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

Институт математики, физики и информационных технологий
(наименование института полностью)

Кафедра «Прикладная математика и информатика»
(наименование кафедры)

02.03.03 Математическое обеспечение и администрирование информационных систем
(код и наименование направления подготовки, специальности)

Технология программирования
(направленность (профиль)/специализация)

БАКАЛАВРСКАЯ РАБОТА

на тему Применение методов Монте-Карло для решения задач теории игр

Студент Н.Р. Алифбекова _____
(И.О. Фамилия) (личная подпись)

Руководитель С.В. Митин _____
(И.О. Фамилия) (личная подпись)

Допустить к защите

Заведующий кафедрой к.т.н., доцент, А.В. Очеповский _____
(ученая степень, звание, И.О. Фамилия) (личная подпись)

«_____»_____20_____г.

Тольятти 2019

АННОТАЦИЯ

Тема выпускной квалификационной работы: «Применение методов Монте-Карло для решения задач теории игр».

Работа была выполнена студентом Тольяттинского государственного университета, института математики, физики и информационных технологий, группы МОб-1501 Алифбековой Назирой Раимбековной. Выпускная квалификационная работа посвящена созданию алгоритмов методов Монте-Карло для решения многокритериальных задач, а также реализации и тестированию полученных алгоритмов.

Объект исследования – математические модели многокритериальных задач.

Предмет исследования – методы нахождения оптимальной стратегии решения многокритериальных задач.

Целью выпускной квалификационной работы является использование методов Монте-Карло для решения многокритериальных задач.

Задачи работы:

1. Описание алгоритма решения частных случаев стохастической игры с отсутствием полной информации.

2. Использование скрытых марковских моделей для описания стохастической игры.

3. Программная реализация полученного алгоритма и его тестирование.

Выпускная квалификационная работа состоит из введения, двух глав, заключения, списка используемых источников. Во введении рассказывается об актуальности исследования по выбранному направлению.

В главе 1 представлено теоретическое описание скрытых марковских моделей, а также предложена модель решения игры двух лиц с неполной информацией. В главе 2 тестируется предложенная модель решения по игре «теннис», также алгоритм тестируется по другой игре со схожими характеристиками. В заключении представлены результаты и выводы о выполненной работе.

В работе использовано 19 таблиц, 10 рисунков и 2 приложения. Список литературы содержит 17 источников, в том числе 12 на иностранном языке. Общий объем выпускной квалификационной работы составляет 46 страниц.

ABSTRACT

The title of the bachelor's thesis is Applying Monte Carlo methods to a solution of game theory problems.

The aim of this work was the use of the Monte Carlo methods for solving multicriteria problems.

The object of the study was mathematical models of multicriteria problems.

The subject of the study was the Monte Carlo problem solving.

The bachelor's thesis is devoted to the application of the Monte Carlo methods for solving multicriteria problems.

The first part of the thesis presents a theoretical description of hidden Markov models, as well as a model for solving a two-person game with incomplete information. In the second part the proposed model of the decision on the game «tennis» is tested, and the algorithm is also tested on a different game with similar characteristics.

In many real-world conflict situations participants do not have all the information about the true states of the process and can evaluate the results of the actions of rivals only on certain limited data sets. Under these conditions, decision making may depend on several factors, including random ones. One of the possible approaches to finding the optimal strategy in such problems is to use probabilistic models to describe the situation under study.

The proposed algorithm was tested on arbitrary data of the game «stone, scissors, paper». During the experiments the algorithm showed high efficiency of the algorithm, which confirms the possibility of using the proposed algorithm in any game with similar characteristics.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ 6
Глава 1 Теоретическое описание скрытых марковских моделей 9
1.1 Обзор работ, связанных с применением методов Монте-Карло для решения задач теории игр 9
1.2 Описание скрытой марковской модели 10
1.3 Постановка задачи и пошаговый алгоритм решения 11
1.4 Описание алгоритмов и инструментарий 14
1.4.1 Алгоритм Баума-Велша 15
1.4.2 Алгоритм Витерби 19
Выводы к первой главе 20
Глава 2 Тестирование методов Монте-Карло для решения задач теории игр 21
2.1 Тестирование на примере игры «Теннис» 21
2.1.1 Вычисление элементов матрицы B 22
2.1.2 Тестирование на данных, представленных в статье 25
2.1.3 Тестирование на данных матрицы B, найденной выше 28
2.1.4 Сравнение с другими подходами решения поставленной задачи . 30
2.2 Применение модели к другой игре со схожими характеристиками 33
Выводы ко второй главе 38
ЗАКЛЮЧЕНИЕ 40
СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ 41
ПРИЛОЖЕНИЕ А Реализация алгоритма Баума-Велша из пакета «HMM» . 43
ПРИЛОЖЕНИЕ Б Реализация алгоритма Витерби из пакета «HMM» 46

ВВЕДЕНИЕ

Многокритериальная задача – это математическая модель принятия оптимального решения одновременно по нескольким критериям. Эти критерии могут отражать оценки различных качеств объекта (или процесса), по поводу которых принимается решение, или оценки одной и той же его характеристики, но с различных точек зрения. В данной работе под многокритериальными задачами понимаются задачи теории игр.

Математическая теория игр – дисциплина, возникшая на стыке 2 наук: математики и экономики. В настоящее время область её практического применения достаточно широка: политические и дипломатические отношения, биология, информатика и другие. Предметом изучения данной дисциплины являются игры и связанные с ними стратегии (т.е. множества доступных игрокам действий), а целью – выработка методов нахождения оптимальных стратегий. Под игрой, как правило, понимается некоторая конфликтная ситуация, т.е. процесс, в котором принимают участие от двух и более сторон преследующих собственные цели [7].

Основополагающей работой в теории игр является книга Джона фон Неймана и Оскара Моргенштерна «Теория игр и экономическое поведение» (1944) [4]. Большая часть данной книги посвящена играм с нулевой суммой. Игры с нулевой суммой – игры, в которых выигрыш одной стороны равен проигрышу другой. Игры с нулевой суммой с участием 2-х игроков называются антагонистическими. К таким играм также можно отнести игры с постоянной суммой, при которых сумма общего выигрыша всех игроков фиксирована, и поэтому увеличение выигрыша одного из них возможно только за счёт уменьшения выигрышей других игроков.

В конце 1960-х годов Джон Харшаньи ввёл понятие игр с неполной информацией и разработал концепцию байесовских равновесий. Д. Харшаньи рассматривал ситуации, когда у одного игрока нет информации о возможных выигрышах другого игрока, и выигрыши приходится оценивать вероятностно.

Игры принято различать по типу. Например, кооперативные и некооперативные игры, с нулевой суммой и ненулевой суммой, параллельные и последовательные, метаигры, игры с полной или неполной информацией [4, 17]. Последние, с одной стороны, наиболее сложны для построения моделей прогнозирования, с другой – представляют значительный практический интерес, т.к., достаточно часто в реальных конфликтных ситуациях, участники не обладают полной информацией о действиях оппонентов [9, 13, 16].

Для дальнейших рассуждений понадобится следующее определение: ход игрока – это выбор действия из некоторого множества возможных действий игрока, определяемых правилами игры.

В работе рассматриваются игры, включающие в себя следующие элементы.

1. Чередование ходов, которые могут быть как личными, так и случайными.

2. Недостаточность информации.

3. Функция выигрыша [6].

Как уже было отмечено ранее, во многих реальных конфликтных ситуациях участники не обладают всей информацией об истинных состояниях процесса и могут оценивать результаты действий соперников только по некоторым ограниченным наборам данных. В этих условиях принятие решений может зависеть от нескольких факторов, в том числе – случайных. Один из возможных подходов к нахождению оптимальной стратегии в таких задачах это использование для описания исследуемой ситуации вероятностных моделей.

В данной работе используются Скрытые Марковские Модели (СММ) для описания состояний игры в различные моменты времени. СММ зарекомендовали себя как мощное средство обработки данных в самых различных областях, с примерами применения можно ознакомиться в [15]. Преимущества СММ перед другими моделями следующие.

1. СММ обладают простой математический структурой.

2. Структура СММ позволяет моделировать сложную цепочку наблюдений.

3. Параметры модели могут быть автоматически выбраны таким образом, чтобы описать имеющийся набор данных для обучения (под обучением СММ понимается определение оценок параметров модели).

В недавно опубликованной работе [14] был предложен метод нахождения оптимальной стратегии для игры с двумя участниками, основанный на применении СММ. В статье был описан подход к решению стратегических игр, в которых игроки могут менять стратегии в течении игры. Целью было увеличить шансы игрока, т.е. узнать какую стратегию принимает соперник в каждый момент времени. Предложенный подход основан на использовании алгоритма Баума-Велша, решения марковской игры поиска смешанного равновесия Нэша.

В бакалаврской работе данный подход получил развитие: предложен альтернативный метод, основанный на применении алгоритма Витерби для получения цепочки скрытых состояний и выбора наиболее вероятного хода вместо решения марковской игры.

Работа алгоритма иллюстрируется на двух демонстрационных примерах: игры в теннис и игры в «Камень, ножницы, бумага».

Данная работа организована следующим образом. В главе 1 дано теоретическое описание скрытых марковских моделей, а также предложена модель решения игры двух лиц с неполной информацией. В главе 2 тестируется предложенная модель решения по игре, описанной в [14], также алгоритм тестируется по другой игре со схожими характеристиками.

Глава 1 ТЕОРЕТИЧЕСКОЕ ОПИСАНИЕ СКРЫТЫХ МАРКОВСКИХ МОДЕЛЕЙ

1.1 Обзор работ, связанных с применением методов Монте-Карло для решения задач теории игр

На практике часто встречаются ситуации, которые в теории игр принято называть повторной игрой (repeated game) т.е. игрой, которая состоит в некотором числе повторений некоторых игровых стадий (stage game). Впервые повторные игры с неполной информацией были введены Ауманом и Машлером в 1960 году [13] Рассматривались игры, в которых начальное состояние выбиралось с вероятностью , и только игрок 1 знает это состояние, в то время как игрок 2 знает все возможные состояния, но не знает фактического состояния. После каждого хода, оба игрока знают исход предыдущего хода и продолжают играть.

В [20] рассматриваются игры с двумя игроками с нулевой суммой, заданные цепью Маркова над конечным множеством состояний и семейством матричных игр. Последовательность состояний подчиняется цепи Маркова. В начале каждого этапа только игроку 1 сообщается о текущем состоянии, затем воспроизводится соответствующая матричная игра, после чего выбранные действия наблюдаются обоими игроками. Такие игры можно охарактеризовать играми с отсутствием информации с одной стороны.

В [14] описана модель скрытой марковской игры (СМИ), при которой второй игрок может наблюдать исходы того или иного хода противника, знает множество возможных скрытых состояний соперника, но не знает скрытых состояний в определённые моменты времени и матрицу переходов из состояния в состояние.

В бакалаврской работе рассмотрен альтернативный метод нахождения оптимальной стратегии, разработанный на основе предложенного в статье [14].

1.2 Описание скрытой марковской модели

Скрытая марковская модель (СММ) с параметрами $\lambda = A, B, \pi$ – это модель марковского процесса, в котором неизвестно, в каком состоянии находится система (состояния скрыты), в то же время каждое состояние может с некоторой вероятностью произвести некоторое событие, которое можно наблюдать (наблюдение).

В бакалаврской работе рассматриваются однородные цепи маркова, то есть такие, матрица переходных вероятностей которых не зависит от номера шага: $A_{ij}\ t = A_{ij}$.

Формальное определение СММ следующее.

Определение 1. Скрытая марковская модель с параметрами $\lambda = A, B, \pi$ описывается следующим образом.

1. Конечное множество состояний $S = s_1, \dots, s_n$.

2. Состояние вероятностей переходов, стохастическая матрица A:

$$A = a_{ij} | a_{ij} = P\ S_T = s_j | S_{T-1} = s_i \quad , 1 \le i, j \le n,$$

где S_T и S_{T-1} – состояния в моменты времени T и $T-1$ соответственно.

Вероятностные характеристики марковского процесса в следующий момент времени зависят только от вероятностных характеристик в текущий момент времени.

3. Конечное множество наблюдений $O = o_1, \dots, o_k$.

4. Состояние конкретного распределения вероятностей, матрица B:

$$B = b_{il} | b_{il} = P\ O_T = o_l | S_T = s_i \quad , 1 \le i \le n, 1 \le l \le k,$$

где O_T – наблюдение в момент времени T, S_T – состояние в момент времени T.

Значение наблюдаемого вектора O_T, взятого в момент времени T, зависит только от скрытого состояния в момент времени T.

5. Вектор начала состояний π

$$\pi = \pi_i | \pi_i = P\ S_1 = s_i \quad , 1 \le i \le n,$$

где S_1 – состояние в начальный момент времени.

Популярным примером, иллюстрирующим использование СММ, является следующая задача о нечестном казино.

Имеется 2 кубика: один честный с вероятностью выпадения каждой грани, равной 1/6, и нечестный с вероятностью выпадения шестёрки, равной 1/2, а остальных граней – 1/10. Крупье подменяет кубики с определённой вероятностью в зависимости от того, какой кубик был брошен в данный момент. Можно наблюдать, что выпадает на кубиках, но неизвестно, какой из кубиков был использован. Таким образом, требуется определить последовательность скрытых состояний (рисунок 1.1).

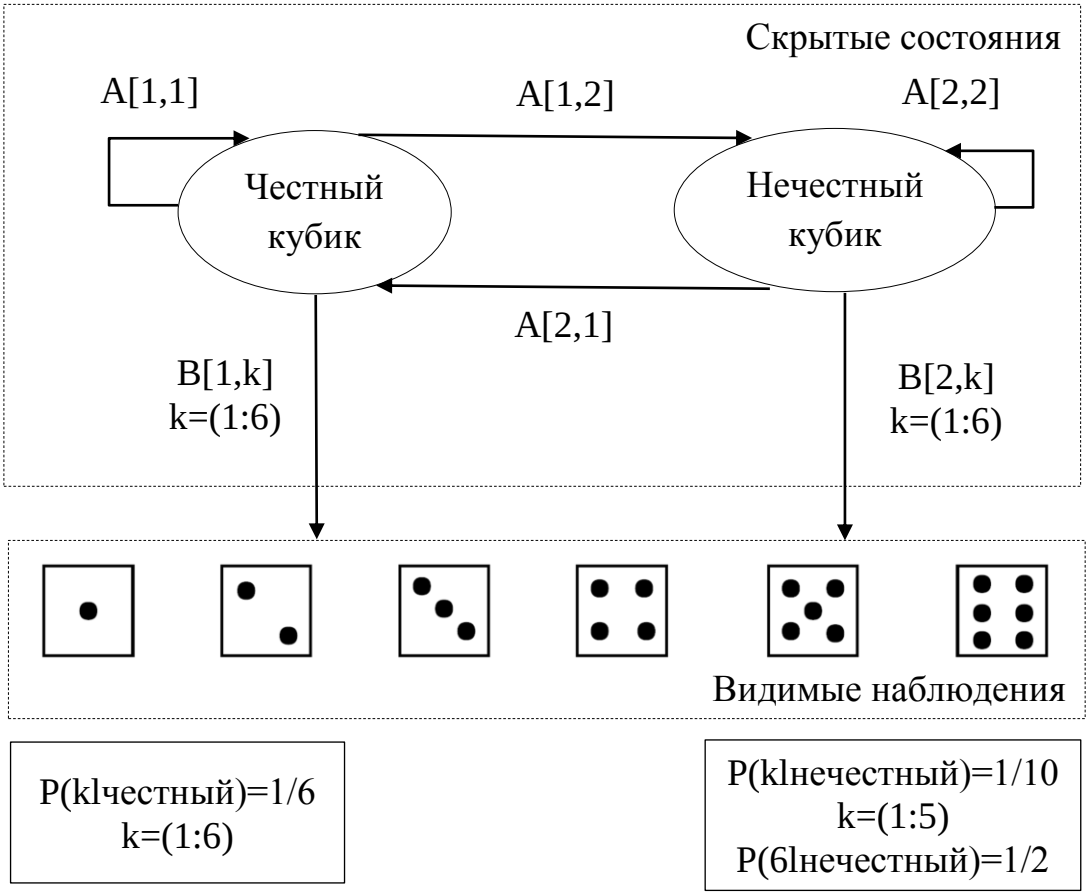

Рисунок 1.1 – Иллюстрация модели для задачи о нечестном казино

1.3 Постановка задачи и пошаговый алгоритм решения

В рассматриваемой статье ([14]) ставилась задача: для игрока i найти вероятность того, что $O_{t_{l+1}} = O_j$ при $1 \leq j \leq k$, используя последовательность

наблюдений $O_{t_0}, \dots, O_{t_l}$. В то же время, в рамках рассматриваемой задачи, неизвестны ни последовательности состояний, ни элементы матриц A и B.

Авторы рассматриваемой статьи на каждом шаге игры выводили оценки параметров модели СММ с помощью алгоритма Баума-Велша, и затем решали марковскую игру.

В бакалаврской работе предлагается вместо решения марковской игры использовать алгоритм Витерби для нахождения состояния другого игрока в текущий момент времени, после чего делать выбор наиболее вероятного хода.

Таким образом, в данной работе предлагается использовать следующий пошаговый алгоритм для решения поставленной задачи.

Входные данные:

- 2 игрока;

- множество доступных игрокам действий (наблюдений) $O = o_1, \dots, o_k$;

- множество состояний 1-го игрока $S = s_1, \dots, s_n$;

- матрицы выплат для каждого из состояний $U_1, \dots, U_k$ размерами $k \times k$.

Замечание: поведение 1-го игрока описывается с помощью СММ с параметрами $\lambda = A, B, \pi$, где A – матрица вероятностей переходов из состояния i $i = 1{:}k$ в состояние j $j = 1{:}n$ размера $n \times n$, B – матрица вероятностей переходов из состояния i $i = 1{:}n$ в наблюдение l $l = 1{:}k$ размера $n \times k$, π – вектор вероятностей появления каждого из состояний в начальный момент времени длины n. Эти параметры являются скрытыми для 2-го игрока. В данной работе они являются входными данными, т.к. по ним моделируются действия 1-го игрока для оценки работы алгоритма.

Результат: на каждом шаге игры алгоритм выдает наиболее вероятную последовательность скрытых состояний 1-го игрока со всеми параметрами его СММ. По итогу считается количество побед 2-го игрока по матрицам выплат (пример представлен в 2.1.2).

1. Вычисляем элементы матрицы B размера $n \times k$, находя смешанные стратегии для первого игрока по матрицам выплат (описание метода нахождения смешанных стратегий представлено в 2.1.1).

2. Первые 3 хода игры второй игрок выбирает случайным образом.

3. Формируется последовательность наблюдений (ходов первого игрока): $O = O_1, O_2, O_3$, где O_1, O_2 и O_3 – какие-либо наблюдения из множества наблюдений O в моменты времени 1, 2 и 3 соответственно.

4. Используем алгоритм Баума-Велша для вывода параметров СММ первого игрока: Входные данные:

- последовательность наблюдений O;

- СММ с параметрами $\lambda = A^p, B, \pi^p$, где матрица A^p и вектор π^p произвольные.

Результат: СММ с параметрами $\lambda^* = \arg max_\lambda P\ O|\lambda$ (подробное описание алгоритма и используемый пакет языка R приведены в 1.4.1).

5. Находим скрытые состояния 1-го игрока, используя алгоритм Витерби. Входные данные:

- последовательность наблюдений O;

- СММ с параметрами λ^*.

Результат: последовательность скрытых состояний противника $S = S_1, \dots, S_t$, где $S_1, \dots, S_t$ – какие-либо состояния из множества состояний S, в моменты времени $1, \dots, t$ соответственно (подробное описание алгоритма и используемый пакет языка R представлены в 1.4.2).

6. Предполагая определённым текущее состояние противника, выбираем для 2-го игрока вариант хода, при котором с наибольшей вероятностью максимизируется его выигрыш: выбираем наиболее вероятное состояние в следующий момент времени находим по матрице B наиболее вероятное наблюдение из этого состояния 1-го игрока. Делаем ход, который по правилам игры является выигрышным.

7. Каждый игрок делает свой ход. Сравниваем наблюдения и вычисляем результат хода по имеющимся матрица выплат.

8. Добавляем результат последнего хода 1-го игрока к цепочке наблюдений O.

9. Повторяем шаги (4)-(8) после каждого хода до окончания игры.

10. Вычисляем результат игры по матрицам выплат (рисунок 1.2).

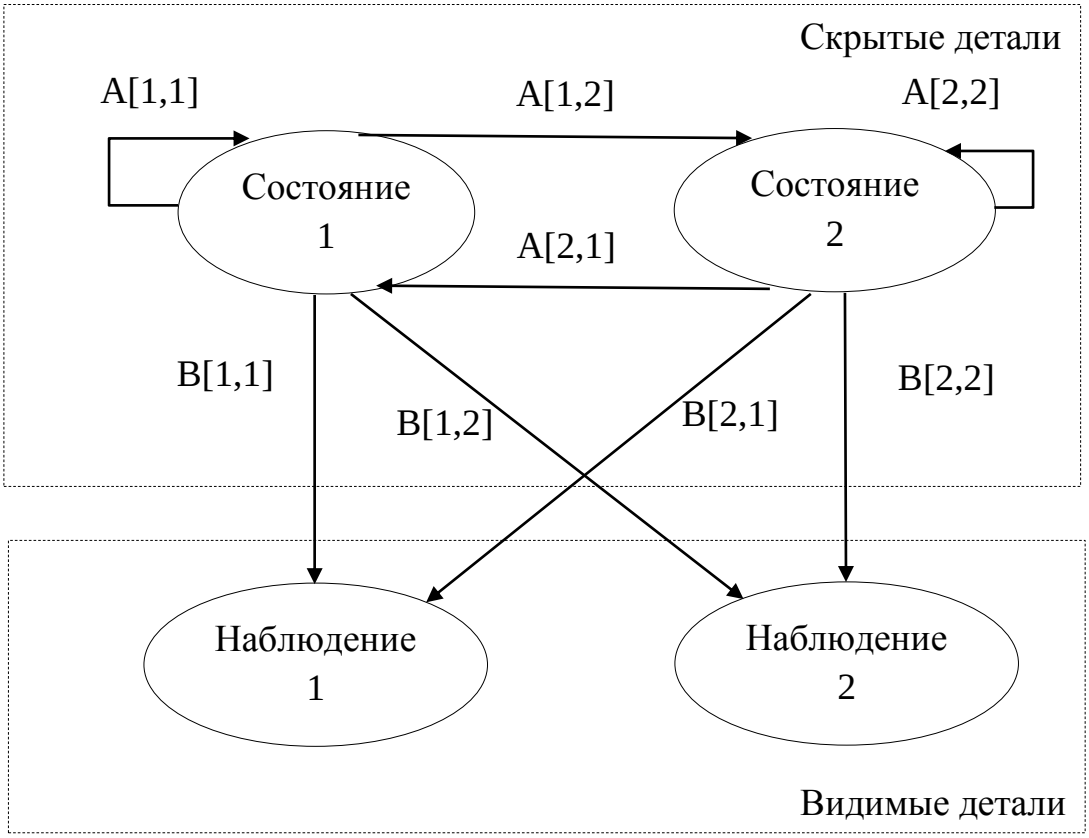

Рисунок 1.2 – Пример модели с двумя состояниями и двумя наблюдениями

1.4 Описание алгоритмов и инструментарий

Все описанные выше методы реализованы в языке программирования R. Выбор данного языка обусловлен наличием большого количества пакетов с открытым кодом, позволяющих решать задачи описанные скрытыми марковскими моделями. Для реализации предложенного алгоритма был использован пакет HMM [11] (код пакета HMM можно посмотреть в [10]), а конкретно функции из него:

- initHMM – инициализация CMM;

- simHMM – генерация цепочки наблюдений и состояний по заданной CMM baumWelch – алгоритм Баума-Велша;

- viterbi – алгоритм Витерби.

Рассмотрим подробнее алгоритмы Баума-Велша и Витерби, использующиеся в данной работе.

1.4.1 Алгоритм Баума-Велша

Входными данными алгоритма являются: последовательность наблюдений O длины t и CMM с параметрами $\lambda = A^p, B, \pi^p$, где матрица A^p и вектор π^p произвольные, а матрица B известна заранее (вычисляется путём нахождения смешанных стратегий). В результате получают CMM с параметрами $\lambda^* = \arg max_\lambda P\ O|\lambda$, т.е. алгоритм пытается настроить модель на данную последовательность наблюдений O.

Сам алгоритм состоит из следующих этапов.

1. Алгоритм прямого-обратного хода: используется для вычисления вероятности $P\ O|\lambda$.

Прямая процедура: вычисляем следующую вероятность.

$$\alpha_r\ i\ = P\ O_1, \dots, O_r, S_r = s_i\ \lambda\ , 1 \le i \le n, \quad (1.1)$$

это вероятность получения последовательности наблюдений O_r, т.е. начальной части подаваемой на вход последовательности O (длина которой t), где $1 < r < t$, при условии, что момент времени r соответствует состоянию s_i.

Для этого:

Инициализация:

$$\alpha_1\ i\ = \pi_i b_{i,1}, 1 \le i \le n, \quad (1.2)$$

Рекурсия:

$$\alpha_{r+1}\ i\ = P\ O_1, \dots, O_{r+1}, S_{r+1} = s_i\ \lambda\ =$$

$$= \sum_{j=1}^{n} P\ O_1, \dots, O_r, O_{r+1}, S_r = s_j, S_{r+1} = s_i\ \lambda\ = {}^{1}$$

$$= \sum_{j=1}^{n} P\ O_1, \dots, O_r, S_r = s_j, S_{r+1} = s_i\ \lambda\ P\ O_{r+1} | S_{r+1} = s_i\ = {}^{2}$$

$$= b_{i,r+1} \sum_{j=1}^{n} P\ O_1, \dots, O_r, S_r = s_j\ \lambda\ P\ S_{r+1} = s_i | S_r = s_j\ = {}^{3}$$

$$= b_{i,r+1} \sum_{j=1}^{n} \alpha_r(j) a_{j,i}\ , 1 \le i, j \le n, 1 \le r \le t-1.$$

Теперь, используя рекурсию, можно найти $\alpha_t\ i\ .$

Таким образом, получаем: $P\ O|\lambda\ = \sum_{i=1}^{n} \alpha_t(i).$

Обратная процедура: вычисляем следующую вероятность.

$$\beta_r\ i\ = P\ O_{r+1}, \dots, O_t | S_r = s_i, \lambda\ , 1 \le i \le n, \qquad (1.3)$$

это вероятность получения последовательности наблюдений O_r, которая в данном случае уже является «концом» последовательности O (размера t), подаваемой на вход, при условии, что мы начинаем в момент времени $r+1$ из исходного состояния s_i и заканчиваем в момент времени t.

Аналогично с прямой процедурой:

Инициализация:

$$\beta_t\ i\ = 1, 1 \le i \le n, \qquad (1.4)$$

Рекурсия:

$$\beta_r\ i\ = P\ O_{r+1}, \dots, O_t | S_r = s_i\ \lambda\ = {}^{4}$$

---

[1]Равенство следует из определения СММ.

[2]$P\ O_{r+1} | S_{r+1} = s_i\ = b_{i,r+1}.$

[3]$P\ S_{r+1} = s_i | S_r = s_j\ = a_{j,i}.$

[4]Равенство следует из определения СММ.

$$= \sum_{j=1}^{n} P\ O_{r+1}, \dots, O_t | S_{r+1} = s_j | S_r = s_i, \lambda\ = [5]$$

$$= \sum_{j=1}^{n} P\ O_{r+2}, \dots, O_t | S_r = s_i, \lambda\ P\ O_{r+1} | S_{r+1} = s_j\ P\ S_{r+1} = s_j | S_r = s_i\ = [6]$$

$$= \sum_{j=1}^{n} \beta_{r+1}(j) b_{i,r+1} a_{i,j}\ , 1 \le i, j \le n, 1 \le r \le t-1.$$

Используя рекурсию, можно вычислить $\beta_1\ i$.

Из формул (1.2) и (1.3) можно получить: $P\ O|\lambda\ = \sum_{i=1}^{n} \alpha_t\ i\ \beta_1\ i\ =$

$$= \sum_{i=1}^{n} \pi_i\, b_{i,1} \beta_1\ i$$

В то же время, из формул (1.1) и (1.3) следует, что $P\ O, S_r = s_i|\lambda\ =$

$$= \alpha_r\ i\ \beta_r\ i\ , 1 \le i \le n, 1 \le r \le t.$$

Из полученной вероятности теперь можно выразить искомую вероятность $P\ O|\lambda$ следующим образом:$P\ O, S_r = s_i|\lambda\ \ \alpha_r\ i\ \beta_r\ i$

$$P\ O|\lambda\ = \sum_{i=1}^{n} P\ O, S_r = s_i|\lambda\ = \sum_{i=1}^{n} \alpha_r\ i\ \beta_r\ i\ , r \in 1, \dots, t\ . \quad (1.5)$$

2. Вычисление вспомогательных переменных.

Через переменную ξ_r обозначим вероятность того, что модель СММ при последовательности наблюдений O в моменты времени r и $r+1$ будет находиться в состояниях s_i и s_j соответственно.

Используя формулы (1.1) и (1.3), можно вычислить данную вероятность следующим образом:

$$\xi_r\ i.j\ = P\ S_r = s_i, S_{r+1} = s_j | O, \lambda\ = \frac{P\ S_r = s_i, S_{r+1} = s_j, O|\lambda}{P\ O|\lambda} =$$

$$= \frac{\alpha_r(i) a_{i,j} \beta_{r+1}(j) b_{j,r+1}}{\sum_{i_s=1}^{n} \sum_{j_s=1}^{n} \alpha_t(i_s) a_{i_s,j_s} \beta_{r+1}(j_s) b_{j_s,r+1}}, 1 \le r \le t-1. \quad (1.6)$$

---

[5]Равенство следует из определения СММ.

[6]$P\ O_{r+1}|S_{r+1} = s_j\ = b_{i,r+1}, P\ S_{r+1} = s_j|S_r = s_i\ = a_{i,j}.$

Через переменную γ_r обозначим вероятность того, что модель СММ, при последовательности наблюдений O в момент времени r будет находиться в состоянии s_i.

Для получения γ используем формулы (1.1), (1.3) и (1.5):

$$\gamma_r\ i\ = P\ S_r = s_i|O,\lambda\ = \frac{P\ O,S_r=s_i|\lambda}{P\ O|\lambda} = \frac{\alpha_r(i)\beta_r(i)}{\sum_{i_S=1}^{n}\alpha_r(i_S)\beta_r(i_S)}, 1 \le r \le t. \quad (1.7)$$

Заметим, что переменные $\xi_r\ i.j\ $ и $\gamma_r\ i\ $ связаны следующим образом:

$$\gamma_r\ i\ = \ \sum_{j=1}^{n}\xi_r\ i.j\ , 1 \le i \le n, 1 \le r \le t-1. \quad (1.8)$$

3. Перевычисление параметров модели λ.

На данном этапе происходит перевычисление параметров модели с целью увеличения $P\ O|\lambda\ $. Новые параметры определяются следующим образом:

$$\pi_i^* = \gamma_1\ i\ , 1 \le i \le n, \quad (1.9)$$

$$a_{ij}^* = \frac{\sum_{r=1}^{t-1}\xi_r\ i.j}{\sum_{r=1}^{t-1}\gamma_r\ i}, 1 \le i \le n, 1 \le j \le n\,. \quad (1.10)$$

В данной постановке задачи не требуется перевычисление элементов матрицы B, так как она дана точная. Поэтому в алгоритме Баума-Велша этот шаг опускаем.

4. Проверка сходимости.

После перевычисления параметров получается следующую СММ: $\lambda^* = \ A^*, B, \pi^*$. Для нее вычисляем $P\ O|\lambda^*$ по одной из формул, представленных ранее на этапе 1.

Замечание 1. Алгоритм Баума-Велша обеспечивает неубывание этой вероятности. На этом шаге проверяем, выполняется ли следующее условие:

$$P\ O|\lambda^*\ - P\ O|\lambda\ < \varepsilon\,, \quad (1.11)$$

где ε – некоторая заданная величина.

В случае отрицательного исхода переобозначаем $\lambda = \lambda^*$, и итерации продолжаются (повторяются все этапы).

Если неравенство выполнено, алгоритм заканчивает работу и на выходе получается λ^*.

Описание алгоритма Баума-Велша можно также посмотреть в [18, 19].

Реализация алгоритма Баума-Велша, осуществлённая в пакете HMM, не предусматривает оценку вектора π. В связи с этим код алгоритма Баума-Велша был доработан с учётом специфики поставленной задачи: была добавлена оценка вектора π и убрана оценка матрицы B в целях уменьшения количества свободных переменных.

1.4.2 Алгоритм Витерби

Алгоритм Витерби позволяет построить наиболее вероятную последовательность предыдущих состояний скрытой марковской модели на основе последовательности наблюдений.

Входные данные: последовательность наблюдений O и СММ с параметрами $\lambda^* = A^*, B, \pi^*$, полученными в алгоритме Баума-Велша. Результатом работы алгоритма будет последовательность скрытых состояний $S = S_1, \dots, S_t$.

Алгоритм состоит из следующих этапов.

1. Вычисление максимальной вероятности того, что при последовательности наблюдений O СММ в момент времени r будет находиться в состоянии s_i. Обозначим эту вероятность следующим образом:

$$\delta_r\ i\ = \max_{S_1,\dots,S_{r-1}} P\ S_1, \dots, S_{r-1}, S_r = s_i, O_1, \dots, O_{r-1}|\lambda\ . \quad (1.12)$$

Инициализация:

$$\delta_1\ j\ = \pi_j b_{j,1},\ 1 \le j \le n. \quad (1.13)$$

Рекурсия:

$$\delta_{r+1}\ j\ = b_{j,r+1} \max_{1\le i\le n}\ \delta_r\ i\ a_{i,j}\ ,\ 1 \le i, j \le n,\ 1 \le r \le t-1. \quad (1.14)$$

Но в данном случае следует запоминать ещё и состояния $\varphi_r\ j$ (s_j в момент времени r). Это те состояния, при которых $\delta_r\ j$ достигает максимума:

$$\varphi_{r+1}\ j\ = arg \max_{1\le i\le n}\ \delta_r\ i\ a_{i,j}\ ,\ 1 \le j \le n,\ 1 \le r \le t-1. \quad (1.15)$$

Используя данную рекурсию, можно вычислить $\delta_t\ i\ , 1 \le i \le n.$

2. На следующем этапе следует вычислять наиболее вероятное конечное состояние системы S_r после шага $t-1$:

$$S_t = arg\max_{1 \le i \le n} \ \delta_t \ i \quad . \tag{1.16}$$

В этой формуле $\max_{1 \le i \le n} \ \delta_t \ i$ – это наибольшая из максимальных вероятностей нахождения системы в момент времени t в состоянии s_i.

3. На последнем этапе вычисляется наиболее вероятная последовательность состояний системы S.

Вычисляется она обратным проходом по массиву состояний φ_r (при которых вероятности $\delta_r \ i$ были максимальны, начиная с наиболее вероятного конечного состояния S_t).

$$S_r = \varphi_{r+1} \ S_{r+1} \ , \ 1 \le r \le t - 1. \tag{1.17}$$

Описание алгоритма Витерби можно также посмотреть в [18, 19].

Выводы к первой главе

1. Представлен алгоритм решения частных случаев стохастической игры с отсутствием полной информации.

2. Использованы СММ (скрытые марковские модели) в качестве инструмента описания данного процесса (стохастической игры).

3. Проведён обзор литературы на близкие темы.

4. Рассмотрены алгоритмы Баума-Велша и Витерби как инструмент решения задач, описанных СММ.

.

Глава 2 ТЕСТИРОВАНИЕ МЕТОДОВ МОНТЕ-КАРЛО ДЛЯ РЕШЕНИЯ ЗАДАЧ ТЕОРИИ ИГР

2.1 Тестирование на примере игры «Теннис»

Протестируем алгоритм на примере игры, описанной в [14], сравним с результатами, полученными авторами статьи, а также проведём дополнительные тесты.

В рассматриваемой работе приведён пример игры в теннис, в которой есть 2 игрока (подающий и принимающий). Набор стратегий для каждого из них:

1) подающий может подать мяч в центр (Центр) или в открытую зону (Открытый);

2) принимающий должен быть готов к одному из этих действий.

У подающего есть 3 состояния: Агрессивный, Спокойный или Защитный. Для каждого состояния имеются свои вероятности подать мяч в центр или в открытую зону (состояния меняются согласно цепи маркова). Задача состоит в том, чтобы предсказать направление подачи (центр или открытая зона).

Прежде чем перейти к непосредственно к вычислениям и тестированию, введём несколько определений:

Определение 2. Матрица выплат – это таблица, в которой отображены выплаты каждому из участников в игре двух лиц за определённый ход. Строки таблицы показывают результаты каждого выбора хода первым игроком, а столбцы – результаты выбора второго игрока.

Определение 3. Смешанная стратегия – заданное вероятностное распределение ходов.

Определение 4. Равновесие в смешанных стратегиях – ситуация, когда ни одному из игроков не выгодно отклоняться от своей смешанной стратегии.

Определение 5. Чистая стратегия – частный случай смешанной стратегии, в котором одна из вероятностей равна 1, а остальные 0.

Определение 6. Седловая точка – вид равновесия в чистых стратегиях, характеризующий тем, что соответствующий элемент матрицы является одновременно наибольшим в своём столбце и наименьшим в своей строке [6].

Используемые матрицы выплат для каждого из состояний в предложенной игре представлены в таблицах 2.1-2.3.

Обозначения:

«Открытый» – открытая зона;

«Центр» – центральная зона.

Таблица 2.1 – Матрица выплат (Агрессивный игрок)

Зона	Открытый	Центр
Открытый	0,35 0,65	0,89 0,11
Центр	0,98 0,02	0,15 0,85

Таблица 2.2 – Матрица выплат (Умеренный игрок)

Зона	Открытый	Центр
Открытый	0,15 0,85	0,8 0,2
Центр	0,9 0,1	0,15 0,85

Таблица 2.3 – Матрица выплат (Защитный игрок)

Зона	Открытый	Центр
Открытый	0,1 0,9	0,55 0,45
Центр	0,85 0,15	0,05 0,95

2.1.1 Вычисление элементов матрицы B

Для вычисления элементов матрицы B найдём равновесие в смешанных стратегиях. Подробное решение будет приведено для матрицы выплат агрессивного игрока (для остальных матриц выплат аналогично).

Рассмотрим матрицу выплат (таблица 2.4).

Таблица 2.4 – Матрица выплат (Агрессивный игрок)

Зона	Открытый	Центр
Открытый	0,35 0,65	0,89 0,11
Центр	0,98 0,02	0,15 0,85

Представленная игра является игрой с постоянной суммой. При игре с постоянной суммой на каждом шаге игры общая сумма выплат игрокам фиксирована. То есть, зная выигрыш первого игрока на каждом шаге игры, с точностью можно сказать о выигрыше второго игрока. В таком случае для поиска равновесия в смешанных стратегиях достаточно рассматривать матрицу выплат, в которой представлены выплаты только для первого игрока (таблица 2.5).

Таблица 2.5 – Матрица выплат первого игрока (Агрессивный игрок)

Зона	Открытый	Центр
Открытый	0,35	0,89
Центр	0,98	0,15

Ситуация равновесия для игры с постоянной суммой описывается следующими неравенствами:

$$H\ x,y\ \leq H\ x,y\ \leq H\ x,y\ ,\forall x \in S_1, \forall y \in S_2, \quad (2.1)$$

где

- $S_1,\ S_2$ – множество стратегий (в данном случае наблюдений) для игрока 1 и игрока 2;

- $x = x_1, x_2\ , y = y_1, y_2$ – смешанные стратегии, где x_1, x_2 – вероятности подать мяч в открытую и центральную зоны первым игроком, а y_1, y_2 – вероятности принять мяч в открытой и центральной зонах вторым игроком;

- $H\ x,y$ – функция выигрыша игрока 1 в ситуации x,y, которая определяется как математическое ожидание его выигрыша [2].

Неравенства (2.1) можно интерпретировать следующим образом: игрок 1 стремится найти стратегию, которая будет приносить ему максимальный выигрыш. В то же время игрок 2 ищет такую стратегию, при которой у него будет минимальный проигрыш.

Прежде чем искать равновесие в смешанных стратегиях, необходимо определить, есть ли равновесие в чистых стратегиях (есть ли седловая точка). В данном случае ситуации равновесия в чистых стратегиях нет.

Таким образом, необходимо найти равновесие в смешанных стратегиях для игрока 1, для чего требуется решить задачу линейного программирования:

$$\mu \to sup$$

$$x_1, x_2 \geq 0$$

$$x_1 + x_2 = 1$$

$$0{,}35x_1 + 0{,}98x_2 - \mu \geq 0$$

$$0{,}89x_1 + 0{,}15x_2 - \mu \geq 0$$

Для решения задачи был использован пакет языка R «Rglpk» [12]. Полученное решение для матрицы выплат агрессивного игрока:

$$x_1 = 0{,}61, x_2 = 0{,}39.$$

Аналогично посчитав для состояний «умеренного» и «защитного» игрока, получаем следующую матрицу B (таблица 2.6).

Таблица 2.6 – Матрица B

Зона / игрок	Открытая зона	Центральная зона
Агрессивный игрок	0,61	0,39
Умеренный игрок	0,54	0,46
Защитный игрок	0,64	0,36

Для сравнения представим матрицу B, полученную в рассматриваемой статье ([14]), без учёта равновесия смешанных стратегий.

Таблица 2.7 – Матрица *B* из статьи «Using HMM in Strategic Games»

Зона / игрок	Открытая зона	Центральная зона
Агрессивный игрок	0,9	0,1
Умеренный игрок	0,6	0,4
Защитный игрок	0,2	0,8

Оценим работу алгоритма на примере двух матриц: рассчитанной выше и представленной в статье.

2.1.2 Тестирование на данных, представленных в статье

Рассмотрим для начала сценарий «Агрессивный игрок» (рисунок 2.1).

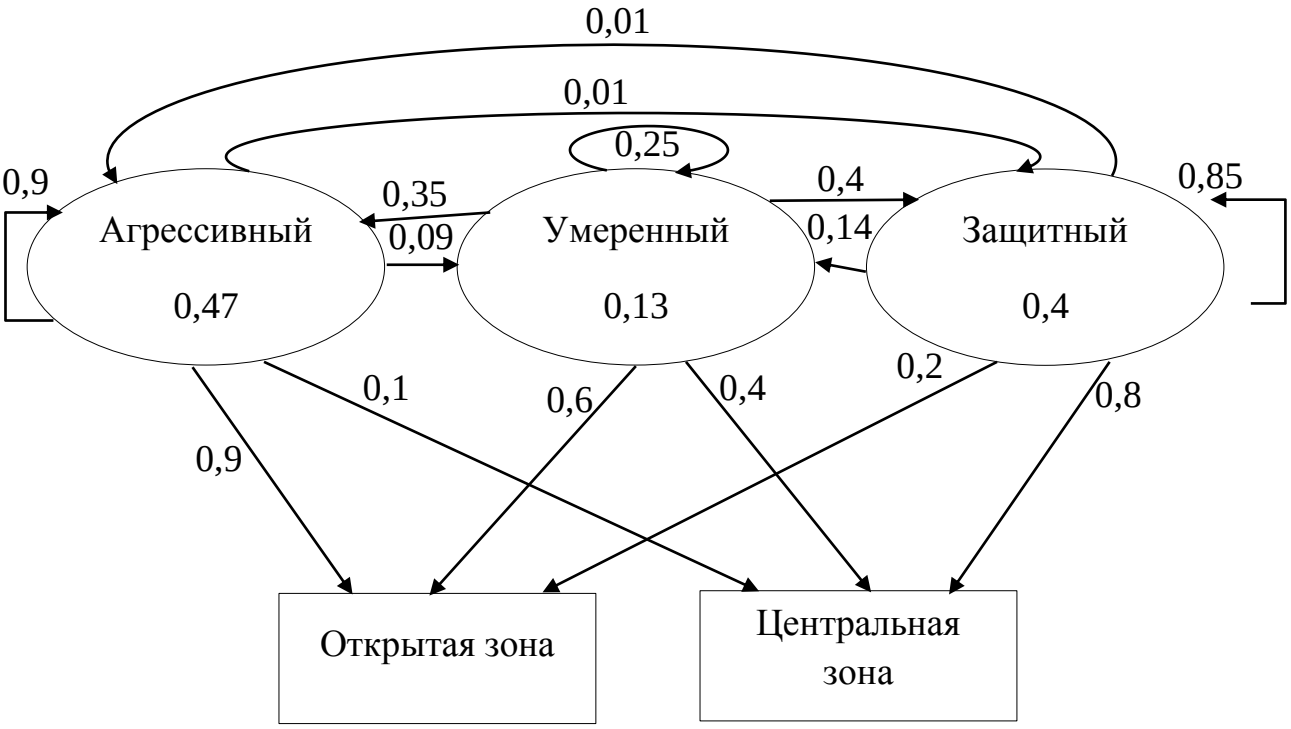

Рисунок 2.1 – Оригинальная CMM (Агрессивный игрок)

На каждом шаге игры применялся алгоритм Баума-Велша. Результат его работы на 1000 шаге можно видеть на рисунке 2.2.

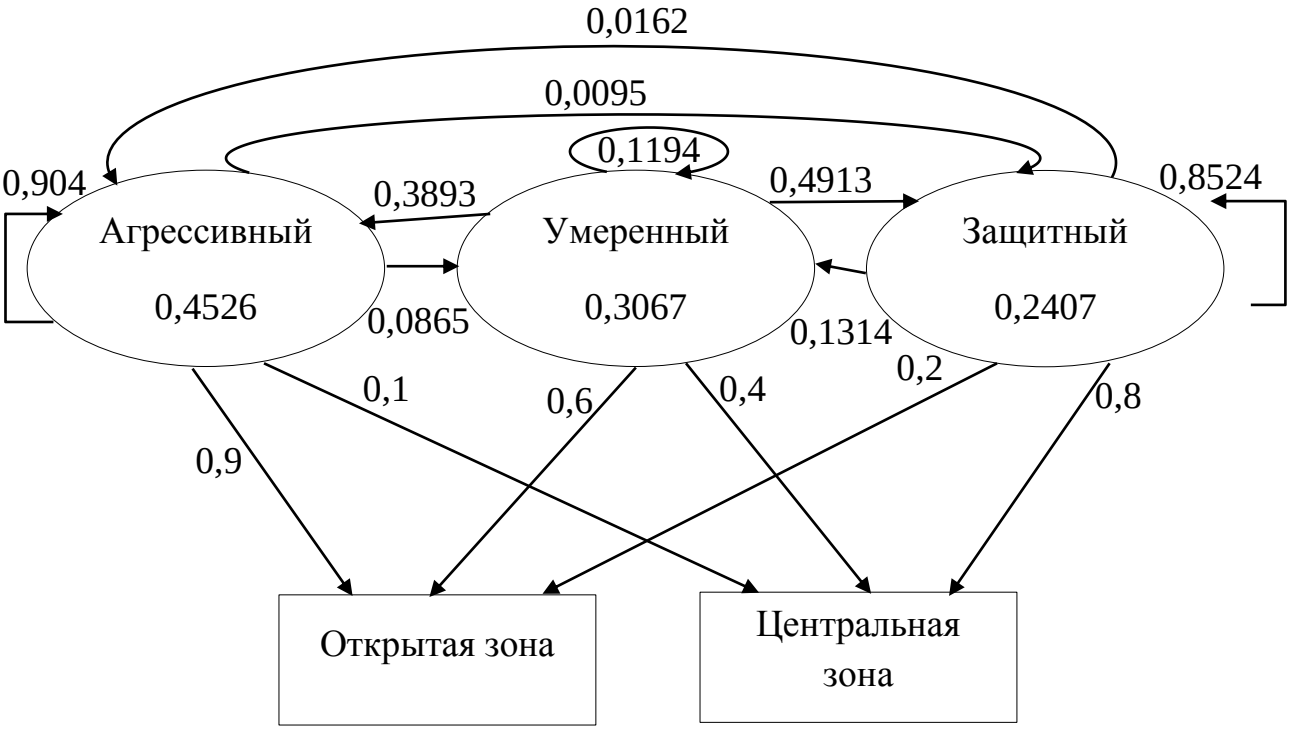

Рисунок 2.2 – Тренированная СММ (Агрессивный игрок) на 1000 шаге игры

После чего, применив алгоритм Витерби для полученной СММ, получаем последовательность предполагаемых состояний первого игрока. Теперь, зная все необходимые сведения об игроке 1, выбираем наиболее вероятное наблюдение для игрока 2. Сравнив полученные наблюдения, считаем результат хода и повторяем алгоритм Баума-Велша и Витерби, но уже на наблюдениях, количество которых на единицу больше.

Используя матрицы выплат, были рассчитаны результаты игры после каждых 200 ходов из 10000 (то есть в 50-ти партиях) и получили 47 выигрышей, что составляет 94% от общего количества игр.

Таблица 2.8 – Сравнение результатов (сценарий агрессивный игрок)

Модели	Предложенная в бакалаврской работе	Предложенная в статье [14]
Количество побед	94%	78%

Рассмотрим теперь аналогичным образом сценарий «Защитный игрок». Оригинальная СММ совпадает с представленной на рисунке 2.1 за

исключением вектора начальных состояний $\pi = 0{,}3; 0{,}13; 0{,}57$. Тренированная СММ, после применения алгоритма Баума-Велша:

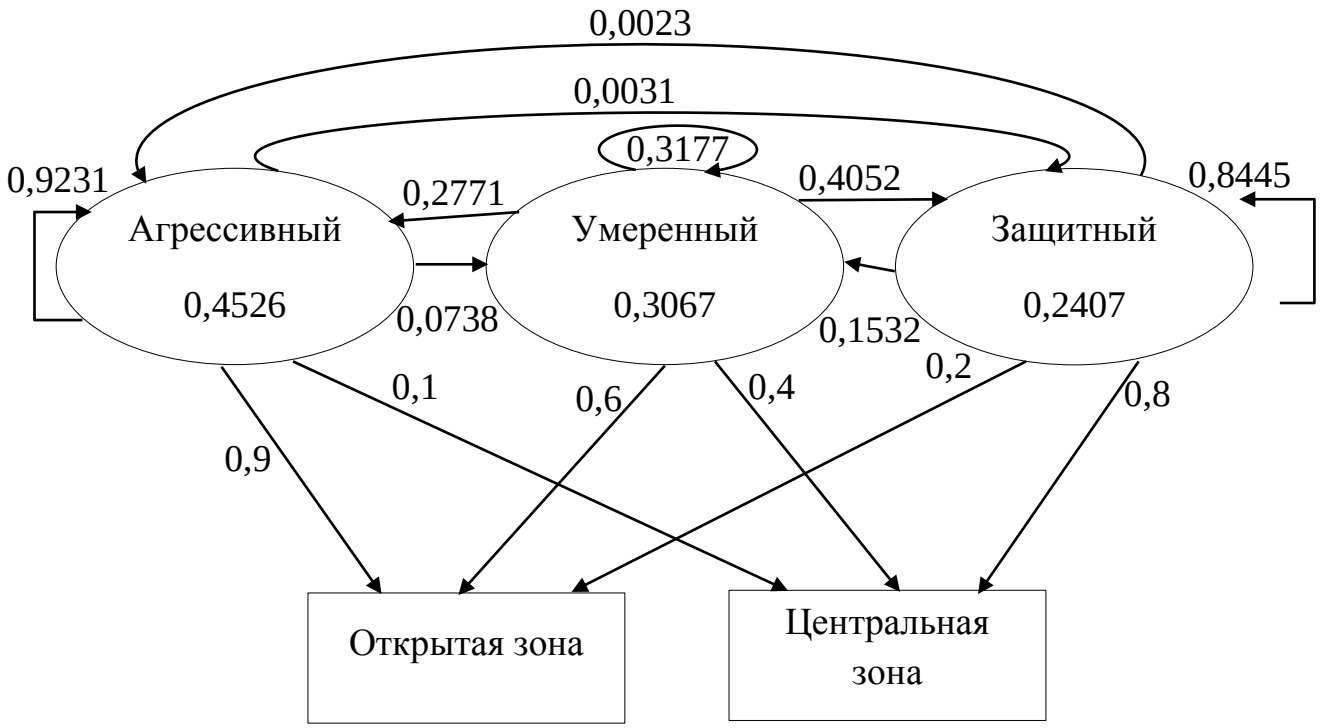

Рисунок 2.3 – Тренированная СММ (Защитный игрок) на 1000 шаге игры

Результат представлен в таблице 2.9.

Таблица 2.9 – Сравнение результатов (сценарий защитный игрок)

Модели	Предложенная в бакалаврской работе	Предложенная в статье [14]
Количество побед	92%	71%

Из полученных результатов видно, что алгоритм, предложенный в бакалаврской работе, сработал лучше, чем представленный в [14].

Особый интерес представляет количество ударов, отражённых принимающим игроком, т.е. сколько ходов из 10000 оказались выигрышными для него. Результат представлен в таблице 2.10.

Таблица 2.10 – Количество отражённых ударов

Сценарий	Агрессивный игрок	Защитный игрок
Количество отражённых ударов	74,53%	73,67%

Также были построены 95%-доверительные интервалы для двух сценариев:

- для сценария агрессивного игрока: $71{,}42\% < x < 76{,}24\%$;
- для сценария защитного игрока: $71{,}18\% < x < 76{,}64\%$.

2.1.3 Тестирование на данных матрицы B, найденной выше

Аналогичным образом алгоритм был протестирован для этой же игры при тех же сценариях (агрессивный игрок и защитный игрок), но уже с матрицей B, найденной выше. Оригинальная СММ для агрессивного игрока представлена на рисунке 2.4.

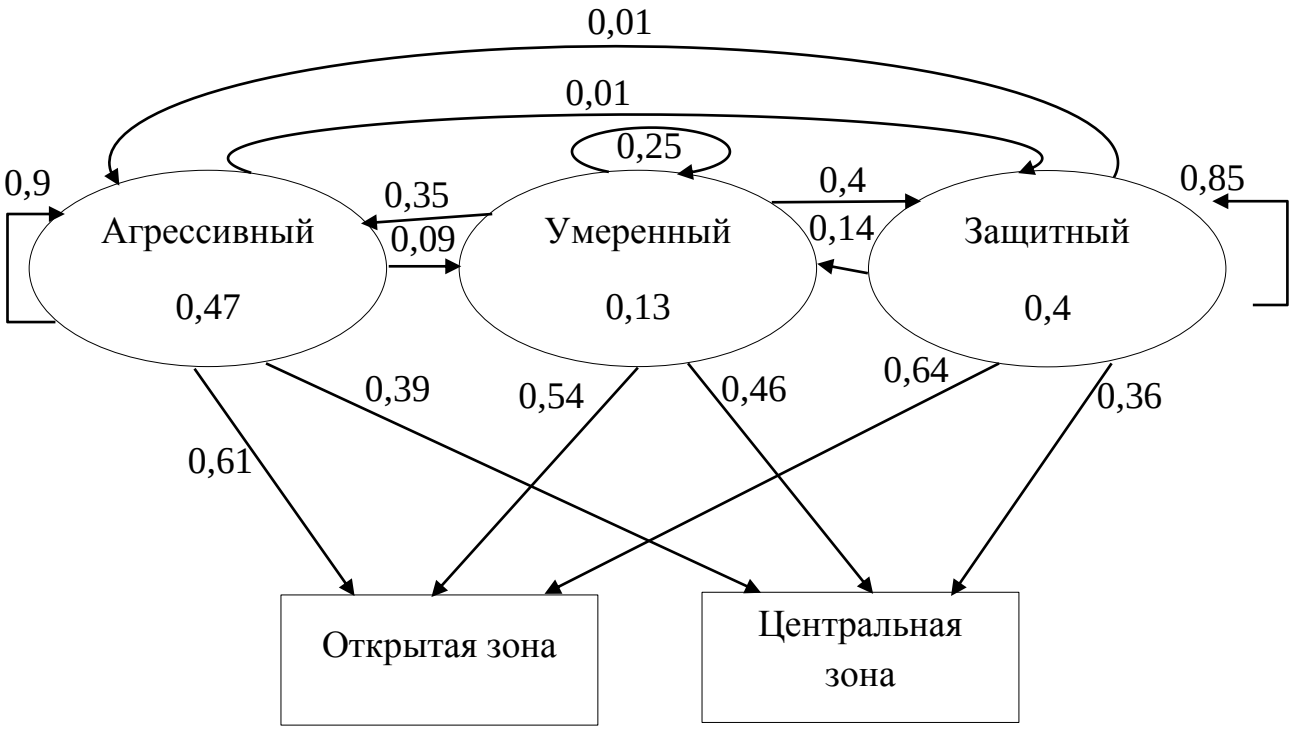

Рисунок 2.4 – Оригинальная СММ (Агрессивный игрок)

Оригинальная СММ защитного игрока совпадает с представленной на рисунке 2.4 за исключением вектора начальных состояний $\pi = \ 0{,}3; 0{,}13; 0{,}57$.

Результаты представлены в таблице 2.11.

Таблица 2.11 – Результаты при тестировании на данных, полученных выше

Сценарий	Результат, посчитанный по матрицам выплат	Количество отражённых ударов
Агрессивный игрок	62%	59,24%
Защитный игрок	62%	61,34%

95%-доверительные интервалы для количества отраженных ударов:

- для сценария агрессивного игрока: $56{,}16\% < x < 63{,}88\%$;

- для сценария защитного игрока: $55{,}79\% < x < 63{,}52\%$.

Видно, что результаты, полученные при тестировании алгоритма на данных, использующих матрицу выплат, найденную в разделе 2.1.1, оказались значительно хуже результатов, полученных при тестировании алгоритма на данных из статьи [14]. Это можно объяснить тем, что в матрице B, посчитанной в данной работе, вероятности подать в открытую зону и центральную очень близки друг к другу. При таких условиях достаточно большую роль в игре играет «случай», который невозможно контролировать. Таким образом, можно сделать следующий вывод: чем равномернее матрица B, тем хуже работает алгоритм, и наоборот.

2.1.4 Сравнение с другими подходами решения поставленной задачи

В данном разделе сравнивается эффективность (количество отражённых ударов) алгоритма решения поставленной задачи, представленного выше, с результатами, полученными с использованием других подходов, а именно:

1) игрок 2 использует равномерную стратегию (его поведение описывается СММ с равномерными матрицами A и B);

2) игрок 2 повторяет последний ход противника.

Эффективность предложенных подходов будет зависеть от равномерности стратегии соперника также, как и в представленном в работе алгоритме. Таким образом, протестируем представленный алгоритм решения, а также и предложенные подходы на нескольких различных СММ первого игрока.

Случай 1. Проверяем на СММ, представленной на рисунке 2.1 (считаем это неравномерным случаем по матрице A и матрице B).

Результаты сравнения с предложенными подходами представлены в таблице 2.12.

Таблица 2.12 – Результаты для случая 1

Подход	Представленный алгоритм	Равномерный подход	Выбор последнего действия
Отражённые удары	75,2%	49,8%	68,4%

Случай 2. Проверяем на СММ, представленной на рисунке 2.4 (считаем это неравномерным случаем по матрице A и равномерным по матрице B).

Результаты сравнения с предложенными подходами представлены в таблице 2.13.

Таблица 2.13 – Результаты для случая 2

Подход	Представленный алгоритм	Равномерный подход	Выбор последнего действия
Отражённые удары	61%	50,9%	52,8%

Случай 3. Рассмотрим также почти равномерную матрицу A и неравномерную матрицу B. Используем СММ, представленную на рисунке 2.5.

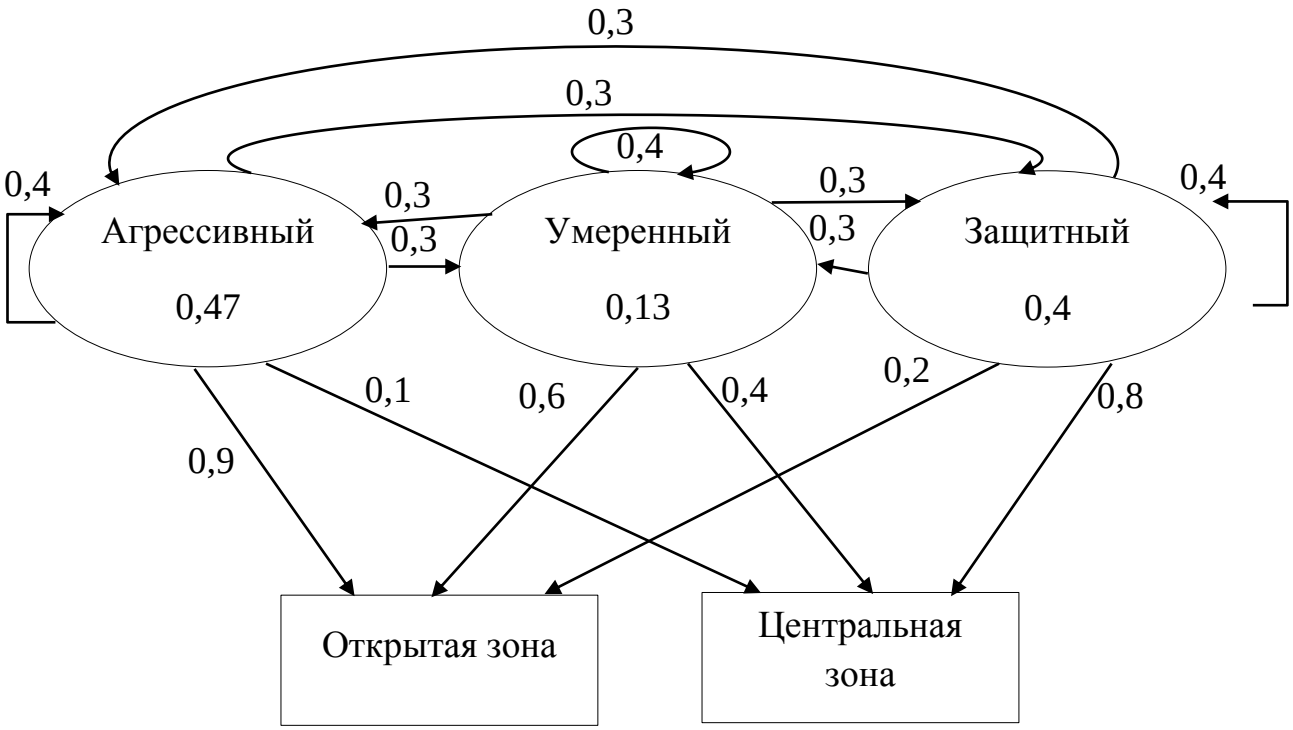

Рисунок 2.5 – СММ для случая 3

Результаты сравнения с предложенными подходами представлены в таблице 2.14.

Таблица 2.14 – Результаты для случая 3

Подход	Представленный алгоритм	Равномерный подход	Выбор последнего действия
Отражённые удары	63,1%	50,6%	53,4%

Случай 4. Последним случаем рассмотрим почти равномерную матрицу A и почти равномерную матрицу B:

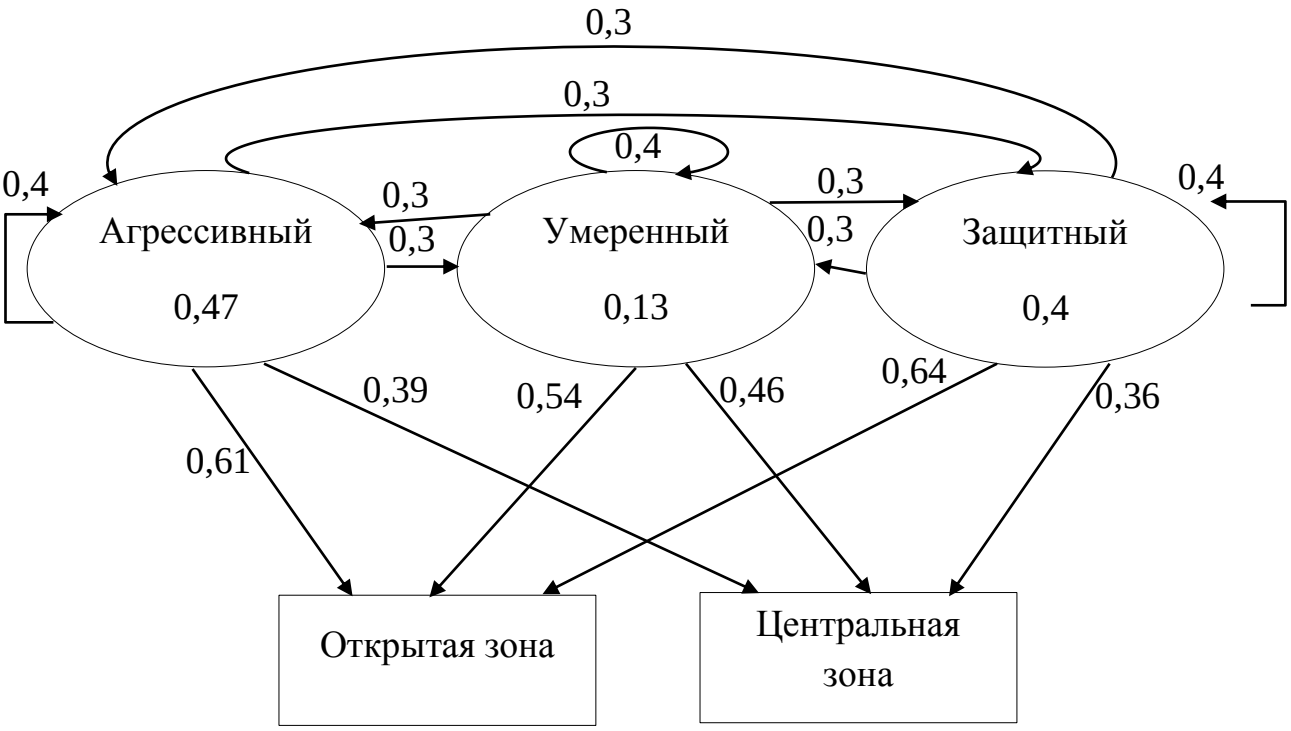

Рисунок 2.6 – СММ для случая 4

Результаты сравнения с предложенными подходами представлены в таблице 2.15.

Таблица 2.15 – Результаты для случая 4

Подход	Представленный алгоритм	Равномерный подход	Выбор последнего действия
Отражённые удары	59,7%	50,8%	51,4%

Сравнение результатов работы предложенного алгоритма с результатами применения других подходов к решению поставленной задачи показало, что алгоритм, предложенный в данной работе, показывает лучшую эффективность работы по сравнению с рассмотренными альтернативными подходами.

2.2 Применение модели к другой игре со схожими характеристиками

В данном разделе в порядке демонстрации неплохой эффективности и универсальности предложенного алгоритма проиллюстрирована его работа на примере другой игры со схожими характеристиками. В качестве примера взята известная игра «Камень, ножницы, бумага». Правила игры следующие.

1. Игроки одновременно делают ход, в котором могут поставить камень, ножницы или бумагу.

2. Побеждает в ходе тот, кто выставил более выигрышную фигуру.

3. Камень побеждает ножницы, ножницы побеждают бумагу, бумага побеждает камень (рисунок 2.7).

4. Считается результат хода.

5. Далее ход может повторяться, то есть игра является повторной.

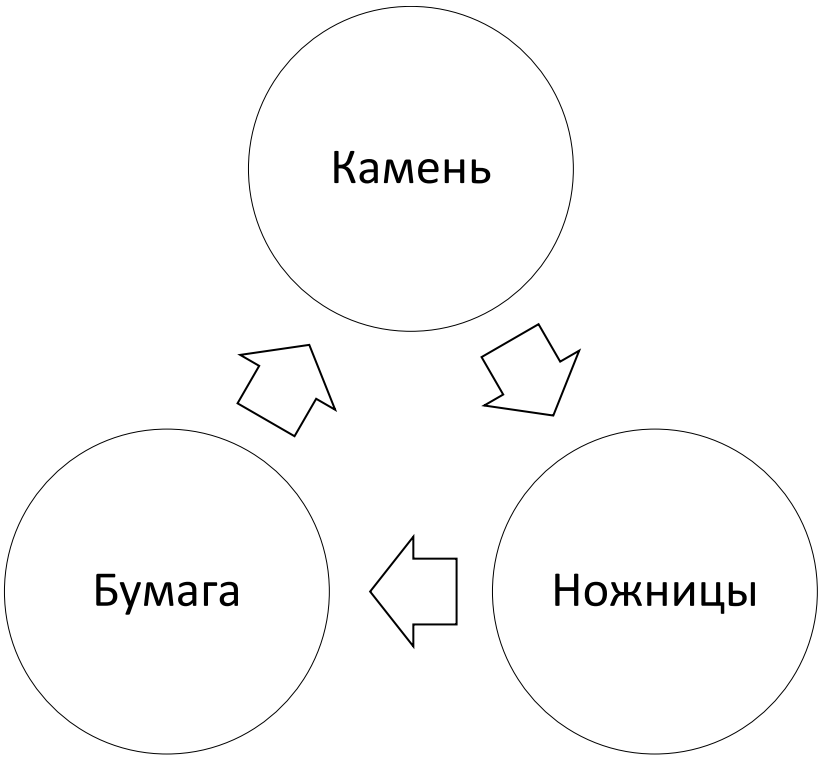

Рисунок 2.7 – Игра «Камень, ножницы, бумага»

Опишем рассматриваемую модель игры.

1. Два игрока, один из которых не знает скрытых состояний и переходов между состояниями и наблюдениями другого.

2. Первый игрок может принимать одно из трёх состояний:

- выгоднее ставить «камень»;

- выгоднее ставить «ножницы»;

- выгоднее ставить «бумагу».

3. Вектор начала состояний $\pi = \frac{1}{3}; \frac{1}{3}; \frac{1}{3}$.

4. Вероятности переходов между состояниями и наблюдениями представлены на рисунке 2.8.

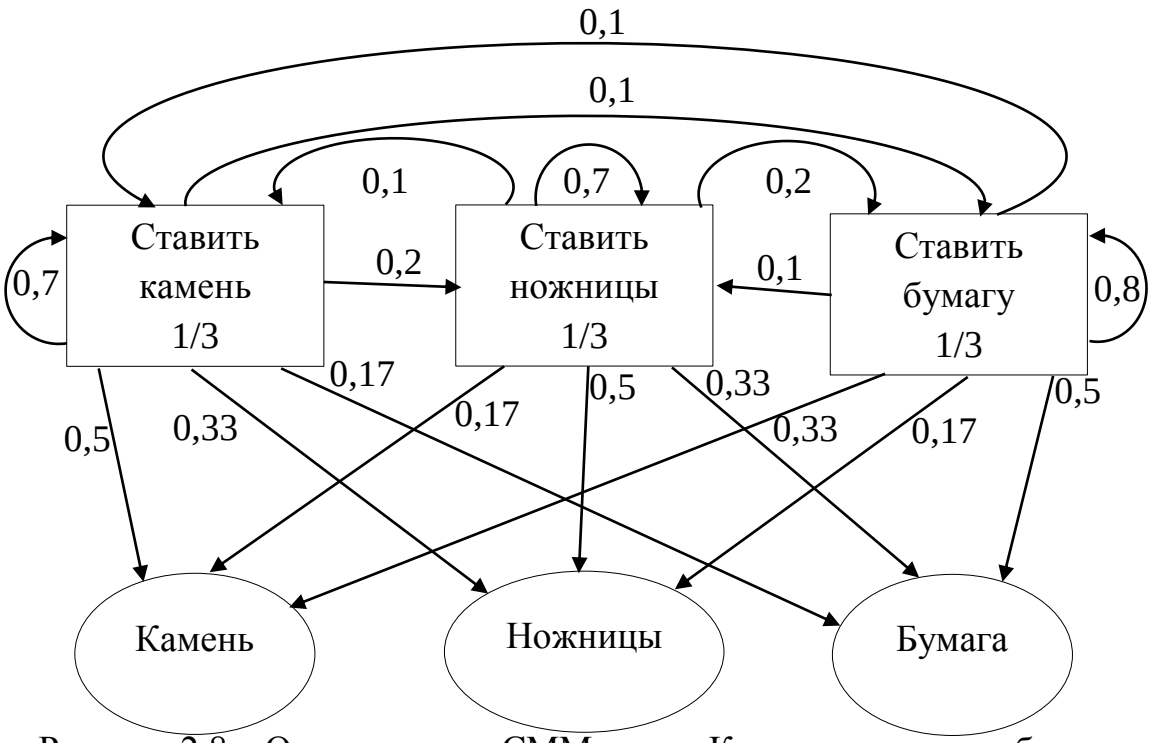

Рисунок 2.8 – Оригинальная СММ игры «Камень, ножницы, бумага»

Использованы следующие матрицы выплат (таблицы 2.16-2.18).

Таблица 2.16 – Матрица выплат (Выгодно ставить камень)

Состояния	Камень	Ножницы	Бумага
Камень	0,5 0,5	1 0	0 1
Ножницы	0,5 0,5	0,5 0,5	1 0
Бумага	1 0	0 1	0,5 0,5

Таблица 2.17 – Матрица выплат (Выгодно ставить ножницы)

Состояния	Камень	Ножницы	Бумага
Камень	0,5 0,5	1 0	0 1
Ножницы	0 1	0,5 0,5	1 0
Бумага	1 0	0,5 0,5	0,5 0,5

Таблица 2.18 – Матрица выплат (Выгодно ставить бумагу)

Состояния	Камень	Ножницы	Бумага
Камень	0,5 0,5	1 0	0,5 0,5
Ножницы	0 1	0,5 0,5	1 0
Бумага	1 0	0 1	0,5 0,5

Множество наблюдений состоит из трёх элементов: $O = \text{Камень}, \text{Ножницы}, \text{Бумага}$.

Тренированная СММ после применения алгоритма Баума-Велша на 1000 шаге игры представлена на рисунке 2.9.

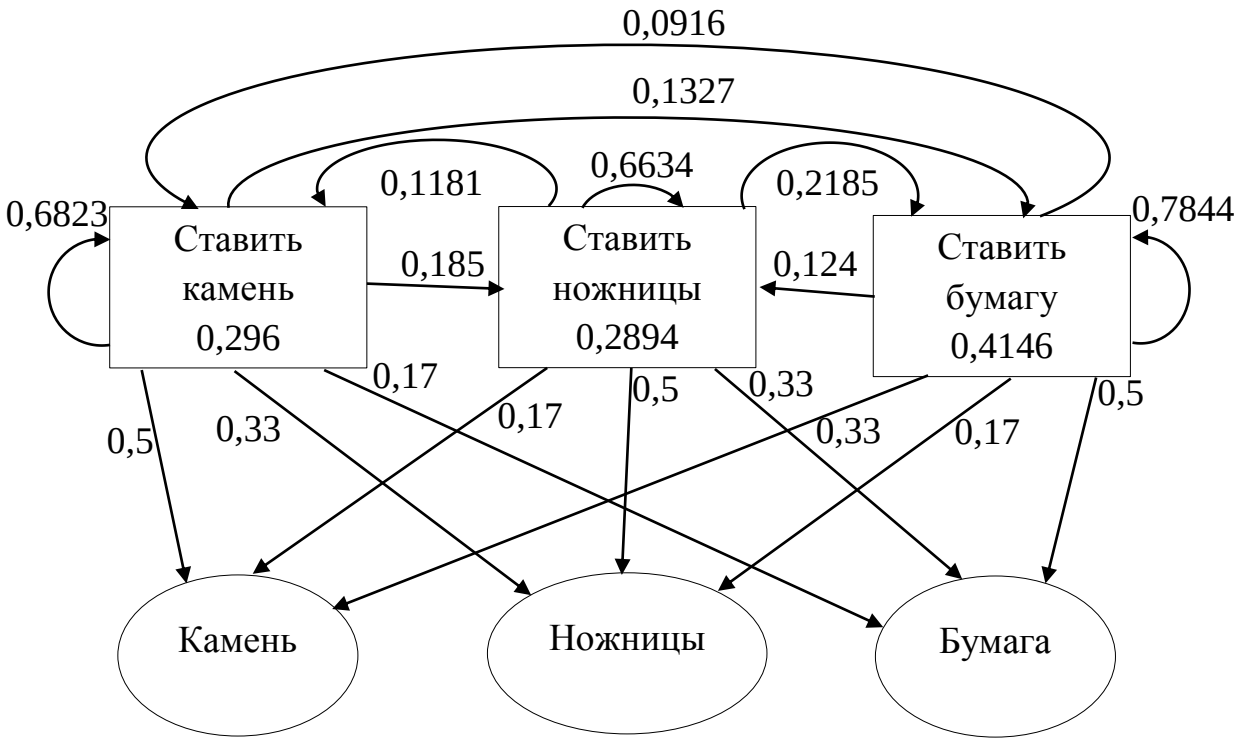

Рисунок 2.9 – Тренированная СММ игры «Камень, ножницы, бумага» на 1000 шаге игры

Сравнение полученных последовательностей состояний для оценки эффективности алгоритма Витерби показывает, что количество «угадываний» составило 79%.

Далее повторяем действия из раздела 2.1 с разницей в том, что для игрока 2 выбираем не наиболее вероятный ход (фигуру), а тот ход, который побеждает наиболее вероятный (побеждает наиболее вероятную фигуру).

Был посчитан результат игры по матрицам выплат после каждых 200 шагов (т.е. в 50-ти партиях), и также посчитано количество выигрышей (количество ситуаций, при которых игрок 2 выставил более выигрышную фигуру), и количество ситуаций «ничья» (игроки выставили одинаковые фигуры). Все результаты представлены в таблице 2.19.

Таблица 2.19 – Результаты работы алгоритма в игре «Камень, ножницы, бумага»

Параметр	По матрицам выплат	Количество «выигрышей»	Количество ситуаций «ничья»
Результат	76,7%	74,3%	15,2%

Полученные результаты оказались неплохими, что предоставляет возможность предполагать, что предложенный алгоритм применим к любой игре со схожими характеристиками. Для подтверждения этой гипотезы был проведён ещё один тест на модели игры «Камень, ножницы, бумага» со следующими шагами:

- игра была запущена 100 раз;

- при каждом запуске игры генерировалась рандомная матрица A, матрицы выплат по которым вычислялись элементы матрицы B и вектор начала состояний π;

- каждая игра содержала 10000 шагов;

- после каждой игры вычислялось количество выигрышей и ситуаций «ничья».

По результатам проведённого теста были построены гистограммы (рисунки 2.10-2.11).

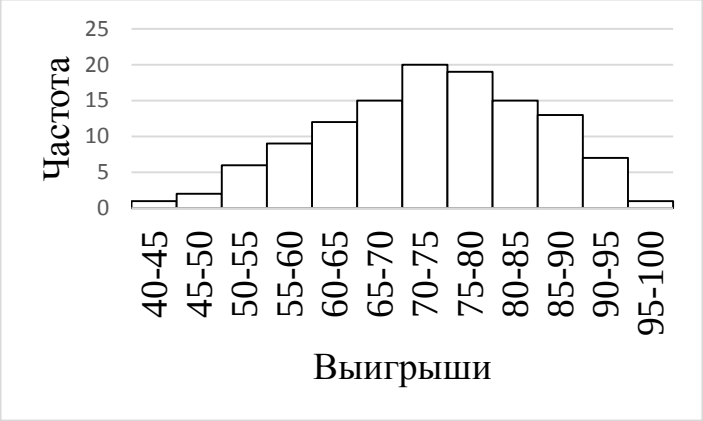

Рисунок 2.9 – Количество выигрышей

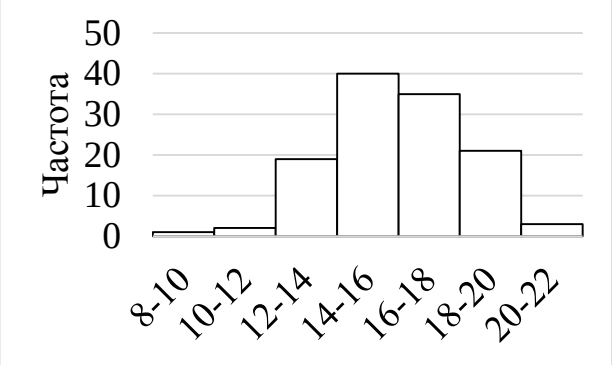

Рисунок 2.10 – Количество ситуаций «ничья»

Диаграммы показывают, что среднее количество выигрышей составило 72.3%, а среднее для ситуации «ничья» составило 14.9%.

Выводы ко второй главе

1. Предложенный алгоритм был протестирован на гипотетическом примере игры в «теннис».

2. Проведено сравнение результатов работы предложенного алгоритма с результатами применения других подходов к решению поставленной задачи.

3. Представленные диаграммы подтверждают гипотезу о том, что предложенный алгоритм применим к любой игре со схожими характеристиками.

ЗАКЛЮЧЕНИЕ

Таким образом, в данной работе:

1) представлен алгоритм решения частных случаев стохастической игры с отсутствием полной информации;

2) использованы СММ (скрытые марковские модели) в качестве инструмента описания данного процесса (стохастической игры);

3) проведён обзор литературы на близкие темы;

4) изучены алгоритмы Баума-Велша и Витерби как инструмент решения задач, описанных СММ.

Предложенный алгоритм был протестирован на гипотетическом примере игры в «теннис», описанном в [14]. Результаты выполнения алгоритма, предложенного в данной работе, оказались лучше результатов, представленных в [14].

Было проведено сравнение результатов работы предложенного алгоритма с результатами применения других подходов к решению поставленной задачи. В качестве альтернативных подходов были взяты следующие: использование равномерной стратегии игроком и выбор последнего хода противника. Было рассмотрено 4 случая СММ противника (комбинаций почти равномерных и неравномерных матриц A и B). В каждом из случаев предложенный алгоритм отработал лучше, чем альтернативные подходы.

Предложенный алгоритм был протестирован на произвольных данных другой игры со схожими характеристиками: «камень, ножницы, бумага». В ходе экспериментов алгоритм показал высокую эффективность работы алгоритма, что подтверждает возможность использования предложенного алгоритма в любой игре со схожими характеристиками.

Возможные направления дальнейшей деятельности:

- применение алгоритма к играм с большим количеством игроков;
- адаптация и применение других подходов к решению такого класса игр.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

Научная и методическая литература

1. Акулич И.Л. Математическое программирование в примерах и задачах: учебное пособие/ И.Л. Акулич – СПб.: Издательство «Лань», 2011. – 352 с.

2. Григорьева, К.В. Методические указания. Часть 1. Бескоалиционные игры в нормальной форме / К.В. Григорьева. – Санкт-Петербург: Факультет ПМ-ПУ СПбГУ, 2007. – 40 с.

3. Данилов, В.И. Лекции по теории игр / В.И. Данилов. – М.: Российская экономическая школа. – 2002. – 140 с.

4. Моргенштерн, О. Теория игр и экономическое поведение / О. Моргенштерн, Дж. Фон-Нейман. – М.: Книга по Требованию, 2012. – 708 с.

5. Морозов, В.В. Исследование операций в задачах и упражнениях / В.В. Морозов, А.Г. Сухарев, В.В. Федоров. – М.: КД Либроком, 2016. – 288 с.

6. Оуэн, Г. Теория игр / Г. Оуэн. – Пер. с англ. И.Н. Врублевский, Г.Н. Дюбина, А.Н. Ляпунова. Под ред. А.А. Корбута. – 2-е изд. – М.: Едиториал УРСС, 2004. – 216 с.

7. Петросян, Л. Теория игр / Л. Петросян, Н. Зенкевич, Е. Шевкопляс. – 2-е изд. – Санкт-Петербург: БХВ-Петербург, 2012. – 432 с.

8. Соколов А.В. Методы оптимальных решений. Общие положения. Математическое программирование / А.В. Соколов, В.В. Токарев. – М.: Физматлит, 2011. – 564 с.

9. Харшаньи, Дж. Игры с неполной информацией // Мировая экономическая мысль. Всемирное признание (лекции нобелевских лауреатов). Т. 5. Кн. 2. – М.: Мысль, 2005. – С. 44-63.

Электронные ресурсы

10. Himmelmann Lin. – 2010. – Режим доступа: https://github.com/cran/HMM/blob/master/R/HMM.r (дата обращения 15.06.2019).

11. Himmelmann Lin. Package «HMM». – 2010. Режим доступа: https://cran.r-project.org/web/packages/HMM/HMM.pdf (дата обращения 15.06.2019).

12. Theussl S. R/GNU Linear Programming Kit Interface / S. Theussl, K. Hornik, C. Buchta, F. Schwendinger. – 2017. – Режим доступа: https://cran.r-project.org/web/packages/Rglpk/Rglpk.pdf (дата обращения 15.06.2019).

Литература на иностранном языке

13. Aumann, R.J. Repeated games with incomplete information / R.J. Aumann, M. Maschler, R.E. Stearns. – Cambridge, MA.: MIT press, 1995. – 360 p.

14. Benevides, M. Using HMM in Strategic Games / M. Benevides, I. Lima, R. Nader, P. Rougemont // Proceedings 9th International Workshop on Developments in Computational Models, Buenos Aires, Argentina, 26 August 2013. Ed. by Mauricio Ayala-Rincon, Eduardo Bonelli, Ian Mackie. – Vol. 144 of Electronic Proceedings in Theoretical Computer Science. – Open Publishing Association, 2014. – Pp. 73-84.

15. Dymarski, P. Hidden Markov Models, Theory and Applications / P. Dymarski. – Rijeka: InTechOpen, 2011. – 314 p.

16. Fudenberg, D. The folk theorem in repeated games with discounting or with incomplete information / D. Fudenberg, E. Maskin // A Long-Run Collaboration On Long-Run Games. – World Scientific Publishers, 2009. – Pp. 209-230.

17. Gibbons, R. A primer in game theory / R. Gibbons. – New York: Harvester Wheatsheaf, 1992. – 267 p.

18. Martin, J.H. Beyond Dictionary Lookup: Text Analysis / J.H. Martin, D. Jurafsky // Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition. – Englewood Cliffs: Prentice Hall, 2009. – Pp. 122-141.

19. Rabiner L.R. A tutorial on hidden Markov models and selected applications in speech recognition // Proceedings of the IEEE. – 1989. – Vol. 77, №2. – Pp. 257-286.

20. Renault J. The value of Markov chain games with lack of information on one side // Mathematics of Operations Research. – 2006. – Vol. 31, №3. – Pp. 490-512.

ПРИЛОЖЕНИЕ А

Реализация алгоритма Баума-Велша из пакета «HMM»

Код взят из пакета «HMM» [10]. Приведённая ниже реализация была уже модифицирована с учётом специфики нашей задачи, а именно: была убрана оценка матрицы B и добавлена оценка вектора π.

```
1 baumWelch = function (hmm, observation, maxIterations=100, delta=1E-9, pseudoCount=0)
2 {
3     tempHmm = hmm
4     tempHmm$transProbs[is.na(hmm$transProbs)]=0
5     tempHmm$emissionProbs[is.na(hmm$emissionProbs)]=0
6     diff=c()
7     for(i in 1:maxIterations)
8     {
9         # Expectation Step (Calculate expected Transitions and Emissions)
10        bw = baumWelchRecursion (tempHmm, observation)
11        T=bw$TransitionMatrix
12        S=bw$startProbs
13        # E=bw$EmissionMatrix
14        # Pseudocounts
15        T[!is.na(hmm$transProbs)] = T[!is.na(hmm$transProbs)] + pseudoCount
16        # E[!is.na(hmm$emissionProbs)] = E[!is.na(hmm$emissionProbs)] + pseudoCount
17        # Maximization Step (Maximise Log-Likelihood for Transitions and Emissions-Probabilities)
18        T = (T/apply (T, 1, sum))
```

```
19          d = sqrt(sum((tempHmm$transProbs-T)^2)) #+
sqrt(sum((tempHmm$emissionProbs-E)^2))
20          diff = c (diff, d)
21          tempHmm$transProbs = T
22          tempHmm$startProbs = S
23          #tempHmm$emissionProbs = E
24          if (d<delta)
25          {
26              break
27          }
28      }
29      tempHmm$transProbs[is.na(hmm$transProbs)]=NA
30      tempHmm$emissionProbs[is.na(hmm$emissionProbs)]=NA
31      return(list(hmm=tempHmm, difference=diff))
32 }
33
34 baumWelchRecursion = function (hmm, observation)
35 {
36      TransitionMatrix = hmm$transProbs
37      TransitionMatrix[,]=0
38      EmissionMatrix=hmm$emissionProbs
39      EmissionMatrix[,]=0
40      f = forward (hmm, observation)
41      b = backward (hmm, observation)
42      probObservations=f[1, length(observation)]
43      for (i in 2: length (hmm$States))
44      {
45          j = f[i, length (observation)]
46          if (j > -Inf)
47          {
```

```
48                  probObservations = j + log (1+exp (probObservations-j))
49              }
50      }
51      for (x in hmm$States)
52      {
53              for (y in hmm$States)
54              {
55                      temp = f[x,1] + log (hmm$transProbs[x,y]) +
56                      log(hmm$emissionProbs[y, observation[1+1]]) + b[y,1+1]
57                      for (i in 2: (length (observation)-1))
58                      {
59                              j = f[x,i] + log (hmm$transProbs[x,y]) +
60                              log(hmm$emissionProbs[y,observation[i+1]])          +
b[y,i+1]
61                              if (j > -Inf)
62                              {
63                                      temp = j + log (1+exp (temp-j))
64                              }
65                      }
66                      temp = exp (temp-probObservations)
67                      TransitionMatrix[x,y]=temp
68              }
69      }
70      bwa <− backward (hmm, observation)
71      fwa <− forward (hmm, observation)
72      pia <− bwa [,1] *fwa[,1]
73      pia <− pia/sum(bwa[,1]*fwa[,1])
74      return    (list(TransitionMatrix=TransitionMatrix,    EmissionMatrix    =
EmissionMatrixstartProbs=pia))
75 }
```

ПРИЛОЖЕНИЕ Б

Реализация алгоритма Витерби из пакета «HMM»

Код взят из пакета «HMM» [11]. Данная реализация алгоритма использовалась без изменений.

```
1 viterbi = function (hmm, observation)
2 {
3       hmm$transProbs[is.na(hmm$transProbs)]=0
4       hmm$emissionProbs[is.na(hmm$emissionProbs)]=0
5       nObservations = length (observation)
6       nStates = length (hmm$States)
7       v = array (NA, c(nStates, nObservations))
8       dimnames(v) = list(states = hmm$States, index = 1: nObservations)
9       # Init
10      for (state in hmm$States)
11      {
12              v[state, 1] = log (hmm$startProbs[state] * hmm$emissionProbs[state, observation[1]])
13      }
14      # Iteration
15      for (k in 2: nObservations)
16      {
17              for (state in hmm$States)
18              {
19                      maxi = NULL
20                      for (previousState in hmm$States)
21                      {
22                              temp = v[previousState, k-1] + log(hmm$transProbs[previousState, state])
23                              maxi = max(maxi, temp )
```

```
24                  }
25                  v[state, k] = log(hmm$emissionProbs[state, observation[k]])
+ maxi
26              }
27      }
28      # Traceback
29      viterbiPath = rep (NA, nObservations)
30      for (state in hmm$States)
31      {
32              if (max(v[,nObservations])==v[state,nObservations])
33              {
34                      viterbiPath[nObservations] = state
35                      break
36              }
37      }
38      for (k in (nObservations-1): 1)
39      {
40              for (state in hmm$States)
41              {
42                      if (max(v[,k] + log(hmm$transProbs[,viterbiPath[k+1]])) ==
v[state, k]+ log (hmm$transProbs[state, viterbiPath[k+1]]))
43                      {
44                              viterbiPath[k] = state
45                              break
46                      }
47              }
48      }
49      return (viterbiPath)
50 }
```