Санкт-Петербургский государственный университет

Токарева Ксения Владимировна

Выпускная квалификационная работа

Реализация алгоритмов контрфактической минимизации сожаления в играх с неполной информацией

Уровень образования: бакалавриат

Направление *02.03.02 «Фундаментальная информатика и информационные технологии»*

ООП *СВ.5003.2018 «Программирование и информационные технологии»*

Научный руководитель:
доц. каф. информационных систем, к.ф.-м.н.
Еремин Алексей Сергеевич

Санкт-Петербург
2022

Содержание

Введение **3**

Постановка задачи **5**

Обзор литературы **6**

Глава 1. Базовые понятия теории игр **8**

Глава 2. Алгоритмы контрфактической минимизации сожалений **11**

Глава 3. Практическая часть **17**

Вывод **33**

Заключение **34**

Список литературы **35**

Введение

Математические модели все чаще используются социологами, экономистами и аналитиками для моделирования реальных процессов в обществе и последующего их анализа. Одними из таких моделей являются модели принятия решений в условиях, когда имеются две и более конкурирующих стороны.

Теория игр – это прикладной раздел математики, исследующий модели принятия решений в условиях конфликта. Изучая эти модели и применяя их на практике, можно оптимизировать многие процессы в экономике, а также предотвращать чрезвычайные ситуации. Так, например, по мнению Л. А. Петросяна и Н. А. Зенкевича, отсутствие исследований в области теории игр привело к финансовому кризису [10].

В 2000 году Харт и Мас-Колелл представили важный для теории игр алгоритм – "Regret matching" [7]. Игроки достигают равновесной игры, отслеживая сожаления за предыдущие игры и действуя в дальнейших раундах пропорционально накопленным положительным сожалениям.

Техника не только проста и интуитивна, она произвела революцию в компьютерном решении некоторых самых сложных игр с блефом. В частности, все побеждавшие с тех пор на ежегодных соревнованиях по компьютерному покеру программы основывались на ней [8].

Игры с несовершенной информацией моделируют стратегические взаимодействия между множеством агентов или игроков, обладающих частичной информацией. Они широко применяются в различных сферах жизни: в переговорах, аукционах, в кибербезопасности. Как правило, в таких играх каждый хочет найти приблизительное равновесие, при котором ни один игрок не сможет улучшить свой результат, отклонившись от равновесия в одиночку.

Контрфактическая минимизация сожаления – самый популярный алгоритм решения антагонистических игр с двумя игроками в развернутой форме с несовершенной информацией и достигает результатов на уровне развития области.

Существует несколько модификаций данного алгоритма, отличаю-

щихся скоростью сходимости. Данная работа посвящена исследованию этих модификаций.

Постановка задачи

Целью работы является изучение, реализация и сравнительный анализ алгоритмов контрфактической минимизации сожаления в играх с неполной информацией на примере игры Дудо.

Для достижения цели были поставлены следующие задачи:

1. Реализовать CFR с семплингом Монте-Карло.
2. Реализовать “Vanilla CFR” в векторной форме.
3. Реализовать CFR+ в векторной форме.
4. Реализовать семейство алгоритмов “Discounted CFR” в векторной форме.
5. Проведение сравнительного анализа эффективности реализованных алгоритмов.

Обзор литературы

Наиболее успешным для игр с неполной информацией стало семейство алгоритмов контрфактической минимизации сожаления — Counterfactual Regret Minimization (CFR) [5]. CFR – это итеративный алгоритм, сходящийся к равновесию по Нэшу в антагонистических играх. Формы табличного CFR были использованы во всех последних значимых работах в мире покера (Bowling et al., 2015; Moravčík et al., 2017; Brown & Sandholm, 2017) и во всех ежегодных соревнованиях по компьютерному покеру [1].

В статье [5] представлена новая техника для решения больших игр, основанная на минимизации сожалений, также представлено понятие контрфактического сожаления, показано, что минимизация контрфактического сожаления уменьшает сожаление в целом. Поэтому самостоятельная игра может быть применена для нахождения равновесия Нэша.

В статье “An Introduction to Counterfactual Regret Minimization” авторы определяют общепринятую нотацию концепции сожаления, объясняют базовые принципы контрфактической минимизации сожаления и детально описывают пример реализации алгоритма на примере игры «Покер Куна» [8]. Данную статью можно рассматривать как отличную вводную статью в предметную область.

Разработка алгоритма CFR+ стала ключевым прорывом в области. Во многих случаях CFR+ хотя бы на порядок быстрее, чем оригинальный CFR (“Vanilla CFR”) [4]. CFR+ был использован, чтобы существенно и приблизительно решить соответственно такие виды покера, как “heads-up limit Texas hold’em” (Bowling et al. 2015) и “heads-up no-limit Texas hold’em (HUNL)”, программа на основе алгоритма CFR+ одержала победу над профессиональными игроками покера [3].

Ноам Браун и Туомас Сандхольм в своей работе [3] представили новый обобщенный вариант CFR — “Discounted Regret Minimization”, который (1) демпфирует сожаления, то есть придает меньшие веса более ранним итерациям, для некоторых случаев для положительных и отрицательных сожалений по-разному, (2) дает разные весовые коэф-

фициенты итерациям для получения стратегий. Это привело к радикальному улучшению производительности во многих случаях. Авторы представили вариант алгоритма, который превзошел CFR+ — предыдущий алгоритм, считавшимся лучшим долгое время — в каждой протестированной игре.

Глава 1. Базовые понятия теории игр

Игра в нормальной форме – это тройка (N, A, u), где:
$N = \{1, ..., n\}$ – конечное множество n игроков,
S_i – конечное множество действий i-го игрока,
$A = \{S_1 \times ... \times S_n\}$ – множество всевозможных комбинаций одновременных действий всех игроков (каждая возможная комбинация одновременных действий называется профилем стратегий),
u – функция, сопоставляющая каждый профиль действий вектору полезности для каждого игрока.

Антагонистическая или *игра с нулевой суммой* — это игра, в которой сумма значений каждого вектора полезности равна нулю. Для двух игроков $u_1 = -u_2$.

Чистая стратегия. Игрок играет с чистой стратегией, если он выбирает одно действие с вероятностью 1.

Смешанные стратегии. Игрок имеет смешанную стратегию, если он имеет хотя бы два действия, реализующиеся с положительной вероятностью.

Через σ будем обращаться к смешанной стратегии. $\sigma_i(s)$ – вероятность того, что игрок i выберет действие $s \in S$.

Чтобы рассчитать *ожидаемую полезность (expected utility)* игры для игрока, нужно просуммировать по всем действиям в профиле стратегий произведение вероятности каждого игрока сыграть это действие и полезности профиля стратегий игрока:

$$u_i(\sigma_i, \sigma_{-i}) = \sum_{s \in S_i} \sum_{s \in S_{-i}} \sigma_i(s)\sigma_{-i}(s^{'})u_i(s, s^{'}).$$

Стратегия наилучшего ответа (best response strategy) для i-го иг-

рока: зная стратегии других игроков, максимизировать ожидаемую полезность (expected utility) для игрока i. Когда каждый игрок действует в соответствии со стратегией наилучшего ответа, комбинация стратегий игроков называется *равновесием по Нэшу*. И ни один игрок не может улучшить свою игру, изменив свою стратегию в одиночку.

Равновесие по Нэшу – это профиль стратегий σ [5]:

$$u_1(\sigma) \geqslant \max_{\sigma_1^{'} \in \sum_1} u_1(\sigma_1^{'}, \sigma_2),\ u_2(\sigma) \geqslant \max_{\sigma_2^{'} \in \sum_2} u_2(\sigma_1, \sigma_2^{'}).$$

Сожаление (regret). Сожаление от непринятия действия – разница между полезностью действия, которое игрок не выбрал, и полезностью действия, которое игрок действительно выбрал.

Regret matching. Действие игрока выбирается случайно в соответствии с распределением, пропорциональным положительным сожалениям от непринятия действий в прошлом.

Алгоритм “regret matching” для некоторого числа итераций:

- вычислить профиль стратегий в соответствии с сожалениями (“regret-matching”) (если все сожаления для игрока неположительные, то использовать равномерно распределённую случайную стратегию);
- добавить профиль стратегий к сумме профилей стратегий;
- выбрать профиль действий игрока в соответствии с профилем стратегий;
- рассчитать сожаления игрока;
- добавить сожаления игрока к его накопленным сожалениям.

После вернуть усредненную стратегию, то есть сумму всех найденных стратегий, поделенную на количество итераций.

Последовательная игра – игра, состоящая из последовательности действий, например, покер. Такие игры могут быть переформулированы как игры одного действия в нормальной форме, если представить, что игроки смотрят на розданные карты и каждый выбирает из чистых стратегий для каждого возможного развития игры заранее как переформулированное мета-действие.

Игры в развернутой форме – это естественная модель для последовательного принятия решений в присутствии других участников, принимающих решения, особенно в ситуациях с несовершенной информацией, где принимающий решение имеет различную информацию о состоянии игры.

Конечная игра в развернутой форме с неполной информацией. Развернутая форма описания игры указывает, какие ходы могут делать игроки, какой информацией они располагают, каковы размеры платежей в конце игры. Игра обычно описывается *деревом игры*; ветви дерева —ходы, которые могут делать игроки в сложившейся обстановке.

Дерево состоит из вершин и соединяющих их рёбер. Вершины подразделяются на терминальные (конечные) и нетерминальные. Каждая нетерминальная вершина характеризуется множеством допустимых ходов и доступной для игрока информацией или информационным набором. Терминальные вершины сообщают о размере выигрыша, получаемого по их достижении.

В развёрнутой форме можно представить и игры неполной информации. В этом случае игра начинается с хода природы, то есть некого случайного события.

Информационное множество (Information set) может содержать информацию об активном игроке и всю информацию, доступную ему в момент решения в игре.

Глава 2. Алгоритмы контрфактической минимизации сожалений

Ниже описана Минимизацию контрфактического сожаления использует алгоритм “Regret-matching”. Дополнительно (1) необходимо учитывать вероятности достижения каждого информационного набора по стратегиям игроков и, (2) так как игра обрабатывается итерационно по последовательности информационных наборов, есть передача «вперед» состояния игры и последовательности действий игроков, а также передача «назад» информации о полезности через эти информационные наборы.

Пусть A – множество всех действий игры, I – информационный набор, а $A(I)$ – множество доступных действий для информационного набор I. Пусть t и T – шаги по времени (t относится к каждому информационному набору и увеличивается с каждым посещением информационного набора). Стратегия σ_i^t для i-го игрока сопоставляет каждый информационный набор I_i i-го игрока и доступное ему действие $a \in A(I_i)$ вероятности того, что игрок выберет а в I_i во время t. Все стратегии вместе во время t формируют профиль стратегий σ^t. Пусть $\sigma_{I \to a}$ профиль стратегий, эквивалентный σ, за исключением того, что действие a будет всегда выбрано в информационном наборе I.

История h – последовательность действий, включая ход природы, начинается от корня дерева игры. Пусть $\pi^\sigma(h)$ вероятность достичь историю h с профилем стратегий σ, $\pi^\sigma(I)$ – вероятность достичь информационный набор I через все возможные истории игры в I, то есть $\pi^\sigma(I) = \sum_{h \in I} \pi^\sigma(h)$. Контрфактическая вероятность достичь информационный набор I, $\pi^\sigma_{-i}(h)$, – это вероятность достичь I с профилем стратегий σ, но в данной работе рассматриваются действия текущего игрока для достижения этого состояния с вероятностью 1. Во всех ситуациях, где говорится «контрфактический», расчет ведется так, как если бы стратегия i-го игрока была изменена, чтобы намеренно сыграть к информационному набору I_i. Иначе говоря, исключаются из вычислений вероятности, которые привели к игре i-го игрока.

Пусть Z – множество всех терминальных историй игры (последовательностей от корня к листу). Тогда правильный префикс $h \subset z$ для $z \in Z$ – нетерминальные истории игры. Пусть $u_i(z)$ полезность для игрока i терминальной истории z. Для упрощения расчетов определим *контрфактическое значение* в нетерминальной истории h:

$$v_i(\sigma, h) = \sum_{z \in Z, h \subset z} \pi^{\sigma}_{-i}(h)\pi^{\sigma}(h, z)u_i(z). \tag{1}$$

Контрфактическое сожаление от непринятия действия a при истории h вычисляется следующим образом:

$$r(h, a) = v_i(\sigma_{I \to a}, h) - v_i(\sigma, h).$$

Контрфактическое сожаление от непринятия действия a при информационном наборе I определяется следующей формулой:

$$r(I, a) = \sum_{h \in I} r(h, a).$$

Пусть $r^t_i(I, a)$ обозначает сожаление непринятия действия в информационном наборе I, принадлежащего игроку i, когда игроки используют σ^t. Накопленное контрфактическое сожаление определим как [8, 2]:

$$R^T_i(I, a) = \sum_{t=1}^{T} r^t_i(I, a),$$

$$R^T(I, a) = \sum_{t=1}^{T} v^{\sigma^t}_p(I \cdot a) - \sum_{t=1}^{T} \sum_{a \in A(I)} \sigma^t(I, a)v^{\sigma^t}_p(I \cdot a).$$

Разница между значением постоянного выбора действия a и ожидаемого значения, когда игроки используют σ, – это сожаление действия, которое тогда взвешено вероятностью, что другой игрок (возможно, игроки, включая шанс) будет играть, чтобы прийти в этот узел дерева. Если определить неотрицательное контрфактическое сожаление как $R^{T,+}_i(I, a) = \max(R^{T,+}_i(I, a), 0)$, то, применяя алгоритм "Regret-matching" Харта и Мас-Колелла к накопленным сожалениям, можно получить

новую стратегию:

$$\sigma_i^{T+1}(I,a) = \begin{cases} \frac{R_i^{T,+}(I,a)}{\sum_{a \in A(I)} R_i^{T,+}(I,a)}, & \sum_{a \in A(I)} R_i^{T,+}(I,a) > 0, \\ \frac{1}{|A(I)|}, & \text{иначе.} \end{cases}$$

Для каждого информационного набора это уравнение используется для подсчета вероятности действий пропорционально накопленным сожалениям. Для каждого действия CFR представляет следующее состояние игры и рекурсивно считает полезность от каждого действия. Сожаления рассчитываются по возвращаемым значениям, и значение игры к определенному узлу подсчитывается и возвращается.

Усредненный профиль стратегий при информационном наборе I, $\overline{\sigma^T}$ сходится к равновесию при $T \to \infty$ [8].

Глава 2.1. CFR с семплингом

Для больших игр проход по полному дереву игры в CFR может оказаться очень «дорогим». Как альтернативу, на каждой итерации алгоритма можно рассматривать меньшую, выборочную часть дерева, используя “Monte Carlo CFR” (MCCFR). Пусть $Q = \{Q_1, ..., Q_K\}$ — множество подмножеств, или блоков, терминальных историй Z таких, что объединение Q составляет Z. Например, “Chance Sampling” (CS), являющийся вариантом MCCFR, разбивает Z на блоки таким образом, что две истории находятся в одном блоке тогда и только тогда, когда все соответствующие случайные действия одинаковы.

На каждой итерации блок Q_j производит выборку с вероятностью q_j, где $\sum_{k=1}^{K} q_k = 1$. В CS блок генерируется выборкой одного действия a для каждой истории $h \in H$ с $P(h) = c$ в соответствии с тем, с какой вероятностью происходит $\sigma_c(h,a)$. В общем виде семплированное контрфактическое значение для игрока i:

$$\widetilde{v}_i(I,\sigma) = \sum_{z \in Z_I \cap Q_j} \frac{1}{q(z)} u_i(z) \pi^{\sigma}(z[I], z),$$

где $q(z) = \sum_{k:z \in Q_k} q_k$ – вероятность с которой z было выбрано. Опреде-

лим семплированное контрфактическое сожаление для действия a при информационном сете I как $\widetilde{r}_i^t(I,a) = \widetilde{v}_i(I, \sigma^t_{I\to a}) - \widetilde{v}_i(I, \sigma^t)$, далее стратегии рассчитываются по накопленным контрфактическим сожалениям $\widetilde{R}_i^T(I,a) = \sum_{t=1}^T \widetilde{r}_i^t(I,a)$.

С MCCFR итерации занимают меньше времени, чем в так называемом "Vanilla CFR" (оригинальный CFR, сформулированный в [5], так как нужно только пройти истории в Q, чтобы рассчитать семплированные контрфактические сожаления [6].

Глава 2.2. Vanilla CFR

CFR может быть применен без какого-либо семплирования, такая разновидность алгоритма называется "Vanilla CFR". В этом варианте просматривается все дерево на каждой итерации без применения семплинга. Есть четыре распространенных способа реализации "Vanilla CFR", они представлены в Таблице 1:

Одновременное обновление / скалярная форма	Переменное обновление / скалярная форма
Одновременное обновление / векторная форма	Переменное обновление / векторная форма

Таблица 1: *Способы реализации "Vanilla CFR"*

В случае одновременного обновления во время каждой итерации за один проход обновляются и сожаления обоих игроков, и их стратегии. В случае попеременного обновления есть два прохода, в каждом из который обновляются сожаления и стратегии только одного игрока [4].

Нам не удалось найти детального описания упомянутой «векторной формы» реализации алгоритмов CFR, поэтому был выведен собственный вариант интерпретации. Его описание представлено в разделе Программная реализация.

Глава 2.3. CFR+

Контрфактическая минимизация сожалений и ее варианты (например, CFR с семплингом и "Pure CFR") известны как наилучшие подхо-

ды к созданию приближенных решений равновесия по Нэшу для игр с несовершенной информацией, таких как покер. Алгоритм CFR+, описанный Oskari Tammelin в [4], обычно превосходит выше описанные алгоритмы на порядок или более с точки зрения времени вычислений, а также потенциально требует меньше памяти. CFR комбинирует алгоритм минимизации сожаления с шагом усреднения, чтобы достичь приблизительное равновесие Нэша. CFR+ изменяет обе эти части. CFR использует алгоритм “regret-matching” и возвращает одинаково взвешенные усредненные стратегии в качестве решения игры. CFR+ использует новый алгоритм минимизации сожаления “regret-matching+” и неравномерно взвешенную усредненную стратегию. Более того, CFR+ обязательно реализуется в векторной форме, с попеременным обновлением.

В CFR+ алгоритм “regret matching”, использующийся в CFR, заменен новым алгоритмом “regret matching+”. Разница между двумя алгоритмами в том, что “regret matching+” обнуляет любые отрицательные сожаления на каждой итерации, в то время как обычный “regret matching” хранит отрицательные значения сожалений, но обрабатывает их как нули во время расчета текущей стратегии [2].

Используя понятие контрфактической величины $v_i(\sigma, I)$, определим накопленное контрфактическое сожаление+ (cumulative counterfactual regret+) при информационном наборе I для действия a до времени T как:

$$R_i^{+,T}(I,a) = \begin{cases} \max\{v_i(\sigma^T_{I\to a}, I) - v_i(\sigma^T, I), 0\}, & T = 1, \\ \max\{R_i^{+,T-1}(I,a) + v_i(\sigma^T_{I\to a}, I) - v_i(\sigma^T, I), 0\}, & T > 1, \end{cases}$$

$$\sigma_i^{T+1}(I,a) = \begin{cases} \frac{R_i^{+,T}(I,a)}{\sum_{a' \in A(I)} R_i^{+,T}(I,a')}, & \sum_{a' \in A(I)} R_i^{+,T}(I,a') > 0, \\ \frac{1}{|A(I)|}, & \text{иначе.} \end{cases}$$

В отличие от CFR, в CFR+ на практике профиль текущей стратегии либо «почти» сходится, либо прямо сходится к приблизительному равновесию по Нэшу, поэтому нет необходимости в усреднении стратегий.

Однако, используя взвешенное усреднение, можно получить более быструю сходимость стратегий. Последовательность весовых коэффициентов, использованная в CFR+, определяется как: $\omega^T = \max\{T - d, 0\}$, где d – *задержка усреднения* в числе итераций [4].

Глава 2.4. Discounted CFR

Семейство алгоритмов "Discounted CFR" с параметрами α, β и γ ($DCFR_{\alpha,\beta,\gamma}$), определяется умножением накопленных положительных сожалений на коэффициент $\frac{t^\alpha}{t^\alpha+1}$, накопленных отрицательных сожалений на коэффициент $\frac{t^\beta}{t^\beta+1}$ и вкладов в усредненную стратегию на $(\frac{t}{t+1})^\gamma$ на каждой итерации t.

В то время как существует неограниченное число демпфирующих схем, которые сходятся в теории, не все они хорошо показывают себя на практике. Один из алгоритмов, неплохо работающий на практике, линейный CFR (linear CFR, LCFR), идентичен CFR, кроме того, что на итерации t обновлению сожаления и средней стратегии задается вес t, то есть итерации взвешены линейно. Аналогично можно умножать накопленные сожаления на $\frac{t}{t+1}$ на каждой итерации.

В этом случае, линейный CFR эквивалентен $DCFR_{1,1,1}$, потому что умножение сожаления на итерации t и вклад в среднюю стратегию на $\frac{t'}{t'+1}$ на каждой итерации $t \leq t' < T$ равносильно взвешиванию итерации t на $\frac{t}{T}$.

CFR+, в котором вклад итерации t в усредненную стратегию пропорционален t^2, эквивалентен $DCFR_{\infty,-\infty,2}$.

Оптимальный выбор коэффициентов α, β и γ зависит от конкретной решаемой игры. Ноам Браун и Туомас Сандхольм в своей работе [3] представили вывод, что DCFR с коэффициентами $\alpha = 3/2$, $\beta = 0$ и $\gamma = 2$ регулярно оказывается более производительным, чем CFR+.

Глава 3. Практическая часть

Глава 3.1. Правила игр

Ввиду сложности покера и общественного интереса к этой игре, алгоритмы CFR часто сравнивают на решении разных вариантов покера.

Данное исследование началось с самого простого покера – покера Куна, для которого известно точное равновесие по Нэшу.

Глава 3.1.1. Покер Куна

Покер Куна – это простая игра на 3 картах, придуманная Гарольдом У. Куном. Два игрока, каждый ставит 1 фишку, то есть ставит 1 фишку вслепую до самой игры. 3 карты, помеченные 1, 2, 3, тасуются, и каждому игроку выдается по одной карте, эта карта – личная информация. Игра начинается с хода первого игрока. По очереди игроки могут делать ставку или пропустить (пас). Игрок, решивший сделать ставку, добавляет одну фишку в банк к совокупности ставок. Если игрок делает пас после ставки другого игрока, то его противник забирает все фишки в банк. Когда произошли 2 ставки или 2 паса подряд, игроки раскрывают свои карты, и игрок с большей картой забирает все фишки из банка [8]. Все возможные итоги игры приведены в Таблице 2.

Последовательные действия			**Распределение фишек**
Игрок 1	Игрок 2	Игрок 1	
пас	пас		+1 игроку с большей картой
пас	ставка	пас	+1 игроку 2
пас	ставка	ставка	+2 игроку с большей картой
ставка	пас		+1 игроку 1
ставка	ставка		+2 игроку с большей картой

Таблица 2: *Распределение фишек в покере Куна в зависимости от действий игроков*

CFR, начинающий с равновесных стратегий, сходится к равновесию по Нэшу в смешанных стратегиях. Среднее значение полезности первого игрока при оптимальной игре должен сходиться к $\frac{1}{18} = -0.05(5)$ при количестве итераций стремящемся к бесконечности.

Глава 3.1.2. Дудо

Дудо – популярная игра в кости, имеющая множество интерпретаций и названий, на сегодняшний день не имеющая единого стандарта правил. В данной работе рассматривается наиболее доступный для понимания вариант.

Игроки сидят вокруг стола, все игроки одновременно кидают кости и аккуратно смотрят выпавшие значения, скрывая их от остальных. Начинающий игрок выдвигает *утверждение* о том, что выпало у всех игроков. И все игроки по кругу по очереди либо делают более сильное *заявление*, либо объявляют «Дудо» (в переводе с испанского «Я сомневаюсь»). Этот вызов (сомнение) заканчивает раунд, игроки поднимают свои колпачки, и один из двух участников, вовлеченных в вызов, теряет кости. Потерянные кости выставляются на обозрение игрокам.

Утверждения содержат положительное число костей и ранг этих костей, иначе их количество. В Дудо ранг 1 называется «джокер», что означает, что ранг 1 считается вместе с остальными рангами в результате. Обозначим утверждение об n костях ранга r как $n \times r$. Обычно, одно утверждение сильнее другого, когда есть увеличение в числе костей и/или в ранге. Исключение представляют утверждения 1 — «джокер».

Рассмотрим ниже игру «Дудо» для двух игроков, у каждого по одной кости. При таких условиях, сила утверждений будет пронумерована следующим образом, как показано в Таблице 3.

Игра идет по кругу от начинающего игрока, после все участники выдвигают строго усиливающиеся утверждения, пока один из игроков не скажет: «Дудо». Тогда все колпачки игроков поднимутся и кости проверяемого утверждения будут подсчитаны вместе с «джокером» и сравнены с утверждением. Далее возможны 3 случая:

Сила $s(n, r)$	0	1	2	3	4	5
Утверждение $n \times r$	1×2	1×3	1×4	1×5	1×6	1×1
Сила $s(n, r)$	6	7	8	9	10	11
Утверждение $n \times r$	2×2	2×3	2×4	2×5	2×6	2×1

Таблица 3: *Распределение фишек в покере Куна в зависимости от действий игроков*

- Настоящий ранг превышает ранг проверяемого утверждения. В этом случае усомнившийся теряет число костей, равное разности между настоящим рангом и рангом утверждения.

- Настоящий ранг меньше, чем ранг вызова. Тогда игрок, в утверждении которого было выражено сомнение, теряет количество костей, равное разности между рангом утверждения и настоящим рангом.

- Настоящий ранг совпадает с проверяемым рангом. В этом случае все игроки, кроме проверяемого игрока, теряют по кости.

Первый круг игры начинает произвольный игрок. Следующий круг начинает игрок, выигравший вызов. Когда игрок теряет все оставшиеся кости, он проигрывает и выходит из игры. Последний оставшийся игрок становится победителем.

Среднее значение полезности первого игрока при оптимальной игре $-\frac{7}{258} \approx -0,027$ [8].

Глава 3.2. Программная реализация

Ниже представлена «векторная форма» реализации алгоритмов CFR.

Каждый узел однозначно определяется своей *историей действий* h. При этом корневым узлом считается состояние игры после раздачи (бросков кубиков в Dudo). Матрица E_h содержит в себе ожидания выигрышей первого игрока для каждой пары рук (поскольку игра антагонистическая, матрица ожидаемых выигрышей второго игрока равна

$-E_h$):

$$E_h = \begin{pmatrix} e_{11h} & e_{12h} & \cdots & e_{1mh} \\ e_{21h} & e_{22h} & \cdots & e_{2mh} \\ \vdots & \vdots & \ddots & \vdots \\ e_{n1h} & e_{n2h} & \cdots & e_{nmh} \end{pmatrix}.$$

Здесь ijh — конкретное состояние игры. Значения e_{ijh} называют так же фукциями *полезности*.

Рассмотрим узлы, где ход совершает первый игрок. Пусть A_h^1 — множество доступных для него действий в узле h. Для каждого действия $a \in A_h^1$ и для каждого инфосета (руки) первого игрока $i = 1, ..., n$ определена вероятность σ_{iha}^1, так что $\sum_{a \in A_h^1} \sigma_{iha}^1 = 1$. Матрица $\Sigma_h^1 = \{\sigma_{iha}^1\}_{i=1,...,n}^{a \in A_h^1}$ представляет собой полную *стратегию* первого игрока в узле h.

Элементы матрицы E_h в этом случае связаны с элементами матриц E_{ha} узлов, которые получаются после хода a первого игрока, следующим соотношением:

$$e_{ijh} = \sum_{a \in A_h^1} \sigma_{iha}^1 e_{ijha}, \quad i = 1, ..., n, \quad j = 1, ..., m.$$

Если теперь обозначить $S_{ha}^1 = \operatorname{diag}(\sigma_{1ha}^1, \sigma_{2ha}^1, ..., \sigma_{nha}^1)$, можно записать предыдущее соотношение в матричном виде:

$$E_h = \sum_{a \in A_h^1} S_{ha}^1 E_{ha}.$$

Аналогично для узлов h, где ход совершает второй игрок, A_h^2 — множество доступных для него действий, σ_{jha}^2 — верятность совершения им хода a для инфосета $j = 1, ..., m$, $\Sigma_h^2 = \{\sigma_{jha}^2\}_{a \in A_h^2}^{j=1,...,m}$ — его полная стратегия, $S_{ha}^2 = \operatorname{diag}(\sigma_{1ha}^2, \sigma_{2ha}^2, ..., \sigma_{mha}^2)$.

Матрица E_h связана с матрицами E_{ha} узлов, которые получаются после хода a второго игрока как

$$E_h = \sum_{a \in A_h^2} E_{ha} S_{ha}^2.$$

Отметим, что любое состояние e_{ijh} имеет вероятность реализации, которая равна произведению вероятностей всех действий, которые к нему ведут (опустим из рассмотрения вероятность реализации событий $e_{ij\circ}$ — получения пары рук ij после раздачи, так как в рассматриваемых играх они все равновероятны и мы можем вынести это число за скобки и использовать его лишь для подсчёта средних выигрышей по всей игре). Эта вероятность равна

$$\pi_{ijh} = \prod_{\substack{h'a \subseteq h \\ h' \in H^1}} \sigma^1_{ih'a} \cdot \prod_{\substack{h'a \subseteq h \\ h' \in H^2}} \sigma^2_{ih'a},$$

где H^1 — множество узлов, где ходит первый игрок, а H^2 — множество узлов, где совершает ход второй. Обозначение $h'a \subseteq h$ означает, что история $h'a$ предшествует или равна истории h. Эти обозначения можно упростить, если положить

$$\sigma^1_{iha} := 1, \quad \forall h \in H^2, \quad \forall a \in A^2_h, \quad i = 1, ...n,$$

$$\sigma^2_{jha} := 1, \quad \forall h \in H^1, \quad \forall a \in A^1_h, \quad j = 1, ...m,$$

и соответственно задать матрицы

$$S^1_h a := I_{n\times n}, \quad \forall h \in H^2, \quad \forall a \in A^2_h,$$

$$S^2_{ha} := I_{m\times m}, \quad \forall h \in H^1, \quad \forall a \in A^1_h.$$

Тогда

$$\pi_{ijh} = \prod_{h'a \subseteq h} \sigma^1_{ih'a} \cdot \prod_{h'a \subseteq h} \sigma^2_{ih'a}.$$

Дополнительно введём вектор-столбцы $P^1_h = \left(\prod_{h'a \subseteq h} S^1_{h'a}\right) \cdot \mathbb{1}$ и вектор-строки $P^2_h = \mathbb{1}^T \cdot \left(\prod_{h'a \subseteq h} S^2_{h'a}\right)$, где $\mathbb{1}$ — вектор подходящей размерности, целиком состоящий из единиц.

В *контрфактической* минимизации мы для каждого узла рассматриваем значения

$$v^1_{ih} = \sum_{j=1}^{m} \pi^2_{jh} e_{ijh},$$

где $\pi^2_{jh} = \prod_{h'a \subseteq h} \sigma^2_{jh'a}$. При этом мы можем записать в матричной форме вектор-столбец из v^1_{ih} как $V^1_h = E_h P^2_h$.

Отметим, что для любого узла $h \in H^1$ справедливо $P^2_h = P^2_{ha}$, для любого действия $a \in A^1_h$, откуда

$$V^1_h = E_h P^2_h = \left(\sum_{a \in A^1_h} S^1_{ha} E_{ha} \right) P^2_h = \sum_{a \in A^1_h} S^1_{ha} V^1_{ha}.$$

В то же время для любого узла $h \in H^2$ выполняется

$$V^1_h = E_h P^2_h = \left(\sum_{a \in A^2_h} E_{ha} S^2_{ha} \right) P^2_h = \sum_{a \in A^2_h} E_{ha} S^2_{ha} P^2_h =$$

$$= \sum_{a \in A^2_h} E_{ha} P^2_{ha} = \sum_{a \in A^2_h} V^1_h.$$

Таким образом, для получения сожалений первого игрока, а стало быть и расчётов его стратегий в любом узле, нам вообще не нужно хранить матрицы из $n \times m$ значений, а достаточно векторов длиной n (кроме разве что процедуры расчёта выигрышей в терминальных узлах z, которые также можно представить процедурой, выдающей контрфактические векторы V^1_z).

Абсолютно аналогично для второго игрока (работаем с векторами-строками):

$$v^2_{ih} = \sum_{i=1}^{n} \pi^1_{ih} e_{ijh},$$

где $\pi^1_{ih} = \prod_{h'a \subseteq h} \sigma^1_{ih'a}$,

$$V^2_h = P^1_h E_h.$$

Для $h \in H^1$

$$V^2_h = \sum_{a \in A^1_h} V^2_h,$$

а для $h \in H^2$

$$V^2_h = \sum_{a \in A^2_h} V^2_{ha} S^2_{ha}.$$

Такой способ хранения и организации расчёта даёт существенную экономию памяти. Следует отметить, что семплинг в его рамках также возможен, но обновляя по одному элементу в векторах V_h и в стратегиях S_{ha} мы лишаемся возможности проводить параллельные операции над векторами.

В данной работе алгоритмы контрфактической минимизации сожаления были реализованы на языке программирования Python 3.9. Для поддержки дерева игры была выбрана библиотека pythreemap, реализованная на основе красно-черных деревьев [9].

Узлы дерева представлены классом Node, имеющим следующие поля: regretSum для хранения накопленных сожалений, strategy для текущей стратегии, strategySum для суммы стратегий для последующего расчета усредненной стратегии, сходящейся к равновесию по Нэшу, infoSet для информационного набора, num_actions для обозначения количества возможных действий в данном узле дерева. В данном классе есть следующие методы: getStrategy для расчета текущей стратегии, getAverageStrategy – для расчета усредненной по всем итерациям стратегии.

Второй класс KuhnTrainer или DudoTrainer используются для нахождения оптимальных стратегий для покера Куна и Дудо соответственно. Каждый класс имеет поле num_actions – максимально возможное в игре количество действий. Есть методы train, предназначенный для обучения, и cfr, являющийся рекурсивной реализацией алгоритма CFR.

Все программы имеют похожую структуру, описанную выше, с некоторыми различиями:

- CFR с семплингом Монте-Карло реализован в скалярной форме с одновременным обновлением,
- “Vanilla CFR” реализован в векторной форме с одновременным обновлением,
- CFR+ реализован в векторной форме с попеременным обновлением,

- “Discounted CFR” реализован в векторной форме с попеременным обновлением.

Метод cfr возвращает для терминальных вершин дерева выигрыш игрока, для остальных вершин возвращает ожидаемое контрфактическое значение согласно формуле 1. Условием выхода из рекурсии является определение того, является ли рассматриваемая вершина терминальной.

Текущая стратегия рассчитывается пропорционально накопленным контрфактическим сожалениям или, если еще нет накопленных положительных сожалений, задается равномерно распределенной.

Различия реализаций вариантов алгоритма CFR:

- Для CFR с CS используется тасовка карт Фишера–Йетса, популяризованная Дональдом Кнутом [8]. После раздачи карт или броска костей рассматривается поддерево игры.

- “Vanilla CFR” реализован без дополнений.

- В CFR+, из-за попеременного обновления, сожаления и накопленные сожаления изменяются только в случае, если в данном проходе по дереву рассматривается игрок, который в данной вершине делает свой ход. То есть сначала расчет новой стратегии идет для первого игрока, при следующей итерации для второго. В CFR+ хранятся только положительные сожаления, происходит взвешивание стратегий.

- В “Discounted CFR” остается попеременное обновление стратегий, добавляются разные весовые коэффициенты для отдельно накопленных положительных и отрицательных сожалений, так же на стратегии.

Глава 3.3. Сравнительный анализ

Глава 3.3.1. Сравнительный анализ на примере покера Куна

Известно, что среднее значение игры первого игрока при оптимальной игре должно сходиться к $\frac{1}{18} = -0.05(5)$ при количестве итераций, стремящемуся к бесконечности. При этом оптимальным существует бесконечное множество оптимальных стратегий первого игрока.

На Рисунках 1—6 представлены результаты применения вариантов алгоритмов CFR для решения игры «Покер Куна».

На Рисунке 1 представлены результаты CFR с CS за сто пятьдесят тысяч итераций и в промежутке [100000, 150000] итераций. Видно, что для стабильности результата требуется несколько сотен итераций, результат сильно зависит от случайности раздачи карт.

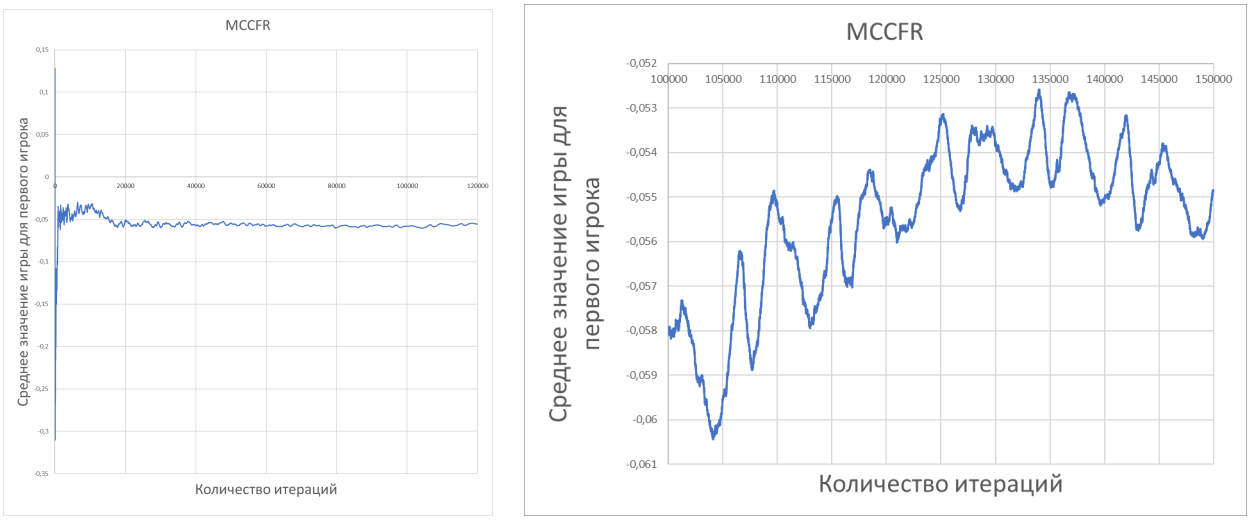

Рис. 1: *Графики зависимости среднего значения полезности первого игрока от количества итераций для алгоритма MCCFR.*

На Рисунке 2 представлены результаты “Vanilla CFR” за пятьдесят тысяч итераций. Среднее значение игры первого игрока колеблется около приблизительного значения игры, разброс сокращается с увеличением количества итераций алгоритма.

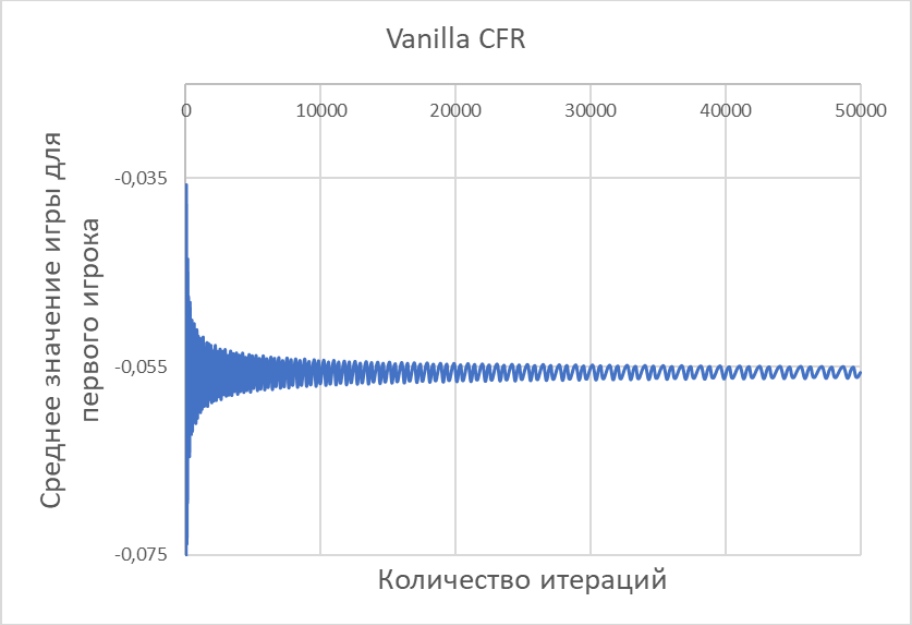

Рис. 2: *График зависимости среднего значения полезности первого игрока от количества итераций для алгоритма "Vanilla CFR".*

На Рисунке 3 представлены результаты $DCFR_{1,1,1}$ за десять тысяч итераций. Из графика видно, что $DCFR_{1,1,1}$ сходится очень быстро, уже после трехтысячной итерации достигается равновесие.

Рис. 3: *График зависимости среднего значения полезности первого игрока от количества итераций для алгоритма* $DCFR_{1,1,1}$.

На Рисунке 4 представлены результаты $DCFR_{\infty,-\infty,2}$ за двадцать пять тысяч итераций. $DCFR_{\infty,-\infty,2}$ сходится к приближенному равновесию по Нэшу гораздо медленнее, чем аналог линейного CFR.

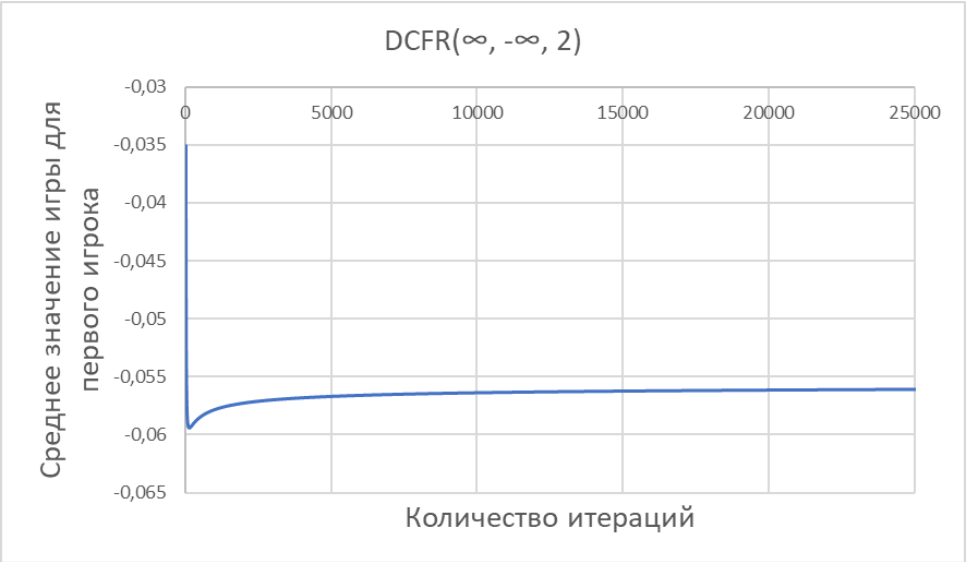

Рис. 4: *График зависимости среднего значения полезности первого игрока от количества итераций для алгоритма* $DCFR_{\infty,-\infty,2}$.

На Рисунке 5 представлены результаты $DCFR_{1.5,0,2}$ за две тысячи пятьсот итераций. Результат $DCFR_{1.5,0,2}$ близок по скорости к алгоритму $DCFR_{1,1,1}$.

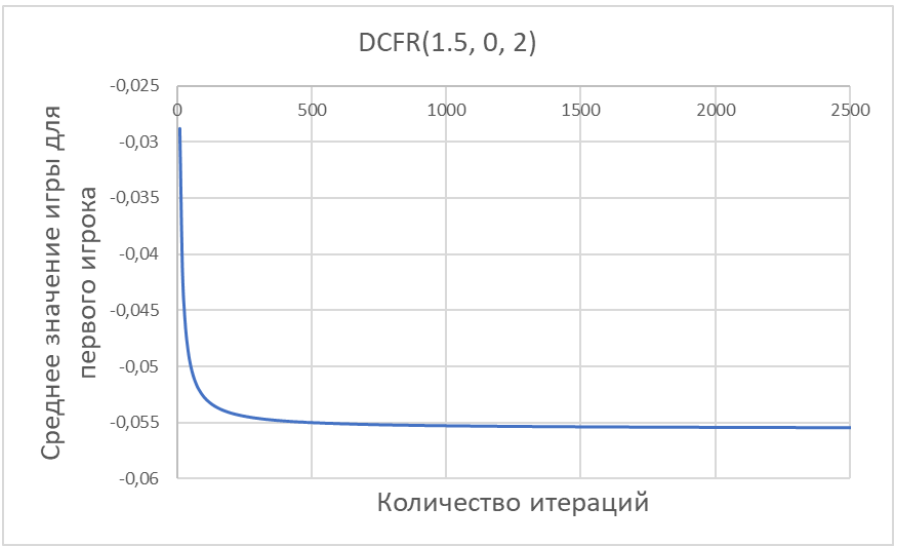

Рис. 5: *График зависимости среднего значения полезности первого игрока от количества итераций для алгоритма* $DCFR_{1.5,0,2}$.

На Рисунке 6 представлены результаты реализованных алгоритмов CFR в векторной форме за двадцать тысяч итераций.

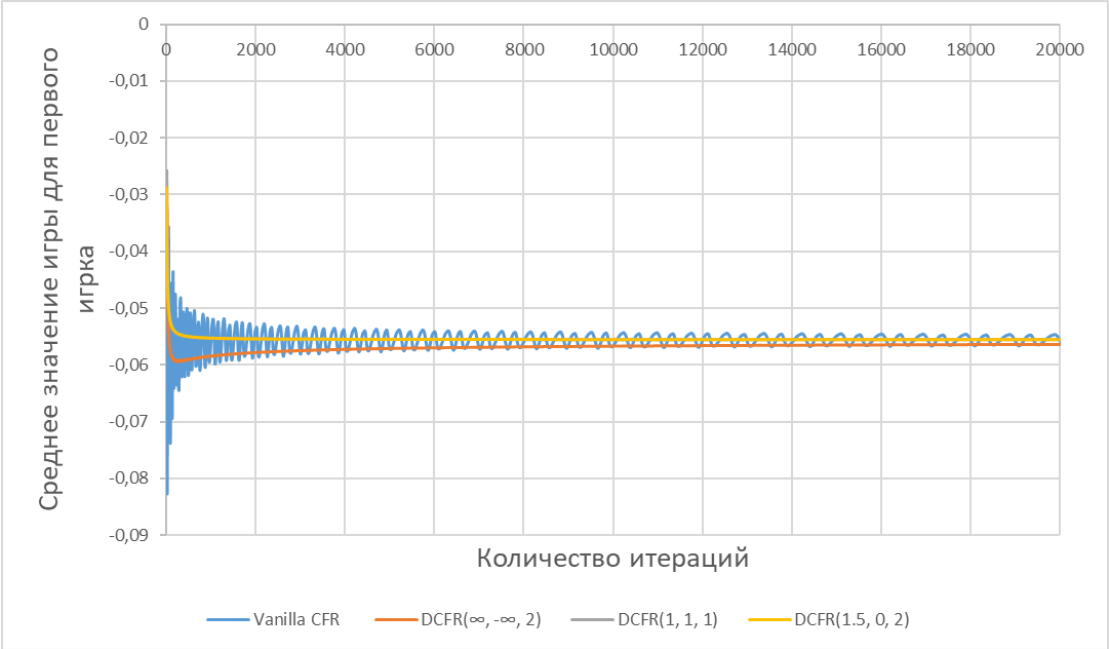

Рис. 6: *График зависимости среднего значения полезности первого игрока от количества итераций для всех векторных алгоритмов.*

На сводном графике линия $DCFR_{1.5,0,2}$ закрывает $DCFR_{1,1,1}$. Экземпляры алгоритма “Discounted CFR” стабильнее сходятся к результату, чем оригинальный CFR. Для покера Куна подтвердилось заявление авторов публикации [3]: $DCFR_{1.5,0,2}$ более быстр, чем $DCFR_{\infty,-\infty,2}$.

Глава 3.3.2. Сравнительный анализ на примере Дудо

Для игры «Дудо» также известно среднее значение игры первого игрока при оптимальной игре $-\frac{7}{258} \approx -0,027$

На Рисунках 7—13 представлены результаты применения вариантов алгоритмов CFR для решения игры «Дудо».

На Рисунке 7 представлены результаты CFR с CS за сто тысяч итераций и в интервале [80000, 100000] итераций. Для приближения к приближенному решению алгоритму нужны десятки тысяч итераций.

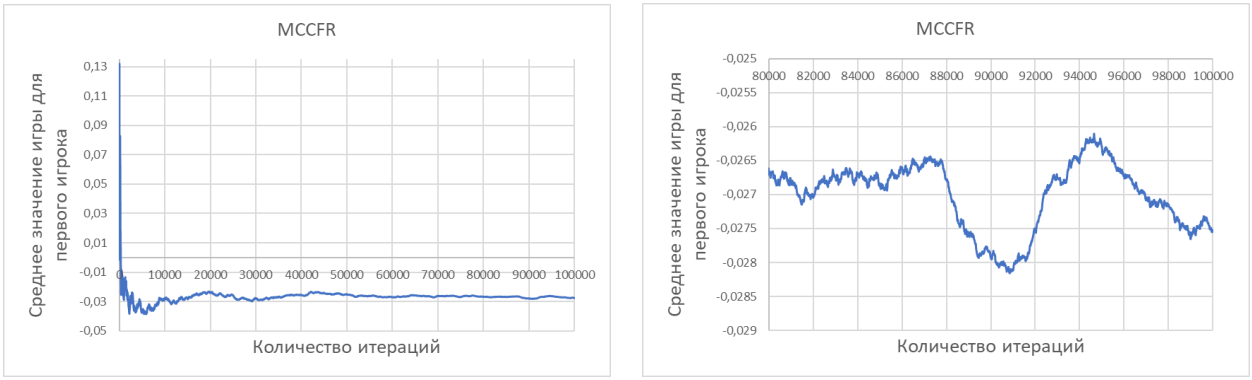

Рис. 7: *Графики зависимости среднего значения полезности первого игрока от количества итераций для алгоритма MCCFR.*

На Рисунке 8 представлены результаты “Vanilla CFR” за восемь тысяч итераций.

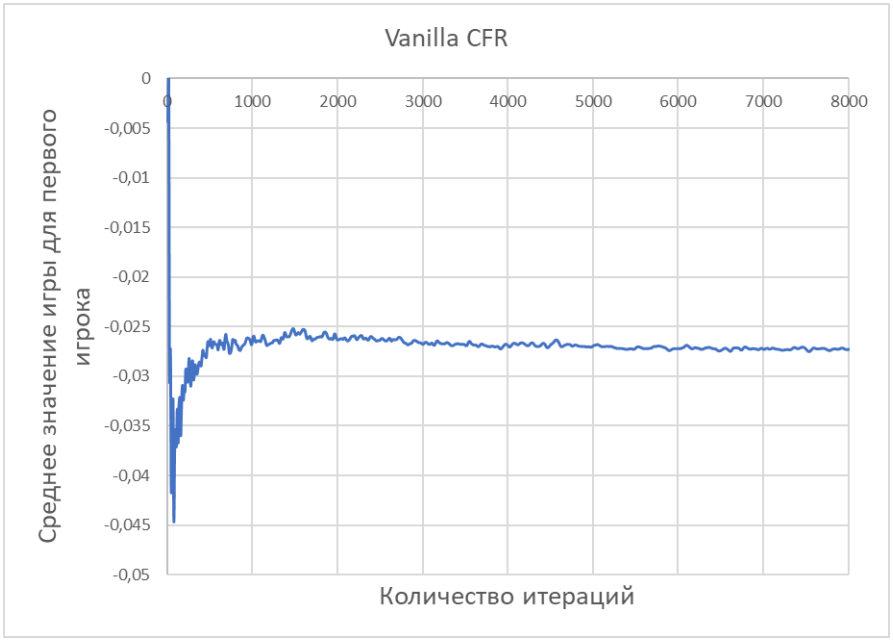

Рис. 8: *График зависимости среднего значения полезности первого игрока от количества итераций для алгоритма “Vanilla CFR”.*

На Рисунке 9 представлены результаты CFR+ за пятнадцать тысяч итераций. Для решения игры «Дудо» CFR+ оказался медленнее, чем оригинальный CFR.

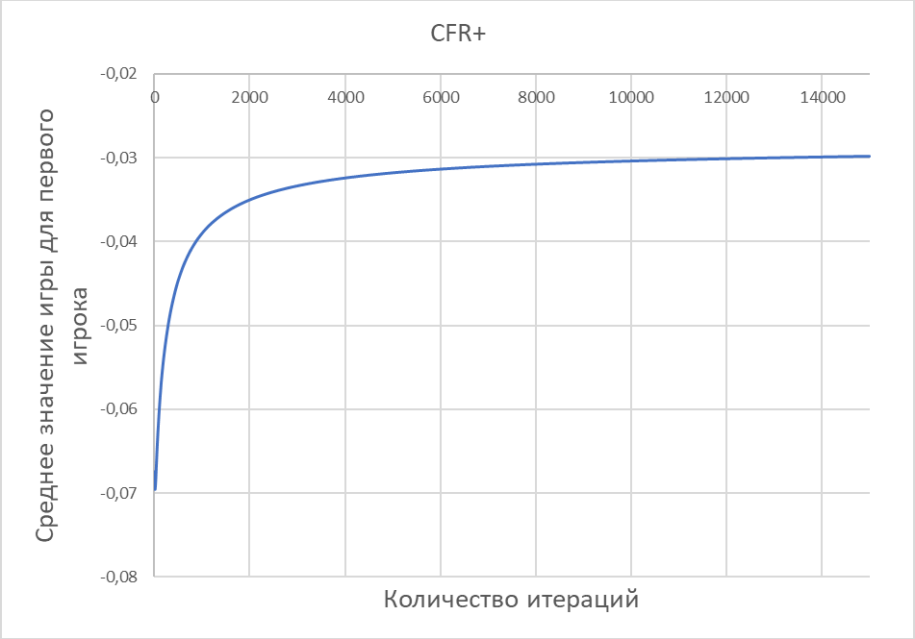

Рис. 9: *График зависимости среднего значения полезности первого игрока от количества итераций для алгоритма CFR+.*

На Рисунке 10 представлены результаты $DCFR_{1,1,1}$ за три тысячи итераций. Аналог линейно взвешенного CFR крайне быстро сходится к приближенному решению игры.

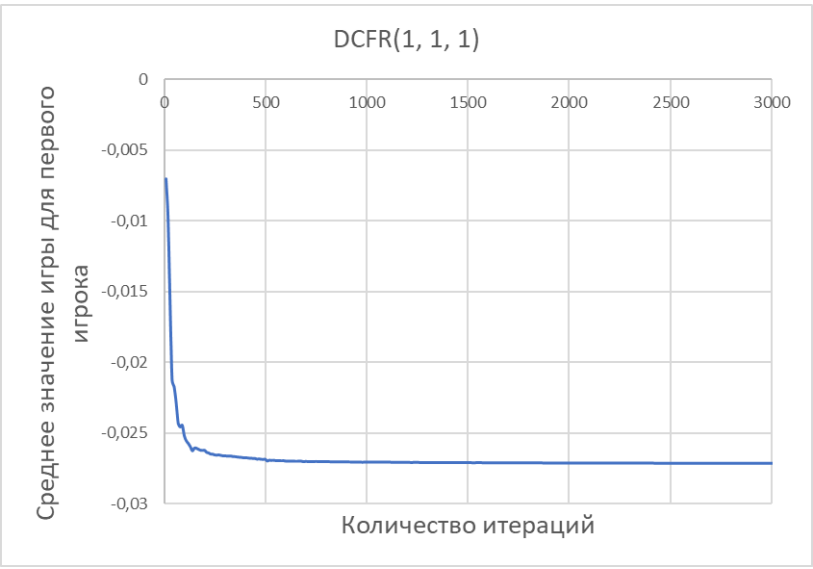

Рис. 10: *График зависимости среднего значения полезности первого игрока от количества итераций для алгоритма* $DCFR_{1,1,1}$.

На Рисунке 11 представлены результаты $DCFR_{\infty,-\infty,2}$ за пятнадцать

тысяч итераций. Взвешивание стратегий пропорционально t^2 не улучшило скорость сходимости, по сравнению с CFR+.

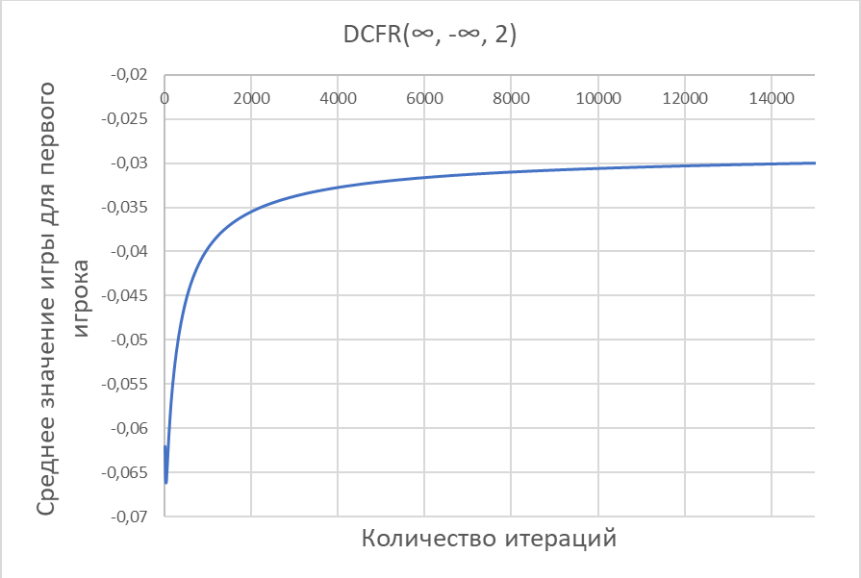

Рис. 11: *График зависимости среднего значения полезности первого игрока от количества итераций для алгоритма* $DCFR_{\infty,-\infty,2}$.

На Рисунке 12 представлены результаты $DCFR_{1.5,0,2}$ за две тысячи итераций. Представленный экземпляр DCFR сравним с $DCFR_{1,1,1}$, превосходит алгоритм CFR+ по скорости в несколько раз.

Рис. 12: *График зависимости среднего значения полезности первого игрока от количества итераций для алгоритма* $DCFR_{1.5,0,2}$.

На Рисунке 13 представлены результаты реализованных алгоритмов CFR в векторной форме за пятнадцать тысяч итераций.

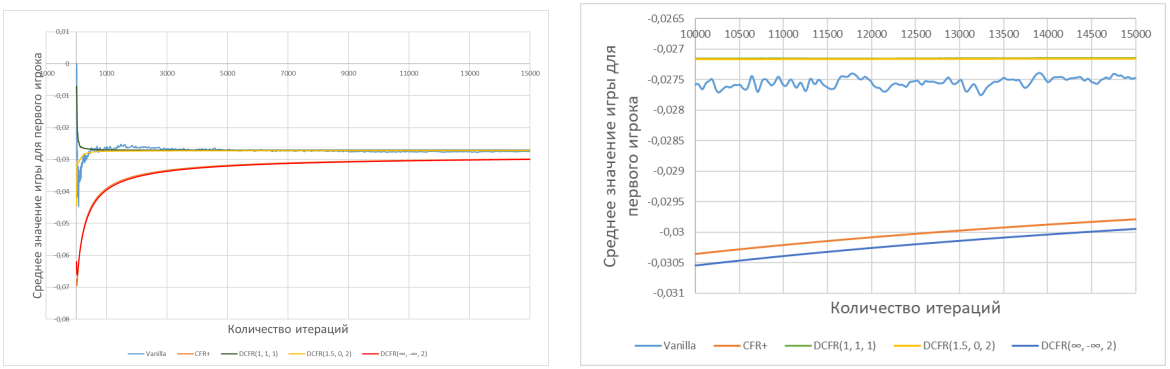

Рис. 13: *Графики зависимости среднего значения полезности первого игрока от количества итераций для всех векторных алгоритмов*

Лучше всего сходятся к равновесию экземпляры алгоритма DCFR: $DCFR_{1.5,0,2}$ и $DCFR_{1,1,1}$. “Vanilla CFR” немного медленнее и колеблется. CFR+ и $DCFR_{\infty,-\infty,2}$ гораздо дольше сходятся к приближенному равновесию.

Вывод

Подводя итоги, можно сказать, что в результате сравнительного анализа реализованных алгоритмов, лучше всего показали себя $DCFR_{1.5,0,2}$ и $DCFR_{1,1,1}$ в обеих играх и превзошли оба варианта CFR+. “Vanilla CFR” в векторной форме уступает по скорости и стабильности семейству алгоритмов “Discounted CFR”.

MCCFR позволил значительно ускорить вычисления за счет прохождения меньшего дерева игры, однако результат теряет стабильность за счет случайности моделирования раздачи карт или броска кубика.

Заключение

В результате работы над ВКР были решены следующие задачи:

1. Реализован CFR с семплингом Монте-Карло.
2. Реализован “Vanilla CFR” в векторной форме.
3. Реализован CFR+ в векторной форме.
4. Реализовано семейство алгоритмов “Discounted CFR” в векторной форме.
5. Проведен сравнительный анализ эффективности реализованных алгоритмов.

Исходный код: https://github.com/AirisFiorentini/CFR.

Список литературы

[1] Noam Brown, Adam Lerer, Sam Gross, Tuomas Sandholm. Deep Counterfactual Regret Minimization // arXiv:1811.00164. — 2019. — URL: https://proceedings.mlr.press/v97/brown19b/brown19b.pdf (online; accessed: 16.05.2022).

[2] Neil Burch. Time and Space: Why Imperfect Information Games are Hard : Ph. D. thesis / Burch Neil ; Department of Computing Science, University of Alberta. — 2017. — URL: https://tinyurl.com/NeilBurch (online; accessed: 25.05.2022).

[3] Noam Brown, Tuomas Sandholm. Solving Imperfect-Information Games via Discounted Regret Minimization // The Thirty-Third AAAI Conference on Artificial Intelligence (AAAI-19). — Computer Science Department, Carnegie Mellon University, 2019. — URL: https://tinyurl.com/DiscountedRegretMinimization (online; accessed: 26.05.2022).

[4] Oskari Tammelin. Solving Large Imperfect Information Games Using CFR+ // arXiv preprint arXiv:1407.5042. — 2014. — URL: https://arxiv.org/pdf/1407.5042.pdf (online; accessed: 15.04.2022).

[5] Martin Zinkevich, Michael Johanson, Michael Bowling, Carmelo Piccione. Regret minimization in games with incomplete information. — MIT Press, Cambridge, MA, 2007. — URL: https://tinyurl.com/RegretMinimization2007 (online; accessed: 25.05.2022).

[6] Richard Gibson. Regret Minimization in Games and the Development of Champion Multiplayer Computer Poker-Playing Agents : Ph. D. thesis / Gibson Richard ; Department of Computing Science, University of Alberta. — 2014. — URL: https://tinyurl.com/PhDGibson (online; accessed: 26.05.2022).

[7] Sergiu Hart, Andreu Mas-Collel. A Simple Adaptive Procedure

Leading to Correlated Equilibrium // Econometrica, Vol. 68. — 2000. — URL: https://www.ma.imperial.ac.uk/~dturaev/Hart0.pdf (online; accessed: 31.05.2022).

[8] Todd W. Neller, Marc Lanctot. An Introduction to Counterfactual Regret Minimization. — 2013. — URL: https://tinyurl.com/CFRInro2013 (online; accessed: 25.05.2022).

[9] Документация библиотеки pytreemap. — URL: https://gavinphr.github.io/pytreemap/ (online; accessed: 30.10.2021).

[10] Петросян Л. А., Зенкевич Н. А. Теория игр и социально-экономическое поведение // Электронная экономическая библиотека. — 1998. — URL: https://lukyanenko.at.ua/_ld/3/331_____.pdf (online; accessed: 29.05.2022).