{k=1}^m \xi_k^t \cdot \alpha_{ki} \geq \sum_{k=1}^m \xi_k^t \cdot \alpha_{kj} = \underline{s}_j^t.$$

Предположим, что на t -ом шаге $\min_p \underline{s}_p^t = \underline{s}_i^t = \underline{s}_j^t, i < j$. Второй игрок в соответствии с

принципами рационального поведения должен выбрать j -ую стратегию на своем ходе, но в соответствии с лексикографическим правилом, он выбирает i -ую стратегию.

Встает вопрос: какое условие должно выполняться, чтобы второй игрок выбрал j -ую стратегию. Рассмотрим такое правило:

$$\begin{cases} \min_l \underline{s}_l^t = \underline{s}_{l_1}^t = \dots = \underline{s}_{l_p}^t, (l_1, \dots, l_p) = P \\ \min_{l \in P} \sum_{k=1}^m \alpha_{kl} = \sum_{i=1}^m \alpha_{ij^*} \end{cases} \quad (3.1.1)$$

Если для $\alpha_{ki} \geq \alpha_{kj}, k = 1, \dots, m$ и существует такое r , что $\alpha_{ki} > \alpha_{kj}, k = r$, то очевидно

$$\sum_{k=1}^m \alpha_{ki} > \sum_{k=1}^m \alpha_{kj} \text{ и условие выбора (3.1.1) указывает на } j\text{-ую стратегию.}$$

Если не существует такого r , что $\alpha_{ki} > \alpha_{kj}, k = r$, то $\sum_{k=1}^m \alpha_{ki} = \sum_{k=1}^m \alpha_{kj}$, значит, стратегии i -ая и

j -ая совпадают. При этом неважно, какая стратегия будет выбрана i -ая или j -ая главное, чтобы выбиралась всегда одна и та же из них. Для этого опять подойдет условие выбора стратегии по формуле (3.1.1), если условие выбора указывает на несколько стратегий, то для однозначности выбора пользуются лексикографическим правилом.

А значит, за условие выбора оптимальной стратегии вторым игроком, при которых не будут выбраны чистые стратегии нестрого доминирующие другие чистые стратегии, подойдут условия:

$$\begin{cases} \min_l \underline{s}_l^t = \underline{s}_{l_1}^t = \dots = \underline{s}_{l_p}^t, (l_1, \dots, l_p) = P \\ \min_{l \in P} \sum_{k=1}^m \alpha_{kl} = \sum_{i=1}^m \alpha_{ij^*} \end{cases}.$$

Второй случай: в задаче встречаются чистые стратегии строго доминируемые другими стратегиями. Без ограничения общности примем пусть i -ая стратегия второго игрока доминируема линейной комбинацией стратегий второго игрока с номерами $g_j \in G$, что

означает $\alpha_{ki} > \sum_{g \in G} \lambda_g \alpha_{kg}, k = 1, \dots, m$, где $\sum_{g \in G} \lambda_g = 1, \lambda_g \geq 0$.

$$\underline{s}_i^t = \sum_{k=1}^m \xi_k^t \alpha_{ki} > \sum_{k=1}^m \xi_k^t \left(\sum_{g \in G} \lambda_g \alpha_{kg} \right) = \sum_{g \in G} \lambda_g \left(\sum_{k=1}^m \xi_k^t \alpha_{kg} \right) = \sum_{g \in G} \lambda_g \underline{s}_g^t$$

Тогда пусть не найдется такое j , что $\underline{s}_i^t > \underline{s}_{g_j}^t$, значит

$$\underline{s}_i^t \geq \underline{s}_{g_j}^t, \forall g \in G \Rightarrow \sum_{g \in G} \lambda_g \underline{s}_g^t \geq \sum_{g \in G} \lambda_g \underline{s}_i^t = \underline{s}_i^t \Rightarrow \sum_{g \in G} \lambda_g \underline{s}_g^t \geq \underline{s}_i^t, \text{ что неверно. Получили}$$

противоречие, следовательно найдется такое j , что $\underline{s}_i^t > \underline{s}_{g_j}^t$. Таким образом выбор стратегии на шаге t по формуле (3.1.1), укажет на чистую недоминируемую стратегию.

Условие аналогичное (3.1.1) для первого игрока примет вид для первого игрока это условие примет вид:

$$\begin{cases} \max_r \bar{s}_r^t = \bar{s}_{r_1}^t = \dots = \bar{s}_{r_q}^t, (r_1, \dots, r_q) = Q \\ \max_{r \in Q} \sum_{j=1}^n \alpha_{rj} = \sum_{j=1}^n \alpha_{r^* j} \end{cases}$$

где $\bar{s}^t = (\bar{s}_1^t, \dots, \bar{s}_m^t)$ — вектор накопленных выигрышей первого игрока на t -ом шаге разыгрывания игры.

ОПРЕДЕЛЕНИЕ ТОЧНОГО РЕШЕНИЯ МАТРИЧНОЙ ИГРЫ МОДИФИЦИРОВАННЫМ МЕТОДОМ БРАУНА-РОБИНСОН

Используя метод Брауна с условием выбора стратегий:

$$\begin{cases} \max_r \bar{s}_r^t = \bar{s}_{r_1}^t = \dots = \bar{s}_{r_q}^t, (r_1, \dots, r_q) = Q \\ \max_{r \in Q} \sum_{j=1}^n \alpha_{rj} \end{cases}$$

для первого игрока

$$\begin{cases} \min_k \underline{s}_k^t = \underline{s}_{k_1}^t = \dots = \underline{s}_{k_p}^t, (k_1, \dots, k_p) = P \\ \min_{k \in P} \sum_{i=1}^m \alpha_{ik} \end{cases}$$

для второго игрока

можно отсечь часть доминируемых стратегий для исходной матрицы игры (см. теорему 3.1.1), получая вспомогательную матрицу игры. Решение вспомогательной игры сведем к решению двух двойственных задач симплекс методом (см. теорему 1.4.1). Из решения вспомогательной игры, восстановим решение исходной задачи по теоремам (1.5.3), (1.5.4). Схема, отражающая работу алгоритма, приведена в приложении 3.

Расчеты показывают, что предложенный алгоритм дает точное решение матричной игры за меньшее время, чем алгоритм симплекс метода и алгоритм решения задач линейного программирования большой размерности. Расчеты проводились в среде Matlab 7.0.1 и приведены в приложении 4.

ВЫВОДЫ:

1. Проведено сравнение итерационных методов решения матричных игр. Получено, что модификация Амвросенко и модификация Борисовой, Магарик сходятся быстрее, чем метод Брауна-Робинсон, монотонный метод и метод симметризации. Но при увеличении размерности матрицы, и точности вычислений, существенно росли время, и число итераций необходимых для получения приближенного решения.
2. В работе было показано, что в методе Брауна-Робинсон с дополнительным условием выбора наилучших стратегий можно применять для уменьшения матрицы игры большой размерности, решать её и восстанавливать по полученному решению решение исходной матрицы. Предлагаемый метод сравнивался с методом нахождения решения игры, путем сведения к двойственным задачам линейного программирования. Причем задачи линейного программирования решались симплекс методом и методом решения задач большой размерности реализованными в Matlab 7.0.1. Расчеты показывают, что предложенный алгоритм дает точное решение матричной игры за меньшее время.
3. Необходимо продолжить исследование для установления возможности введения дополнительного условия в модификации метода Брауна-Робинсон для усечения исходных матриц большой размерности.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. В.Г. Хорошевский, В.В.Власюк теоретико-игровой подход к организации стохастически оптимального функционирования распределенных вычислительных систем. РАН. Автометрия. 2000г. Т.39 №3 с.17-25
2. В.Г. Хорошевский, С.Н. Мамойленко Стратегии стохастически оптимального функционирования распределенных вычислительных систем. РАН. Автометрия. 2003г. Т.39 №2 с.81-91
3. А.А. Васин, В.В. Морозов. Теория игр и модели математической экономики. М.:МаксПресс 2005 —271 с.
4. И.Д. Протасов. Теория игр и исследование операций. М.: «Гелиос АРВ», 2003 —368 с.
5. Петросян Л.А., Зенкевич Н.А., Семина Е.А. Теория игр. М.: Высшая школа, 1998
6. Мулен Э.Теория игр. С примерами из математической экономики. М.: Мир, 1985 —198с.
7. Воробьев Н.Н. Основы теории игр: Бескоалиционные игры. 1984
8. Ф.П. Васильев А.Ю.Иваницкий. Линейное программирование. М.: Факториал пресс, 1998.
9. Воробьев Н.Н. Конечные бескоалиционные игры. с.21-55 УМН Т.14, вып.4 1959
10. Дж. Робинсон: Итеративный метод решения игр. с. 110—117 — Матричные игры/Ред. Н. Н. Воробьев. — М.: Физматгиз, 1961. —280 с.
11. Г.Н. Шапиро. Замечание о вычислительном методе в теории игр. с.118-127.
Матричные игры/Ред. Н. Н. Воробьев. — М.: Физматгиз, 1961. —280 с.
12. В.В. Амвросенко Ускорение сходимости метода Брауна решения матричных игр. с.570-575. Экономика и математические методы Т.1 вып. 4 1965
13. Э. П. Борисова, П. В. Магарик О двух модификациях метода Брауна решения матричных игр. //Москва. Экономика и математические методы 1966, Т2, №.5 с.732-738.
14. Садовский А.Л. Монотонный итеративный алгоритм решения матричных игр//ДАН СССР, 1978 Т.238 №3 с.538-540.
15. Беленький В.З., Волконский В.А., Иванков С.А., Поманский А.Б., Шапиро А. Д. Итеративные методы в теории игр и программировании. — М.: Наука, 1974. —239 с
16. Беленький В.З., Волконский В.А., Иванков С.А. Об одном общем подходе к исследованию сходимости итеративных процессов. //Москва. Экономика и математические методы 1974, Т10, №.1 с.140-158.
17. Д.Гейл, Х.У.Кун. А.У. Таккер. О симметричных играх. с.62-71 Матричные игры/Ред. Н. Н. Воробьев. — М.: Физматгиз, 1961. —280 с.

Приложение 1 Программная реализация итерационных методов

Программная реализация метода Брауна-Робинсон

```
function Res=brown_posledov(MATR,epsilon);
[n1, n2]=size(MATR);
if n2==1,
    [Value, I]=max(MATR);
    X=zeros(1,n1);
    X(I)=1;
Res=struct('iter',1,'epsilon',eval('epsilon'),'Xprob',X,'Yprob',1,'Value',Value);
    return
end
if n1==1
    [Value, J]=min(MATR);
    Y=zeros(1,n2);
    Y(J)=1;
Res=struct('iter',1,'epsilon',eval('epsilon'),'Xprob',1,'Yprob',Y,'Value',Value);
    return
end
Koll=zeros(1,n1);
Kol2=zeros(1,n2);
iter=0;
END=0;
Vprev=NaN;
if n2==1
    Fmin=MATR;
else
    Fmin=min(MATR');
end
[Fmax, first]=max(Fmin);
Koll(first)=1;
while(END~=1);
    %второй игрок
    %disp('второй игрок')
    [Smin, J]=min(Koll*MATR);
    Kol2(J)=Kol2(J)+1;
    [Fmax, I]=max(MATR*Kol2');
    iter=iter+1;
    %формируется условие выхода
    if (Fmax/(iter)-Smin/(iter))<epsilon
        END=END+1;
    else
        END=0;
    end
    Vprev=Vaver;
    if END~=1,
        Koll(I)=Koll(I)+1;
    end
end
Res=struct('iter',iter,'epsilon',eval('epsilon'),'Xprob',Koll/(iter),'Yprob',Kol2/(iter),
'Value',Vaver);
```

Программная реализация метода Амвросенко

```
function RES=Amvros(MATR,epsilon)
%алгоритм Амвросенко- модификация метода Брауна
%за счет кратного повторения стратегий игроками
[n1,n2]=size(MATR);
if n2==1,
    [Value, I]=max(MATR);
    X=zeros(1,n1);
    X(I)=1;
    RES=struct('iter',1,'epsilon',eval('epsilon'),'Xprob',X,'Yprob',1,'Value',Value);
    return
end
if n1==1
    [Value, J]=min(MATR);
    Y=zeros(1,n2);
    Y(J)=1;
    RES=struct('iter',1,'epsilon',eval('epsilon'),'Xprob',1,'Yprob',Y,'Value',Value);
    return
end
%строки накопления принятия стратегий игроками 1,2
```

Приложение 1 Программная реализация итерационных методов

```

Kol1=zeros(1,n1);
Kol2=zeros(1,n2);
Vf_max=-inf;
Vc_min=inf;
iter=0;
END=0;
%Vprev=NaN;
if n2==1
    Fmin=MATR;
else
    Fmin=min(MATR');
end
[X, first] =max(Fmin);
Kol1(first)=1;
%второй игрок
[ Smin, J] =min(Kol1*MATR);
Kol2(J)=Kol2(J)+1;
while(Vc_min-Vf_max>epsilon);
    %на k-ом шаге делаем расчет, о том какие стратегии принимаем
    %для этого находим вектора, показывающие, сколько и какие стратегии
    %мы принимаем несколько раз подряд первым и вторым игроком
    %определяем стратегии k1,k2
    [ a, k1] =min(Kol1*MATR);
    A=a/sum(Kol1);
    [ b, k2] =max(MATR* (Kol2'));
    B=b/sum(Kol2);
    %определяем вектора m1,m2 по которым в дальнейшем определим
    %количество раз принятия одной и той же стратегии двумя игроками
    m2=inf*ones(1,n1);
    for j=1:n1,
        if MATR(j,k1)>MATR(k2,k1)
            m2(1,j)=floor(((MATR(k2,:)* (Kol2'))-MATR(j,:)* (Kol2')) / (MATR(j,k1)-
MATR(k2,k1)))+1;
        end
    end
    m1=inf*ones(1,n2);
    for j=1:n2,
        if MATR(k2,k1)>MATR(k2,j)
            m1(1,j)=floor(((Kol1*MATR(:,j))-Kol1*MATR(:,k1)) / (MATR(k2,k1)-MATR(k2,j)))+1;
        end
    end
    %m-сколько раз используется стратегия
    m=min([ m1,m2]);
    %изменения в стратегиях
    Kol1(1,k2)=Kol1(1,k2)+m;
    Kol2(1,k1)=Kol2(1,k1)+m;
    iter=iter+1;
    %определение нижней и верхней цен игры
    VfFloor=min(Kol1*MATR)/sum(Kol1);
    if VfFloor>Vf_max
        Vf_max=VfFloor;
        Kol1_best=Kol1;
    end
    Vceil=max(MATR* (Kol2'))/sum(Kol2);
    if Vceil<Vc_min,
        Vc_min=Vceil;
        Kol2_best=Kol2;
    end
end
RES=struct('iter',iter,'epsilon',eval('epsilon'), 'Xprob',Kol1_best/sum(Kol1_best), 'Yprob',
,Kol2_best/sum(Kol2_best), 'Value', (Vc_min+Vf_max)/2);

```

Программная реализация метода Борисовой, Магарик

```

function RES=Borisova_var1(MATR,epsilon)
%(MATR,epsilon)
%алгоритм Борисовой, Магарик- модификация метода Брауна
%в котором определяется некоторым образом тяжесть принимаемой стратегии
%first- оптимальная стратегия первого игрока
%second- оптимальная стратегия второго игрока

```

Приложение 1 Программная реализация итерационных методов

```

[ n1,n2]=size(MATR);
if n2==1,
    [ Value, I] =max(MATR);
    X=zeros(1,n1);
    X(I)=1;
    RES=struct('iter',1,'epsilon',eval('epsilon'),'Xprob',X,'Yprob',1,'Value',Value);
    return
end
if n1==1
    [ Value, J] =min(MATR);

    Y=zeros(1,n2);
    Y(J)=1;
    RES=struct('iter',1,'epsilon',eval('epsilon'),'Xprob',1,'Yprob',Y,'Value',Value);
    return
end
%стратегии первого и второго игроков
X=zeros(1,n1);
Y=zeros(1,n2);
%множества определяющие оптимальные стратегии
I=zeros(1,n1);
J=zeros(1,n2);
Umin_best=-inf;
Vmax_best=inf;
END=0;
%первый игрок
if n2==1
    Fmin=MATR;
else
    Fmin=min(MATR');
end
[ Fmax, first]=max(Fmin);
X(1,first)=1;
%второй игрок
[ Smin, Second]=min(X*MATR);
Y(1,Second)=1;
%определение множества J-оптимальных стратегий
iter=1;
while (Vmax_best-Umin_best)>epsilon
    %первый игрок
    %1- выбирает себе оптимальную стратегию
    [ Fmax, first]=max(MATR*Y');
    Ei=zeros(1,n1);
    %оптимальная стратегия
    Ei(first)=1;
    %2- выбирает себе вес данной оптимальной стратегии
    U=(X*MATR);
    Umin=min(U);
    if Umin>Umin_best
        Xbest=X;
        Umin_best=Umin;
    end
    %j- вектор из 1 и inf, 1-элемент i-ой строки входит в множество J, inf
    %- не входит в множество J
    J=inf*ones(1,n2);
    for i=1:n2,
        if U(i,i)==Umin,
            J(i,i)=1;
        end
    end
    %определяем минимальный элемент Aij в строке MATR номера столбцов, которой
    %принадлежат множество J
    [ Aij,Jindex]=min(MATR(first,:).*J);
    %затем определяем lambda1
    lambda1=inf;
    for j=1:n2
        if MATR(first,j)<Aij
            lambda_help=(U(j)-U(Jindex))/(U(j)-U(Jindex)+Aij-MATR(first,j));
            if lambda_help<lambda1
                lambda1=lambda_help;
            end
        end
    end
    %установка стратегии для игрока 2
    Y=zeros(1,n2);
    for j=1:n2
        if lambda1<lambda_help
            Y(j)=1;
        end
    end
    %обновление стратегии игрока 1
    X=zeros(1,n1);
    for i=1:n1
        if Y(i)>0
            X(i)=1;
        end
    end
    %обновление оптимальных стратегий
    Fmax=MATR*X;
    Fmin=min(Fmax);
    Vmax_best=max(Fmax);
    Umin=min(Fmin);
    if Vmax_best-Umin<epsilon
        END=1;
    end
end

```

Приложение 1

Программная реализация итерационных методов

```

    end
end
if lambda1==inf
    lambda1=1/(iter+1);
end
%составление оптимальной стратегий
X=X*(1-lambda1)+lambda1*Ei;
%второй игрок
%2-выбирает себе оптимальную стратегию
[Smin, second] =min(X*MATR);
Ej=zeros(1,n2);
%оптимальная стратегия
Ej(1,second)=1;
%2-выбирает себе вес данной оптимальной стратегии
V=(MATR*Y)';
Vmax=max(V);
if Vmax<Vmax_best;
    Vmax_best=Vmax;
    Ybest=Y;
end
%j- вектор из 1 и inf, 1-елемент i-ой строки входит в множество J, inf
%- не входит в множество J
I=-inf*ones(1,n1);
for j=1:n1,
    if V(1,j)==Vmax,
        I(1,j)=1;
    end
end
%определеняем максимальный элемент Bij в строке MATR номера столбцов, которой
%принадлежат множеству J
[Bij,Iindex]=max(MATR(:,second).*I');
%затем определяем lambda1
mul=inf;
for i=1:n1
    if MATR(i,second)>Bij
        mu_help=(V(1,Iindex)-V(1,i))/(V(1,Iindex)-V(1,i)+MATR(i,second)-Bij);
        if mu_help<mul
            mul=mu_help;
        end
    end
end
if mul==inf
    mul=1/(iter+1);
end
%составление оптимальной стратегий
Y=Y*(1-mul)+mul*Ej;
iter=iter+1;
end
RES=struct('iter',iter,'epsilon',eval('epsilon'),'Xprob',Xbest,'Yprob',Ybest,'Value',(Um
n_best+Vmax_best)/2);

```

Программная реализация монотонного метода

```

function RES=monoton_XY(MATR,eps)
%монотонный метод два раза решаем игру с матрицей A
%и матрицей Trans(-A)
[n1, n2]=size(MATR);
RES=struct('iter',0,'help_iter',0,'epsilon',eval('eps'),'Xprob',zeros(1,n1),'Yprob',zeros
(1,n2),'Value',0);
resX=monotonous(MATR,eps);
resY=monotonous(-MATR',eps);
RES.iter=resX.iter+resY.iter;
RES.help_iter=resX.help_iter+resY.help_iter;
RES.Xprob=resX.Xprob;
RES.Yprob=resY.Xprob;
resX.Value;
resY.Value;
RES.Value=(resX.Value-resY.Value)/2;

function Res=monotonous(A,epsilon)
[n1, n2]=size(A);

```

Приложение 1

Программная реализация итерационных методов

```

Brown_epsilon=eval('epsilon');
%возвращаемая структура
Res=struct('iter',0,'help_iter',0,'epsilon',eval('epsilon'),'Xprob',zeros(1,n1),'Value',0
);
%искомые вектора
Xprob=zeros(1,n1);
Chelp=zeros(1,n2);
if n2==1
    Fmin=A;
else
    Fmin=min(A');
end
[ X, I]=max(Fmin);
Xprob(I)=1;
Chelp=A(I,:);
iter=1;
END=0;
Xprev=Xprob;
Vaver=min(Chelp);
Vprev=Vaver;
while(END~=1)
    Note=zeros(1,n2);
    for i=1:n2,
        if abs(Chelp(1,i)-Vprev)<epsilon
            Note(1,i)=1;
        end
    end
    Under_A=[];
    for i=1:n2
        if Note(1,i)==1
            Under_A=[Under_A,A(:,i)];
        end
    end
    %вызов под функции решающей игру методом Брауна-Робинсон
    Ans=brown_posledov(Under_A,Brown_epsilon);
    Res.help_iter=Res.help_iter+Ans.iter;
    Xdif=Ans.Xprob;
    Cdif=(Xdif*A);
    Ans=brown_posledov([ Cdif,Chelp],Brown_epsilon);
    Res.help_iter=Res.help_iter+Ans.iter;
    Alpha=Ans.Xprob;
    %не обязательная проверка
    if (abs(1-Alpha(1,1)-Alpha(1,2))>epsilon
        %iter
        disp('ошибка');
        return;
    end
    Xprob=(1-Alpha(1,1))*Xprev+Alpha(1,1)*Xdif;
    Chelp=(1-Alpha(1,1))*Chelp+Alpha(1,1)*Cdif;
    Vaver=min(Chelp);
    if (abs(Vaver-Vprev)<epsilon)
        END=END+1;
    end
    Vprev=Vaver;
    Xprev=Xprob;
    iter=iter+1;
end
Res.iter=iter;
Res.Xprob=Xprob;
Res.Value=Vaver;

```

Программная реализация метода симметризации

```

function RES=method_symmetr(A,epsilon)
%алгоритм решения игры путем сведения исходной матрицы к симметричной
MATR=A';
[m,n]=size(MATR);
Minel=min(min(MATR));
if(Minel<=0)
    Matr=MATR+(-Minel+1)*ones(m,n);
else

```

Приложение 1 Программная реализация итерационных методов

```

    - Matr=MATR;
end
Smatr=symmetr(Matr);
X=zeros(m+n+1,1);
%Определяем минимаксную стратегию
maxSmatr=max(Smatr);
[ MinMax, J]=min(maxSmatr);
X(J)=1;
%Ro - переменная помогающая от зацикливания
Ro=0.025;
sigma=inf;
iter=0;
while (iter<1000000)&& (sigma>epsilon)
    ksi=Smatr*X;
    [ sigma,maxI]=max(ksi);
    h_matr=zeros(1,m+n+1);
    for j=1:n+m+1
        if j~=maxI
            if Smatr(j,maxI)<=0
                h_matr(j)=inf;
            else
                h_matr(j)=(sigma-ksi(j))/(Smatr(j,maxI)+sigma-ksi(j));
            end
        else
            h_matr(j)=inf;
        end
    end
    %К- номер следующей стратегии, которую будем уменьшать
    [ Alpha,K]=min(h_matr);
    if Alpha<Ro*sigma,
        Alpha=Ro*sigma;
    end
    e=zeros(n+m+1,1);
    e(maxI)=1;
    X=(1-Alpha)*X+Alpha*e;
    iter=iter+1;
end

ProbX=zeros(m,1);
ProbY=zeros(n,1);
ProbX=X(1:m,1)/X(m+n+1,1);
ValueY=1/sum(ProbX);
YY=ProbX*ValueY;
ProbY=X(m+1:m+n,1)/X(m+n+1,1);
ValueX=1/sum(ProbY);
XX=ProbY*ValueX;
if Minel<=0
    ValueG=(ValueX+ValueY)/2+(Minel-1);
else
    ValueG=(ValueX+ValueY)/2
end
RES=struct('iter',iter,'epsilon',eval('epsilon'),'Xprob',XX,'Yprob',YY,'Value',ValueG);

function Smatr=symmetr(MATR)
%алгоритм симметризации матрицы
[m,n]=size(MATR);
Smatr=zeros(m+n+1);
Smatr(1:m,m+1:m+n)=-MATR;
Smatr(m+1:m+n,1:m)=MATR';
Smatr(m+n+1,1:m)=-ones(1,m);
Smatr(m+n+1,m+1:m+n)=ones(1,n);
Smatr(1:m,m+n+1)=ones(m,1);
Smatr(m+1:m+n,m+n+1)=-ones(n,1);

```

Приложение 2

Все методы реализованы в среде Matlab 7.0.1, там же проводились расчеты.

Примеры решения игр полковника Блотто, матрица которой задается формулами

$$\alpha_{ij} = K(x_i, y_j) \begin{cases} n+2, & \text{если } m-i > n-j, i > j, \\ n-j+1, & \text{если } m-i > n-j, i = j, \\ n-j-i, & \text{если } m-i > n-j, i < j, \\ -m+i+j, & \text{если } m-i < n-j, i > j, \\ j+1, & \text{если } m-i = n-j, i > j, \\ -m-2, & \text{если } m-i < n-j, i < j, \\ -i-1, & \text{если } m-i = n-j, i < j, \\ -m+i-1, & \text{если } m-i < n-j, i = j, \\ 0, & \text{если } m-i = n-j, i = j, \end{cases}$$

Игра полковника Блотто размера 11×14

		Брауна-Робинсон	Модификация Амвросенко	Модификация Борисовой	Монотонный	Симметризации
$\epsilon=10^{-2}$	N	4334	245	1111	10(22901)	42616
	T(c)	0.23	0.17	0.19	1.36	14.17
	V	-4.9960	-4.9964	-4.9987	-5.0012	-4.9169
$\epsilon=10^{-3}$	N	43020	1343	11048	527(374800)	-
	T(c)	2.14	0.7650	1.83	17.89	-
	V	-4.9996	-4.9995	-4.9999	-5.0005	-

Игра полковника Блотто размера 7×10

		Брауна-Робинсон	Модификация Амвросенко	Модификация Борисовой	Монотонный	Симметризации
$\epsilon=10^{-2}$	N	1834	233	565	7(4258)	153405
	T(c)	0.5160	0.1560	0.3250	0.2500	40.4530
	V	-3.4965	-3.4968	-3.4965	-3.1181	-3.3630

Игра полковника Блотто размера 11×12

		Брауна-Робинсон	Модификация Амвросенко	Модификация Борисовой	Монотонный	Симметризации
$\epsilon=10^{-2}$	N	2812	80	606	7(82025)	63485
	T(c)	0.1570	0.0620	0.0910	3.7810	19.0470
	V	-4.9972	-4.9985	-4.9973	-5.0046	-5.0180

При увеличении размерности матрицы, и точности вычислений, существенно росли время, и число итераций необходимых для получения приближенного решения.

Приложение 3

Ниже приведена схема определение точного решения матричной игры модифицированным методом Брауна-Робинсон



Приложение 4

Расчеты проводились в среде Matlab 7.0.1. Предлагаемый метод сравнивался с методом нахождения решения игры, путем сведения к двойственным задачам линейного программирования. Причем задачи линейного программирования решались симплекс методом и методом решения задач большой размерности реализованными в Matlab 7.0.1. Пример решения игры разность, матрица которой задается формулами

$$\alpha_{ij} = |i - j|$$

		Симплекс метод	Метод решения задач большой размерности	Предлагаемый метод
N=50	T	0.32	0.20	0.03
N=200	T	4.42	2.78	0.53
N=500	T	42.67	36.67	4.61

Пример решения игры дуэль, матрица которой задается формулами

$$\alpha_{ij} = \frac{n(i-j) + ij}{n^2}$$

N=50	T	2.73	0.60	0.54
N=200	T	291.57	18.31	15.87

Пример решения игры в пальцы, матрица которой задается формулами

$$\alpha_{(i+j \cdot m)(p+q \cdot m)} = \begin{cases} j+q+2, & i = q+1, j+1 \neq p \\ -(j+q+2), & i \neq q+1, j+1 = p \\ 0, & \text{в остальных случаях} \end{cases}$$

N=64	T	3.83	0.85	0.74
N=256	T	450.57	30.31	25.45

Расчеты показывают, что предложенный алгоритм дает точное решение матричной игры за меньшее время, чем алгоритм симплекс метода и алгоритм решения задач линейного программирования большой размерности.