Санкт-Петербургский государственный университет
информационных технологий, механики и оптики

Кафедра «Компьютерные технологии»

А.А. Терескин, Б.М. Ярцев

Метод выделения факторов, влияющих на решения объекта моделирования на примере игры «Покер техасский холдем»

Санкт-Петербург
2010

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В наше время моделирование поведения является важной и очень востребованной областью. Модели могут использоваться для предсказывания и прогнозирования. В контексте этих задач игры могут рассматриваться как хорошая тестовая площадка для проверки новых методов и технологий искусственного интеллекта.

Любой модели для хранения информации о реальных процессах необходим способ уменьшать все пространство состояний. Зачастую ограничения накладываемые на модели позволяют существенно упростить представление реального процесса. К примеру, в играх с неполной информацией построение абстракции дерева игры за счет объединения информационных множеств позволяет уменьшить объем информации хранимой в модели. За счет этого обучение происходит быстрее.

В настоящей работе приведен новый способ построения абстракции для эффективного представления состояний игры. Важно, что метод позволяет строить абстракцию игры индивидуально для каждого оппонента в зависимости от полученной в процессе игры информации о стратегии игрока, а также в зависимости от общего числа сыгранных партий. Данная абстракция используется при моделировании поведения агента. От качества построенной абстракции напрямую зависит результат моделирования оппонента, а следовательно и игры против него.

В третьей главе будут приведены результаты применения метода построения абстракции для построения контрстратегий против существующих агентов для игры в покер.

Глава 1. Основные понятия

В данной главе будут приведены основные понятия, которые в дальнейшем будут использованы в работе для описание алгоритма и результатов. Также в данной главе будет приведен краткий обзор существующих работ на тему моделирования и построения абстракций.

1.1. Представление игры в экстенсивной форме

Существуют разные типы представления игр. Самый привычный — в виде набора правил игры. Несмотря на то, что этот способ является простым, для того, чтобы можно было делать выводы о динамике игры и выигрыше, необходимо более формальное определение.

Достаточно часто используется способ представления игры в *экстенсивной форме*. Экстенсивная или расширенная форма игры дается в виде дерева, в котором вершины представляют состояния игры и корень дерева — начальное состояние. Есть два типа вершин:

- Вероятностные вершины — вершины, в которых происходит случайный процесс, переводящий игру в другое состояние
- Вершины принятия решения — состояния игры, в которых игрок совершает действие

На рисунке 1.1 приведен фрагмент дерева представления игры в экстенсивной форме для двух игроков. Вершина дерева (изображена в виде ромба) — вероятностная вершина, в которой происходит случайный процесс. Круглые вершины — вершины принятия решений одним игроком, квадратные — другим. Овалами, объединяющими несколько вершин, обозначены информационные множества.

Формальное определение многошаговой позиционной игры с неполной информацией [1]:

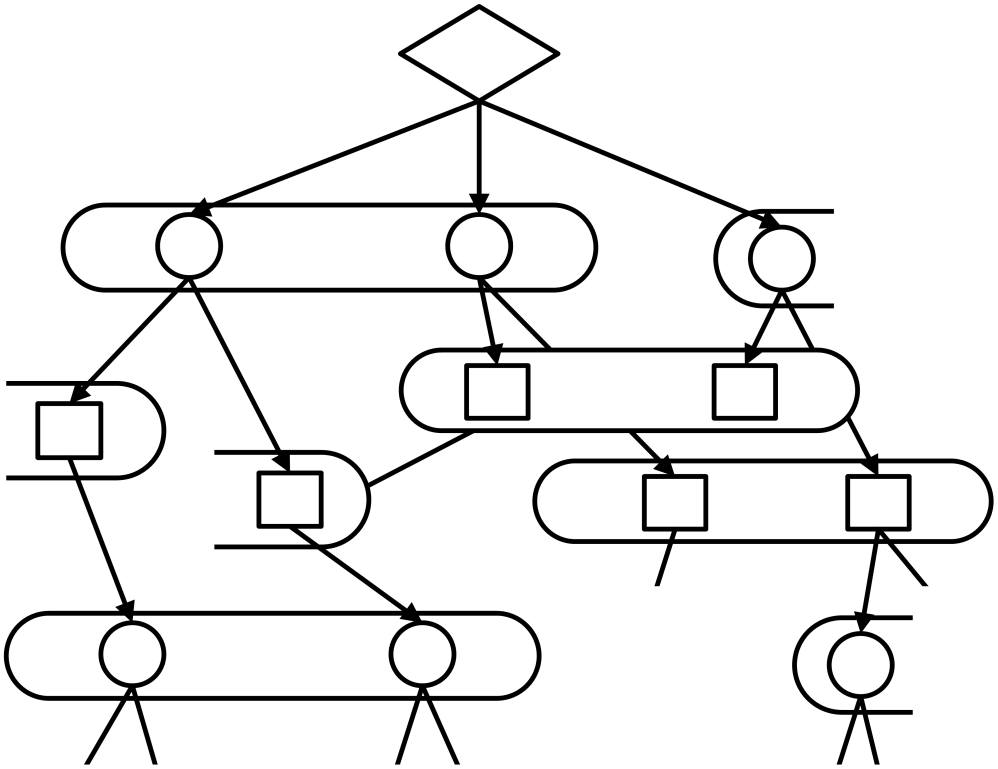

Рис. 1.1. Пример дерева представления игры с неполной информацией в экстенсивной форме

Определение: Многошаговая позиционная игра Γ n лиц определяется:

1. Заданием множества N всех игроков ($|N| = n$).

2. Заданием древовидного графа $G = (X, F)$ (где X— множество вершин в дереве, а $F : X \to X$, для любого $x \in X$ $F(x)$ — множество потомков вершины x в дереве) с начальной вершиной x_0, называемой начальной позицией игры.

3. Разбиением множества всех вершин X на $n + 2$ множеств $X_1, X_2, \ldots, X_n, X_{term}, X_c$, где множество X_i называется множеством очередности i-го игрока ($i = 1, \ldots, n$), а множество $X_{term} = \{x : F_x = \varnothing\}$ — множеством окончательных позиций. X_c — множество вершин случая.

4. Заданием функции f_c сопоставляющей каждой вершине случая $x \in X_c$ вероятностную меру $f_c(y|x)$ на $y \in F(x)$ ($f_c(y|x)$ — вероятность

из вершины x в дереве G перейти в вершину y). Все такие меры независимы.

5. Подразбиением каждого множества $X_i, i = 1, \ldots, n$ на k непересекающихся множеств $I_i^j = X_i^j \ j = 1, \ldots, k$, называемые информационными множествами i-го игрока ($|I_i| = k$). При этом для любых позиций одного и того же информационного множества множество следующих за ним вершин должно содержать одно и то же число вершин, т.е. для любых $x, y \in I_i^j : |F_x| = |F_y|$, и никакая вершина информационного множества не должна следовать за некоторой другой вершиной этого же множества, т.е. если $x \in I_i^j$, то не существует другой вершины $y \in I_i^j$ такой, что $y \in F_x$

6. Заданием вектор-функции $K(x) = (K_1(x), \ldots, K_n(x))$ на множестве окончательных позиций $x \in X_{term}$; функция $K_i(x)$ называется выигрышем i-го игрока.

Также стоит заметить, что листы дерева из определения являются терминальными состояниями, показывающими, что игра закончена и не может быть совершено никаких действий.

Наличие скрытой информации отличает игры с неполной информацией от игр с полной информацией. Наличие скрытой информации важно, потому что оно означает, что для каждого игрока, различные вершины принятия решения становятся неразличимыми — принимая игровое решение о каком-либо действии, игрок точно не знает, какой вершине информационного множества соответствует текущее состояние игры. Также можно сказать что игра с полной информацией является вырожденным случаем игры с неполной информацией, так как в случае полной информации каждое информационное множество содержит только одну вершину.

1.2. Факторы

1.2.1. Определение

Дадим формальное определение понятия фактор, используемого в данной работе для описания некоторого подмножества информационных состояний (см. определение представления игры в экстенсивной форме 1.1).

Определение: Пусть $G = (X, F)$ — древовидный граф игры с неполной информацией n лиц, представленной в экстенсивной форме, тогда фактором $Fact_k$ для игрока i называется отображение $Fact_k : I_i \to B$, где B — множество значений фактора, I_i — множество всех информационных множеств игрока i (фактор принимает конечное множество значений $|B| = n_k$, не уменьшая общности можно считать, что это отрезок ряда целых чисел $0..n_k - 1$).

Таким образом каждый фактор задает для каждого игрока разбиение всех информационных множеств в дереве игры на некоторые классы. Если мы рассмотрим множество факторов, то на него не накладывается никаких дополнительных ограничений, т.е. вполне возможно, что какому-то подмножеству всех информационных множеств будет соответствовать два различных значения двух факторов. Это создает некоторую избыточность описания состояния игры набором факторов.

Простой пример фактора — текущая стадия игры. Допустим, в некоторой игре есть n стадий, правила игры на каждой стадии имеют некоторые особенности, тогда фактор разбивающий множество всех информационных множеств на стадии будет каждому информационному множеству сопоставлять целое число из диапазона $0, \ldots, n - 1$.

1.2.2. Пространство состояний

Представление игр в экстенсивной форме имеет существенные недостатки. При наивном подходе к моделированию представление модели оппонента в памяти будет иметь состояние для каждой вершины в дереве

игры. В результате размер такого представления будет сопоставим по размеру с полным деревом игры, что для реальных игр делает работу с такими моделями практически невозможной.

Еще одной причиной, почему дерево игры необходимо сокращать, является необходимость ускорить обучение модели. Если пространство состояний будет слишком большим, то при обучении модели понадобится большое число наблюдений, прежде чем в каждом из состояний модели будет достаточное число действий оппонента.

В этом разделе будут представлены некоторые методы для работы с пространствами многих состояний. Сначала будет дана идея представления состояний для больших пространств. После этого мы сосредоточимся на уменьшении размера представления с помощью *объединения состояний*. Идея заключается в том, чтобы уменьшить эффективный размер пространства состояний за счет группировки состояний, которые являются эквивалентными по отношению к некоторому критерию, учитывающему выигрыш в игре и оптимальные действия.

Пространство состояний можно представить как перечисление всех состояний $S = \{s_1, s_2, \ldots, s_n\}$. Такое представление называется *экстенсивным* или *явным представлением*. Однако, множество состояний возможно представить без необходимости их перечисления, если мы будем говорить о свойствах состояний. Такой способ представления называется *неявным*. Часто, все пространство состояний будет являться декартовым произведением нескольких дискретных свойств или *факторов*, по этой причине достаточно часто используется термин *факторное пространство*. Большое преимущество неявного представления в том, что оно может быть гораздо меньше.

Достаточно сильно с явным представлением связаны понятия *абстракция* и *агрегация*. Абстракцию можно рассматривать как процесс удаления свойств состояний (возможно, каких-либо определенных состояний), ко-

торые считаются неуместными или мало влияющими. Термин агрегация относится к процессу группировки или объединения состояний в соответствии с определенным правилом эквивалентности. Получающиеся в результате этого процесса агрегированные состояния могут быть использованы для представления сгруппированных состояний в уменьшенной модели.

Наиболее подробно в данном разделе будут рассмотрены факторизованные представления. Далее, будут представлены методы для работы с факторизованными представлениями.

1.2.3. Факторизованное представление

Как уже было сказано, факторизованные представления базируются на идее, что состояние может быть описано набором свойств или факторов. Пусть $Fact = \{Fact_1, Fact_2, \ldots, Fact_k\}$ — набор факторов. Множество значений каждого фактора — некоторое дискретное множество.

Состояние игры (информационное множество) представляется набором из k значений факторов и все пространство состояний состоит из всех возможных комбинаций допустимых значений.

1.3. Абстракция

Под термином абстракция можно понимать не только процесс удаления свойств состояний в представлении игры, но также результат этого процесса — построенное пространство состояний. Важно, чтобы это пространство имело меньший размер по сравнению с исходным пространством состояний. Основную сложность при работе с деревом игры — большой размер дерева в реальных играх. К примеру, в случае отсутствия абстракции, при моделировании нужно очень много времени, чтобы в одном состояние увидеть достаточное число действий оппонента. При использовании абстракции делается предположение о том, что во всех информационных множествах, соответствующих одному состоянию абстракции, можно на-

блюдать одни и те же распределения вероятностей действий оппонента. В общем случае это не верно.

1.3.1. Определение

Дадим формально определение понятия абстракция, используемого в данной работе.

Определение: Рассмотрим граф представления игры в экстенесивной форме $G = (X, F)$, множество факторов. Каждый из факторов задает разбиение всех информационных множеств I_i игрока i на классы. Назовем абстракцией отображение $A : Fact \to Abs$, где $Fact = (Fact_1, Fact_2, \ldots, Fact_k)$ — вектор значений факторов информационного множества I_i^j, Abs — некторое множество состояний абстракции. Пусть выполняется $A(Fact(I_i^k)) = A(Fact(I_i^j))$, тогда для всех таких k должно выполняться: $P(I_i^k) = P(I_i^j)$ — функция распределения вероятности возможных переходов из информационного множества I_i^k.

1.3.2. Постановка задачи

Необходимо построить абстракцию игры, которая содержала бы существенно меньшее число состояний, чем число информационных состояний в дереве игры. При построении абстракции необходимо:

- Необходимо минимизировать число состояний — чем меньше состояний, тем меньше действий оппонента необходимо наблюдать, чтобы построить модель.

- Абстракция должна обладать похожими стратегическими свойствами по сравнению с полным деревом игры. Под похожестью в данном случае понимаются похожие распределения действий информационных множеств, находящихся в одном состоянии абстракции.

1.3.3. Существующие подходы

В данном подразделе будет рассмотрено несколько подходов к сокращению дерева игры в покере.

Изоморфизм карт

Самый простой способ построить абстракцию для покера — объединить состояния с одинаковыми рангами карт, но различными мастями. Пример: 4♠9♠, 4♡9♡, 4♢9♢ и 4♣9♣ можно объединить в одно состояние. Такой метод построения абстракции не будет в результате терять информацию об игре, так как нет никакой стратегической разницы в выборе действий с комбинациями карт, отличающимися только мастью. Данный способ позволяет уменьшить число состояний по крайней мере в 4 раза [2]. Однако, такого сокращения пространства состояний игры все равно будет недостаточно.

Изоморфизм деревьев

В статье [3] авторы описывают метод уменьшения размерности дерева игры для поиска эквилибриум-стратегии. Метод использует тот факт, что некоторые поддеревья можно рассматривать как одно. В результате объединения строится уменьшенная абстракция игры, в которой находится оптимальная стратегия.

Разбиение на корзины

Также для сокращения размера дерева игры используется разбиение на корзины. Данный подход использует несколько другое определение абстракции игры, тем не менее с его помощью также решается задача уменьшения всего пространства состояний игры в экстенсивной форме. На каждой стадии игры все возможные карты, которые могут быть сданы игроку, а

также общие карты разбиваются на фиксированный набор корзин на основе предположения, что карты имеющие похожие стратегические свойства попадают в одну и ту же корзину. Общий подход в покере — разбиение карт на основе силы комбинации, которую можно с ними собрать, таким образом, чтобы слабые комбинации попадали в корзину с меньшим номером, а сильные — в корзину с большим номером. Также есть модификации данного подхода, учитывающие номер корзины на предыдущей стадии игры, а также использующие различные метрики для сравнения силы получившихся комбинаций, учитывающие не только текущую силу комбинации карт, но также потенциальную возможность эту комбинацию улучшить.

1.4. Существующие подходы для выделения факторов

Задачи выделения свойств (*feature selection* и *feature extraction*) занимают одно из центральных мест в машинном обучении. Различные методы выделения свойств входных данных находят активное применение в распознавании образов, классификации, анализе генов. Выделение свойств, определяя характерные свойства входных данных для обучения, позволяет алгоритму машинного обучения сфокусироваться на тех аспектах входных данных, которые являются наиболее важными для анализа и предсказания. Конечно, здесь уместнее было бы вместо термина свойства использовать термин факторы, но эта замена была сделана сознательно, чтобы не путаться в терминологии, и в качестве факторов понимать только то, что удовлетворяет определению в этой главе.

Можно выделить два класса методов для решения данной задачи: «фильтр» и «wrapper» [4]. Достаточно подробный обзор различных методов этих двух классов дан в работе [5]. В данном разделе будут приведены основные идеи этих двух классов методов. Подход «фильтр» являлся предпосылкой для создания алгоритма выделения факторов, который будет описан в разделе 2.4 данной работы. Насколько известно автору, методы выделе-

ния важных свойств еще не использовались для выделения факторов и уменьшения размерности абстракции игры.

На рисунке 1.2 приведена общая схема работы методов выделения свойств в двух разных подходах. Под алгоритмом понимается алгоритм машинного обучения, использующий представления входных данных в виде свойств для обучения и предсказания.

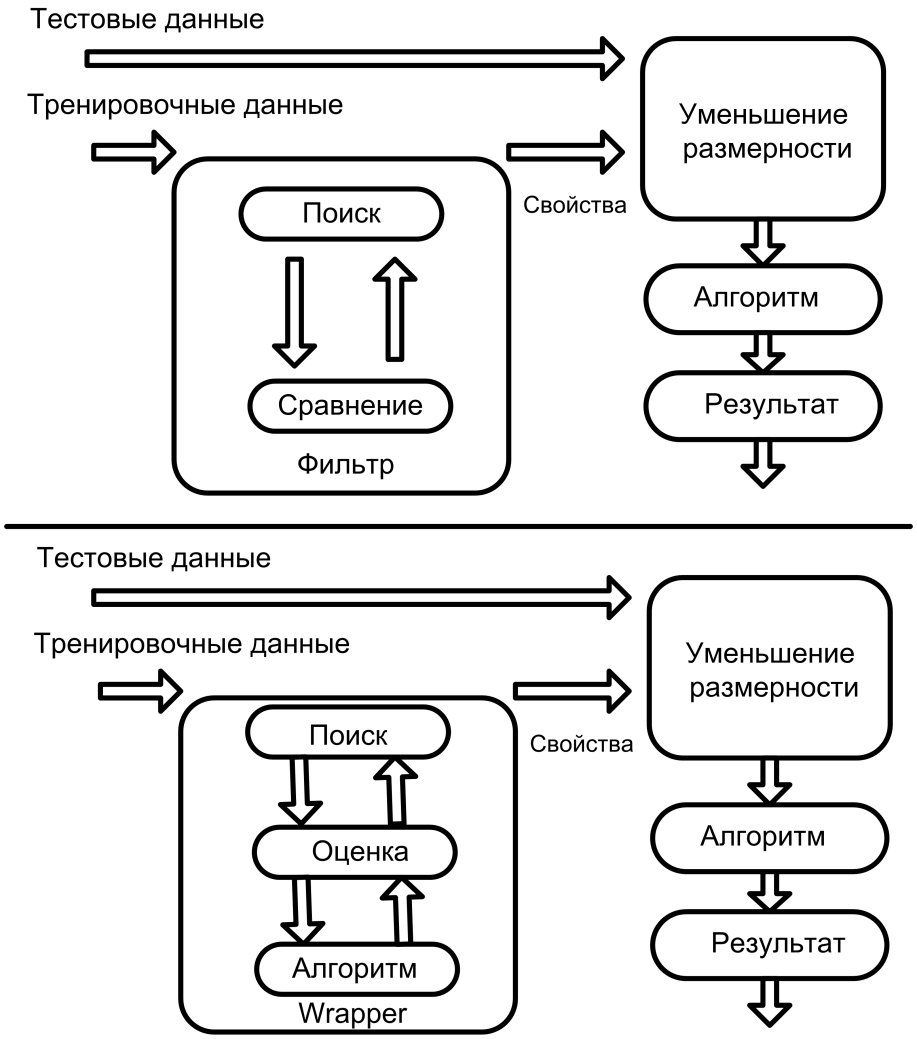

Рис. 1.2. Схемы работы подходов выделения свойств: «фильтр» и «wrapper»

1.4.1. Подход «wrapper»

Данный подход основан на методологии Кохави и Джона [4], которые предложили простой метод решения проблемы выбора свойств входных данных основываясь на выбранном методе машинного обучения. Фактически, алгоритм обучения воспринимается как черный ящик. Метод исполь-

зует результат предсказания алгоритма для оценки важности подмножеств переменных (свойств). В случае небольшого числа переменных может быть использован метод последовательного перебора всех подмножеств. Но известно, что эта проблема является NP-трудной [6]. Существуют методы использующие данный подход без полного перебора подмножеств. Варианты жадных стратегий: метод прямого отбора (*forward selection*) и метод обратного исключения (*backward elimination*). В основе метода прямого отбора лежит идея включения на каждом шаге в модель переменной, лучше всего удовлетворяющей критерию. Метод обратного исключения похож на предыдущий метод, но с тем отличием, что все переменные изначально включены в модель и постепенно осуществляется отсеивание тех из них, которые не проходят проверку на значимость.

1.4.2. Подход «фильтр»

Данный подход в отличие от «wrapper» не использует в качестве оценки результаты выбора свойств какой-либо внешний алгоритм машинного обучения. Методы, использующие данный подход, основываются на подсчете общих характеристик входных данных. Обычно используются различные статистические функции переменных (свойств) такие как взаимная информация (*mutual information*), корреляция Пирсона (*Pearson correlation*).

1.5. Классификация моделей

В покере, всем игрокам доступна публичная информация об игре, такая как ставки и общие карты, однако наличие приватных карт игрока подтверждает, что есть часть информации об игре известная только одному игроку. Наличие такой скрытой информации и делает покер очень интересной игрой для изучения.

Скрытая информация каждого из игроков играет главную роль в определении того, какие действия будут совершены в течение партии, а также

кто будет победителем, если игра дойдет до финальной стадии партии, когда оба игрока откроют приватные карты. Получение информации об оппоненте, позволяющей отвечать на два последних вопроса является моделированием оппонента. В этом разделе будут даны краткие определения и общие идеи двух видов классификации моделей по типу получаемой информации и по типу работы алгоритмов моделирования.

1.5.1. Явные и неявные модели

Модели оппонента могут быть явными или неявными. Достаточно много методов в области моделирования оппонентов используют неявное представление модели. Это означает, что алгоритм скрывает знание об оппоненте в своем внутреннем представлении либо в логике процесса принятия решения. Таким образом, игрок, натренированный против определенного оппонента, хранит полученные знания как «черный ящик». Основная задача, решаемая при использовании неявных моделей в играх — победить определенного оппонента в игре, однако при этом остается неясным каким образом можно распространить уже имеющуюся базу знаний на нового оппонента без дополнительного обучения.

В отличие от неявного типа моделей, явные используют для обучения некоторый функциональный процесс (например, нейронную сеть), получает на вход информацию об оппоненте и выдает на выходе представление оппонента. Результат работы алгоритма может получен либо в виде предсказания возможных действий оппонента в определенной игровой ситуации с учетом возможных характерных игровых особенностей стратегии оппонента, либо в виде возможных используемых оппонентом стратегий. Таким образом агент может использовать результат явного моделирования для построения контрстратегии. Явное моделирование распространяет существующие знания на новых оппонентов, распознавая их без дополнительной тренировки.

1.5.2. Стратегические модели

Данный класс имеет такое название из-за того, что модель данного типа должна представлять стратегию поведения оппонента. Термин «стратегия поведения» был взят из теории игр и обозначает сопоставление каждой точке принятия решения игроком (информационное множество в экстенсивном представлении игры) распределения вероятностей доступных игроку действий.

Модели этого класса представляют процесс выбора оппонентом действия явным способом. Это означает, что информация в модели обновляется согласно действиям оппонента, используя уже имеющуюся информацию об оппоненте на момент совершения действия. Таким образом модель строится последовательно в течение нескольких итераций — на каждой итерации информация об игроке добавляется в модель с учетом данных в модели, полученных на предыдущих итерациях.

Существенную сложность при построении подобного рода моделей в играх с неполной информацией представляет моделирование ситуации, когда скрытая информация об оппоненте не открывается. Если игрок сбрасывает карты, то не до конца ясно как обновлять модель.

1.5.3. Модели, использующие наблюдения

Описание метода моделирования оппонента в игре покер с использованием наблюдений приводится в работе [7].

Для того, чтобы обойти проблему моделирования ситуации, когда оппонент сбросил карты, присущую стратегическому классу моделей, может быть использован альтернативный подход, основанный на моделировании наблюдений действий оппонента. Агент, принимающий решения, учитывает только информацию доступную ему в каждой вершине дерева игры и моделирует вероятности наблюдения действий оппонента в этих состояниях.

Все действия в игре рассматриваются с позиции агента, который принимает решения, используя модель. При этом рассматриваются как действия оппонента, так и действия случайного процесса (в покере это раздача карт). В результате, вся информация, которая должна быть добавлена в модель, известна и не может быть скрыта. Например, в случае, если оппонент сбросил карты, единственная информация, которая необходима модели — сам факт того, что оппонент сбросился. В этом существенное отличие от модели стратегического класса, для которой также важны возможные сброшенные карты.

Выводы по главе 1

В данной главе были даны основные определения и понятия, которые в дальнейшем будут использованы при описании предлагаемого подхода для выделения факторов при построении абстракции в моделях стратегического типа.

Глава 2. Построение абстракции игры покер «Техасский холдем»

Покер неоднократно использовался в качестве модели для исследования игр. Работы фон Неймана и Моргенштерна, Джона Нэша содержат примеры и выводы относительно свойств этой игры. Сама игра имеет довольно много разновидностей. Даже внутри одного вида покера есть определенное разделение, связанное с числом игроков, размером максимальных ставок и другими правилами игры.

Класс игр в покере «Техасский холдем» является в настоящее время наиболее популярным как в казино так и на покерных сайтах. Популярность игры привела в свою очередь к интересу исследователей. Основная задача — создать агента, который мог бы играть наравне, или даже обыгрывать лучших профессиональных игроков-людей.

Автором данной работы была выбрана именно эта разновидность покера, при игре именно в нее есть возможность провести сравнение результатов игры агента против существующих на настоящее время агентов.

2.1. Правила игры «Техасский холдем»

«Техасский холдем» — это разновидность игры в покер. Если говорить более точно, то это не одна определенная разновидность, а класс схожих игр; существует насколько вариантов «Техасского холдема». Во всех вариантах играют от двух до десяти игроков за одним столом (мы ограничимся одним столом и не будем рассматривать варианты турнирного покера).

Главное отличие между вариантами «Техасского холдема» в сумме денег, на которую игроки могут сделать собственную ставку (*bet*) или перебить чужую ставку (*raise*). По этому признаку можно выделить лимитный, безлимитный покер и покер с лимитом равным размеру текущего банка. Везде, где далее будет упоминаться игра покер (если не будет особых замечаний), следует понимать, что речь идет именно о разновидности «Техас-

ского холдема» — лимитном покере для двух игроков. В лимитной версии «Техасского холдема» имеют значение две суммы ставок — при этом одна сумма обычно в два раза больше второй (например $2/$4). Меньшая сумма определяет значение единичной ставки или повышения в первых двух раундах игры, большая сумма — в последних двух раундах.

Всего в игре есть четыре раунда. Так как раунды торговли занимают центральное место в игре в покер, то сначала будет дано описание того, как происходит игра в одном раунде ставок.

В раунде ставок первый игрок может выбирать из двух возможных действий: пропустить (*check*) или сделать ставку (bet). Когда игрок пропускает, он не увеличивает размер банка, когда делает ставку (например $2) — общий банк игры увеличивается на размер ставки. Множество допустимых действий для второго игрока зависит от того, что выбрал первый игрок. Если первый игрок пропустил, второй игрок также как и первый вначале может тоже пропустить либо сделать ставку. Если первый игрок ставит, то второй может сбросить карты (*fold*), уравнять (*call*) или повысить (raise). Если игрок сбрасывает карты, это означает, что он отказывается от дальнейшей борьбы, и его оппонент выигрывает банк. Когда игрок уравнивает, он увеличивает банк на сумму достаточную, чтобы уравнять ставку оппонента. Повышение означает, что игрок за один ход уравнивает ставку оппонента и сверх этого делает свою ставку. Например, если ставка стоит $2, то повышение означает что игрок докладывает в банк $4.

Раунд ставок заканчивается, когда ни один из игроков не делает повышения ставки на очередном круге торговли, т.е. оба игрока пропускают ход (check) или последняя ставка (bet или raise) были уравнена (call). Также на каждом раунде можно суммарно максимально сделать четыре ставки.

Рассмотрим общую структуру игры. Сначала игрокам необходимо заплатить анте (*ante*) — вынужденную ставку, которая еще называется слепой ставкой (*blind bet*). После этой стадии все игроки получают по две

карты из общей колоды, состоящей из 52 карт. Эти карты видят только сами игроки. Далее следует раунд торговли. Когда первый раунд ставок завершается, на стол помещаются три видимые для всех игроков карты - эта стадия игры называется флоп (*flop*). Далее следует второй раунд ставок и, когда он завершается, на стол выкладывается еще одна общая карта. Эта стадия называется тёрн (*turn*). Начинается предпоследний и последний раунды ставок, это означает, что размер единичной ставки увеличивается в два раза. Третий раунд торговли завершается появлением последней пятой общей карты на столе — ривера (*river*). После ривера происходит последний раунд торговли, также с удвоенным размером ставок.

Если к окончанию партии никто из игроков не сбрасывает карты, то следует завершающая стадия игры - шоудаун (*showdown*) — все игроки открывают свои карты. В результате сыгранной партии выигрыш в виде банка получает игрок получивший самую старшую комбинацию из всех доступных ему карт (двух приватных и пяти общих). Список всех возможных комбинаций карт отсортированный по старшинству приведен в таблице 2.1

Так как покер является игрой на деньги, где задача каждого игрока максимизировать свой выигрыш, обычно играют сессии игр против одних и тех же оппонентов. Повторение нескольких партий дает игроку возможность изучить стратегию игры оппонента с целью использования этого знания для выявления слабостей, что в свою очередь позволяет еще больше увеличить выигрыш. Как результат, каждое решение в каждой игре оказывает влияние на результат.

2.1.1. Свойства игры

Таким образом соответствующая приведенным правилам разновидность покера обладает следующими свойствами:

1. Последовательность — все действия совершаются игроками последо-

Название комбинации	Описание
Королевский флеш *Royal flush*	$AKQJ10$ одной масти пример: $A\spadesuit K\spadesuit Q\spadesuit J\spadesuit 10\spadesuit$
Стрейт флеш *Straight flush*	пять последовательных карт одной масти пример: $4\clubsuit 5\clubsuit 6\clubsuit 7\clubsuit 8\clubsuit$
Каре *4-of-a-kind*	четыре карты одного достоинства пример: $A\spadesuit A\diamondsuit A\clubsuit A\heartsuit 3\heartsuit$
Фул-хаус *full house*	три и две карты одного достоинства пример: $J\spadesuit J\diamondsuit J\clubsuit 6\heartsuit 6\clubsuit$
Флеш *flush*	пять карт одной масти пример: $Q\clubsuit 4\clubsuit 8\clubsuit J\clubsuit 10\clubsuit$
Стрейт *straight*	пять последовательных карт пример: $7\diamondsuit 8\heartsuit 9\spadesuit 10\heartsuit J\heartsuit$
Тройка *3-of-a-kind*	три карты одного достоинства пример: $7\diamondsuit 7\heartsuit 7\spadesuit A\heartsuit 8\heartsuit$
Две пары *2-pair*	две пары одинаковых по достоинству карт пример: $6\diamondsuit 6\clubsuit 4\spadesuit 4\diamondsuit J\spadesuit$
Пара *pair*	две карты одного достоинства пример: $4\spadesuit 9\heartsuit 10\diamondsuit K\spadesuit K\clubsuit$
Старшая карта *high-card*	у игрока комбинация со старшей картой пример: $2\diamondsuit 5\clubsuit 7\spadesuit 8\heartsuit Q\clubsuit$

Таблица 2.1. Список покерных комбинаций с примерами, отсортированный по старшинству (вверху таблицы старшие комбинации)

вательно.

2. Неполнота информации — игроку известны только его приватные карты, а также общие карты на столе, но не известны приватные карты оппонента.

3. Случайность — раздача карт происходит случайным образом и все возможные последовательности карт в течение партии равновероятны.

4. Нулевая сумма игры.

5. Частичная информация — в случае, если игрок сбрасывает карты, информация о них никогда не будет доступна оппоненту.

6. Основная задача игроков — максимизировать выигрыш.

Дерево игры в экстенсивной форме содержит $3.16 * 10^{17}$ вершин и $3.16 * 10^{14}$ информационных множеств [8].

2.2. Набор факторов для представления игры

В данном разделе будет перечислен набор факторов, используемых для представления абстракции в модели для игры «Техасский холдем».

Определение факторов и факторного пространства было дано в разделе 1.2 данной работы.

Важно заметить, что создание факторов для описание игры или процесса является областью, где необходимы экспертные знания. В связи с этим построенные набор не претендует на оптимальность и полноту, однако он позволил получить некоторые результаты, которые будут приведены в разделе 3.

2.2.1. Текущая стадия игры

Данный фактор имеет всего четыре возможных значения:

- 0 — префлоп
- 1 — флоп
- 2 — терн

- 3 — терн

По мнению автора, данный фактор является важным, так как он хорошо разделяет множество всех действий. Также стратегии на различных стадиях игры могут отличаться.

2.2.2. Последовательность действий на текущей стадии

Данный фактор имеет десять возможных значений:

- 0 — на текущей стадии игры еще не было действий;
- 1 — check;
- 2 — check bet;
- 3 — check bet bet;
- 4 — check bet bet bet;
- 5 — check bet bet bet bet;
- 6 — bet;
- 7 — bet bet;
- 8 — bet bet bet;
- 9 — bet bet bet bet.

2.2.3. Число ставок на префлопе

По правилам игры на префлопе одна ставка (большой блайнд) делается еще до раздачи карт игрокам. Данный фактор имеет 4 значения:

- 1 — одна ставка;
- 2 — две ставки;

- 3 — три ставки;
- 4 — четыре ставки.

2.2.4. Число ставок на флопе

По правилам игры на флопе возможно от нуля до четырех ставок. Значения фактора идентичны значениям 2.2.3.

2.2.5. Число ставок на терне

По правилам игры на терне возможно от нуля до четырех ставок. Значения фактора идентичны значениям 2.2.3.

2.2.6. Число ставок на ривере

По правилам игры на ривере возможно от нуля до четырех ставок. Значения фактора идентичны значениям 2.2.3.

2.2.7. Инициатива на предыдущей стадии игры

Важный параметр, который часто учитывается при игре в покер — наличие инициативы. Инициатива — ставка, которую оппонент вынужден уравнивать, чтобы иметь возможность продолжить игру. В случае если оппонент в ответ на ставку делает повышение, инициатива переходит к нему.

Таким образом возможно 3 значения данного фактора:

- 0 - никто не делал ставок, соответственно никто из игроков не обладает инициативой;
- 1 - инициатива у первого игрока, т.е. последняя ставка была сделана первым игроком;
- 2 - инициатива у второго игрока, т.е. последняя ставка была сделана вторым игроком.

2.2.8. Тип общих карт

Автором был добавлен фактор, учитывающий так называемую структуру общих карт. Было сделано предположение, что наличие среди общих карт какой-то определенной комбинации или большая вероятность дополнения общих карт до сильной комбинации может влиять на стратегию оппонента. Есть мнение, что данный фактор более применим к игре с людьми, чем к игре с компьютерными агентами.

Данный фактор может принимать следующие значения:

- 0 - общие карты включают в себя карты как минимум двух разных мастей и не идут подряд, а также не включают старшие карты (туз, король, дама);
- 1 - из общих карт можно собрать без использования приватных карт игрока уже готовую комбинацию флеш или стрейт (см. таблицу 2.1);
- 2 - есть возможность, дополнив общие карты еще двумя картами, получить комбинации стрейт или флеш (см. таблицу 2.1);
- 3 - есть возможность, дополнив общие карты еще одной картой, получить комбинации стрейт или флеш (см. таблицу 2.1);
- 4 - среди общих карт есть туз, король или дама;
- 5 - среди общих карт есть пара.

2.2.9. Суммарный выигрыш

Данный фактор показывает насколько большой выигрыш получит победитель текущей партии.

2.2.10. Раздача старшей общей карты

Данный фактор в меньшей степени применим против игры с агентами и в большей степени с людьми. Каждое из значений фактора показывает,

что в результате раздачи общих карт появилась одна из старших карт (туз, король). Фактор необходим для отыскания стратегии оппонента, когда в случае выхода старшей общей карты делается ставка.

2.3. Моделирование оппонента

В данном разделе будет кратко описана схема построения модели оппонента стратегического типа для игры покер «Техасский холдем», а также схема использования модели во время игры. Более подробно алгоритм описан в работе Е. Долгих [9]. В данном разделе для простоты изложения будет описана версия алгоритма для случая, когда известны карты оппонента. Для ситуации, когда карты оппонента неизвестны, общая схема работы остается той же, но есть некоторые модификации.

2.3.1. Обучение

Пусть A — абстракция игры, то есть отображение из множества значений факторов в некоторое пространство состояний. H — последовательность вершин дерева игры, в которых действовал оппонент. Так как по окончании партии алгоритму моделирования известна скрытая информация оппонента, то в каждом информационном множестве, в котором оппонент принимал решение возможно определить точно состояние игры. Далее необходимо для каждого $x \in H$ выполнить следующие действия:

- найти состояние в абстракции $A(Fact(I_x))$ $(x \in I_x)$
- изменить значения вероятностей действий в данном состоянии, используя метрику для скрытых параметров оппонента

2.3.2. Игра

Во время игры агент находит состояние абстракции, соответствующее текущему состоянию игры. Это состояние содержит распределения вероятностей действий оппонента для различных значений метрики скрытых

параметров оппонента. Изначально (перед первым действием игрока) все скрытые параметры оппонента равновероятны, но с каждым новым действием вероятности изменяются в соответствии с действием оппонента и вероятностью этого действия для различных значений метрики скрытых параметров.

2.4. Работа с абстракцией

В данном разделе будет описан алгоритм построения абстракции игры для модели оппонента. Абстракция основана на факторном представлении игры 1.2.

Кроме алгоритма, приведенного здесь, автор протестировал модификацию алгоритма, когда изначально на основе экспертных знаний задается некоторая структура дерева абстракции, однако данный подход принес такие же или менее хорошие результаты по сравнению с подходом, когда в самом начале работы алгоритма есть только одна вершина в дереве.

2.4.1. Построение

Абстракция представляет собой дерево, в каждой вершине которого хранятся распределения действий игрока и ссылки на действия. Ребро между вершинами V_a и V_b имеет ассоциированное с этим ребром значение одного из факторов. Наличие ребра E с ассоциированным значением фактора $Fact_t = g$ между вершинами V_a и V_b (V_a — родитель) означает, что все действия игрока, совершенные в состояниях игры I(информационных множествах) со значениями фактора $Fact_t(I)$ равными g переносятся из узла дерева абстракции V_a в узел V_b. Соответственно совокупность перенесенных действий строит в состоянии абстракции V_b распределения отличные от V_a.

Алгоритм работает последовательно, начиная с дерева, состоящего только из одной вершины. На каждой итерации просматривается каждый узел

дерева и делается разбиение узла по значению одного из факторов. После разбиения получается две вершины, для новой вершины считается определенная оценочная (*fitness*) функция. В результате, для каждой вершины будет построено некоторое число возможных разбиений по значениям факторов — выбирается разбиение с максимальным значением оценочной функции.

Ниже приведена схема работы одной итерации алгоритма построения абстракции:

Require: $Tree$;

1: **for all** $Node \in Tree$ **do**
2: $parts := \{\}$
3: **for all** $Fact_i \in Fact$ **do**
4: **for** $value = 1, \ldots, MaxValue(Fact_i)$ **do**
5: **if** $canSplit(Node)$ **then**
6: $Node_{parent}, Node_{new} := split(Node)$
7: $fitness := Fitness(Node_{parent}, Node_{new})$
8: $parts = parts + \{Fact_i, value, Node_{new}, Node_{parent} fitness\}$
9: $\{Fact_max, value, Node_{new}, Node_{parent} fitness\} = Max_{fitness}(parts)$
10: $Node = filter(Fact_i, value)$
11: $Node.addChild(Fact_i, value, Nodenew, fitness)$

Итерации алгоритма выполняются, пока можно разбить хотя бы одну вершину.

2.4.2. Поиск состояния

Все дети вершины дерева хранятся в упорядоченном списке в порядке их выделения.

Схема поиска состояния в дереве:

Require: $Tree$,I;

1: $Root := Tree.root$

2: $FactorValues := \{Fact_i(I)\}$

3: $Current := Root$

4: $found := true$

5: **while** found **do**

6: $found := false$

7: **for all** $Edge \in Current.childs$ **do**

8: **if** $Edge.factor \in FactorValues$ **then**

9: $FactorValues := FactorValues - Edge.factor$

10: $Current := Edge.node$

11: $found = true$

2.5. Сравнение состояний

Важный момент в работе алгоритма — сравнение получившихся после выделения фактора состояний — так называемая оценочная или fitness функция. Автором было использовано два подхода. В данном разделе будет описан каждый из них. Автором были протестированы оба подхода, наилучшие результаты на небольших тестовых примерах давал подход, использующий сравнение распределений действий, поэтому в дальнейшем для построения абстракций существующих агентов был использован именно он.

2.5.1. Максимизация вероятности действия

Данный подход основан на предположении, что следует выделять состояния, в которых для всего отрезка сил рук в каждой точке есть характерное явно выделенное действие. При таком подходе для подсчета оценочной функции нового состояния нет необходимости сравнивать его с состоянием, из которого оно было выделено.

Метод сравнения двух состояний:

Require: $state_{new}$;

1: $p_{raise} := state_{new}.getActionProbability(Raise)$

2: $p_{call} := state_{new}.getActionProbability(Call)$

3: $p_{fold} := state_{new}.getActionProbability(Fold)$

4: **return** $\sum_{x=0}^{1} Max(p_{raise}(x), p_{call}(x), p_{fold}(x))$

2.5.2. Сравнение распределений действий

Данный подход основан на предположении, что следует выделять состояния, в которых распределения вероятностей действий отличаются от родительских вершин. При использовании описанной здесь функции имеет значение распределение вероятностей действий, которое получается в родительской вершине после выделения вершины, а также распределение вероятностей действий в новой вершине.

Для сравнения распределений использовалась следующая функция:

$$R_{prob}(p_1,\, p_2) = \int p_2(x) log\left(\frac{p_1(x)}{p_2(x)}\right)\, dx$$

Легко убедиться, что в случае, если распределения равны, данная функция будет возвращать 0. Также данная функция не может являться метрикой на множестве распределений, так как она не обладает свойством симметричности.

Метод сравнения двух состояний:

Require: $state_{new}, state_{parent}$;

1: $result := 0.0$

2: **for** $act \in \{Raise, Call, Fold\}$ **do**

3: $\quad p_1 := state_{new}.getActionProbability(act)$

4: $\quad p_2 := state_{parent}.getActionProbability(act)$

5: $\quad result := result + R_{prob}(p_1, p_2)\sqrt{\frac{state_{new}.getActionViewed(act)}{state_{new}.getTotalActionsViewed()}}$

6: **return** $result$

2.5.3. Пример

На рисунке 2.1 изображены две пары различных распределений вероятностей действия игрока, снизу написаны значений функции сравнения распределений (сравнивается верхняя вершина с нижней) и значения функции подсчета максимума распределений для каждой из групп распределений действий.

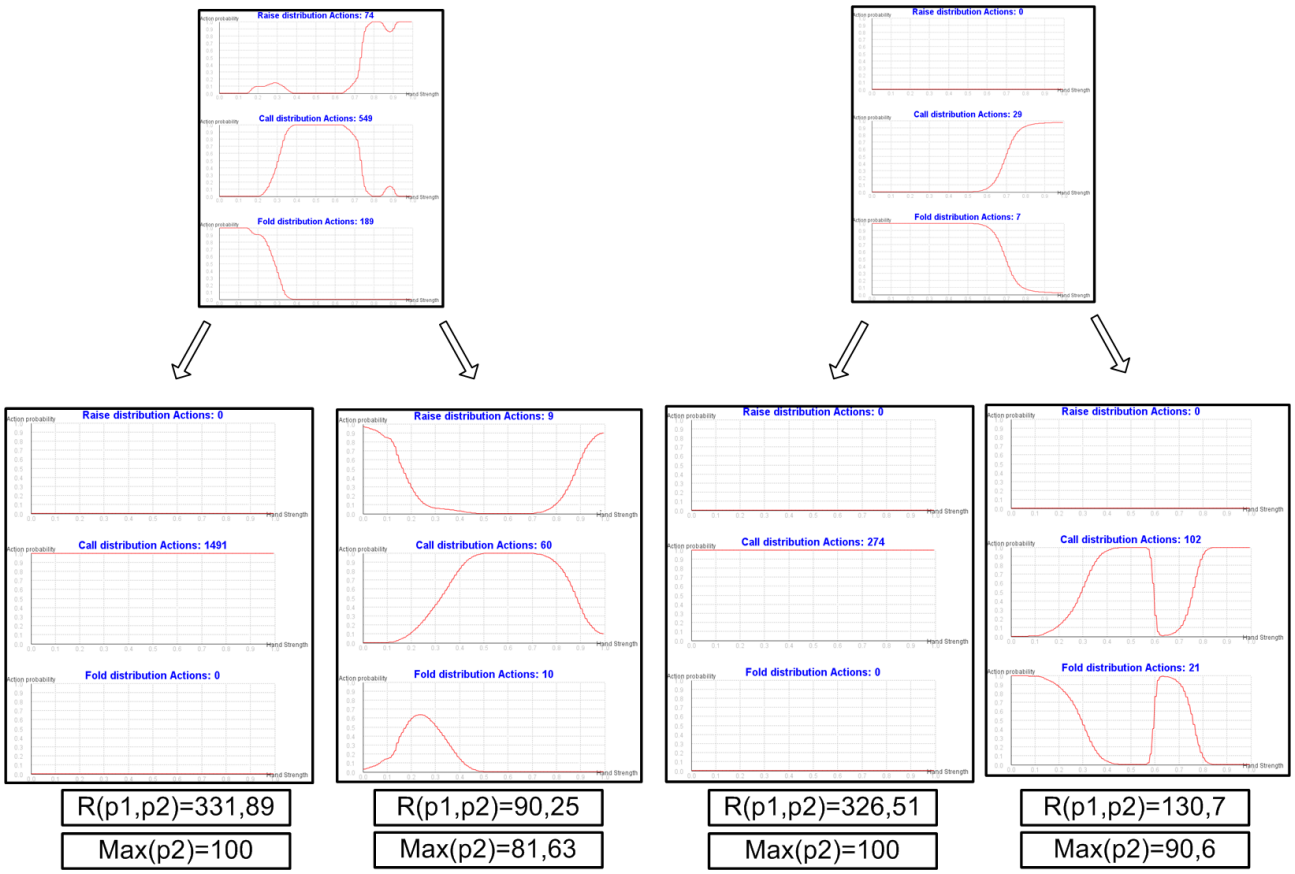

Рис. 2.1. Пример сравнения распределений. R(p1,p2) — сравнение двух групп распределений действий, Max(p2) — вычисление функции максимума по трем распределениям действий (Raise, Call, Fold)

2.6. Особенности реализации

Для тестирования алгоритма был использован агент реализующий стратегическое моделирование в применении к покеру [9]. Агент был реализован как подключаемый модуль к программе PokerAcademyPro [10]. Абстракция строилась заранее на основании данных об уже сыгранных оппонентом партиях. Во время игры дообучения модели не происходило.

Выводы по главе 2

В данной главе описан процесс моделирования оппонента в игре покер «Техасский холдем» с использованием модели стратегического класса, а также описан алгоритм построения дерева абстракции и поиска в этом дереве состояния по известному набору значений факторов.

Глава 3. Результаты

В данной главе будут приведены основные результаты, позволяющие сделать выводы об эффективности предложенного автором метода построения абстракции для представления игры. Под агентом в игре следует понимать программу, реализующую некоторые алгоритмы и принимающую решения в соответствии с правилами игры.

3.1. Выявление особенностей в игре тестовых агентов

В данном разделе будут описаны результаты применения метода построения абстракции для выделения факторов влияющих на игру определенного набора тестовых агентов. Подход, использованный здесь, можно разделить на следующие шаги:

1. Создание упрощенной версии агента с явными заранее определенными игровыми свойствами.

2. Игра агента против *исследующего агента* для формирования достаточного числа историй игровых партий.

3. Построение абстракции и модели на агента, созданного на шаге 1.

4. Сравнение информации в абстракции с определенными на шаге 1 характерными особенностями оппонента.

Выбор исследующего агента был сделан на основе тех же требований, что были изложены в работе [11], где исследующий агент был использован для получения истории игр с исследуемым агентом для последующего обучения. А именно, под определением исследующего агента понимается агент, который позволяет получить в результате в истории игр с оппонентом записи о действиях своего оппонента в максимальном числе информационных множеств. Единственный способ повлиять на информационные

множества оппонента — через историю действий в партии. Таким образом исследующий агент должен выбирать действия таким образом, чтобы покрыть как можно больше различных историй ставок. Так как исследующий агент не должен никогда сбрасывать карты, для его построения необходимо выбрать распределение возможных действий между повышением и уравниванием ставки.

Автором был выбран исследующий агент, делающий ставку в 60% случаев и уравнивающий ставку оппонента (либо пропускающий ход) в оставшихся 40%.

3.1.1. Тестовый агент №1

Данный агент играет по следующей стратегии: на префлопе, флопе и ривере он всегда уравнивает ставку оппонента, на терне в случае, если перед ним было только одно действие и оппонент пропустил ход (check), агент делает ставку, иначе — уравнивает.

Было сыграно 500 партий с исследующим агентом. После этого на истории сыгранных партий был запущен алгоритм построения абстракции, использующий для сравнения состояний функцию сравнения распределений действий оппонента от силы руки в каждом из состояний. В результате было построено дерево состояний абстракции, состоящее из трех вершин, приведенное на рисунке 3.2. Рядом с каждым из состояний построенной абстракции изображены распределения вероятностей действий агента. В вершине дерева осталось 2047 действий call, в вершине выделенной по значению фактора «последовательность действий на текущей стадии игры» 382 действия call и в вершине, выделенной по значению фактора «стадия игры» 122 действия raise. На рисунке 3.1 приведено текстовое описание построенной абстракции.

```
Node count: 3
|   <Node: 1>
|   Total actions: 2047
|   (0) HeadsUpActionLineFactor 1 (Check)
|   |   Compare distributions: 0.0
|   |   <Node: 2>
|   |   Total actions: 382
|   |   (0) StreetFactor 2
|   |   |   Compare distributions: 755.49
|   |   |   <Node: 3>
|   |   |   Total actions: 122
```

Рис. 3.1. Описание дерева абстракции тестового агента №1

3.1.2. Тестовый агент №2

Данный агент играет по следующей стратегии:

- На префлопе в 50% агент делает ставку, в 50% уравнивает.
- На флопе, в случае если у него была инициатива на префлопе (он последний сделал ставку), то 80% — ставка, 20% — уравнивание, в противном случае 20% — ставка, 80% — уравнивание.
- На терне, в случае если у него была инициатива на флопе, то 80% — ставка, 20% — уравнивание, в противном случае 20% — ставка, 80% — уравнивание.
- На ривере в 50% агент делает ставку, в 50% уравнивает, но если число ставок больше одной, то он сбрасывает карты.

Было сыграно 500 партий с исследующим агентом. После этого на истории сыгранных партий был запущен алгоритм построения абстракции, использующий для сравнения состояний функцию сравнения распределений действий оппонента. Следует заметить, что увеличением числа анализируемых партий можно уменьшить дисперсию распределений и увеличить точ-

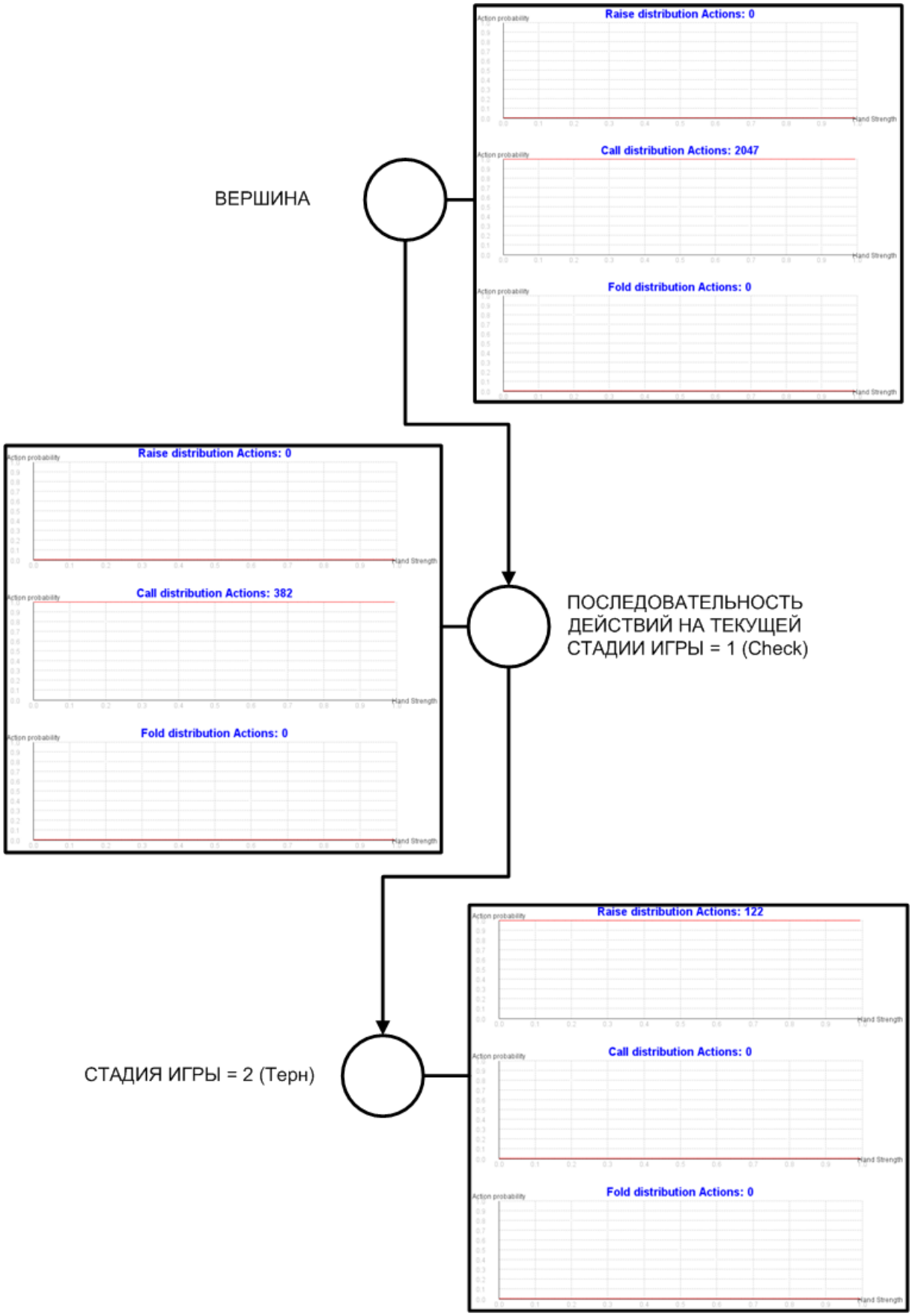

Рис. 3.2. Результат построения абстракции для тестового агента №1

ность результата, однако автор считает, что выходных данных алгоритма достаточно для понимания стратегии оппонента даже с учетом небольших неточностей.

В результате применения метода была построена абстракция из 8 состояний (была использована функция сравнения распределений с установленным параметром — минимальное число увиденных в состоянии действий, равным шестидесяти).

Процесс построения абстракции проходил следующим образом:

1. Итерация 0 — выделено состояние по значению фактора текущая стадия игры=ривер.

2. Итерация 1 — выделено состояние по значению фактора ставки на текущей стадии игры=две ставки (bet bet). Также выделено состояние по значению фактора инициатива на предыдущей стадии игры.

3. Итерация 2 — выделено состояние из вершины по значению фактора текущая стадия игры=префлоп, также из состояния инициатива на предыдущей стадии игры было выделено состояние по значению фактора число ставок на терне.

4. Итерация 3 — выделено состояние по значению фактора текущая стадия игры=префлоп.

5. Итерация 4 — выделено состояние по значению фактора число ставок на терне=0.

На рисунке 3.3 приведено текстовое описание построенного дерева абстракции. На рисунке 3.4 изображены тройки распределений действий тестового агента №2, при этом номер блока с распределениями соответствует номеру состояния на рисунке 3.3.

Из рисунков можно сделать следующие выводы о структуре выделенной абстракции:

```
Node count: 8
|   <Node: 1>
|   Total actions: 196
|   (0) StreetFactor 3
|   |   Compare distributions: 46.70
|   |   <Node: 2>
|   |   Total actions: 636
|   |   (0) HeadsUpActionLineFactor 7 (BetBet)
|   |   |   Compare distributions: 219.88
|   |   |   <Node: 3>
|   |   |   Total actions: 74
|   (1) PrevStreetInitiativeFactor 1
|   |   Compare distributions: 74.19
|   |   <Node: 4>
|   |   Total actions: 150
|   |   (0) TurnRaisesNumberFactor 0
|   |   |   Compare distributions: 9.49
|   |   |   <Node: 6>
|   |   |   Total actions: 449
|   (2) StreetFactor 0
|   |   Compare distributions: 33.91
|   |   <Node: 5>
|   |   Total actions: 692
|   (3) TurnRaisesNumberFactor 0
|   |   Compare distributions: 13.81
|   |   <Node: 7>
|   |   Total actions: 537
|   |   (0) PreflopRaisesNumberFactor 4
|   |   |   Compare distributions: 37.47
|   |   |   <Node: 8>
|   |   |   Total actions: 108
```

Рис. 3.3. Описание дерева абстракции тестового агента №2.

Рис. 3.4. Распределения вероятностей действия в состояниях абстракции, построенной для тестового агента №2

- Состояния 2 и 3 единственные состояния, в которых число действий fold ненулевое. Данные состояния соответствуют стадии игры ривер. В состоянии 2 значения распределений действий call и raise относятся в среднем как 50/50. Состояние 3 соответствует ситуации, когда на ривере было сделано больше одной ставки, в этом случае агент всегда сбрасывает карты. Наличие в состоянии 2 небольшого числа действий fold объясняется тем, что есть последовательности действий отличные от bet bet, например check bet bet и bet bet bet. Числа действий для данных последовательностей ставок оказалось недостаточно для выделения отдельных состояний, поэтому эти действия остались в состоянии 2.

- В состоянии 5, соответствующем стадии игры префлоп, распределения действий call и raise относятся примерно как 50/50.

- Состояния 4 (у агента была инициатива на предыдущей стадии игры)

и 6 (то же, что 4, а также число ставок на терне ноль) достаточно похожи. Значение функции сравнения распределений для этих двух состояний 9.49. Выделение отдельного состояния 6 обусловлено некоторыми отличиями распределений, существующими из-за достаточно большой дисперсии в связи с выборкой небольшого числа историй партий оппонента, используемых для анализа. Распределения call и raise относятся примерно как 20/80.

- Состояние 7 соответствует значению фактора число ставок на терне равному нулю. Следует пояснить, что значение этого фактора равно нулю на всех стадиях игры кроме ривера, поэтому в данное состояние были объединены все действия на флопе и терне, в которых агент не обладал инициативой. В состоянии 8 были выделены действия, совершенные после того, как на префлопе было четыре ставки. Распределения в состоянии 8 отличаются от 7 на 37.47.

- Все действия, которые не были выделены в процессе построения абстракции, остались в состоянии 1. Отношение распределений call и raise 80/20.

3.2. Результаты игры против существующих агентов

Для проверки результатов работы метода, предложенного в данной работе, было проведено тестирование агента, использующего модификацию алгоритма Expectimax для выбора действия и использующего стратегическую модель для предсказания действия оппонента. Алгоритм агента подробно описан в работе [9]. Для хранения результатов моделирования была использована абстракция, построенная в результате анализа историй партий оппонента. Для каждого оппонента строилась своя абстракция. Тестирование проводилось в программе [10]. Данная программа поставляется в комплекте с агентами Sparbot(PsOpti4) и Poki. PokerAcademyPro2 имеет

программный интерфейс Meercat-API для подключения агентов, который был использован автором для подключения своего агента в эту среду. Кроме того, есть свободно распространяемая версия агента FellOmen, работающая в данной программной среде.

Для оценки математического отклонения выигрыша была использована формула приведенная в работе [8]:

$$\frac{6}{\sqrt{N}}$$

где N — число сыгранных партий. Данная оценка является достаточно грубой, однако в программной среде PokerAcademy2 не предусмотрено методов для более точной оценки (к примеру, повторная игра одной и той же партии с обменом приватных карт между оппонентами).

Далее в этой главе будут перечислены результаты игры агента, использующего моделирование с абстракцией, заранее построенной для каждого из оппонентов. Ввиду того, что число состояний в этих абстракциях гораздо больше, чем в абстракциях для тестовых агентов, сами представления построенных абстракций в текстовом или графическом виде в данной работе представлены не будут.

3.2.1. Poki

Краткое описание агента

Poki[2] является одним из наиболее известных агентов, использующих стратегию с фиксированными правилами. Данный агент использует моделирование на основе экспертных правил. Для выбора действия с наибольшим матожиданием используется метод Монте-Карло. Изначально данный агент был рассчитан для игры за столом 10 игроков, однако поздее стратегия была адаптирована для игры двух лиц.

Абстракция

Было проанализировано 5000 историй партий. При обучении агент видел карты оппонента по окончании каждой партии даже в том случае, если оппонент сбрасывал карты.

Построенная абстракция имеет 140 состояний.

Эксперименты

В общей сложности было сыграно 10500 партий, результат — 0.72 малых ставок за одну партию. А с учетом оценки отклонения: 0.72 ± 0.06. В работе [9] приводится результат игры против этого агента с использованием простой абстракции в виде декартова произведения значений всех факторов: 0.41 ± 0.06. Можно сделать вывод, что в данном случае после построения абстракции результат игры против этого агента стал лучше.

График выигрыша агента от числа сыгранных партий представлен на рисунке 3.5.

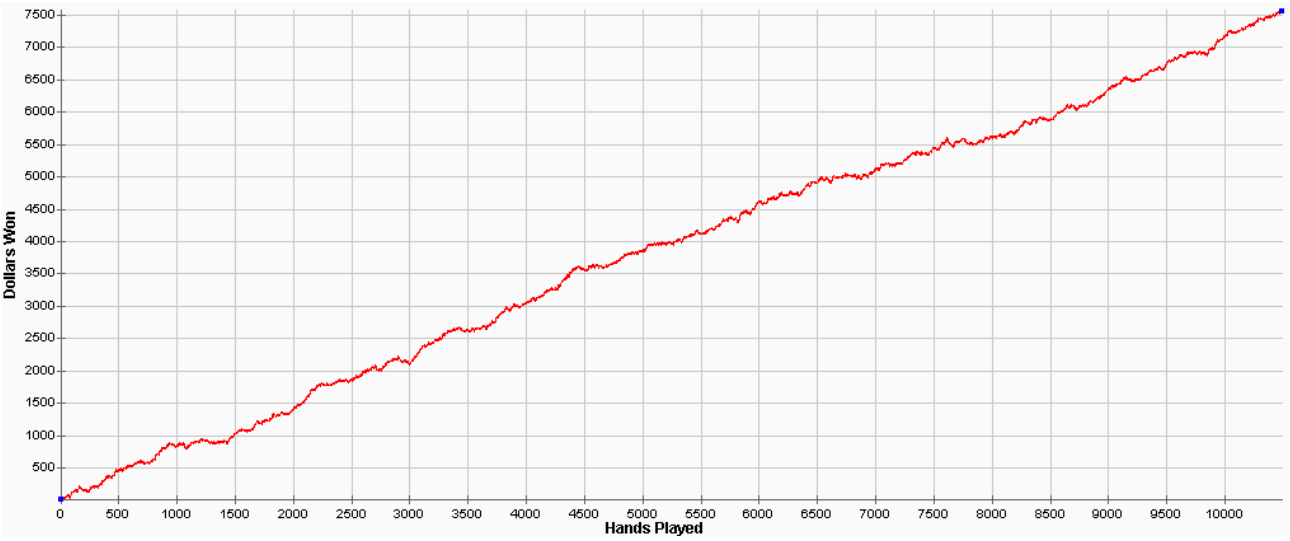

Рис. 3.5. График выигрыша против агента Poki

3.2.2. Sparbot (PsOpti4)

Краткое описание агента

Sparbot также известен как PsOpti4. Данный агент использует ϵ-эквилибриу
стратегию, построенную, используя методы описанные в работе [8]. Уязви-
мость данной программы оценивается в 0.11 малых ставок за одну партию
в собственной абстракции и 0.27 — в полной игре.

Абстракция

Было проанализировано 65000 историй партий. При обучении агент ви-
дел карты оппонента по окончании каждой партии даже в том случае, если
оппонент сбрасывал карты.

Построенная абстракция имеет 278 состояний.

Эксперименты

В общей сложности было сыграно 28300 партий, результат — 0.16 ма-
лых ставок за одну партию. А с учетом оценки отклонения: 0.16 ± 0.03.
В работе [9] приводится результат игры против этого агента с использо-
ванием простой абстракции в виде декартова произведения значений всех
факторов: 0.13 ± 0.03. Можно сделать вывод, что в данном случае после по-
строения абстракции результат игры против этого агента по крайней мере
не ухудшился.

График выигрыша агента от числа сыгранных партий представлен на
рисунке 3.6.

3.2.3. Fellomen

Краткое описание агента

Для построения данного агента был использован подход фиктивная иг-
ра. Основная идея данного метода заключается в итеративном построении

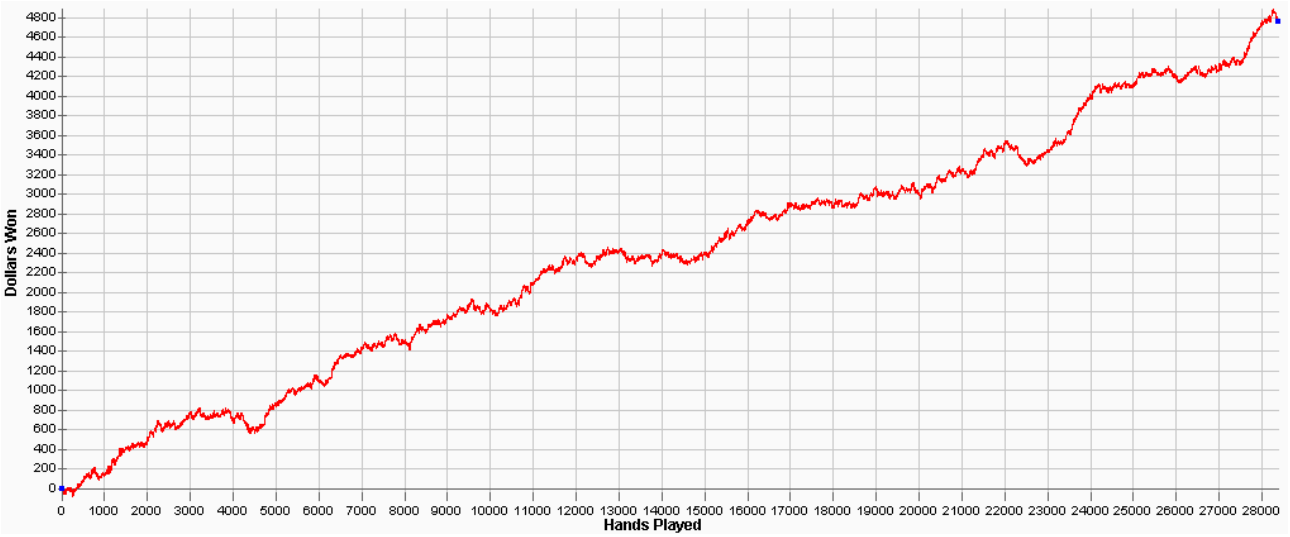

Рис. 3.6. График выигрыша против агента Sparbot(PsOpti4)

стратегий. На каждом шаге к уже построенному агенту строится контр-стратегия, затем эта контр-стратегия включается в агента в качестве смешанной стратегии. Таким образом будет строится множество стратегий, каждая из которых будет учитываться при принятии решения агентом с определенной вероятностью.

Построенная абстракция

Для построения абстракции было проанализировано 85000 историй партий. При обучении агент видел карты оппонента по окончании каждой партии даже в том случае, если оппонент сбрасывал карты.

Построенная абстракция имеет 695 состояний.

Эксперименты

В общей сложности было сыграно 24500 партий, результат — 0.12 малых ставок за одну партию. А с учетом оценки отклонения: 0.12 ± 0.04. В работе [9] приводится результат игры против этого агента с использованием простой абстракции в виде декартова произведения значений всех факторов: 0.01 ± 0.03. Можно сделать вывод, что метод выделения состояний абстракции позволил определить состояния с характерными распре-

делениями. В результате улучшились предсказательные свойства модели оппонента и удалось получить на достаточно большом числе игр выигрыш, позволяющий с уверенность говорить о превосходстве построенного агента.

График выигрыша агента от числа сыгранных партий представлен на рисунке 3.7

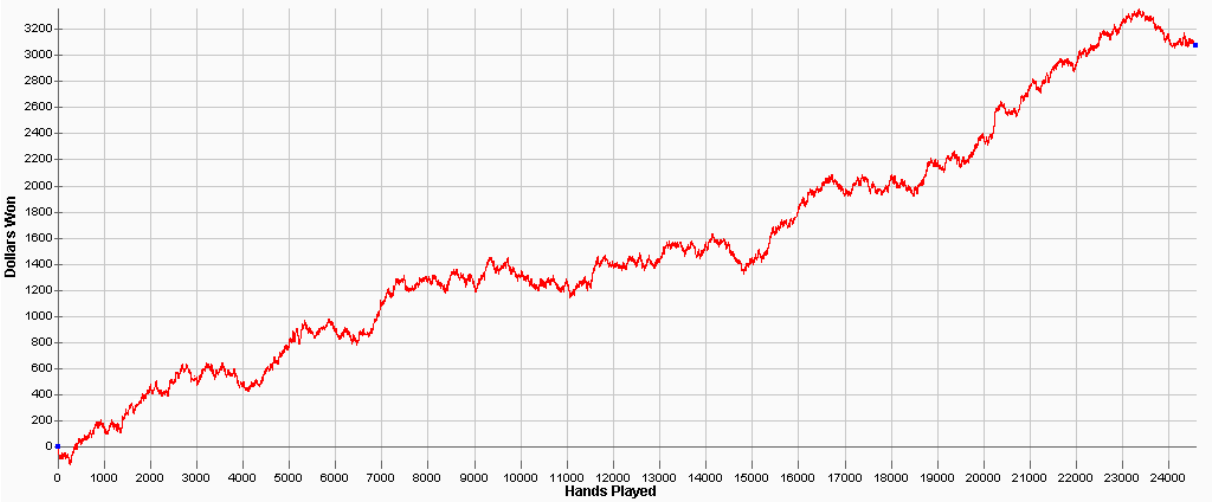

Рис. 3.7. График выигрыша против агента FellOmen

Выводы по главе 3

В данной главе приведены результаты проведения экспериментов двух видов:

- Построение абстракции по истории партий агента с заранее заданной смешанной стратегией.
- Построение абстракции для игры против существующих агентов.

На основании этих результатов можно сделать вывод, что алгоритм приведенный в данной работе позволяет строить абстракцию с небольшим набором факторов, соответствующую стратегии оппонента. В свою очередь экспериментально было показано, что наличие хорошей абстракции помогает успешно моделировать поведение оппонентов и увеличивать выигрыш в игре против них.

ЗАКЛЮЧЕНИЕ

Основные результаты работы:

1. Построен набор факторов для игры «Покер техасский холдем».
2. Реализован алгоритм построения и изменения абстракции.
3. Выведены критерии оценки качества полученной абстракции.
4. Построена абстракция для представления стратегий существующих агентов.
5. Качество построенной абстракции подтверждено улучшением в игре агента против существующих агентов.

Дальнейшие направления работы:

1. Расширить набор факторов представления игры.
2. Модифицировать оценочную функцию, использующуюся для оценки выделенных состояний.

ИСТОЧНИКИ

1. *Петросян, Л. А.* Теория игр / Л. А. Петросян, Н. А. Зенкевич, Е. А. Семина. — Москва: Книжный дом «Университет», 1998.

2. *Billings, D.* Algorithms and assessment in computer poker, phd thesis / D. Billings. — 2006.

3. *Gilpin, A.* Better automated abstraction techniques for imperfect information games, with application to texas hold'em poker / A. Gilpin, T. Sandholm // In International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS. — 2007.

4. *Kohavi, R.* Wrappers for feature subset selection / R. Kohavi, G. H. John // *Artif. Intell.* — 1997. — Vol. 97, no. 1-2. — Pp. 273–324.

5. *Hall, M. A.* Correlation-based feature selection for machine learning, phd thesis / M. A. Hall. — 1999.

6. *Amaldi, E.* On the approximability of minimizing nonzero variables or unsatisfied relations in linear systems / E. Amaldi, V. Kann // *Theor. Comput. Sci.* — 1998. — Vol. 209, no. 1-2. — Pp. 237–260.

7. *Schauenberg, T. C.* Opponent modelling and search in poker, msc. thesis / T. C. Schauenberg. — 2006.

8. Approximating game-theoretic optimal strategies for full-scale poker / D. Billings, N. Burch, A. Davidson et al. // Proc. International Joint Conference on Artificial Intelligence, Acapulco, Mexico. — 2003. — Pp. 661–668.

9. *Долгих, Е. А.* Разработка системы моделирующей поведение агента в стохастической игре с неполной информацией на примере игры покер «техасский холдем», бакаларская работа / Е. А. Долгих. — 2010.

10. PokerAcademyPro2. — http://www.poker-academy.com/.

11. *Johanson, M. B.* Robust strategies and counter-strategies: building a champion level computer poker player, msc. thesis / M. B. Johanson. — 2007.