МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Тольяттинский государственный университет»

Кафедра «Прикладная математика и информатика»
(наименование)

01.03.02 Прикладная математика и информатика
(код и наименование направления подготовки, специальности)

Компьютерные технологии и математическое моделирование
(направленность (профиль))

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему: «Применение теории игр к моделированию современных экономических процессов»

Обучающийся А.И. Цибикина
(И.О. Фамилия) (личная подпись)

Руководитель М.А. Тренина
(ученая степень, звание, И.О. Фамилия)

Консультант к.п.н., доцент С.А. Гудкова
(ученая степень, звание, И.О. Фамилия)

Тольятти 2024

Аннотация

Тема выпускной квалификационной работы: «Применение теории игр к моделированию современных экономических процессов». В данной работе представлена программа, в которой реализовано решение экономической задачи для автомобильного завода при помощи матричной игры.

Актуальность работы заключается в том, что любую конкурентную ситуацию можно решить при помощи математической формулировки. Объектом исследования работы является матричная игра и способы её решения.

Данная выпускная квалификационная работа состоит из: введения, заключения, трёх разделов и списка используемой литературы.

В ведении описывается связь между экономикой и теорией игр, раскрывается поставленная задача и ставятся задачи для её решения.

В первом разделе исследуется теория игр и выделяется вид – матричных игр, а также описываются различные способы решения.

Во втором разделе описывается выбор языка программирования и проектирование программы с учетом теоретической части.

В третьем разделе производится реализация и тестирования приложения, а также демонстрируется результат работы на готовом примере.

Заключение состоит из вывода о проделанной работе.

В работе использовано 1 таблица, список литературы, который содержит 25 литературных источников, 27 рисунков. Общий объем выпускной квалификационной работы составляет 56 страницы.

Abstract

The topic the graduate qualification work: «The application of game theory to the modeling of modern economic processes».

The graduation work consists of an introduction, three chapters, a conclusion, a list of used literature, including foreign sources.

The key issue of the thesis is the implementation of a program to solve an economic problem using a matrix game. We address the problem of competitive situations between two players and solve them using game theory.

The purpose of the final qualification work is to create a console application for solving a matrix game and finding an equilibrium situation for the task.

The thesis can be divided into several parts, which include problem situation analysis, studying theoretical material on the topic of game theory and choosing the appropriate type of game, analyzing various methods for solving matrix games, designing one of the solution methods for the task, implementing a console application and testing a ready-made application using data taken from the economics of an automobile plant.

The management describes the task and sets tasks for its solution.

In the first section, game theory is investigated and the type of matrix games is highlighted. The second section describes the design of the program taking into account the theoretical part. The third section provides implementation and testing. The conclusion consists of a conclusion on the work done.

The work uses 1 table, the list of references contains 25 literary sources, 27 drawings. The total volume of the final qualifying work is 56 pages.

Содержание

Введение

Тема выпускной квалификационной работы «Применение теории игр при моделирование современных экономических задач». В наше время экономика и математика тесно связаны, поскольку математика помогает принимать эффективные решения в сложных экономических ситуациях, где необходимо учитывать множество факторов, включая взаимодействие между конкурирующими сторонами. Наилучшим примером конфликтной ситуации является ситуация борьбы между предприятиями, которые производят одинаковую продукцию.

Вариант, который рассмотрен в выпускной квалификационной работе – матричная игра. В ней противодействует два игрока, у которых стратегии при различных альтернативах конечны.

Актуальность данной темы обусловлена тем, что теория игр применяется во многих сферах, так как обеспечивает основу для понимания того, как может вести себя фирма или отдельные лица в условиях конкуренции.

Объектом исследования является матричная игра для решения проблемы в экономике автомобильного завода.

Предметом исследования методы решения матричных игр.

Цель выпускной квалификационной работы создание консольного приложения для решения матричной игры и нахождения равновесной ситуации по поставленной задачи.

Чтобы осуществить данную цель, мне необходимо решить задачи:

- проанализировать проблемную ситуацию и выполнить постановку задачи;
- изучить теоретический материал по теме теории игр и выбрать подходящий тип игры;
- провести анализ различных методов для решения матричных игр;

– спроектировать один из методов решения на поставленной задаче;
– реализовать консольное приложение с учетом алгоритма построенного ранее;
– протестировать готовое приложение на примере данных, взятых из экономики автомобильного завода.

В первом разделе рассматривается понятия теории игр, их классификация и пояснение выбора матричной игры, а также рассматривается решение матричной игры при помощи различных методов. Во втором разделе определяется язык программирования и проектируется программа для решения матричной игры. В третьем разделе с помощью алгоритма описанного во втором разделе реализуется консольное приложение и тестируется на готовых данных.

1 Теория игры в экономике

1.1 Актуальность проблемы и постановка задачи

Автомобильный завод 1 столкнулся с снижением спроса на свою продукцию из-за популярности автомобилей автомобильного завода 2. Чтобы увеличить прибыль завода 1 нужно увеличить спрос продукции. Предположим, что каждый завод имеет возможность производить автомобили с применением различных технологий, представлены в таблице 1. С учетом качества производства по разным технологиям предприятия назначает определенную цену за автомобиль (среднее число). Но также у каждого предприятия есть затраты на производство одного автомобиля (запчасти, оборудование, зарплата персоналу). А также зависимость продажи автомобиля, напрямую зависит от соотношения цен на автомобили отечественного бренда и конкурента [18].

Таблица 1 – Описание технологий

Технология	Описание
1. Линейное производство	Эта технология производства автомобилей, где процессы производства разделены на последовательные этапы.
2. Использование передовых материалов	Применяет передовые материалы, такие как легкие сплавы, для создания более легких и прочных автомобилей.
3. Цифровизация и автоматизация	Завод внедряет цифровые технологии и системы автоматизации в производственные процессы. Это позволяет повысить точность и скорость производства.

В результате маркетингового исследования рынка автомобилей была определена функция спроса на продукцию:

$$Y = A - B * X\,, \qquad (1)$$

где Y – количество продукции, которую приобретёт население, д.е.;

A – константа, отражающая уровень спроса при нулевой цене;

B – коэффициент, отражающий изменение спроса на единицу товара в ответ на изменение цены;

X – средняя цена автомобиля заводами, д.е [1].

В данной формулировке задачи важно определить сколько продукции будет реализовано в каждом предприятии и какая прибыль за счет этого будет наибольшей. Следовательно, одной из основных задач является максимизация собственной прибыли. Но в данном случае более важной проблемой является конкурентная борьба. В конкурентной борьбе победителем становится не тот, кто получает наибольшую прибыл, а тот, кто получает наибольшую разницу в прибыли по сравнению с конкурентами. Такой конфликт можно рассматривать как соревнование двух игроков с нулевой суммой, где выигрыш одного игрока равен проигрышу другого, и в данном случае теория игр становится одним из основных инструментов для решения данной задачи.

1.2 Основные понятия в теории игр

«Теория игр – раздел прикладной математики, точнее исследование операций, где изучают стратегические взаимодействия между лицами, принимающими рациональные решения» [7, с. 14]. Следовательно, игра эта борьба за выигрыш каждого игрока, но игроком считается только тот, кто принимает решение в игровой ситуации. Каждый игрок в процессе игры применяет свою стратегию для достижения выигрыша – оценка результата игры. Исход игры представляет собой совокупность всех возможных состояний, которые могут быть достигнуты в результате выбора оптимальной стратегий игроками. Значит математическая формулировка игрового подхода будет выглядеть следующим образом:

$$I = (K, S, W), \quad (2)$$

где $K = 1, \dots, k -$ количество игроков, участвующих в игре;

$S = (S_{ij}, \dots, S_{nm}) -$ стратегии игроков, для каждого игрока $i\ (1, \dots, n)$ ход его стратегии зависит от хода его противника $j\ (1, \dots, m)$;

$W = (W_1, \dots, W_n) -$ выигрыш игрока.

Для реализации игровых ситуация и нахождения оптимальных стратегий выделяются типы игр, представленные на рисунке 1.

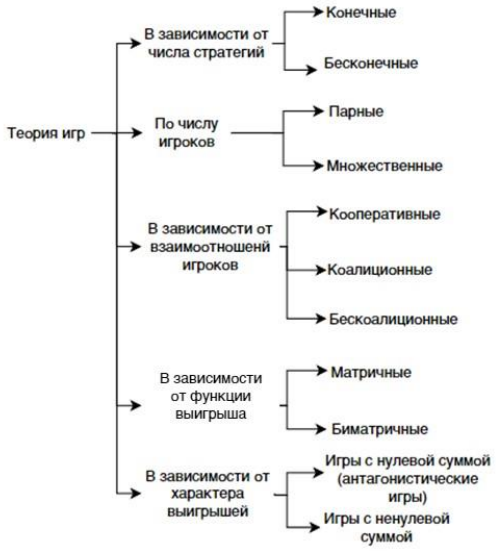

Рисунок 1 – Классификация игр

У каждого типа игры есть свои подклассы, их можно классифицировать, опираясь на их признаки и методы решения. Игры имеют стратегии и от их числа зависит ход решения, поэтому выделяют подкласс, зависящий от количества стратегий: конечные игры, где число стратегий ограничено и бесконечные. Также в играх может участвовать два и более игрока, поэтому выделяется подкласс, отвечающий за количество игроков в игре: парные и множественные игры с n – игроками, где $n > 2$. В играх также есть взаимоотношение между игроками, они могут собираться в круг лиц по

интересам в процессе игры или разделится на них в самом начале: коалиционные, бескоалиционные, кооперативные. Игры могут быть классифицированы по характеру выигрышей. Игры с нулевой суммой, или антагонистические, представляют собой ситуацию, когда выигрыш одного игрока происходит за счет проигрыша другого, причем сумма выигрышей в каждой партии равна нулю. В отличие от этого, в играх с ненулевой суммой, или неантагонистических играх, критерии для игроков различны, и сумма выигрышей может быть ненулевой [16]. Также игры можно разделить по виду функций выигрыша. Матричные игры представляют собой конечные игры двух игроков с нулевой суммой, которые задаются в виде одной матрицы, где выигрыши одного игрока равны проигрышам другого. Биматричные игры являются конечными играми с ненулевой суммой, где выигрыши каждого игрока представлены матрицей данного игрока, задаваемой двумя матрицами.

В поставленной задачи мы имеем дело с двумя игроками, и количество наших шагов будет конечными, также от выигрыша одного из игроков, будет зависит проигрыш другого. Поэтому вариант игры, который подойдет для решения данной задачи – матричная игра.

1.3 Матричные игры. Составление модели игры

«Матричные игры – это математическая модель антагонистической игры, в которой участвуют два игрока, каждый из которых принимает решение о своей стратегии, стремясь максимизировать свой выигрыш, при этом их количество шагов конечно» [6, с. 8]. Основные понятия, связанные с матричными играми, включают в себя платежную матрицу.

Платежная матрица – это основной инструмент анализа матричных игр. Она представляет собой таблицу, в которой обычно строки соответствуют стратегиям первого игрока, а столбцы – стратегиям второго игрока [4].

В платежной матрице элементами могут выражаться в разных эквивалентах. Матрица будет выглядеть следующим образом:

$$V = \begin{pmatrix} v_{11} & v_{12} & \cdot\cdot & v_{1n} \\ v_{21} & v_{22} & \cdot\cdot & v_{2n} \\ \cdot\cdot & \cdot\cdot & \cdot\cdot & \cdot\cdot \\ v_{m1} & v_{m2} & \cdot\cdot & v_{mn} \end{pmatrix}, \quad (3)$$

где $m -$ количество чистых стратегий первого игрока;

$n -$ количество чистых стратегий второго игрока.

«Стратегия игрока – это набор правил или действий, которые выбирает игрок для достижения оптимального результата в игре, если первый игрок выбирает стратегию i, а второй выбирает стратегию j, тогда выигрыш для первого игрока является при стратегии $s = (i, j)$, что соответствует выигрышу $V_1(s) = V_1(i, j) = v_{ij}$» [16, с. 69] . Поскольку это игра с нулевой суммой, то в данной игре второй выигрыш игрока $V_2(s) = -V_1(s)$.

Главной задачей игрока 1 увеличить (максимизировать) свой результат. Для начала нужно определить наилучшую стратегию стороны игрока 1, при том, что после его хода, у игрока 2 будут уменьшаться выигрыши. Значит, когда игрок 1 будет ходить при стратегии i, то игрок 2 на каждый ход со стороны 1 отвечает своей стратегией j_*, которая будет не выгодна для игрока 1 и даст ей меньший выигрыш. Поэтому мы находим минимальный элемент в каждой строке:

$$v_{ij_*} = min_j\, v_{ij}. \quad (4)$$

Чтобы найти для 1 игрока оптимальную стратегию, нужно найти максимальный выигрыш из v_{ij_*} при самой разной стратегии игрока противника. Данная величина обозначения как нижняя цену игры или максимин и находится по формуле:

$$v_{i_* j_*} = \max_i v_{i j_*} = \max_i \min_j v_{ij}. \quad (5)$$

Похожим алгоритмом будет пользоваться игрок 2 и при выборе стратегии он будет выбирать наибольший выигрыш, значит игрок 2 при любой его стратегии j, при том что игрок 1 отвечает стратегией i_* невыгодной для игрока 2. Следует проигрыш будет наибольшим. Поэтому находим максимальный элемент в каждом столбце:

$$-v_{i j_*} = v_{i_* j} = \max_i v_{ij}. \quad (6)$$

Чтобы найти для игрока 2 оптимальную стратегию, нужно найти минимальный проигрыш из $v_{i_* j}$ при самой разной стратегии игрока 1. Данная величина находится по формуле 8.

$$-v_{i_* j_*} = \max_j(-v_{i_* j}) = \max_j(-\max_i v_{ij}) = -\min_j \max_i v_{ij}, \quad (7)$$

получим:

$$v_{i_* j_*} = \min_j \max_i v_{ij}. \quad (8)$$

Эта пара стратегий называется седловой точкой. Она включает в себя метод минимакса и максимина, который предполагает, что каждый игрок выбирает стратегию так, чтобы минимизировать потенциальный проигрыш или максимизировать свой выигрыш. Поэтому метод седловой точки находит оптимальное решение для каждого игрока в ситуации равновесия.

1.4 Ситуация равновесия. Матричная игра в чистых стратегиях

Допустим в платежной матрице V есть седловая точка, тогда рассмотрим такие величины, как нижняя цена – h_* и верхняя цена – h^*. Эти величины находятся через стратегии, которые достигаются $\max_i \min_j v_{ij}$ и $\min_j \max_i v_{ij}$, для игрока 1 эта стратегия называется – максимин, для игрока 2 – минимакс:

$$h_* = \max_i \min_j v_{ij}, \tag{9}$$

$$h^* = \min_j \max_i v_{ij}. \tag{10}$$

Ситуация равновесия по Нэшу в матричной игре для платежной матрицы V, выглядит как $(s_1^{(i^*)}, s_2^{(j^*)}) = (i^*, j^*)$ и представляет неравенством:

$$v_{ij_*} \leq v_{i_*j_*} \leq v_{i_* j}, \tag{11}$$

где $i = 1,2, \dots m$;

$j = 1,2, \dots n$.

«Те стратегии (i_*, j_*), которые удовлетворяет неравенству и будут являться седловой точкой матрицы. Так как у нас бывают разные вариации матрицы, то седловая точка не всегда присутствует» [16, с. 99]. Разберемся с свойствами матричных игр:

- если у нас есть выигрыш в матричной игре $V(i,j) = v_{ij}$, то в ситуации равновесия выигрыш $V(i_*, j_*) = v_{i_*j_*} = h$. Значение $h -$ это теперь цена игры;
- нижная цена игры будет всегда меньше, чем верхняя цена $(\max_i \min_j v_{ij} \leq \min_j \max_i v_{ij})$;
- $h = \max_i \min_j v_{ij} = \min_j \max_i v_{ij} = v_{i_*j_*}$;

– стратегии $(s_1^{(i^*)}, s_2^{(j^*)})$ являются оптимальными для каждого игрока, когда матричная игра имеет седловую точку (i_*, j_*). Каждый игрок будет придерживаться своей оптимальной стратегии, чтобы улучшить свой ход.

В математическом виде нахождение седловой точки можно рассмотреть в виде следующей схемы:

$$V = \begin{pmatrix} v_{11} & v_{12} & \dots & v_{1n} \\ v_{21} & v_{22} & \dots & v_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ v_{m1} & v_{m2} & \dots & v_{mn} \end{pmatrix} \begin{matrix} \to \min_j v_{1j} \\ \to \min_j v_{2j} \\ \vdots \\ \to \min_j v_{mj} \end{matrix} \to \max_i \min_j v_{ij} \qquad (12)$$

$$\begin{matrix} \downarrow & \downarrow & & \downarrow \\ \max_i v_{i1} & \max_i v_{i2} & \dots & \max_i v_{in} \end{matrix}$$

$$\downarrow$$

$$\min_j \max_i v_{ij}$$

Определение седловой точки можно также описать при помощи алгоритма, представленного на рисунке 2 и применять его на разные вариации матричной игры.

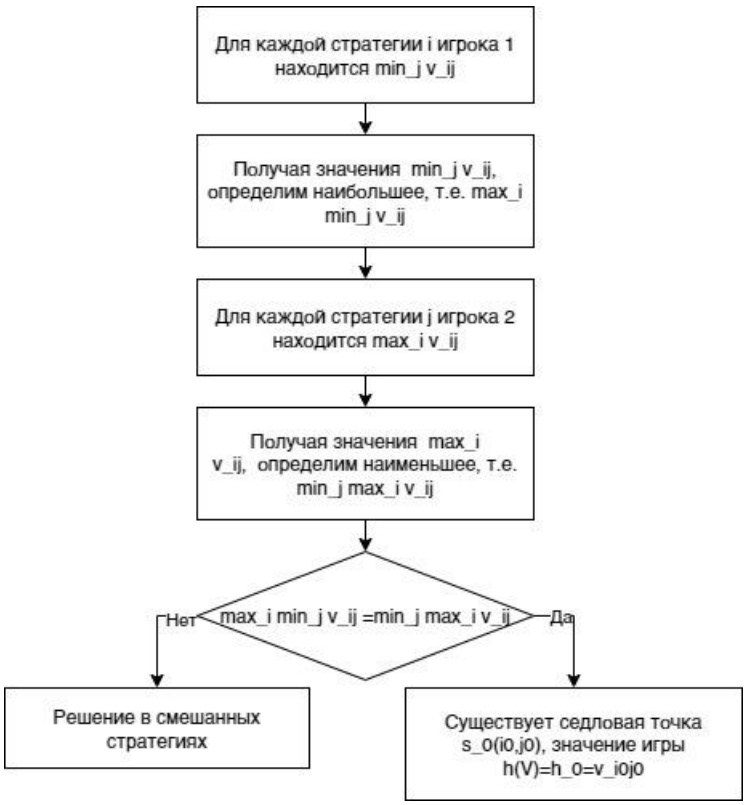

Рисунок 2 – Алгоритм нахождения седловой точки

Пример 1. Представлена платежная матрица

$$V = \begin{matrix} \begin{pmatrix} 5 & 1 & 1 & 2 \\ 4 & 1 & 3 & 5 \\ 3 & 2 & 4 & 3 \end{pmatrix} \begin{matrix} \to 1 \\ \to 1 \\ \to 2 \end{matrix} \to \max_i \min_j v_{ij} = 2 \\ \begin{matrix} \downarrow & \downarrow & \downarrow & \downarrow \\ 5 & 2 & 4 & 3 \end{matrix} \\ \downarrow \\ \min_j \max_i v_{ij} = 2 \end{matrix} \qquad (13)$$

Следовательно, нижняя и верхняя цена игры равны $h = \max_i \min_j v_{ij} = \min_j \max_i v_{ij} = 2$, тогда оптимальные стратегии в данной игре $(i_*, j_*) = (2,3)$, она составлена из стратегии 2 игрока 1, и стратегией 3 игрока 2, они считаются оптимальными для каждого из них.

Таким образом, наличие седловой точки определяет решается матричная игра в чистых стратегиях или нет, если мы не имеем седловой точки, тогда задача решается в смешанных стратегиях и имеет иное решение.

1.5 Матричная игра в смешанных стратегиях

В матричных играх не всегда есть ситуация равновесия, бывает, что нижняя и верхняя цена игры не равны. Это говорит о том, что игрок 1 может создать для себя ситуацию гарантированного выигрыша, а игрок 2 не может ему ответить гарантированным проигрышем [3]. Поэтому наша сумма выигрышей уже не является нулевой $h_* - h^* = \max_i \min_j v_{ij} - \min_j \max_i v_{ij} > 0$. В данном случае каждый игрок должен выбирать свои стратегии с определенной вероятностью. В отличие от варианта в чистых стратегиях, такие стратегия называются смешанными. Представим смешанную стратегию в виде вектора:

$$X = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{pmatrix} = (x_1, x_2, \dots, x_m)^T, \quad (14)$$

где $m -$ количество чистых стратегий первого игрока;

x_i –вероятность выбора игрока стратегии i, $x_i \geq 0$, $\sum_{i=1}^{m} x_i = 1$;

X^T –транспонированная матрица для вектора X.

В матричных игра при смешанных стратегиях, указывается вероятность с которой выбирается чистая стратегия. Решение задачи в чистых стратегиях также можно рассматривать с помощью вектора вида – единичной длины, так как чистая стратегия является вероятностью 1, а все остальные 0.

Рассмотрим ситуация, когда нам дана платёжная матрица V при размерности $m \times n$, игрок 1 и 2 выбирают свои стратегии $X^T = (x_1, x_2, \dots, x_m)^T$

и $Y^T = (y_1, y_2, \dots, y_n)^T$. Формулировка решения матричной задачи будет заключаться в паре смешанных стратегий (X, Y).

«В решении в смешанными стратегиях ситуация уже не будет реализована так, как это было в чистых стратегиях (i, j). Поскольку, каждая вариация происходит с определенной вероятностью, то она имеет вид $x_i y_j$» [16, с. 100]. Тогда при вероятности $x_i y_j$ выигрыш игрока будет иметь вид v_{ij}, а математическое ожидание выигрыша будет равно:

$$M\varepsilon = \sum_{i=1}^{m} \sum_{j=1}^{n} v_{ij} x_i y_j = X^T V Y. \quad (15)$$

В смешанных стратегиях ситуацию равновесия возьмём как значение (X^*, Y^*), то цена игры будет выглядеть как $h = V(X^*, Y^*) = X^{*T} V Y^*$, при выполнении неравенства:

$$V(X, Y^*) \leq V(X^*, Y^*) \leq V(X^*, Y). \quad (16)$$

Также в смешанных стратегиях, есть теорема фон Нейман, которая отвечает за то, что при оптимальной стратегии игрока 1 его выигрыш и цена игры остается неизменным, при любых стратегиях игрока 2. Рассмотрим данную теорему на платёжной матрице V при размерности $m \times n$, при функции выигрыша $V(X, Y) = X^T V Y$, которая имеет седловую точку при условии

$$\max_X \min_Y X^T V Y, \leq \min_Y \max_X X^T V Y. \quad (17)$$

Таким образом, было показано, что для игр с конечным числом стратегий всегда существует решение в чистых или смешанных стратегиях. После теоретического описания, рассмотрим на практике решение небольших

матричных игр, и выявим более эффективный метод и алгоритм для решения подобных задач.

1.6. Способы решения матричных игр

1.6.1 Аналитическая форма записи для задачи игроков

Самое главное действие в решение матричной игры заключается в нахождении оптимальной стратегии в чистых или смешанных стратегиях. Для этого нужно описать аналитическая форму решения для каждого игрока [20]. В самом начале мы определяем, что игрок 1 должен из всех своих стратегий X, найти ту стратегию, которую можно будет назвать оптимальной X^0 при условия неравенства:

$$X^T v_{(j)} \geq h \text{ для } \forall\, j = 1,2,\dots,n, \tag{18}$$

где $h -$ наибольшее значение из $h(V)$;

$v_{(j)} -$ столбцы платежной матрицы V.

Из вектора $(x_1, x_2, \dots, x_m)^T$ нужно найти , те значения стратегии, которые оптимальные для игрока 1 $X^0 = (x_1^0, x_2^0, \dots, x_m^0)$, и в которые будут доставлять $\max h$ и удовлетворять неравенствам:

$$\sum_{i=1}^{m} v_{ij}\, x_i \geq h \text{ для } \forall\, j = 1,2,\dots,n, \tag{19}$$

$$\sum_{i=1}^{m} x_i = 1,\ x_i \geq 0, \text{для } \forall\, i = 1,2,\dots,m. \tag{20}$$

Аналитическая форма для решения второго игрока говорит о том, что игрок 2 должен из всех своих стратегий Y, найти ту стратегию, которую мы назовем оптимальной Y^0 при условия неравенства:

$$Y^T v^{(i)} \geq h \text{ для } \forall\, i = 1,2,\dots,m, \tag{21}$$

где $h -$ наименьшее значение из $h(V)$;

$v^{(i)} -$ строки платежной матрицы V [20].

Нам нужно найти из вектора $(y_1, y_2, \dots, y_n)^T$,те оптимальные стратегии игрока 2 $Y^0 = (y_1^0, y_2^0, \dots, y_n^0)$, которые будут доставлять $min\, h$ и удовлетворять неравенствам:

$$\sum_{j=1}^{n} v_{ij}\, y_j \le h \text{ для } \forall\, i = 1,2, \dots, m, \quad (22)$$

$$\sum_{j=1}^{n} y_j = 1, y_j \ge 0, \text{для } \forall\, j = 1,2, \dots, n. \quad (23)$$

После того, как мы нашли значения оптимальных стратегий игрока 1 $X^0 = (x_1^0, x_2^0, \dots, x_m^0)$, а также игрока 2 $Y^0 = (y_1^0, y_2^0, \dots, y_n^0)$, можно найти значение игры $h(V) = {X^0}^T H Y^0$.

Представление матричной игры при помощи аналитической формы помогает интерпретировать платежную матрицу в неравенства, чтобы в дальнейшем можно было перейти к сведению ЛПР или визуально показать, как это решение будет выглядеть на графике.

1.6.2 Графический метод решения матричной игры

Чтобы решить матричную игру при помощи графического метода в самом начале нам нужно воспользоваться построением неравенств, как это показано в аналитической форме записи.

Пусть у нас есть V – платежная матрица 2×2 и мы не можем найти оптимальное решение в чистых стратегиях, тогда нам нужно найти оптимальное решение через смешанные стратегии, где (x_1, x_2) удовлетворяют неравенствам:

$$\begin{cases} v_{11}x_1 + v_{21}x_2 \ge h, \\ v_{12}x_1 + v_{22}x_2 \ge h, \\ x_1 + x_2 = 1, \\ x_1, x_2 \ge 0 \end{cases}. \quad (24)$$

Также ее можно представить в ином виде, где у нас $v_{11}x_1 + v_{21}x_2 = h$, $v_{12}x_1 + v_{22}x_2 = h$ и вычислить дальнейшие значения h, x_1, x_2:

$$h^0 = \frac{v_{11}v_{22} - v_{12}v_{21}}{v_{11} + v_{22} - v_{12} - v_{21}}, \quad (25)$$

$$x_1^0 = \frac{v_{22} - v_{21}}{v_{11} + v_{22} - v_{12} - v_{21}}, \quad (26)$$

$$x_2^0 = \frac{v_{11} - v_{12}}{v_{11} + v_{22} - v_{12} - v_{21}}. \quad (27)$$

А если мы решаем графически, то мы считаем, что игрок 1 выбирая стратегии значение будет равно $x_1 = x$, а у игрока 2 значение будет равно $x_2 = 1 - x$, при этом выборе математическое ожидание 2 игрока при стратегии $i = 1$ будет равно:

$$M\varepsilon = h_1^{(1)} = v_{11}x + v_{21}(1 - x), \quad (28)$$

при $i = 2$

$$M\varepsilon = h_2^{(1)} = v_{12}x + v_{22}(1 - x). \quad (29)$$

Аналогично, мы будем находить значения для игрока 2 и его значений.

Пример 2.1. Решим графическим методом платежную матрицу 2×2:

$$V = \begin{pmatrix} 30 & 40 \\ 50 & 20 \end{pmatrix}. \quad (30)$$

В самом начале мы определим находится ли решение в чистых стратегиях $\max_i \min_j v_{ij} = 30$, $\min_j \max_i v_{ij} = 40$, мы видим, что решение будет проходить в смешанных стратегиях и значение игры будет находится в диапазоне $30 \leq h(V) \leq 40$.

Построим графическое решение для 1 игрока (рисунок 3).

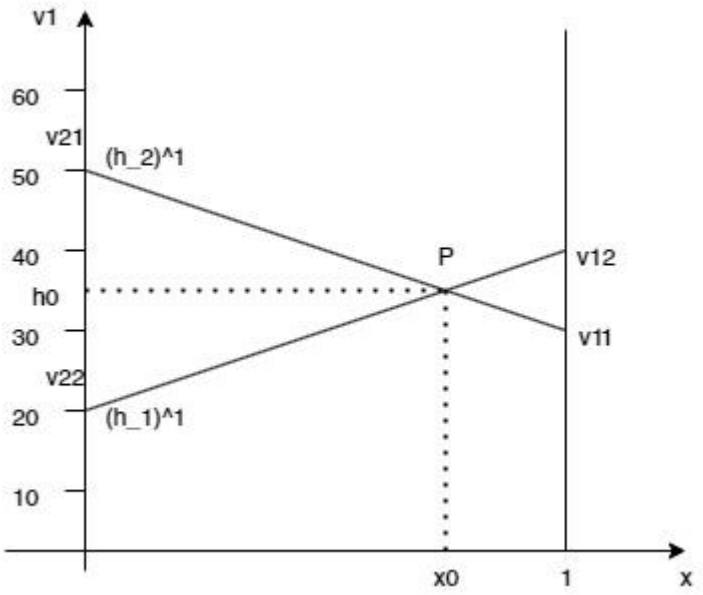

Рисунок 3 – Графическое решение для игрока 1

Точка пересечения P, находится на пересечении $h_1^{(1)}$ и $h_2^{(1)}$, для этого данные значение нужно приравнять и найти точку:

$$30x + 50(1-x) = 40x + 20(1-x) \rightarrow x = \frac{3}{4}. \qquad (31)$$

Получим вероятность для оптимальных стратегий для первого игрока равные $x_1 = x_0 = 0{,}75\,, x_2 = 1 - x_0 = 0{,}25\,,\ h(H) = h_0 = 35.$

Решим задачу для 2 игрока (рисунок 4).

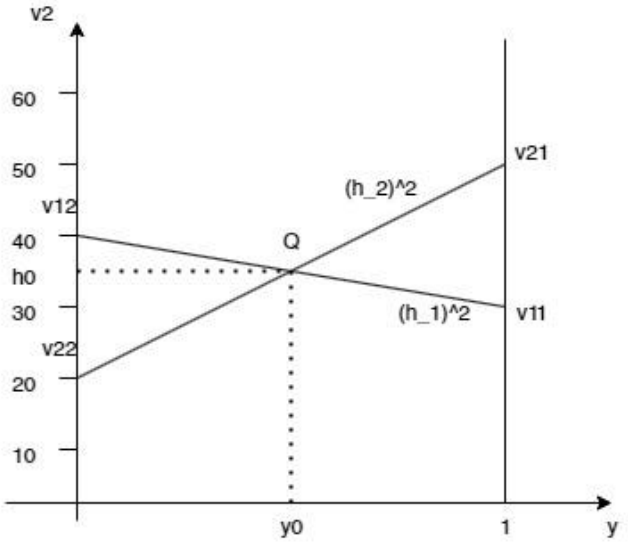

Рисунок 4 – Графическое решение для игрока 2

Точка пересечения Q, находится на пересечении $h_1^{(2)}$ и $h_2^{(2)}$, для этого данные значение нужно приравнять и найти точку:

$$30y + 40(1-y) = 50y + 20(1-y) \rightarrow y = \frac{1}{2}. \qquad (32)$$

Получим вероятность для оптимальных стратегий для второго игрока равные $y_1 = y_0 = 0{,}5\,, y_2 = 1 - y_0 = 0{,}5\,,\ h(H) = h_0 = 35.$

Графический метод очень удобен в решение, когда у нас матрицы небольшого размера, так как это поддаётся решению. Если же мы имеем дело с большой размерностью, то основным инструментом для сокращения размерности и приведения его к более простому виду является доминирование строк и столбцов в матрице.

1.6.3 Доминирование строк и столбцов в платежной матрице

При работе с матрицами у нас бывает большая размерность, с которой будет тяжело справится при решении в графическом методе, а также при решение иных методах. Поэтому платежную матрицу можно сократить. Выберем определенный столбец – k и выполним определенные условия:

$$v_{(k)} \geq \sum_{r=1}^{s} \gamma_r v_{(j_r)}, \quad \sum_{r=1}^{s} \gamma_r = 1, \quad \gamma_r > 0\,, \quad r = 1,2,\dots,s, \qquad (33)$$

где $j_1, j_2, \dots j_s$ −отличные столбцы от k-ого столбца матрицы V.

Если условия совпадают, то считается, что остальные столбцы не являются для 2 игрока существенным и их можно исключить. Это означает, что $v_{(k)}$ столбец доминирует в матрице V среди других столбцов [20].

Выберем определенную строку – k и выполним определенные условия:

$$v^{(k)} \geq \sum_{r=1}^{s} \gamma_r v^{(j_r)}, \quad \sum_{r=1}^{s} \gamma_r = 1, \quad \gamma_r > 0\,, \quad r = 1,2,\dots,s, \qquad (34)$$

где $j_1, j_2, \dots j_s$ −отличные строки от k-ой строки матрицы V.

Если условия совпадают, то считается, что остальные строки не являются для 1 игрока существенным и их можно исключить. Это означает, что $v^{(k)}$ строка доминирует в матрице V среди других строк. Рассмотрим на примере доминирование строк и столбцов в платежной матрице.

Пример 2.2 Уменьшим размерность платежной матрицы 3 × 4:

$$V = \begin{pmatrix} 30 & 10 & 20 & 10 \\ 30 & 20 & 20 & 20 \\ 40 & 50 & 40 & 10 \end{pmatrix}. \qquad (35)$$

Для начала рассмотрим k_1 строку и будем сравнивать с остальными строками . Значения k_1 строки $\leq$ k_2 строки, а также значения k_1 строки $\leq k_3$ строки, поэтому k_1 доминирует и её можно исключить, также следует, что вероятность стратегии $y_1 = 0$. Рассмотрим строку k_2, она явно на один элемент не меньше, значит она существенная и мы не можем её убрать:

$$V = \begin{pmatrix} 30 & 20 & 20 & 20 \\ 40 & 50 & 40 & 10 \end{pmatrix}. \qquad (36)$$

Перейдем к значению столбцов, k_4 столбец $\leq k_3$ столбца, а также k_4 столбец $\leq k_2$ столбца, k_4 столбец $\leq k_1$ столбца. Значит данный столбец доминирует, и мы его можем исключить. Теперь обратимся к следующему столбцу k_3, где значения k_3 столбец $\leq k_2$ столбца, k_3 столбец $\leq k_1$ столбца. И мы также имеем дело с доминируемым столбцом и можем его исключить, а вероятность стратегий будет равна $x_3 = 0, x_4 = 0$.

Столбцы k_1 и k_2 существенны и мы их не можем удалить и конечный результат платежной матрицы выглядит следующим образом:

$$V = \begin{pmatrix} 30 & 40 \\ 50 & 20 \end{pmatrix}. \quad (37)$$

Так мы свели задачу к платежной матрице размерности 2×2 , данная платежная матрица решена в примере 1.1 с помощью графического метода и ответы оптимальных стратегий равны: $x_1 = 0{,}75\,, x_2 = 0{,}25\,, x_3 = 0\,, x_4 = 0$ $h(H) = 35;$ $y_1 = 0,$ $y_2 = 0{,}5\,, y_3 = 0{,}5\,,$ $h(H) = 35.$

Таким образом, анализ доминирования строк и столбцов является одним из вариантов упрощения матричной игры, позволяя сократить число возможных стратегий игроков и тем самым ускорить процесс нахождения оптимальных решений.

1.6.4 Решение матричной игры при помощи задачи ЛПР

Если платежная матрица имеет большую размерностью и графическое решение является не эффективным, даже после доминирования строк, тогда нашу аналитическую форму мы сводим к решению ЛПР [3]. Для этого нужно задать вероятности игрока 1, с которыми он будет ходит по стратегиям $(p_1, p_2, \dots, p_m)$, которые будут удовлетворять неравенствам:

$$\sum_{i=1}^{m} v_{ij}\, p_i \geq h \text{ для } \forall\, j = 1,2, \dots, n, \quad (38)$$

$$\sum_{i=1}^{m} p_i = 1, p_i \geq 0, \text{для } \forall\, i = 1,2, \dots, m. \quad (39)$$

При соблюдении этих неравенств мы найдем оптимальные стратегии $P = ({p_1}^0, {p_2}^0, \dots, {p_m}^0)$, доставляющая значения $\max(h)$ [16].

Упростим решение неравенств при помощи замены значения $x_i = \frac{p_i}{h}$, тогда значения $p_i = x_i h \; \forall \; i = 1,2, \dots, m$:

$$\sum_{i=1}^{m} v_{ij}\, p_i \geq v = h \sum_{i=1}^{m} v_{ij}\, x_i \geq h = \sum_{i=1}^{m} v_{ij}\, x_i \geq 1 \text{ для } \forall\, j = 1,2, \dots, n, \quad (40)$$

$$\sum_{i=1}^{m} p_i = 1 = h \sum_{i=1}^{m} x_i = 1 = \sum_{i=1}^{m} x_i = \frac{1}{h}, x_i \geq 0, \text{для } \forall\, i = 1,2, \dots, m. \quad (41)$$

Используем $\max\left(\frac{1}{h}\right) = \min h$, поэтому мы получим задачу для решения $X = (x_1, x_2, \dots, x_m)$ при неравенствах:

$$\sum_{i=1}^{m} v_{ij}\, x_i \geq 1 \text{ для } \forall\, j = 1,2, \dots, n, \quad (42)$$

$$x_i \geq 0, \text{для } \forall\, i = 1,2, \dots, m. \quad (43)$$

Найдем оптимальные стратегии $X = ({x_1}^0, {x_2}^0, \dots, {x_m}^0)$, при условии:

$$\sum_{i=1}^{m} x_i \to min. \quad (44)$$

Вернемся к замене и найдем значение игры, $h(V) = \frac{1}{\sum_{i=1}^{m} {x_i}^0}$, тогда вероятность оптимальных стратегий, которые выбирает игрок будет равна:

$$p_i = {x_i}^0 * \; h(V) \; \forall \; i = 1,2, \dots, m. \quad (45)$$

Аналогично найдем оптимальные вероятности для игрока 2, он которыми он будет ходит по стратегиям $(q_1, q_2, \dots, q_n)$, которые будут удовлетворять неравенствам:

$$\sum_{j=1}^{n} v_{ij}\, q_j \leq h \text{ для } \forall\, i = 1,2, \dots, m, \quad (46)$$

$$\sum_{j=1}^{n} q_j = 1, q_j \geq 0, \text{для } \forall\, j = 1,2,\dots,n. \quad (47)$$

При соблюдении этих неравенств мы найдем оптимальные стратегии $Q = ({q_1}^0, {q_2}^0, \dots, {q_n}^0)$, доставляющая значения $\min(h)$ [16].

Упростим решение неравенств при помощи замены значения $y_j = \frac{q_j}{h}$, тогда значения $q_j = y_j h \; \forall\, j = 1,2,\dots,n$:

$$\sum_{j=1}^{n} v_{ij}\, q_j \leq h = h \sum_{j=1}^{n} v_{ij}\, y_j \leq h = \sum_{j=1}^{n} v_{ij}\, y_j \leq 1 \text{ для } \forall\, i = 1,2,\dots,m, \quad (48)$$

$$\sum_{j=1}^{n} q_j = 1 = h \sum_{j=1}^{n} y_j = 1 = \sum_{j=1}^{n} y_j = \frac{1}{h}, y_j \geq 0, \text{для } \forall\, j = 1,2,\dots,n. \quad (49)$$

Используем $\min h = \max\left(\frac{1}{h}\right)$, поэтому мы получим задачу для решения $Y = (y_1, y_2, \dots, y_n)$ при неравенствах:

$$\sum_{j=1}^{n} v_{ij}\, y_j \leq 1 \text{ для } \forall\, i = 1,2,\dots,m, \quad (50)$$

$$y_j \geq 0, \text{для } \forall\, j = 1,2,\dots,n. \quad (51)$$

Найдем оптимальные стратегии $Y = ({y_1}^0, {y_2}^0, \dots, {y_n}^0)$, при условии:

$$\sum_{j=1}^{n} y_j \to max. \quad (52)$$

Вернемся к замене и найдем значение игры, $h(V) = \frac{1}{\sum_{j=1}^{n} {y_j}^0}$, тогда вероятность оптимальных стратегий, которые выбирает игрок будет равна:

$$q_j = {y_j}^0 * \; h(V) \; \forall\, j = 1,2,\dots,n. \quad (53)$$

Таким образом, для более сложной платежной матрицы одним из способов решения является сведение задачи к ЛПР. Поскольку, в решение

матричных игр есть множество способов решения, одним из них является решение с позиций метода анализа иерархий. Рассмотрим на сколько этот способ эффективный в отличие от классического решения.

1.6.5 Решение матричной игры при помощи метода анализа иерархии

Это метод рассматривает применение МАИ для решения матричных игр. Данный подход позволяет не только найти оптимальные стратегии игроков, но и оценить относительную важность различных стратегий, влияющих на исход игры. «Описание подобных ситуаций в рамках матричной игры достаточно просто вкладывается в определенную схему метода анализа иерархий (МАИ), широко известного метода принятия многокритериальных решений » [8, с. 162].

Для решения матричной игры с позиции МАИ в самом начале важно оценить стратегии первого и второго игрока с точки зрения иерархии. Рассмотрим общий вид решения с точки зрения иерархии для игрока 1 при платежной матрице размерностью $m \times n$. Первый этап формирует m матриц парных сравнений (PI) для стратегий $i..n$ относительно стратегий $j..m$ (с использованием шкалы парных сравнений). Данные матрицы являются обратно симметричными:

$$PI = \begin{pmatrix} 1 & \cdots & a_{1m} \\ \cdots & \cdots & \cdots \\ \frac{1}{a_{1m}} & \cdots & 1 \end{pmatrix}, \quad (54)$$

где a_{1m} − интенсивная относительная важность.

Чтобы правильно рассчитать нормированные веса, приведем каждую матрицу к нормированному виду. «Разделим элементы каждого столбца на сумму элементов этого столбца (то есть нормализируем столбец) и запишем нормированную матрицу N. Затем сложим элементы каждой полученной

строки матрицы N и разделим эту сумму на число элементов строки и запишем полученные нормированные веса в матрицу V » [8, с. 163] :

$$N = \begin{pmatrix} \frac{1}{1+\frac{1}{a_{1m}}} & \cdots & \frac{a_{1m}}{1+a_{1m}} \\ \cdots & \cdots & \cdots \\ \frac{\frac{1}{a_{1m}}}{1+\frac{1}{a_{1m}}} & \cdots & \frac{1}{1+a_{1m}} \end{pmatrix}, \quad (55)$$

$$V = \begin{pmatrix} \frac{\frac{1}{1+\frac{1}{a_{1m}}}+\cdots+\frac{a_{1m}}{1+a_{1m}}}{m} \\ \cdots \\ \frac{\frac{\frac{1}{a_{1m}}}{1+\frac{1}{a_{1m}}}+\cdots+\frac{1}{1+a_{1m}}}{m} \end{pmatrix}. \quad (56)$$

Так производится расчет нормированных весов для стратегий $i..n$ относительно стратегий $j..m$, которые фиксируются в матрицу W_{1-2}. Далее, при использовании метода анализа иерархий (МАИ) в решении матричных игр, можно применить известную технику формирования линейных взвешенных оценок. Эта техника позволяет вычислить вектор-столбец $W_{1-\text{ПР}}$ нормированных весов для стратегий игрока 1 относительно фокуса «Принятие решения» (ПР), исходя из иерархической структуры, которая описывалась на первом этапе:

$$W_{1-\text{ПР}} = W_{1-2} * W_{2-\text{ПР}}, \quad (57)$$

где $W_{2-\text{ПР}}$ − вектор-столбец нормированных весов для стратегий $j..m$ относительно фокуса «Принятие решения».

Аналогичным образом происходит решение для двойственной иерархии второго игрока, где при схожих вычислениях мы находим вектор-столбец $W_{2-\text{ПР}}$ нормированных весов для стратегий игрока 2.

В ходе вычисления данные нормированные веса можно рассматривать как вероятности выбора стратегий игрока, что схоже с решением матричной игры при помощи метода ЛПР. Только при помощи классического решения получаются более точные значения, поскольку «в методе МАИ есть особенности в формировании матриц парных сравнений на основе экспертных оценок и эти особенности отражают условия неопределенности для реальных ситуаций» [8, с. 165].

Таким образом, матричную игру можно решать при помощи различных инструментов, но точность решения будет меняться с учетом специфики этого метода. Поэтому после анализа выявилось, что наиболее эффективным и точным методом решения является сведение задачи к ЛПР.

Выводы по первому разделу

В данном разделе мы рассмотрели экономическую задачу для автомобильного завода, была обоснована актуальность данной проблемы и выявлен вид игры из теории игр, который можно применить при решении данной задачи. Подробно изучены матричные игры, их классификация на игры в чистых и смешанных стратегиях. Рассмотрены условия существования ситуации равновесия в таких играх. Были описаны различные методы решения матричных игр: аналитическая форма записи, графический метод, анализ доминирования строк и столбцов, решение задачи при помощи метода анализа иерархии, а также сведение к задаче линейного программирования. Рассмотренные в данном разделе теоретические основы матричных игр создают необходимую базу для дальнейшего практического применения теории игр в решение поставленной задачи.

2 Проектирование программы для решения поставленной задачи

2.1 Общая структура программного кода

Для того, чтобы спроектировать данный программный продукт, остановимся на выборе языка программирования. Я остановилась на объектно-ориентированном языке программирования – Java, его основными преимуществами в отличии от других языков является:

- кроссплатформенность, есть возможность работать на разных ОС без перекомпиляции;
- безопасность, помогает предотвратить уязвимости;
- надежность, автоматическое управление памятью, что снижает количество ошибок;
- масштабируемость, подходит для разработки крупномасштабных приложений;
- производительность, благодаря технологии Just-In-Time (JIT) компиляции, Java обеспечивает высокую производительность [21];
- простота, в отличие от тяжелых языков программирования (C++), является простым языком для изучения.

Программа будет разработана в среде разработки Eclipse Luna. Доступная и обширная IDE для разработки различных приложений [13] . Для поставленной задачи было выбрано консольное приложение – это программа, которая работает через командную строку среды разработки и представляет все выводы происходят в текстовых режимах. Консольные приложения часто используются для автоматизации задач, управления операционной системой, обработки данных и других задач, не требующих графического интерфейса. Для математического вычисления и преобразования поставленной задачи в информационный вид, данный тип приложения является самым подходящий.

Для разработки консольных приложений на Java в среде разработки Eclipse использовались следующие подходы:

1) создание Java Project, а также создание основного класса с методом main(), который будет точкой входа в приложение [21];
2) определение набора библиотек, которые будут использоваться в приложение:
   - import java.util.Scanner, для чтения ввода пользователя в консоли используется данный класс;
   - import static java.lang.Math.pow, это статический класс отвечающий за степень, в которую возводит число, нужен для читаемости и упрощения кода;
   - import static java.lang.System.out, этот импортируемый класс позволяет использовать out без необходимости указывать полное имя;
3) определение основного класса и описание полностью зависимости других созданных классов, если они будут, составление диаграммы и построение алгоритма программы.

Таким образом, в данном разделе я определилась с выбором языка и выбором среды разработки, описала поэтапно структуру создания консольного приложения, теперь для полного проектирования нужно создать диаграмму классов и описать каждый из них.

2.2 Проектирование программного кода

Чтобы смоделировать программу нам нужно построить диаграмму классов. Она заключается в том, предоставляет набор графических обозначений, которые позволяют разработчикам создавать диаграммы для понимания, проектирования и документирования структуры и поведения системы. Диаграмма UML – это построение классов и описание их взаимодействий между собой. Каждый класс имеет свои атрибуты, операции.

Атрибуты – это экземпляр класса, который описывает диапазон значений, в которых может находится этот экземпляр.

Операции – это методы класса, которые обычно могут изменять атрибуты.

Взаимодействие между классами происходит при помощи различных отношений, такие как:

– зависимость, данная связь, говорит о том, что изменение данных у независимого объекта ведет к изменению данных зависимого объекта;
– ассоциация, данная связь, говорит о том, что объекты одного класса связаны с объекта другого класса, так что их можно перемещать между собой;
– наследование, данная связь, говорит о том, что элемент потомок будет строится по тем же спецификациям, как и у родителя.

UML – диаграмма, для построение моего программного кода выглядит следующим образом, обратимся к рисунку 5.

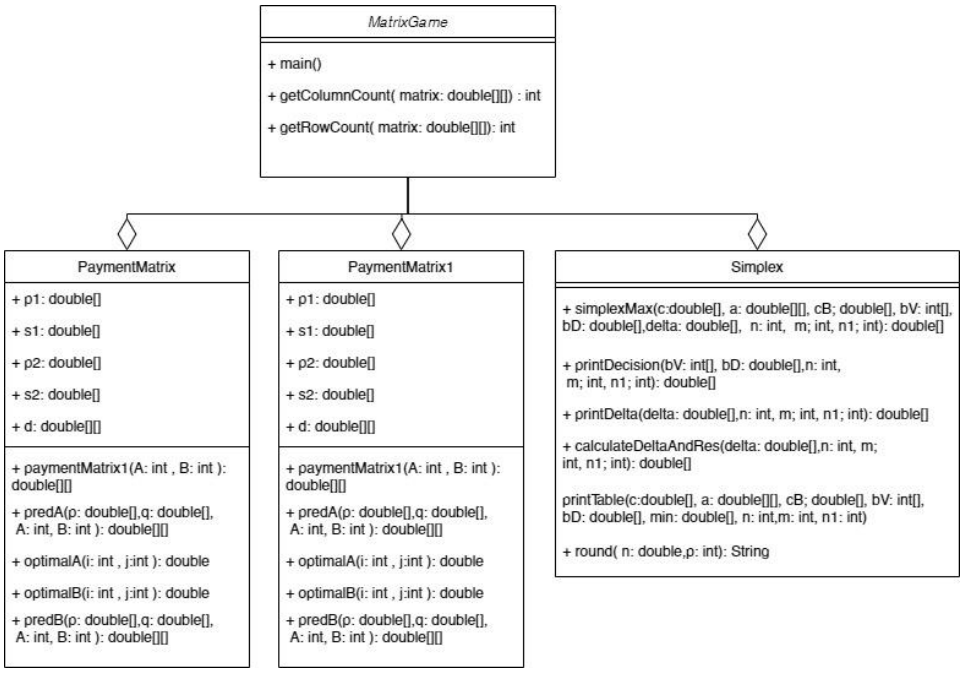

Рисунок 5 – Диаграмма классов для программного кода

В диаграмме присутствует 4 класса. Основным классом является MatrixGame – он использует функциональность всех объектов других классов для выполнения своих собственных задач, таких как вывод результатов и определение равновесия по Нэшу. PaymentMatrix и PaymentMatrix1 предоставляют функциональность для создания платежной матрицы и нахождения оптимальных стратегий для игроков A и B. Это основная функциональность, необходимая для игры. Simplex предоставляет функциональность для решения задач линейного программирования при помощи симплекс таблицы. Это используется для расчета выигрышей игроков при заданных стратегиях. Все классы связанны с помощью агрегации, так как она позволяет разделить функциональность на более мелкие компоненты, которые можно легко использовать повторно или заменять по мере необходимости [21] . То есть при замене класса PaymentMatrix другим классом, который предоставляет ту же функциональность, без необходимости переписывать код класса MatrixGame. Аналогично и для других присутствующих классов. В целом, агрегация - это подходящий тип зависимости в данной ситуации, поскольку она позволяет создавать более гибкие и модульные программы.

Воспользуемся графическим изображением последовательности выполнения программы, для того, чтобы посмотреть какие шаги предпринимает пользователь, который начал работу с ней. Для этого воспользуемся блок-схемой алгоритма программы, представленный на рисунке 6.

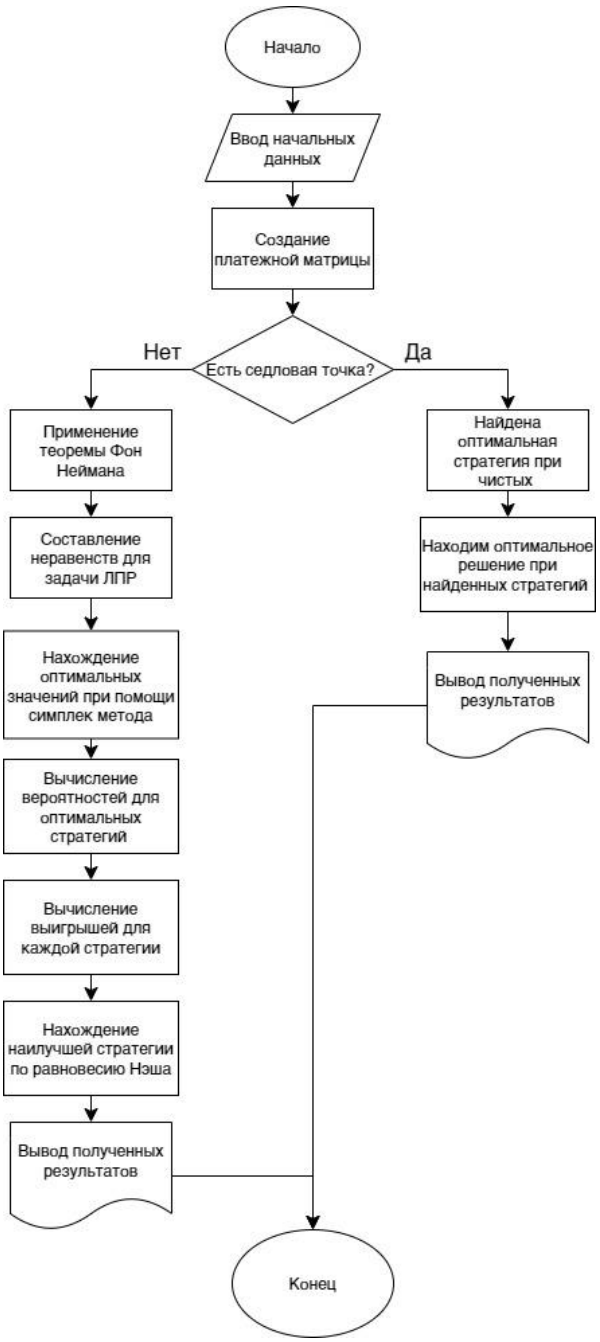

Рисунок 6 – Блок-схема алгоритма программы

Таким образом, был представлен алгоритм программы и классы, которые фигурируют в нем. Использование диаграмм классов и блок-схем алгоритмов в процессе разработки программного обеспечения позволяет более эффективно планировать, проектировать и реализовывать программные системы. Также стоит отметь, что для полного обзора проектирования нужно разобрать каждый класс, который был представлен в диаграмме.

2.2.1 Проектирование класса для ввода данных и составления платежной матрицы

Класс, отвечающий за ввод данных, является одним из главных, поскольку именно в нем происходит построение платежной матрицы с помощью, которой решается поставленная задача. В программе есть два класса, которые похожи по функциональности и вычислению, только класс PaymentMatrix, отвечает за ввод данных через клавиатуры, следовательно, все данные мы можем задать сами, а класс PaymentMatrix1 работает уже с готовыми данными, которые будут даны в примере.

Основные атрибуты классов:

- p1, цена реализации на выходе за автомобиль при выборе стратегии игрока 1;
- p2, цена реализации на выходе за автомобиль при выборе стратегии игрока 2;
- s1, себестоимость за единицу автомобиля при выборе стратегии игрока 1;
- s2, себестоимость за единицу автомобиля при выборе стратегии игрока 2;
- d, доля автомобилей игрока 1, купленной населением при ценах p1 и p2.

Основные операции классов:

- paymentMatrix(), отвечает за построение платежной матрицы, при помощи математической модели поставленной задачи;
- predA(), отвечает за вычисление выигрыша игрока 1 при оптимальной вероятности 1 игрока, чтобы в дальнейшем оценить его оптимальную стратегию;
- predB(), отвечает за вычисление выигрыша игрока 2 при оптимальной вероятности 2 игрока, чтобы в дальнейшем оценить его оптимальную стратегию;

– optimalA(), отвечает за вычисление количества прибыли игрока 1 при его оптимальных стратегиях;
– optimalB(), отвечает за вычисление количества прибыли игрока 2 при его оптимальных стратегиях;

Для того, чтобы в методе paymentMatrix() вычислить все элементы платежной матрицы, нужно составить математическую модель для подсчёта. Исходя из поставленной задачи, мы находим платежную матрицу для игрока 1, значит элементы в ней — это прибыль 1–го игрока минус прибыль 2–го игрока. Прибыль же будет находится по обычной формуле доходы минус расходы, оба значения зависят от количества купленной населением автомобилей с помощью формулы спроса (описанной в 1 разделе)

Значит математическая модель для подсчёта будет выглядеть следующим образом:

$$\left(9{,}8 - 0{,}4 * \left(\frac{p1 + p2}{2}\right)\right) * 1000 * \left(\mathrm{d} * (\mathrm{p1} - \mathrm{s1}) - (1 - \mathrm{d}) * (\mathrm{p2} - \mathrm{s2})\right). \quad (58)$$

Данная формула будет фигурировать не только в методе paymentMatrix(), но и во всех остальных методах, только для них будут иные условия подсчета.

Таким образом, были рассмотрены детально классы, отвечающие за ввод данных и создание платежной матрицы, на основе математической модели, составленной исходя заданному заданию.

2.2.2 Проектирование класса для решения ЛПР при помощи симплекс-метода

После анализа способов решения матричной игры в смешанных стратегиях самым эффективным является сведение задачи к ЛПР. Для этого построим двойственную задачу для нахождения оптимальных стратегий. Составим симметричную пару двойственной задачи, для примера возьмем матрицу коэффициентов из платежной матрицы с размерностью 3×3:

$$F(x) = x_1 + x_2 + x_3 \to min,$$

$$\begin{cases} a_{11}x_1 + a_{21}x_{2,} + a_{31}x_3 \geq 1 \\ a_{21}x_1 + a_{22}x_{2,} + a_{23}x_3 \geq 1 \\ a_{31}x_1 + a_{32}x_{2,} + a_{33}x_3 \geq 1, \\ x_1, x_{2,} x_3 \geq 0 \end{cases} \quad (59)$$

$$Z(x) = y_1 + y_2 + y_3 \to max,$$

$$\begin{cases} a_{11}y_1 + a_{12}y_2 + a_{13}y_3 \leq 1 \\ a_{21}y_1 + a_{22}y_2 + a_{23}y_3 \leq 1 \\ a_{31}y_1 + a_{32}y_2 + a_{33}y_3 \leq 1 \\ y_1, y_2, y_3 \geq 0 \end{cases}. \quad (60)$$

Решаем задачу по нахождению максимума при помощи симплекс метода, где основным инструментом решения является симплекс таблица. Приведем данную систему к каноническому виду:

$$Z(x) = y_1 + y_2 + y_3 \to max,$$

$$\begin{cases} a_{11}y_1 + a_{12}y_2 + a_{13}y_3 + y_4 = 1 \\ a_{21}y_1 + a_{22}y_2 + a_{23}y_3 + y_5 = 1 \\ a_{31}y_1 + a_{32}y_2 + a_{33}y_3 + y_6 = 1 \end{cases}. \quad (61)$$

Вводим базисные переменные y_4, y_5, y_6. И составляем первый опорный план, где свободные переменные равны 0, а базисы равны 1. Составляем симплекс таблицу для решения данного метода, при опорном плане (0,0,0,1,1,1). Основной задачей симплекс метода – является нахождения оптимального плана, когда значения из целевой функции будут положительными.

Обозначим основные переменные, которые будут фигурировать в операциях класса Simplex:

- c, коэффициенты из целевой функции;
- a, коэффициенты из ограничений;
- cB, коэффициенты при базисных переменных;
- bV, базисные переменные;

– bD, базисное решение.

Основные операции класса Simplex:

– simplexMax (), отвечает за реализацию симплекс-метода для решения ЛПР с целью нахождения максимума целевой функции;
– printDecision(), отвечает за вывод найденного оптимального плана после завершения симплекс метода;
– printTable(), отвечает за вывод таблицы симплекс метода;
– calculateDeltaAndRes (), отвечает за нахождения двойственной задачи, через теорему о двойственности;
– round (), отвечает за форматирование числа, до заданной точности и выводит значение в виде строки для визуализации результатов в таблице.

Разберем более подробно алгоритм решения, где описывается пошаговое выполнение симплекс-метода для поиска максимума. Он включает в себя вычисление дельт, определение разрешающей строки, обновление базисного решения и вывод промежуточных и итоговых результатов (рисунок 7).

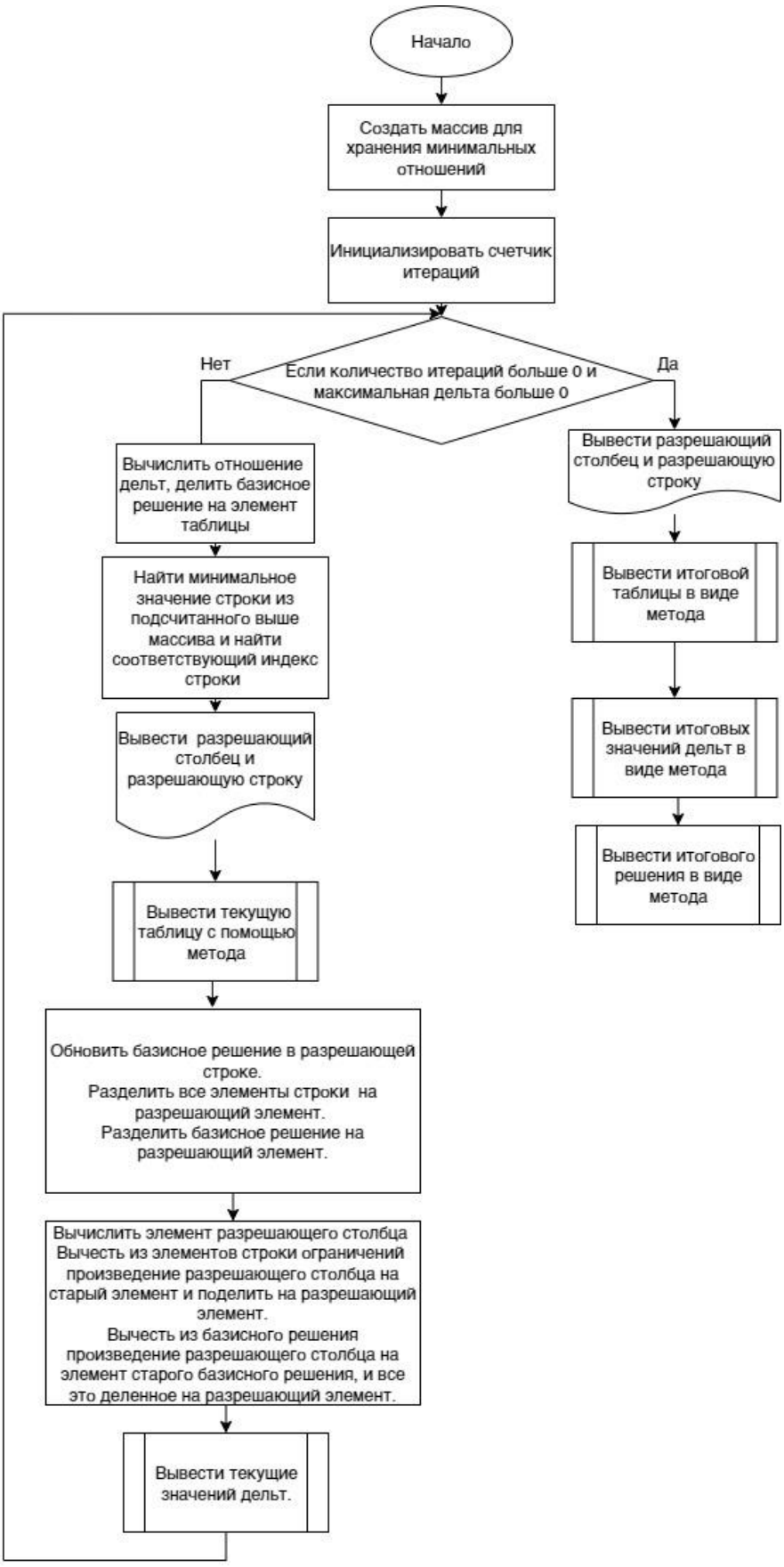

Рисунок 7 – Алгоритм для решения симплекс метода

Чтобы найти двойственную задачу для минимума, используем последнюю итерацию прямой задачи и с помощью теоремы двойственности найден значения оптимального плана для нахождения минимума (рисунок 8).

Рисунок 8 – Алгоритм нахождения минимума целевой функции

Таким образом, в данном класса спроектирован алгоритм нахождения оптимальных значений, с помощью которых можно перейти к следующему шагу и найти оптимальные вероятности хода игрока определенной стратегией.

2.2.3 Проектирование основного класса

Программа имеет основной класс, в котором описывается большинство действий. Этот класс отвечает за последовательность выполнения каждого фрагмента кода и за связь между классами, описанными выше. Данный класс является основным, потому что в нем определен метод main(), который

является точкой входа в приложение, остальные классы не имеют данного метода [22]. Для начала мы должны определиться с размерностью матрицы и задать нужные данные, чтобы составить платежную матрицу, как говорилась выше это делается с помощью классов PaymentMatrix и PaymentMatrix1 (рисунок 9).

Рисунок 9 – Алгоритм. ввода данных в основном классе

После переходим к расчёту седловой точки, и определяем в каким стратегиях будет решаться задача: чистых или смешанных. Если задача оказалась легкой, то решение находится в чистых стратегиях (рисунок 10).

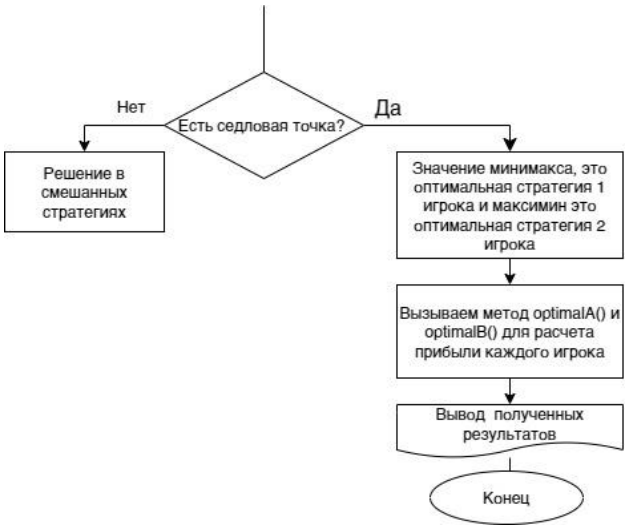

Рисунок 10 – Алгоритм решения в чистых стратегиях

Для решения задачи в смешанных стратегиях используется способ сведения задачи к ЛПР и применение класса Simplex. Для того чтобы перейти к следующему этапу решения, мы сводим платежную матрицу к более простому виду при помощи теоремы Фон Неймана и далее уже находим значения вероятности оптимальных стратегий каждого игрока.

Так как каждый игрок разумный и делает ход так, чтобы было выгодно для него, важно определить оптимальную стратегию для каждой стороны. Для этого можно воспользоваться равновесием по Нэшу, здесь «каждый игрок выбирает свою стратегию таким образом, что ни один из них не может улучшить свой результат, изменив свою стратегию в одностороннем порядке» [2, с. 11]. Для начала нужно вычислить ожидаемый выигрыш, вычисленный при помощи классов PaymentMatrix и PaymentMatrix1, каждого игрока для каждой комбинации стратегий при оптимальных вероятностях (рисунок 11).

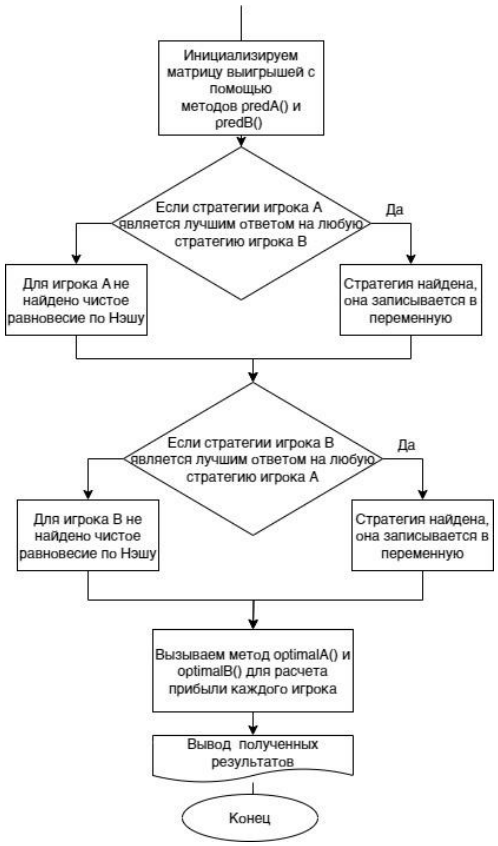

Рисунок 11 – Алгоритм для нахождения прибыли по равновесию Нэша

Таким образом, был описан основной класс для решения поставленной задачи, показано как он взаимодействует с другими классами и какие методы были вызваны. Также более подробно разобрано как решается задача в смешанных стратегиях и для чего применяется равновесие по Нэшу в данной ситуации.

Вывод по второму разделу

В данном разделе я определилась на каком языке программирования будет реализован мой проект, были разобраны основные детали, с которых важно начать проектировать программу. Также рассмотрено важность применения диаграммы классов и написана собственная диаграмма для решения поставленной задачи. Для четкого понятия как работает программа был описан поэтапный алгоритм действия и разобран на основные моменты, которые подробно были описанные в разделах. Данное проектирование проекта поможет в дальнейшем правильно и структурированно реализовать программу и протестировать её на примере.

3 Реализация программы и тестирование на примере экономической задачи для автомобильного завода

3.1 Программная реализация приложения

После проектирования проекта и выбора языка перейдем к реализации программы. Для реализации используется синтаксис языка программирования Java. Для реализации данной программы, я задаю собственную размерность исходя из данных, которые были даны в пункте 1.1, «чтобы правильно осуществить выбор ввода данных я воспользовалась оператором управления потоков – switch()» [13, с. 56]. Он позволяет совершить выбор из нескольких вариантов при вводе значения. Классы PaymentMatrix и PaymentMatrix1, отвечающие за ввод данных вызываются как объекты в основном классе MatrixGame (рисунок 12).

```
System.out.println("1. Ввод новых данных");
System.out.println("2. Ввод готовых данных");
int choice =scanner.nextInt();
switch (choice) {
case 1:
    PaymentMatrix matrix = new PaymentMatrix();
    paymentMatrix = matrix.paymentMatrix(n,m);
    break;
case 2:
    PaymentMatrix1 matrix1 = new PaymentMatrix1();
    paymentMatrix = matrix1.paymentMatrix1(n,m);
    break;
default:
    System.out.println("Неверный выбор");
    break;
}
```

Рисунок 12 – Листинг вызова объектов классов PaymentMatrix и PaymentMatrix1

После того как мы получили платежную матрицу в зависимости от нашего выбора, следующим этапом является нахождение седловой точки, задаются значения minimax, maximin, а сам фрагмент нахождения этих точек, представлен на рисунке 13.

```
double[] mins = new double[n]; // минимакса
double[] maxs = new double[m]; // максимина
for (int i = 0; i < n; i++) {
    mins[i] = paymentMatrix[i][0];
    for (int j = 0; j < m; j++) {
        if (paymentMatrix[i][j] < mins[i]) {
            mins[i] = paymentMatrix[i][j];
        }
    }
    if (mins[i] > mins[mini]) {
        mini = i;
    }
}
minimax =mins[mini];
for (int j = 0; j < m; j++) {
    maxs[j] = paymentMatrix[0][j];
    for (int i = 0; i < n; i++) {
        if (paymentMatrix[i][j] > maxs[j]) {
            maxs[j] = paymentMatrix[i][j];
        }
    }
    if (maxs[j] < maxs[maxj]) {
        maxj = j;
    }
}
maximin=maxs[maxj];
```

Рисунок 13 – Фрагмент кода по нахождению максимина и минимакса

На рисунке 14 представлен фрагмент, когда мы переходим к решению в смешанных стратегиях и упрощаем нашу матрицу с помощью теоремы Фон Неймана, чтобы найти наименьший элемент в матрице, я воспользовалась константой Double.MAX_VALUE - это самое большое положительное число, которое может быть представлено в типе данных double в Java [22].

```
//переводим матрицу по теореме фон Неймана

double min_value = Double.MAX_VALUE;
for (int i = 0; i < paymentMatrix.length; i++) {
    for (int j = 0; j < paymentMatrix[i].length; j++) {
        min_value = Math.min(min_value, paymentMatrix[i][j]);
    }
}

// Прибавляем минимальное значение по модулю ко всем остальным элементам
for (int i = 0; i < paymentMatrix.length; i++) {
    for (int j = 0; j < paymentMatrix[i].length; j++) {
        paymentMatrix[i][j] += Math.abs(min_value);
    }
}
```

Рисунок 14 – Фрагмент кода с применением теоремы Фон Неймана

Следующий шаг в выполнении – это правильное сведение матричной игры к решению задачи ЛПР, для этого нужно правильно задать значения, применяемые в симплекс таблице, смотрите на рисунок 15.

```
System.out.print("_____\n");
System.out.println("Построим симплекс таблицу");
for (int i = 0; i < basic; i++) {
   if (i<m){
   c[i] =1;
   }
   else {
   c[i]=0;
   }
}

for (int i = 0; i < n; i++) {
      for (int j = 0; j < basic; j++) {
          if (j < m) {
              matrixAd[i][j] = paymentMatrix[i][j];
          } else {
              if (i == j - m) {
                  matrixAd[i][j] = 1;
              } else {
                  matrixAd[i][j] = 0;
              }
          }
      }
  }
for (int i = 0; i < n; i++) {
   cB[i] =0;
   bV[i]=m+i;
   bD[i] = 1;
}
```

Рисунок 15 – Фрагмент кода, где вычисление коэффициентов для симплекс таблицы

На рисунке 16 представлен фрагмент кода, где мы вызываем методы simplexMax(), calculateDeltaAndRes(), они предназначены для нахождения решения x – double[] xx , решение y – double[] yy .

```
Simplex simplexMethod = new Simplex();
yy = simplexMethod.simplexMax(c, matrixAd, cB, bV, bD,delta,basic,m,n);
xx = simplexMethod.calculateDeltaAndRes(delta,basic,m,n);
```

Рисунок 16 – Фрагмент кода с вызовом методов решения ЛПР

Метод simplexMax() в классе Simplex решает задачу ЛПР при помощи симплекс таблицы, на рисунке 17 показано как мы находим значения для разрешающего столбца, для этого воспользуемся константой Double.NEGATIVE_INFINITY – это особое значение, которое не равно

любому другому числу, включая – Double.MAX_VALUE [22]. Оно используется для представления ситуаций, когда значение выходит за пределы диапазона.

```
double[] simplexMax(double[] c, double[][] a, double[] cB, int[] bV,
        double[] bD,double[] delta, int n, int m, int nl) {
    double[] min = new double[m];

    out.println("\n\nПоиск максимума\n\n");
    int i, j, r = -1, s = -1;
    double[][] tmpA = new double[m][];
    double[] tmpBD = new double[m];

    int k = 0;
    while (true) {
        ++k;

        // Считаем дельты, выделяем разрешающий столбец
        // (столбец с максимальной оценкой)
        double deltaMax = Double.NEGATIVE_INFINITY;
        r = -1;
        for (j = 0; j < n; ++j) {
            double z = 0;
            for (i = 0; i < m; ++i) {
                z += cB[i] * a[i][j];
            }
            delta[j] = c[j] - z;
            if (deltaMax < delta[j]) {
                deltaMax = delta[j];
                r = j;
            }
        }
```

Рисунок 17 – Фрагмент метода simplexMax()

Чтобы правильно найти оптимальную стратегию по равновесию Нэша вычисляем при помощи методов predA(), predB() выигрыши каждого игрока при определенных стратегиях, для этого используем изначально формулу, по которой считали платежную матрицу, только добавляем вероятности: double[] p, double[] q (рисунок 18).

```
double[][] predB(double[] p, double[] q, int A, int B){
    double[][] winB = new double[A][B];
    for (int i = 0; i < A; i++) {
          for (int j = 0; j < B; j++) {
              winB[i][j] = (9.8 - 0.4 * ((p[i]*p1[i] + q[j]*p2[j]) / 2))
                      * 1000 * d[i][j] * (1 - d[i][j]) * (q[j]*p2[j] - q[j]*s2[j]) ;
          }
    }
    return  winB;
}
```

Рисунок 18 – Фрагмент кода метода predB()

Для того, чтобы увидеть результат находим воспользуемся алгоритмом нахождения по равновесию Нэша оптимальной стратегии. Для этого введем флаг isBestResponseA, данная переменная будет помогать определять, когда нужная стратегия будет найдена (рисунок 19).

```
for (int i = 0; i < n; i++) {
     boolean isBestResponseA = true;
     for (int j = 0; j < m; j++) {
         if (winA[i][j] < winA[k][j]) {
             isBestResponseA = false;
             break;
         }
     }
     if (isBestResponseA) {
         k = i;
         System.out.println("Равновесие по Нэшу для игрока A - выбор стратегии A[" + k + "]");
     }
 }
```

Рисунок 19 – Фрагмент кода, где реализовано нахождение оптимальной стратегии

Таким образом, была реализована программа со всеми входящими в неё классами, выбраны нужные переменные и написаны методы в соответствии с проектированным алгоритмом из 2 раздела. Представленный готовый проект нужно протестировать на готовых данных, которые относятся к экономики автомобильного завода.

3.2 Тестирование программы на примере автомобильного завода

Результатом тестирования консольного приложения будет результат прибыли каждого лица в задачи. Поскольку мы решаем поставленную задачу для автомобильного завода, то с помощью программы мы должны определить прибыль данного завода и именно она должна быть больше, чем у конкурента.

На рисунке 20 консольное приложение предлагает воспользоваться готовыми данными или вручную вводит все значения для каждого игрока.

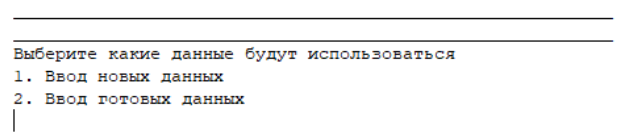

Рисунок 20 – Первый этап «Выбрать ввод данных»

На рисунке 21 нам показывают данные, которые были внесены в нашу программу.

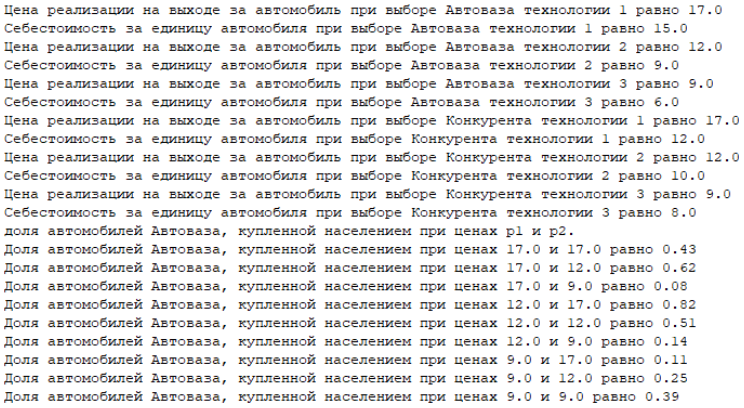

Рисунок 21 – Второй этап «Обозреваем используемые данные»

На рисунке 22 составляется платежная матрица по внесенным данным и находится максимин и минимакс этой матрицы. После чего определяется при каких стратегиях будет решаться программа.

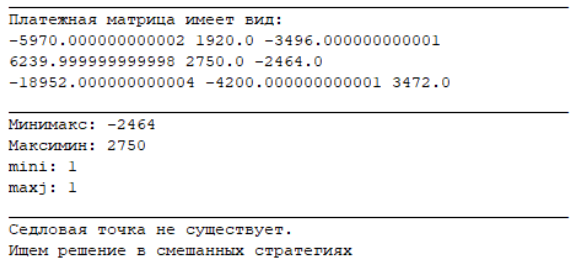

Рисунок 22 – Третий этап «Определяем наличие седловая точка»

На рисунке 23 изменяется по теореме Фон Неймана матрица и подготавливается к решению с помощью ЛПР. Составляется система для решения.

```
Измененная матрица
12982.000000000002 20872.000000000004 15456.000000000004
25192.0 21702.000000000004 16488.000000000004
0.0 14752.000000000004 22424.000000000004

Игра сводится к двоичной задаче линейного програмирования
Max = y0 + y 1 + y 2
| 12982.000000000002*y0 20872.000000000004*y 115456.000000000004*y 2 <= 1
| 25192.0*y0 21702.000000000004*y 116488.000000000004*y 2 <= 1
| 0.0*y0 14752.000000000004*y 122424.000000000004*y 2 <= 1

Двоично
Min = x0 + x 1 + x 2

| 12982.000000000002*x0 25192.0*x 10.0*x 2 >= 1
| 20872.000000000004*x0 21702.000000000004*x 114752.000000000004*x 2 >= 1
| 15456.000000000004*x0 16488.000000000004*x 122424.000000000004*x 2 >= 1
```

Рисунок 23 – Четвертый этап «Сводим задачу к ЛПР»

На рисунке 24 представлен фрагмент последней симплекс таблице, где найдено готовое решение для x и y, до этого было построено ещё две таблицы пока не было найдено оптимальное решение

```
-----------------------------------------------------
Разрешающий столбец: r = 1
Разрешающая строка:  s = 3
                            1.0     1.0     1.0     0.0     0.0     0.0     C[j]
       cB      bV      bD   y[0]    y[1]    y[2]    y[3]    y[4]    y[5]    bD[i] / a[i][r]
       0.0     y[4]    0.17 0.0     5110.14 0.0     1.0     -0.52   -0.31   0.0
       1.0     y[1]    0.0  1.0     0.43    0.0     0.0     0.0     0.0     0.0
       1.0     y[3]    0.0  0.0     0.66    1.0     0.0     0.0     0.0     0.0
                            0.0     -0.09   0.0     0.0     0.0     0.0     delta[j]
Все оценки неположительны, подсчет завершен.
Решение y = (0.0, 0, 0.0, 0.17, 0, 0)Решение x = (0.0, 4.0E-5, 2.0E-5, 0.0, 0.08876, 0.0, )
```

Рисунок 24 – Пятый этап «Составление симплекс таблицы»

На рисунке 25 показано нахождение значение вероятностей для каждой стратегии игрока 1 – автомобильного завода и игрока 2 – конкурента.

```
Оптимальная стратегия игрока A:
A[0] с вероятностью 0.0
A[1] с вероятностью 0.7203803649447444
A[2] с вероятностью 0.2796196350552556
Оптимальная стратегия игрока B:
B[0] с вероятностью 0.19069647905422774
B[1] с вероятностью 0.0
B[2] с вероятностью 0.8093035209457723
```

Рисунок 25 – Шестой этап «Находим вероятности оптимальных стратегий»

На рисунке 26 показаны данные выигрыша автомобильного завода при использовании разных стратегий, аналогично находится выигрыш для конкурента, чтобы посчитать в конечном итоге лучшую стратегию для двоих сторон.

```
Выигрыш Автоваза при стратегииA[0] B[0] равно 0.0
Выигрыш Конкурента при стратегииA[0] B[0] равно 2138.722985273135
Выигрыш Автоваза при стратегииA[0] B[1] равно 0.0
Выигрыш Конкурента при стратегииA[0] B[1] равно 0.0
Выигрыш Автоваза при стратегииA[0] B[2] равно 0.0
Выигрыш Конкурента при стратегииA[0] B[2] равно 496.96372798703
Выигрыш Автоваза при стратегииA[1] B[0] равно 13154.065483341654
Выигрыш Конкурента при стратегииA[1] B[0] равно 1044.6289606107212
Выигрыш Автоваза при стратегииA[1] B[1] равно 8895.80661261473
Выигрыш Конкурента при стратегииA[1] B[1] равно 0.0
Выигрыш Автоваза при стратегииA[1] B[2] равно 2001.2333164309564
Выигрыш Конкурента при стратегииA[1] B[2] равно 644.5023181780626
Выигрыш Автоваза при стратегииA[2] B[0] равно 798.0189160007267
Выигрыш Конкурента при стратегииA[2] B[0] равно 807.2851282912993
Выигрыш Автоваза при стратегииA[2] B[1] равно 1949.6516782397423
Выигрыш Конкурента при стратегииA[2] B[1] равно 0.0
Выигрыш Автоваза при стратегииA[2] B[2] равно 2564.8748092537844
Выигрыш Конкурента при стратегииA[2] B[2] равно 1509.449256233917
```

Рисунок 26 – Седьмой этап «Вычисление выигрышей при разных стратегиях»

На рисунке 27 представлена оптимальная стратегия и найдена прибыль для каждого участника, после чего модно сделать вывод по тем данным, которые были введены в начале программы.

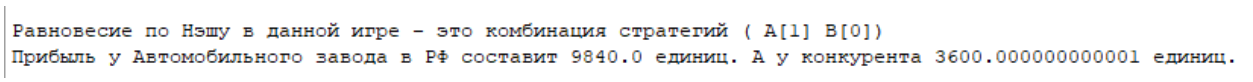

Рисунок 27 – Завершающий этап программы

Таким образом, в результате проведенных работ был разработан алгоритм решения матричной игры в смешанных стратегиях при сведении её к задаче ЛПР. После успешного тестирования можно сделать вывод, что цель работы была достигнута и разработанное приложение готово к использованию.

Вывод по третьему разделу

Данный раздел посвящен реализации программы и тестированию на определенном примере. Исходя из тестирования программы можно увидеть, что она была успешно реализована. Это говорит о том, что созданное приложение работает корректно и готово к использованию.

Заключение

При вы выполнении выпускной квалификационной работы мной были изучены теоретические данные связанные с теорией игр, а также был проведен анализ по выбору более подходящего вида игр для решения поставленной задачи.

После проведения исследования предметной области и представления выбора были рассмотрены различные способов решения матричной игры, с учетом всех данных, предоставленных из теоретического раздела, была построена математическая модель решения.

На основе полученной модели была реализована программа на языки программирования Java в среде разработки Eclipse. Эта программа представлена в виде обычного консольного приложения, которая рассчитывает прибыль автомобильного завода при определённых условиях.

Во время тестирования был предоставлен готовый пример, на котором была протестирована программа, а также каждый модуль входящий в неё. Отметим, что в результате данного тестирования программа работала без каких-либо ошибок и результат полученный на выходе соответствовал ожиданиям. Следовательно, программа спроектирована и реализована успешно и её можно пользоваться в дальнейшем.

Делая вывод, можно сказать, что для разработки данного консольного приложения нужно иметь знания в теории игр и понимать, как экономическая задача может решаться при помощи данного раздела. А также иметь опыт в проектировании приложения и умение свободно пользоваться языком программирования Java.

Таким образом, в результате проделанной работы было разработано консольное приложение для решения матричной игры.

Список используемой литературы

1. Алексеенко, Н. А. Экономика промышленного предприятия / Н.А. Алексеенко, И.Н. Гурова. - М.: Издательство Гревцова, 2021. - 264 с (дата обращения: 23.03.2024).
2. Бинмор, К. Теория игр. Очень краткое введение / К. Бинмор. - М.: ИД "Дело" РАНХиГС, 2019. - 256 с.
3. Буткевич, И. В. решение матричных игр в смешанных стратегиях / И. В. Буткевич // Международный студенческий научный вестник. – 2023.
4. Ганичева, А. В. Модели теории игр в экономике и бизнесе / А. В. Ганичева, А. В. Ганичев // Эпоха науки. – 2019.
5. Гуськова, О.И. Объектно ориентированное программирование в Java: учебное пособие / О. И. Гуськова. - Москва: МПГУ, 2018. - 240 с.
6. Деорнуа, П. Комбинаторная теория игр / П. Деорнуа. - М.: МЦНМО, 2017. - 40 с (дата обращения: 1.04.2024).
7. Диксит А., Нейлбафф Б. Теория игр. Искусство стратегического мышления в бизнесе и жизни - М.: МАНН, 2021, 464 с (дата обращения: 18.02.2024).
8. Ерасов, И. В. О решении матричных игр в рамках метода анализа иерархий / И. В. Ерасов // Теория, методология и концепция модернизации в экономике, управлении проектами, политологии, педагогике, психологии, праве, природопользовании, медицине, философии, филологии, социологии, математике, технике, физике: Сборник научных статей по итогам международной научно-практической конференции, Санкт-Петербург, 26–27 сентября 2013 года. – Санкт-Петербург: Общество с ограниченной ответственностью "Редакционно-издательский центр "КУЛЬТ-ИНФОРМ-ПРЕСС", 2017. (дата обращения: 1.05.2024).
9. Зверева, Т. Е. Применение методов теории игр в экономике / Т. Е. Зверева, С. С. Плотникова // Потенциал российской экономики и инновационные пути его реализации: материалы всероссийской научно-

практической конференции студентов и аспирантов, Омск, 28 апреля 2022 года / Под редакцией Т.В. Ивашкевич, А.И. Ковалева, О.В. Фрик, Д.В. Саврасовой. – Омск: Омский филиал федерального государственного образовательного бюджетного учреждения высшего профессионального образования "Финансовый университет при Правительстве Российской Федерации", 2022 (дата обращения: 10.03.2024).

10. Кисельчук, А. А. Теория игр в экономике. Как теорема Нэша помогает решать экономические задачи / А. А. Кисельчук, В. В. Емельянов // Математические методы и модели техники, технологий и экономики: сборник материалов Всероссийской научно-практической студенческой конференции, Санкт-Петербург: Политех-Пресс, 2021.

11. Кощегулова, И. Р. Применение математического аппарата линейного программирования с переменными коэффициентами к решению матричных игр / И. Р. Кощегулова, М. А. Стрельцов // Экономика и управление: научно-практический журнал. – 2021.

12. Миляева, Л. Г. Экономика организации (предприятия). Практикоориентированный подход. Учебное пособие / Л.Г. Миляева. - М.: КноРус, 2021. - 224 с (дата обращения: 13.02.2024).

13. Нелеп, В. В. Основные понятия теории игр в экономике / В. В. Нелеп // Трибуна ученого. – 2020 (дата обращения: 13.04.2024).

14. Нимейер, П. Программирование на Java / П. Нимейер, Д. Леук. - М.: Эксмо, 2018. - 448 с (дата обращения: 26.04.2024).

15. Перевозчиков, К. О. Важные аспекты теории игр и вклад Джона Нэша в развитие экономики / К. О. Перевозчиков // Фундаментальная и прикладная наука: состояние и тенденции развития: Сборник статей XXXV Международной научно-практической конференции, Петрозаводск, 23 ноября 2023 года. – Петрозаводск: Международный центр научного партнерства "Новая Наука", 2023.

16. Прудников, И. М. Новый подход к поиску точек равновесия в экономике и теории игр / И. М. Прудников // Системы компьютерной математики и их приложения. – 2019.

17. Сигал, А.В. Теория игр и ее экономические приложения: Учебное пособие / А.В. Сигал. - М.: Инфра-М, 2017. - 413 с.

18. Спирин, В. И. Стандартная модель международной торговли, теория игр и деиндустриализация российской экономики / В. И. Спирин // Свободная мысль. – 2023.

19. Юрлов, Ф. Ф. Возможности применения теории кооперативных игр в экономике / Ф. Ф. Юрлов, Н. Я. Леонтьев, Д. А. Самаров // Актуальные вопросы экономики, менеджмента и инноваций: Материалы Международной научно-практической конференции, Нижний Новгород, 16 ноября 2022 года. – Нижний Новгород: Нижегородский государственный технический университет им. Р.Е. Алексеева, 2022 (дата обращения: 30.03.2024).

20. Gorelik, V. A. Methods for Determining Optimal Mixed Strategies in Matrix Games with Correlated Random Payoffs / V. A. Gorelik, T. V. Zolotova // Chebyshevskii Sbornik. – 2023 (дата обращения: 21.03.2024).

21. Beginning Java Objects: From Concepts to Code ISBN-13// Jacquie Barker Fairfax, VA, USA. – 2023 (дата обращения: 17.04.2024).

22. Java Cookbook by Ian F. Darwin (2020) RejmiNet Group, Inc. All rights reserved. Printed in the United States of America (дата обращения: 25.04.2024).

23. Head First Java Third Edition by Kathy Sierra, Bert Bates, and Trisha Gee (2022) by Kathy Sierra and Bert Bates. Printed in the United States of America.

24. Harbuzova, N. D. Nudging from "Nash" to "Pareto": choice optimization problem in the language of game theory / N. D. Harbuzova //, 28 февраля 2020 года, 2020.

25. Ross, Don, "Game Theory", The Stanford Encyclopedia of Philosophy (Spring 2024 Edition), Edward N. Zalta & Uri Nodelman (eds.). URL : https://plato.stanford.edu/archives/spr2024/entries/game-theory/ (дата обращения: 30.03.2024).