

Менжулин Сергей Алексеевич

Тема: Исследование и разработка методов представления и обработки
знаний в сетевых играх

Утверждена приказом по университету № 2184/2 от 11.09.2001

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

по направлению высшего профессионального образования
552800 “Информатика и вычислительная техника”

Факультет Автоматики и Вычислительной Техники

Руководитель:

Гаврилов Андрей Владимирович,
к.т.н., доц. каф. ВТ НГТУ

Новосибирск, 2003 г.

Министерство образования Российской Федерации

НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

УТВЕРЖДАЮ
Заведующий кафедрой ВТ
д.т.н., профессор
Губарев В.В.

“ ” _____ 2003 г.

ЗАДАНИЕ

на магистерскую диссертацию

Студенту *Менжулину С.А.* факультета Автоматики и Вычислительной
Техники, обучающемуся по направлению 552800 “Информатика и
вычислительная техника”

Магистерская программа (специализация) 552819, “Компьютерный
анализ и интерпретация данных ”

Тема: Исследование и разработка методов представления и обработки
знаний в сетевых играх

Цель работы: Целью магистерского исследования является изучение
способов представления и обработки знаний в сетевой игре, разработка
алгоритма, по которому виртуальный игрок мог бы играть в эту игру.

Руководитель:

Гаврилов А.В.,
к.т.н., доц. каф. ВТ НГТУ

(подпись)

“ ” _____ 2003 г.

Реферат

Работа представлена на 80 страницах, содержит список литературы из 11 названий, 8 рисунков и 8 формул, 2 таблицы.

Ключевые слова: компьютерные игры, сетевые игры, искусственный интеллект, обработка знаний, генетическое программирование, теория игр, коалиционные игры, дерево решений, критерии полезности, стратегия.

Работа посвящена проблеме формирования оптимального поведения виртуальных игроков в сетевых компьютерных играх, извлечению знаний из действий игроков – людей, эволюционным методам оптимизации поиска выигрышных стратегий в играх.

Содержание

<u>. Применение искусственного интеллекта в играх</u>	8
<u>1.1. Классификация игр с точки зрения использования искусственного интеллекта</u>	10
<u>1.1.1. Игры – приключения</u>	10
<u>1.1.2. Ролевые игры</u>	12
<u>1.1.3. Игры жанра action</u>	13
<u>1.1.4. Стратегические игры</u>	13
<u>1.2. Методы представления знаний и принятия решений</u>	17
<u>1.2.1. Таблицы решений</u>	17
<u>1.2.2. Таблицы переходов между состояниями</u>	18
<u>1.2.3. Деревья решений</u>	18
<u>1.2.4. Сценарии</u>	19
<u>1.3. Методы реализации искусственного интеллекта в играх</u>	21
<u>1.3.1. Конечные автоматы</u>	21
<u>1.3.2. Скриптовые языки</u>	22
<u>1.3.3. Метод перебора</u>	23
<u>1.3.4. Эволюционное моделирование</u>	25
<u>1.4. Обучаемость виртуальных игроков</u>	29
<u>1.5. Выводы</u>	31
<u>2. Разработка алгоритма для компьютерного игрока</u>	33
<u>2.1. Выбор и описание игры</u>	33
<u>2.2. Математическая модель</u>	37
<u>2.3. Математические методы решения задачи</u>	40
<u>2.4. Применение генетического программирования</u>	41
<u>2.4.1. Общая теория</u>	41
<u>2.4.2. Адаптация генетического алгоритма</u>	49
<u>2.4.3. Выбор способа кодирования решений</u>	49
<u>2.4.4. Формулировка правил скрещивания и мутации</u>	50
<u>2.4.5. Правило отбора</u>	51

	5
2.5. Описание принципа работы полученного алгоритма	52
2.6. Выводы	54
Заключение	55
Список литературы:	56
Приложение 1	57

Введение

Развитие компьютерной техники стимулировало развитие различных областей ее применения. Разработка компьютерных игр на некотором этапе своего развития перешла из академической области в коммерческую сферу. На настоящее время игровая индустрия является прибыльной и перспективной для дальнейшего развития. После коммерциализации игровой индустрии практически прекратилась разработка новых алгоритмов, в основном использовался математический аппарат и базовые принципы, существовавшие в момент отделения. Сейчас на рынке игрового программного обеспечения сложилась такая ситуация, что рынок насыщен играми с реалистичной графикой, моделируемой с качеством, приближающимся к сложности сцен из реальной жизни, но поведение управляемых компьютеров персонажей по сложности сопоставимо с разработками 60-х годов. Увеличение скорости работы вычислительной техники позволило моделировать более сложное поведение компьютерных оппонентов в играх. Стало возможным, например, использование более глубокого перебора в пространстве возможных решений, как это произошло с шахматными играми, но в настоящее время этот путь развития практически исчерпал себя. На рынке компьютерных игр наблюдается уменьшение продаж, потребители отказываются покупать игры с примитивным, предсказуемым поведением компьютерных оппонентов. Разработчики компьютерных игр нуждаются в разработке новых алгоритмов, методов использования искусственного интеллекта, что стимулирует научные исследования и обеспечивает их финансирование.

Целью магистерского исследования является изучение способов представления и обработки знаний в сетевой игре (более подробно о выбранной игре будет описано в работе), разработка алгоритма, по которому виртуальный игрок мог бы играть в эту игру. Особенностью сетевых игр является то, что в них может играть одновременно более одного человека, а так же является обязательным наличие сервера, централизованно участвующего в моделировании игровой ситуации. В данной работе было выдвинуто предположение, что игрок – человек

способен научиться играть в некоторую игру и находить оптимальные решения (последовательности действий). Его игру можно оценивать и использовать полученные знания для формирования поведения компьютерных оппонентов.

Научная новизна исследования заключается в том, что был предложен метод оценивания стратегий человека при помощи грубой математической модели для выделения пригодных для использования (подражание). Полученные стратегии используются и модифицируются в алгоритме проверки, реализованном на разновидности генетического программирования. Конечной целью разработанного алгоритма является получения некоторого набора эффективных стратегий.

Практическая ценность работы заключается в возможности использования разработанных алгоритмов в коммерческих проектах по разработке компьютерных игр.

Основные положения и результаты диссертационной работы докладывались и обсуждались на конференции “Дни науки НГТУ” (г. Новосибирск, 2002). По результатам исследования планируется публикация статьи.

Диссертация состоит из введения, двух глав, заключения, списка литературы из 11 наименований. Работа содержит 57 страниц основного текста, 8 иллюстраций.

В первой главе приведен обзор существующих методов использования искусственного интеллекта в играх, указаны недостатки и обозначена сфера интересов для дальнейшего исследования.

Во второй главе приведено детальное описание игры, формализация задачи с помощью математического аппарата теории игр, построена грубая модель игры. Разработан генетический алгоритм для программирования поведения компьютерных игроков.

В заключении формулируются основные выводы по результатам исследования.

. Применение искусственного интеллекта в играх

До недавнего времени главной целью создателей игр было достижение реалистичной и подробно детализированной 3D-графики. Когда же уровень программного и аппаратного обеспечения позволил это сделать, а вычислительный ресурс процессора высвободился для новых задач, стал актуальным вопрос о наделении игр более реалистичным интеллектом.

Каждый человек имеет свое субъективное понятие интеллекта и наделяет это слово своим смыслом, поэтому трудно дать исчерпывающее определение интеллекту, хотя значительный интерес представляет следующее определение, данное в словаре Вебстера [1], тому, что принято называть интеллектуальным поведением:

- 1) Способность успешно реагировать на любую, особенно новую, ситуацию путем надлежащих корректировок поведения.
- 2) Способность понимать взаимосвязи между фактами действительности для выработки действий, ведущих к достижению поставленной цели.

По свидетельству Сорена Джонсона (Soren Johnson) [2], разработка искусственного интеллекта в академической среде и в игровой индустрии существенно отличается. Сорен Джонсон, работая в компании Firaxis, отвечал за систему искусственного интеллекта в популярной игре-стратегии Civilization III. По его словам, создание искусственного интеллекта в игре - это словно большой цикл: сделать небольшой фрагмент искусственного интеллекта, посмотреть его в работе, внести улучшения, опять посмотреть в работе, и так снова и снова. В академическом же сообществе дело обстоит иначе, поскольку здесь у ученых нет никаких естественных причин для изнурительных итерационных процессов. Тут главное — чтобы создаваемый искусственный интеллект был максимально приближен к модели, которую исследователи пытаются изучать. Еще одно связанное с этим преимущество — ученым нет необходимости доводить разработку до

совершенства, поскольку она не является составной частью коммерческого продукта.

В игровой индустрии доминирует прагматичный подход, так что на конференции GDC-2002 [2] доклады и презентации разработчиков были сфокусированы не столько на революционных новациях, сколько на усовершенствовании уже хорошо известных технологий.

Рассмотрим некоторые жанры компьютерных игр с точки зрения использования искусственного интеллекта и актуальности вопросов моделирования "разумного" поведения.

1.1. Классификация игр с точки зрения использования искусственного интеллекта

1.1.1. Игры – приключения

В принципе приключения - жанр, в котором потребность в применении теории искусственного интеллекта практически не возникает. Все локации¹ predeterminedены заранее разработчиками игры, головоломки продуманы сценаристами, поэтому создание даже не просто игры, а целого инструментария для создания игр такого жанра вообще является полностью технической проблемой.

Однако есть в приключениях и узкое место - диалоги. Во время многочисленных встреч с "обитателями" различных локаций играющему приходится постоянно общаться с ними, постепенно узнавая различную информацию. Элементарное разнообразие, вносимое программистами с помощью датчика случайных чисел и нескольких условных структур, немного вносило разнообразие в игру, но не более.

Интересное продолжение в развитии интерфейса было реализовано в игре "Under a Killing Moon", содержащей несколько гигабайт интерактивного видео. В процессе игры игроком выбираются не конкретные фразы из фиксированного списка, а общие темы для разговора, а уже виртуальный персонаж сам будет говорить на заданную тему. Задача играющего - удержать диалог в нужном русле. Такая реализация лишь усиливает внешнюю иллюзию интеллектуальности, но реально остается простой игрой случая и встроенных в программу сюжетов.

Задача синтаксического разбора введенного предложения даже с ограниченным словарем в сотню слов является типичной задачей искусственного интеллекта. Необходимо отметить, что даже для английского языка, который служит основой для всех современных языков программирования в силу своей лаконичности и достаточно формальной семантики, до настоящего времени не удалось создать интеллектуальную

¹ Локация - от англ. location – местность. В большинстве компьютерных игр локацией называется целая зона, участок карты и т.п.

программу, способную понимать смысл фраз из достаточно больших областей знаний, например, нашего обыденного мира.

Для достаточно формальных миров и конкретных наборов предметов, обладающих однозначными, недвусмысленными характеристиками и названиями, такая система была создана, причем уже давно. Это знаменитая программа Винограда SHRDLU [4], разработанная им в 1971-м году, представляет собой замкнутую систему, некий абстрактный мир, состоящий из разноцветных кубиков и пирамид. В этом мире функционирует "однорукий" робот, способный общаться с человеком на естественном (английском) языке и совершать манипуляции с объектами своего замкнутого мирка. Этот робот также способен понимать достаточно сложные выражения и самостоятельно планировать свою деятельность и объяснять мотивы тех или иных действий. Вот небольшой пример - фрагмент реального диалога с этой системой:

- Найди блок, который выше, чем тот, что в твоей руке, и положи его в коробку.

- ОК.

- Поставь, пожалуйста, один из красных блоков на другой, а сверху либо зеленый блок, либо пирамиду.

- ОК.

- Служит ли самая короткая вещь, которая опирается на опору самой высокой пирамиды, опорой для чего-то еще?

- Да, для зеленой пирамиды.

- Когда ты брался за зеленую пирамиду?

- В то время, когда я строил башенку из красного куба, большого красного блока и большого зеленого куба.

- Почему?

- Чтобы от нее избавиться.

- Почему?

- Чтобы освободить красный куб.

Многочисленное повторение вопроса "Почему?" в итоге приведет к ответу: "Потому что вы меня об этом попросили".

Самым интересным здесь является то, что программа не отвечает на заранее заданные типовые фразы. SHRDLU, можно сказать, понимает, о чем его спрашивают, и отвечает достаточно разумно. Конечно, ничего подобного ни в одной из известных игр найти нельзя, поскольку практическая реализация такого подхода в какой-нибудь приключенческой игре будет не совсем тривиальна.

Что же касается русского языка, то возможность реализации для него подобной системы сегодня практически равна нулю. Сложная падежная система, склонения, времена, все это создает массу проблем, и, если еще программе понять достаточно формальную фразу может и удастся, то сформулировать грамматически корректный ответ очень трудно.

1.1.2. Ролевые игры

Пожалуй, именно в ролевых играх, как ни в каких других играх, можно использовать практические возможности искусственного интеллекта. С помощью сложной целочисленной системы подсчета очков удается учитывать множество нюансов поведения и схваток в некоем виртуальном фантастическом мире.

AD&D немного напоминает систему Винограда, когда действие разворачивается в количественно и, в принципе, неограниченном, но качественно замкнутом условном виртуальном мире. В рамках AD&D можно разыгрывать массу неповторяемых сюжетов, хотя попытки добавления новых объектов в AD&D как правило заканчивались неудачей, потому что тщательно сбалансированная и отлаженная на практике система начинает как бы жить собственной жизнью и автоматически отторгать чужеродные элементы, не вписывающиеся в ее внутреннюю структуру. Даже небольшое изменение числового значения какого-нибудь параметра, не говоря уже о добавлении новых возможностей, приводит к непредсказуемым возмущениям в совсем других местах в процессе игры, настолько плотно увязаны игровые таблицы, внешне, казалось бы, достаточно автономные.

Система правил AD&D позволяет разыгрывать игровые партии самого разного объема и продолжительности, хотя и число участников является

одним из требующих настройки параметров. И, несмотря на то, что AD&D не моделирует непосредственно поведение конкретных объектов, она формально описывает внутреннюю структуру виртуального мира, в котором эти объекты могут корректно существовать, жить и развиваться, и поэтому с полным правом может быть отнесена к системам искусственного интеллекта.

1.1.3. Игры жанра action

Игр этого жанра очень много, но в них можно выделить то, что интересует нас в отношении необходимости моделирования поведения - это действия компьютерных противников, нападающих на игрока со всех сторон. Применение методов искусственного интеллекта сводится к использованию различных алгоритмов поиска кратчайшего пути, передвижения по лабиринту, преследования. Все эти алгоритмы являются достаточно изученными и интереса для дальнейшего исследования не представляют.

1.1.4. Стратегические игры

Очевидно, стратегические игры (как правило, военные) - это именно та область компьютерных игр, в которой применение методов искусственного интеллекта является наиболее актуальным на сегодняшний день.

Все алгоритмы, используемые в военно-стратегических играх, независимо от способа организации игрового времени - пошаговые или в реальном времени, очевидно, являются алгоритмами нахождения оптимального хода в игре двух лиц с полной информацией (когда расстановка сил известна заранее и не требуется исследования игрового пространства) или неполной информацией (когда большая часть карты закрыта). Методы решения подобных задач очень хорошо исследованы, но из этого не следует, что их эффективная практическая реализация лежит на поверхности. К сожалению, т.н. теория численных методов, описывающая способы алгоритмического, компьютерного нахождения решения для ряда

задач пока не в состоянии предложить более-менее эффективные способы, примером здесь могут служить шахматные программы.

В 1949 году Клод Шеннон предложил две алгоритмические схемы для шахмат и похожих на них игр. Первая - это полный перебор на глубину в N полуходов (ходов одной из сторон) с числовой оценкой конечных позиций с учетом материального соотношения и позиционных факторов типа "конь в центре", "открытая линия" и т.п.

Вторая - это неполный перебор, когда программа включает в рассмотрение не все, а только небольшое число разумных с ее точки зрения ходов. Все сегодняшние игровые программы, в которых в реальной партии максимальное число ходов из возникающих позиций практически не превышает 40-50 (шахматы, шашки, реверси и т.п.) используют исключительно полный перебор на большую глубину, а их авторы докладывают о числе позиций, оцениваемых в единицу времени.

В других играх, где число ходов еще меньше, программы уже играют с человеком-чемпионом на равных или побеждают его. Например, в известной игре "Реверси" (или "Отелло") в среднем 5-7 ходов.

Очевидно, алгоритм полного перебора для игр с ограниченными возможностями маневра, подходит как нельзя лучше. Но для игр, в которых игровые доски даже ненамного больше шахматной, возникает множество проблем.

Практикой доказано, что для более-менее сильной игры на уровне первого разряда - кандидата в мастера в любую игру, в том числе и в коммерческие компьютерные военно-стратегические игры, необходим полный перебор вариантов на глубину не менее 8 полуходов (ходов за одну из сторон). Если взять шахматы, то 40 (среднее число ходов) в восьмой степени будет 6500 миллиардов позиций. С использованием современных средств ограничения перебора удастся уменьшить количество позиций в достаточно оптимальных ситуациях в примерно квадратный корень из этого числа. Тогда получается примерно два с половиной миллиона позиций.

Но если взять игру Го, в которой доска несколько больше - 19x19 полей, хоть все равно несравнима по величине с игровыми полями

компьютерных стратегических игр, но число возможных ходов превосходит шахматное количество на порядок - примерно 300 ходов, ситуация выглядит совсем пессимистично. 300 в восьмой степени - это 65 миллиардов миллиардов (два раза) позиций.

В то же время, сильнейшие программы для игры Го играют все же достаточно сильно. Это достигается использованием алгоритмов на базе второй схемы Шеннона. В перебор включаются порядка 7-10 ходов, отбираемых по максимальному уровню статической оценки, и поэтому удается осуществлять расчет более глубоко. Однако если этот метод и подходит для игры с соперниками среднего уровня, то при встречах с профессионалами, когда исход партии подчас решает тонкий нетривиальный ход, программа проигрывает, не включая в перебор выигрывающие ходы.

В военных стратегических компьютерных играх, несмотря на большие размеры игрового поля и игровые единицы с различными характеристиками, тактика ведения игрового боя, пусть отдаленно, но все же схожая с реальным сражением, достаточно далека как от шахматной стратегии, так и от игр типа Го. Она является значительно более простой и основывается на так называемой "системе продукций", одним из фундаментальных методов в теории искусственного интеллекта.

Суть "системы продукций" заключается в том, что стратегия принятия решений хранится в большой базе знаний, состоящей из набора эвристик, свойственных и получаемых на основе опросов людей-экспертов в моделируемой предметной области. Эти эвристики представляют собой условные операторы типа "ЕСЛИ (накоплено 300 единиц ресурсов И можно построить завод по производству тяжелых боевых машин И такого завода пока нет) ТО построить завод".

Как показывает практика, в большинстве стратегических игр после трех-пяти полных проходов до победы с учетом различных рекомендаций из документации, человек быстро обнаруживает базовые скрытые закономерности игры и в дальнейшем руководствуется небольшим набором правил, позволяющих ему вести игру на достаточно высоком уровне.

Причем отличие человека от программы заключается не столько в различии этих правил, сколько в умении живого игрока гибко применять эти правила в зависимости от ситуации, складывающейся на доске.

Кроме того, человек, как правило, мыслит в подобных ситуациях достаточно долгосрочными категориями, сначала продумывая план действий, а затем реализуя его с помощью стандартных для его индивидуальности приемов. В шахматах есть основополагающий принцип, гласящий - "лучше иметь плохой план, чем никакого плана". Правильность этого принципа подтверждают практически все реализации военных стратегических игр.

1.2. Методы представления знаний и принятия решений

1.2.1. Таблицы решений

Таблицы решений представляют собой частный случай так называемых продукционных систем. В этих системах правила вычислений представляются в виде продукций. Продукции представляют собой операторы специального вида и состоят из двух основных частей, для краткости называемых обычно "ситуация — действие".

"Ситуация" содержит описание ситуации, в которой применима продукция. Это описание задается в виде условий, называемых посылками продукции. "Действие" — это набор инструкций, подлежащих выполнению в случае применимости продукции.

Соответствующая таблица решений содержит две графы — слева приведены описания ситуаций, справа — соответствующие действия. Предполагается, что программист разработал интерпретирующую программу для подобных таблиц. Эта программа работает следующим образом. Для конкретной ситуации осуществляется последовательный просмотр ситуаций, указанных в таблице, и сравнение их с входными эталонами. Если тестируемый образец соответствует некоторой ситуации, то выполняется действие, указанное для этой ситуации. Пример такой таблицы для некоторой игры приведен в таблице 1.

Таблица 1.

Пример таблицы решений

Ситуация	Действие
Противник находится вне зоны поражения	Ждать
Противник находится в зоне поражения	Напасть на противника
Количество здоровья меньше 20%	Сбежать

1.2.2. Таблицы переходов между состояниями

Таблицы переходов между состояниями используются для синтеза конечных автоматов. Таблица переходов описывает:

- Множество всех состояний, в которых может находиться система.
- Условия смены текущего состояния
- Действие, которое необходимо совершить для перехода в новое состояние
- Состояние, в котором окажется система после перехода.

Пример таблицы переходов для описания поведения некоторого виртуального игрока приведен в таблице 2.

Таблица 2

Таблица переходов, описывающая поведение игрока

Текущее состояние	Следующее состояние	Условие перехода	Действие
Ожидание	Бой	Можно напасть на противника	Атаковать
Бой	Преследование	Противник убегает	Начать преследование
Преследование	Бой	Можно напасть на противника	Атаковать
Бой	Покой	Показатель здоровья меньше 20%	Сбежать
Бой	Ожидание	Противник побежден	Ждать

1.2.3. Деревья решений

Деревья решений - это универсальное средство представления знаний, близкое к продукциям. Они создают иерархическую структуру классифицирующих правил типа "ЕСЛИ... ТО..." (if-then), имеющую вид дерева. Для принятия решения, к какому классу отнести некоторый объект или ситуацию, требуется ответить на вопросы, стоящие в узлах этого

дерева, начиная с его корня. Вопросы имеют вид "значение параметра А больше х?". Если ответ положительный, то осуществляется переход к правому узлу следующего уровня, если отрицательный - к левому узлу; затем снова следует вопрос, связанный с соответствующим узлом. Пример дерева решений приведен на рис. 1

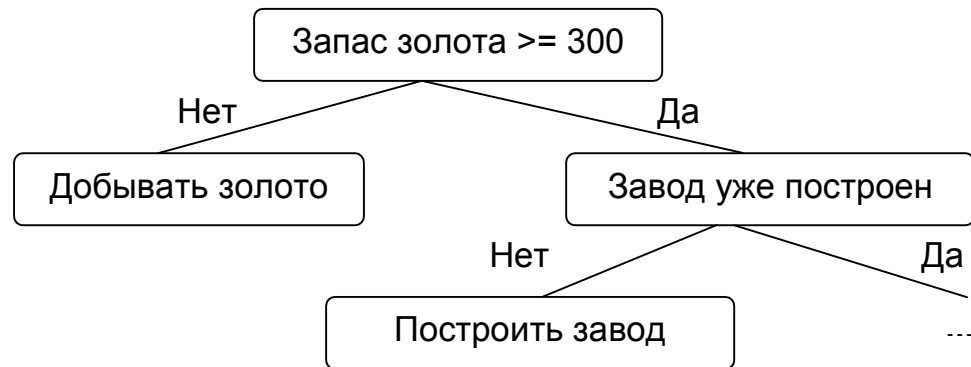


Рис. 1 Фрагмент дерева решений

Деревья решений принципиально не способны находить "лучшие" (наиболее полные и точные) правила в данных. Они реализуют простой принцип последовательного просмотра признаков и "цепляют" фактически осколки настоящих закономерностей, создавая лишь иллюзию логического вывода.

1.2.4. Сценарии

Существует масса различных способов представления сценариев. Наиболее часто используются два из них – текстовое описание (сценарии, основанные на качественных оценках) или таблицы и схемы для обобщения количественных данных, в том числе получаемых с применением сложных математических моделей (сценарии, основанные на количественных оценках). Сценарий указывает:

- Набор объектов действия.
- Последовательно указывает основные действия, которые выполняют объекты и основную реакцию на все действия.
- Описывает основные свойства текущей сцены действия.

Сценарий не описывает все то, что в нем не написано - подробные детали. То есть, имея определенную степень детализации, он возлагает все детали на изобретательность и творческий вкус режиссера использующего этот сценарий, то есть то, что для постановщика/зрителя ("пользователя") данного сочинения не имеет принципиального значения или просто не столь важно для изменения.

1.3. Методы реализации искусственного интеллекта в играх

1.3.1. Конечные автоматы

Не принадлежащие игроку персонажи могут управляться специальными алгоритмами, с подачи Алана Тьюринга [3] именуемыми машинами с конечным числом состояний или просто конечными автоматами. В этой системе искусственного интеллекта для управляемого компьютером персонажа предусмотрено ограниченное количество моделей поведения или состояний, которые заранее определены разработчиком. То или иное состояние, которое персонаж-агент выбирает по ходу игры с кажущейся разумностью, зависит от выполнения критериев, также заранее заложенных разработчиком. Эти критерии могут включать следующие параметры: видит или нет агент игрока, как много оружия осталось у противника, как много сил у самого персонажа и так далее. Для добавления в поведение агента элемента непредсказуемости многие разработчики включают в модель принятия решений нечеткую логику или случайный выбор значений (весов) некоторых из параметров.

Проще всего пояснить функционирование конечного автомата в игре на примере. Допустим, на игрока нападает вражеский монстр-охранник с мечом. Монстр начинает из состояния «Ждать», когда со стопроцентным здоровьем дожидается появления в поле зрения игрока. После чего переходит в состояние «Нападение» и устремляется к гостю. Когда же расстояние уменьшается до определенного уровня, монстр переходит в режим «Атака» и пытается уробить игрока мечом. Если это удастся — режим «Победный танец», игра окончена. Но если в ходе атаки умелому игроку удастся подорвать здоровье монстра, доведя его до уровня меньше 20%, охранник переходит в режим «Бегство» и пытается спасти то, что от здоровья осталось. Бегство может закончиться либо гибелью монстра, либо его исчезновением из поля зрения игрока — в этом случае происходит автоматическое переключение в режим «Ждать».

Основными достоинствами конечных автоматов является простота реализации и скорость выполнения. Из недостатков можно выделить отсутствие возможности обучения и сложность модификации

1.3.2. Скриптовые языки

Особенности сценариев (обычно именуемых на англоязычный манер скриптами) — в том, что управление общим ходом событий и действиями персонажей не зашито жестко в само тело игры, а выполняется с помощью внешних модулей - скриптов, написанных на языке высокого уровня. Такой подход позволяет разработчикам, устанавливающим общую цель и методы ведения конкретной игры, работать как бы полунезависимо от программистов, занятых непосредственным воплощением игрового механизма игры. Прежде всего, такой подход существенно сокращает время разработки, но также и позволяет эффективно встраивать в игру элементы искусственного интеллекта. Управляющие поведением персонажей скрипты могут очень сильно различаться по сложности. Например, тривиальный сценарий волшебника-помощника просто объяснит игроку, какова магическая сила того или иного предмета. Но могут быть и скрипты весьма изощренных в интеллекте персонажей, реализующие или самостоятельный конечный автомат с большим числом состояний, или иной, более экзотический тип системы искусственного интеллекта. Сценарии могут быть написаны как на специализированных языках, созданных под конкретную игру, так и на стандартных языках общего назначения, типа Perl или Java.

Первое достоинство скриптового языка: легкость модификации работы программы без коренной переделки. Как и любой прочий язык, скриптовой язык выполняется некоторой машиной. Только в отличие от обычных, где после перевода в команды нижнего уровня, такой машиной является процессор, скриптовой язык выполняется некоторой виртуальной машиной, которую вам надо будет заложить в вашей программе.

Это значит, что любую команду или операцию будет выполнять программа, но последовательность этих команд будет определять ее автор

не на момент компиляции, а уже после создания самой программы. И соответственно изменения, которые автор не сможет внести, не потребуют перекомпиляции всего проекта или отдельных его частей.

Второе достоинство скрипта: Возможность ограничения области назначения.

Третье достоинство: (из второго) возможность ограничения доступа к внутренностям программы без сильного ущемления функциональности изменения поведения программы.

Четвертое достоинство: (из третьего) возможность предоставления должной функциональности без доступа к внутренней (может быть даже критичной к изменениям), жесткой структуре программы (к ядру).

К недостаткам можно отнести скорость выполнения. Если машинные команды выполняются непосредственно, то одна команда скриптового языка сначала будет рассмотрена виртуальной машиной, а затем выполнена (то есть, преобразована в набор тех же машинных команд). Кроме того, как и у конечных автоматов, отсутствует возможность обучения.

1.3.3. Метод перебора

При реализации различных игр часто встает задача нахождения элемента конечного множества, обладающего заданными свойствами (если такой элемент существует). Эта задача в принципе может быть решена с помощью перебора. Есть мнение [7], что такая принципиальная возможность имеет только теоретическое значение, так как трудоемкость перебора слишком велика. Между тем, любой общий метод, в том числе и метод перебора, может оказаться как хорошим, так и плохим. Все зависит от конкретной задачи и способов реализации. При этом реализация данного метода очень зависима от достижений вычислительной техники, а она в свою очередь не перестает делать всё большие и большие успехи. Так что если сегодня что-то мы ещё не можем пересчитать, не исключено, что это станет возможным уже завтра.

В сущности, перебор - это решение задач, возникающих из заданной, когда значение некоторого искомого параметра фиксируется различным образом, и производится выбор того из рассмотренных значений, которое дает наиболее подходящее решение. Часто каждая из рассматриваемых задач с фиксированным значением данного параметра в свою очередь решается переборными методами. Тогда говорят о многоуровневом или иерархически устроенном алгоритме. Основываясь на столь общем определении перебора, можно дать лишь тривиальные рекомендации о способах его выполнения (однако, и они иногда полезны). К счастью, рассматриваемые задачи, кроме возможности их решения методами перебора, обычно имеют и другие характерные общие особенности. Это позволяет создавать, изучать и применять общие методы перебора.

Основная задача при реализации переборных методов заключается в нахождении такого порядка элементов перебора, при котором искомый элемент встретится как можно раньше. Это задача сокращения перебора. Существуют и другие проблемы, связанные с этим методом, например, проблема оптимизации генерации элементов перебора, но о них здесь речи не пойдет.

Элементы перебора и иерархические связи между ними образуют объект, называемый конечным деревом. Поиск нужного элемента в этом случае так или иначе сводится к обходу этого дерева. Один из способов сокращения перебора тогда - это отсечение бесперспективных ветвей. При этом реально никакого отсечения может и не происходить. Если, скажем, осуществлять обход дерева вглубь, выбирая ветви слева на право, то отсечение ветви - это такое изображение дерева, при котором отсекаемая ветвь окажется самой правой. Если же обход некоторой ветви не осуществляется совсем, то только в случае, когда формально доказывается отсутствие на ней решения. В связи с этим метод перебора часто называют методом полного перебора. При этом не имеется ввиду, что обязательно будут просмотрены все вершины дерева. Это лишь подразумевает, что исключена возможность не найти решение, если оно существует в одной из вершин этого дерева. Оптимизация перебора путем его сокращения

зачастую имеет критическое значение, так как все дерево перебора может содержать просто слишком много вершин.

Наиболее действенными методами сокращения перебора являются так называемые методы искусственного интеллекта. К их числу относятся, в частности, эвристические методы, а также метод самообучения. Эвристикой называют правило, которое позволяет сделать выбор при отсутствии точных теоретических обоснований. Метод самообучения заключается в изменении алгоритмом своих свойств на основании полученных им результатов работы. Одной из наиболее типичных задач, решаемых методом полного перебора, является задача выбора наилучшего хода в антагонистической позиционной игре. В то же время, в случае достаточно сложных игр, решение этой задачи не может обойтись без эффективных способов сокращения перебора, в частности без методов искусственного интеллекта.

1.3.4. Эволюционное моделирование

Данный подход является попыткой смоделировать не то, что есть, а то, что могло бы быть, если бы эволюционный процесс направлялся в нужном направлении и оценивался предложенными критериями.

Идея эволюционного моделирования сводится к экспериментальной попытке заменить процесс моделирования человеческого интеллекта моделированием процесса его эволюции. При моделировании эволюции предполагается, что разумное поведение предусматривает сочетание способности предсказывать состояние внешней среды с умением подобрать реакцию на каждое предсказание, которое наиболее эффективно ведет к цели.

Этот метод открывает путь к автоматизации интеллекта и освобождению от рутинной работы. Это высвобождает время для проблемы выбора целей и выявления параметров среды, которые заслуживают исследования. Такой принцип может быть применен для

использования в диагностике, управлении неизвестными объектами, в игровых ситуациях.

Генетические алгоритмы - адаптивные методы поиска, которые в последнее время часто используются для решения задач функциональной оптимизации. Они основаны на генетических процессах биологических организмов: биологические популяции развиваются в течении нескольких поколений, подчиняясь законам естественного отбора и по принципу "выживает наиболее приспособленный" (survival of the fittest), открытому Чарльзом Дарвином. Подражая этому процессу генетические алгоритмы способны "развивать" решения реальных задач, если те соответствующим образом закодированы. Например, генетические алгоритмы могут использоваться, чтобы проектировать структуры моста, для поиска максимального отношения прочности/веса, или определять наименее расточительное размещение для нарезки форм из ткани. Они могут также использоваться для интерактивного управления процессом, например на химическом заводе, или балансировании загрузки на многопроцессорном компьютере. Вполне реальный пример: израильская компания Schema разработала программный продукт Channeling для оптимизации работы сотовой связи путем выбора оптимальной частоты, на которой будет вестись разговор. В основе этого программного продукта и используются генетические алгоритмы.

Основные принципы генетического алгоритма были сформулированы Голландом [4] и хорошо описаны во многих работах [5 - 7]. В отличие от эволюции, происходящей в природе, генетические алгоритмы только моделируют те процессы в популяциях, которые являются существенными для развития. Точный ответ на вопрос: какие биологические процессы существенны для развития, и какие нет? - все еще открыт для исследователей.

В природе особи в популяции конкурируют друг с другом за различные ресурсы, такие, например, как пища или вода. Кроме того, члены популяции одного вида часто конкурируют за привлечение брачного партнера. Те особи, которые наиболее приспособлены к окружающим

условиям, будут иметь относительно больше шансов воспроизвести потомков. Слабо приспособленные особи либо совсем не произведут потомства, либо их потомство будет очень немногочисленным. Это означает, что гены от высоко адаптированных или приспособленных особей будут распространяться в увеличивающемся количестве потомков на каждом последующем поколении. Комбинация хороших характеристик от различных родителей иногда может приводить к появлению "суперприспособленного" потомка, чья приспособленность больше, чем приспособленность любого из его родителей. Таким образом, вид развивается, лучше и лучше приспособляясь к среде обитания.

Генетические алгоритмы используют прямую аналогию с таким механизмом. Они работают с совокупностью "особей" - популяцией, каждая из которых представляет возможное решение данной проблемы. Каждая особь оценивается мерой ее "приспособленности" согласно тому, насколько "хорошо" соответствующее ей решение задачи. Например, мерой приспособленности могло бы быть отношение силы/веса для данного проекта моста. (В природе это эквивалентно оценке того, насколько эффективен организм при конкуренции за ресурсы.) Наиболее приспособленные особи получают возможность "воспроизводит" потомство с помощью "перекрестного скрещивания" с другими особями популяции. Это приводит к появлению новых особей, которые сочетают в себе некоторые характеристики, наследуемые ими от родителей. Наименее приспособленные особи с меньшей вероятностью смогут воспроизвести потомков, так что те свойства, которыми они обладали, будут постепенно исчезать из популяции в процессе эволюции.

Так и воспроизводится вся новая популяция допустимых решений, выбирая лучших представителей предыдущего поколения, скрещивая их и получая множество новых особей. Это новое поколение содержит более высокое соотношение характеристик, которыми обладают хорошие члены предыдущего поколения. Таким образом, из поколения в поколение, хорошие характеристики распространяются по всей популяции. Скрещивание наиболее приспособленных особей приводит к тому, что

исследуются наиболее перспективные участки пространства поиска. В конечном итоге, популяция будет сходиться к оптимальному решению задачи.

В последние годы, реализовано много генетических алгоритмов и в большинстве случаев они мало похожи на классический генетический алгоритм. По этой причине в настоящее время под термином "генетические алгоритмы" скрывается не одна модель, а достаточно широкий класс алгоритмов, подчас мало похожих друг от друга. Исследователи [8, 9] экспериментировали с различными типами представлений, операторов скрещивания и мутации, специальных операторов, и различных подходов к воспроизводству и отбору.

Хотя модель эволюционного развития, применяемая в генетических алгоритмах, сильно упрощена по сравнению со своим природным аналогом, тем не менее, генетический алгоритм является достаточно мощным средством и может с успехом применяться для широкого класса прикладных задач, включая те, которые трудно, а иногда и вовсе невозможно, решить другими методами. Однако, генетические алгоритмы, как и другие методы эволюционных вычислений, не гарантируют обнаружения глобального решения за полиномиальное время. Генетические алгоритмы не гарантируют и того, что глобальное решение будет найдено, но они хороши для поиска "достаточно хорошего" решения задачи "достаточно быстро". Там, где задача может быть решена специальными методами, почти всегда такие методы будут эффективнее генетических алгоритмов и в быстродействии и в точности найденных решений. Главным же преимуществом генетических алгоритмов является то, что они могут применяться даже на сложных задачах, там, где не существует никаких специальных методов. Даже там, где хорошо работают существующие методы, можно достигнуть улучшения сочетанием их с генетическими алгоритмами.

1.4. Обучаемость виртуальных игроков

Литературе [8] используется следующее определение: обучение - процесс изменения свойств системы, позволяющий ей достигнуть наилучшего или, по крайней мере, приемлемого функционирования в изменяющихся условиях, называется адаптацией. В то же время обучение определялось как передача знаний или приемов решения задач от обучающего к обучаемому. Из этого можно сделать вывод, что самообучением в таком случае можно назвать получение системой информации, помогающей решать задачи, из собственного опыта работы. Теперь, если в определении адаптации изменить соответствующее условие на "в постоянных или изменяющихся условиях", а как субъект процесса изменения свойств указать саму систему, то полученное определение будет также подходить к понятию самообучения. Именно в таком смысле применяется понятие самообучения в современных работах по сложным позиционным играм [9, 10].

Самообучением в настоящее время принято называть выбор структуры или подбор параметров системы на основе исследования их оптимальности путем непосредственной подстановки, оценки полученных характеристик системы и соответствующей корректировки. Имеется в виду, что система в процессе работы улучшает свои характеристики, изменяя параметры и приближая их к идеальным, и таким образом "учится" на своем опыте. В применении к сказанному выше можно сказать, что игровую программу можно обучать, изменяя некоторые ее параметры, заставляя ее играть с экспертом или другой программой и закрепляя благоприятные изменения. Наиболее подходящими (но не единственными) параметрами для таких манипуляций являются параметры оценочной функции, так как их изменение не подразумевает никаких изменений в структурах данных или алгоритмах и, соответственно в тексте самой программы. В то же время, эти параметры существенно влияют на качество игры программы.

В современных играх используются различные подходы к обучению. Наиболее распространенными являются игры без возможности какого-либо

обучения, что объясняется их реализацией, т.е. автоматы с жесткой логикой, языки сценариев. Более сложными являются игры, использующие самообучение. Самообучение реализуется корректировкой таблиц, подбором параметров оценочных функций (шахматы, шашки). И, наконец, самые сложные игры, использующие последние разработки в области искусственного интеллекта, способны обучаться сложному поведению, изначально не предусмотренному программой. Так, игра Black&White предоставляет игроку возможность обучить виртуальное существо выполнять некоторые последовательности действий. Причем, какое именно действие и в какой ситуации выполнять задает игрок в процессе обучения. Метод интересен в научном плане (создание аниматов²), но поскольку игра – это коммерческий продукт, то разработчики не раскрывают свои разработки и можно лишь догадываться об использованных принципах.

² анимат – термин происходит от слов animal и robot

1.5. Выводы

Основной фактор для реализации искусственного интеллекта во всех играх заложен в понимании того, какие результаты должны быть достигнуты в той или иной ситуации. Для описания игры с математической точки зрения можно использовать теорию игр, которая позволяет описать игру и составить ее модель, а также сформулировать предпочтения игроков. Методы перебора применимы лишь в частных случаях, поскольку обладают низкой эффективностью при наличии временных ограничений и в большинстве требуют больших вычислительных мощностей. Скриптовые программы достаточно эффективны, могут изменяться (в отличие от конечных автоматов с жесткой логикой), но они требуют от автора знания оптимальных стратегий и не имеют возможностей для самообучения. Следовательно, наиболее пригодные к использованию методы, использующие какие-либо эвристики, перестают работать, либо начинают работать хуже после изменений правил игр, что характерно для так называемых "непрерывно модифицирующихся игр". Подобные игры получили распространение в связи с развитием сетевых технологий, поскольку правила модифицируются централизованно. В связи со всем перечисленным можно сделать вывод, что разработка алгоритмов, использующих различные формы обучения, является актуальной.

2. Разработка алгоритма для компьютерного игрока

2.1. Выбор и описание игры

Возьмем для рассмотрения сетевую компьютерную игру, основанную на правилах AD&D. Это ролевая настольная игра, которая была перенесена на сетевую платформу.

Пожалуй, именно в ролевых играх, как ни в каких других играх, можно использовать практические возможности искусственного интеллекта. С помощью сложной целочисленной системы подсчета очков удастся учитывать множество нюансов поведения и схваток в некоем виртуальном фантастическом мире.

AD&D немного напоминает систему Виноградова [10], когда действие разворачивается в количественно и, в принципе, неограниченном, но качественно замкнутом условном виртуальном мире. В рамках AD&D можно разыгрывать массу неповторяемых сюжетов, хотя попытки добавления новых объектов в AD&D как правило заканчивались неудачей, потому что тщательно сбалансированная и отлаженная на практике система начинает как бы жить собственной жизнью и автоматически отторгать чужеродные элементы, не вписывающиеся в ее внутреннюю структуру. Даже небольшое изменение числового значения какого-нибудь параметра, не говоря уже о добавлении новых возможностей, приводит к непредсказуемым возмущениям в совсем других местах в процессе игры, настолько плотно увязаны игровые таблицы, внешне, казалось бы, достаточно автономные.

Система правил AD&D позволяет разыгрывать игровые партии самого разного объема и продолжительности, хотя и число участников является одним из требующих настройки параметров. И несмотря на то, что AD&D не моделирует непосредственно поведение конкретных объектов, она формально описывает внутреннюю структуру виртуального мира, в котором эти объекты могут корректно существовать, жить и развиваться, и поэтому с полным правом может быть отнесена к системам искусственного интеллекта.

Основой игры является игровой сервер, имеющий связь с Internet. Именно сервер моделирует игровой мир и отвечает за поведение виртуальных игроков. Игроки – люди соединяются с сервером при помощи специальной программы – клиента, организующего поддержку протокола telnet. В игре отсутствует графика, все взаимодействие происходит в текстовом режиме. Поэтому, иногда такой тип игр называют интерактивной книгой, которая пишется в режиме реального времени. Структура игровой системы представлена на рис. 1.

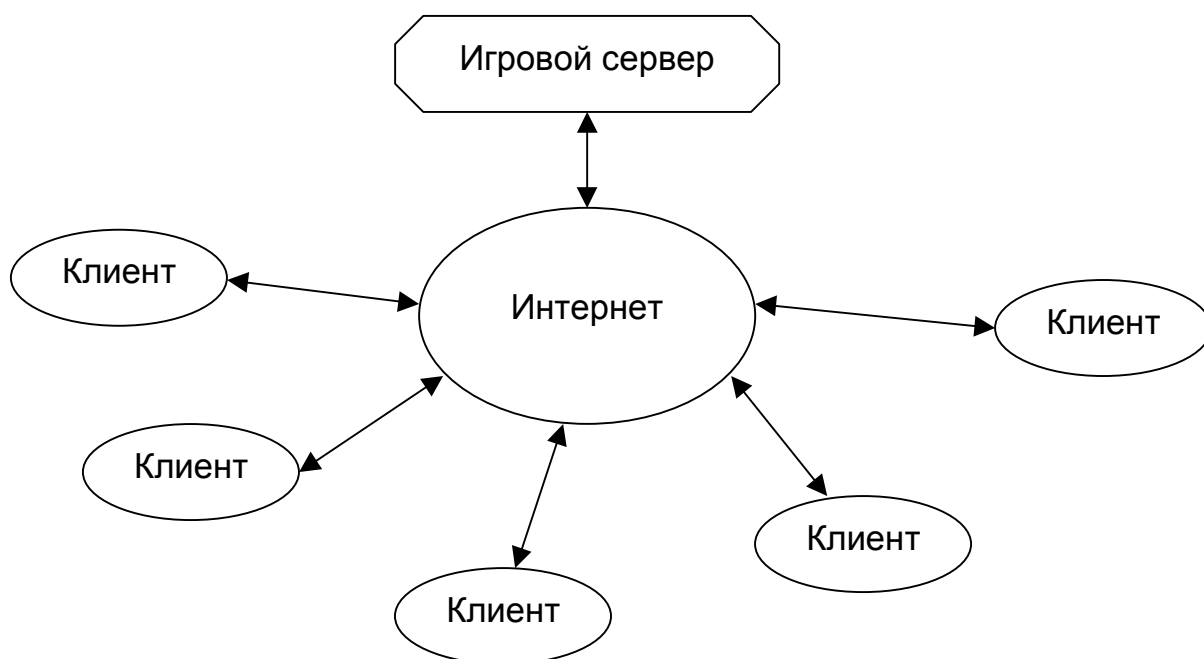


Рис.1 Структура взаимодействия клиентов с игровым сервером

Рассматриваемая игра изначально являлась настольной ролевой игрой, затем была перенесена, в связи с развитием компьютерных сетей, на сетевую платформу. Согласно правилам игры, в нее могут играть одновременно не менее двух игроков, количество участников сверху не ограничено, но по статистике их число не превышает нескольких десятков, исходя из этого, можно принять число участников конечным. Каждый игрок обладает такими параметрами, как количество здоровья (выражено в числовом эквиваленте), показатель защиты (функция, определяющая, насколько уменьшается показатель здоровья игрока при нанесении ему некоторого вреда) и некоторый конечный набор всех возможных действий. На каждом ходу игры

игрок может увеличить свои показатели здоровья и защиты, расширить текущий набор доступных действий, но не более чем полный набор доступных действий для данного игрока. Соответственно, он может уменьшить аналогичные показатели и набор доступных действий своего противника. Характер воздействия определяется совершенным ходом.

Все игроки, участвующие в игре принадлежат одной из двух коалиций, игроки из разных коалиций являются противниками, игроки из одной коалиции являются союзниками и действия игрока могут быть направлены на увеличение показателей здоровья, защиты и набора доступных действий своего союзника, если ему позволяет это его набор допустимых ходов.

Игрок выбывает из игры, если его показатель здоровья после очередного хода игры становится отрицательным, игроку засчитывается поражение. Игра заканчивается после того, как в одной из коалиций не останется больше игроков.

Каждый ход игры, в ходе которого игроки должны сделать свой выбор из числа возможных альтернатив длится фиксированный интервал времени, по истечении которого игроки узнают о действиях, совершенных другими игроками. Так как в самой игре отсутствует зависимость от времени, то временное ограничение будет учитываться лишь в алгоритмах поиска оптимального решения.

Необходимо формализовать данную игру, это позволит воспользоваться существующими методами решения подобных задач, либо адаптировать наиболее подходящий метод.

Классифицируем игру, это пригодится при составлении математической модели [11]:

1. Число участников не менее двух, так как мы приняли его конечным, то обозначим его как n .
2. Число стратегий конечно, так как каждый игрок обладает ограниченным набором допустимых ходов, следовательно, игра является конечной.
3. В игре присутствует конфликт между игроками, из чего можно сделать вывод, что игра является антагонистической.
4. Согласно правилам игры, в каждой игре всегда присутствует две коалиции, причем коалиция может состоять из одного игрока. Наличие коалиций свидетельствует о том, что это кооперативная игра.
5. Данная игра является игрой с полной информацией, поскольку в ней нельзя сделать одновременно несколько ходов одним из игроков, а также участникам известны выборы, сделанные при предшествующих ходах.

Правила игры допускают существование таких ходов, выполнение которых может привести с некоторой вероятностью к завершению игры. Для того чтобы упростить задачу и не учитывать фактор риска, введем в модель допущение, запрещающее подобные ходы. В дальнейшем исследовании задача может быть усложнена снятием данного ограничения.

2.2. Математическая модель

Обозначим множество игроков через N , всего игроков n .

X_i - множество стратегий игрока i , $i = \overline{1, n}$

X_i^t - текущий набор стратегий игрока i . В начале игры $X_i^t = X_i$

$x^i \in X_i^t$ - стратегии игрока i

K_j - коалиции игроков, $\bigcap_{j=1,2} K_j = \emptyset$, $\bigcup_{j=1,2} K_j = N$.

S - множество всех возможных исходов принимаемых решений

$S(x^i)$ - множество возможных исходов, если игрок применяет одну из стратегий

x^i

h_i - количество жизни i -го игрока

a_i - защита i -го игрока, параметр используется в функции f_1

Функция, описывающая изменение количества жизни, получаемое игроком i при использовании на нем стратегии x :

$$h_i = f_1(i, x), \forall x \in \{X_i\}_{i=1}^n. \quad (1)$$

Функция, описывающая изменение защиты, получаемое игроком i при использовании на нем стратегии x :

$$a_i = f_2(i, x), \forall x \in \{X_i\}_{i=1}^n. \quad (2)$$

Функция, описывающая изменение числа доступных стратегий, получаемое игроком i при использовании на нем стратегии x . На сколько количество изменилось число стратегий, описывает параметр m_i :

$$m_i = f_3(i, x), \forall x \in \{X_i\}_{i=1}^n. \quad (3)$$

Согласно правилам игры, выбор любой из стратегий (совершение одного хода игроком) сказывается следующим образом: стратегия оказывает влияние на результат только одной из функций f_1, f_2, f_3 , и изменяет соответствующий параметр, для всех прочих функций значение параметра не изменяется. Примем, что решение является оптимальным, если оно максимизирует сумму результатов функций f_1, f_2, f_3 для любого из игроков, входящих в ту же коалицию, что и игрок i . Это соответствует равновесию

коалиционных игр по Нэшу, поскольку цель игрока заключается в улучшении состояния любого из игроков, с которым он находится в одной коалиции, включая себя. Таким образом, игрок не ставит на первое место личные интересы. Если выбранная стратегия не оказывает влияния на игроков - союзников, то следует попытаться минимизировать сумму результатов функций f_1, f_2, f_3 для всех прочих игроков.

Тогда отношение предпочтения можно выразить как:

$$\sum_{x^i} = f_1(l, x^i) + f_2(l, x^i) + f_3(l, x^i) - f_1(g, x^i) - f_2(g, x^i) - f_3(g, x^i), \quad (4)$$

для всех l принадлежащих той же коалиции что и i , и всех g не принадлежащих этой коалиции.

В результате получаем игру, которую можно записать как:

$$\Gamma = \langle N, S, \{X^i\}, \{S(x^i)\}, \{\sum_{x^i}\} \rangle. \quad (5)$$

В конце каждого хода игры вычисляется значение h_i по формуле f_1 , новое значение a_i по формуле f_2 , определяется новое значение X^i , если $m_i \neq 0$, которые используются на следующем ходу игры.

Условие для дальнейшего участия игрока i в игре:

$$h_i > 0, i = \overline{1, n}. \quad (6)$$

Условие продолжения игры:

$$K_j \neq \emptyset, j = \overline{1, 2}. \quad (7)$$

Полученное описание соответствует следующим требованиям:

- 1) Перечислены ходы, которые могут делать игроки.
- 2) Определена информация, которой располагают игроки в процессе игры.
- 3) Определены возможные варианты действий игроков.
- 4) Возможно определение предельных размеров платежей в конце игры (единственно важный показатель в конце игры – значения h_i может быть вычислен точно).

Следовательно, игра находится в развернутой (экстенсивной) форме и ее описание может быть составлено в виде дерева игры. В данном случае предпочтительнее использование функций полезности, чем составление громоздкой платежной матрицы. Существование функции полезности является необходимым и достаточным условием определенности оптимального выбора в данной задаче, следовательно, нет необходимости строить платежную матрицу, для поиска оптимального решения на каждом из шагов игры можно пользоваться функцией оценки альтернатив – функцией полезности.

2.3. Математические методы решения задачи

Для поиска решения на дереве игры можно использовать различные методы. Наиболее простой из них – метод полного перебора. Поскольку игра конечна (см. классификацию), то можно утверждать, что решение будет найдено. Но этот метод имеет наименьшую эффективность и может требовать для решения много времени. Согласно условиям, время на каждый ход строго лимитировано, следовательно, метод полного перебора не подходит.

Существуют методы, использующие фиксированную глубину перебора дерева. Они экономят вычислительные мощности, но не гарантируют нахождения оптимального решения.

Более точные методы – используют различные эвристические процедуры, что позволяет находить более полезные решения, чем метод с принудительным ограничением глубины или ширины поиска, но требуют дополнительного исследования для нахождения необходимого набора эвристик.

Наиболее быстрым (без поиска эвристик) в рассматриваемом случае является метод граней и оценок. Данный метод специально разрабатывался для анализа игровых деревьев и гарантирует высокую эффективность при более низких затратах на вычисления, чем метод полного перебора и поиска с фиксированной глубиной (при достаточно большой глубине, при малой глубине метод поиска с фиксированной глубиной менее требователен к ресурсам), при этом точность метода граней и уступок выше.

Но все эти методы не подходят в рассматриваемом случае, поскольку ограничение времени, отводимое на каждый ход, оговорено явно в правилах игры. Следовательно, данная игра не может быть решена методом перебора, даже с использованием оптимизаций. Используем полученную модель для реализации другого метода.

2.4. Применение генетического программирования

2.4.1. Общая теория

Как известно, эволюционная теория утверждает, что жизнь на нашей планете возникла вначале лишь в простейших формах - в виде одноклеточных организмов. Эти формы постепенно усложнялись, приспособляясь к окружающей среде, порождая новые виды, и только через много миллионов лет появились первые животные и люди. Каждый биологический вид с течением времени совершенствует свои качества так, чтобы эффективней справляться с важнейшими задачами выживания, самозащиты, размножения и т.д. Таким путем возникла защитная окраска у многих рыб и насекомых, панцирь у черепахи, яд у скорпиона и многие другие полезные приспособления.

С помощью эволюции природа постоянно оптимизирует все живое, находя неординарные решения. Неясно, за счет чего происходит этот прогресс, однако ему можно дать научное пояснение, базируясь всего на двух биологических механизмах - естественного отбора и генетического наследования.

Ключевую роль в эволюционной теории играет естественный отбор. Его суть состоит в том, что наиболее приспособленные особи лучше выживают и приносят больше потомков, чем менее приспособленные. Заметим, что сам по себе естественный отбор еще не обеспечивает развитие биологического вида. Поэтому очень важно понять, каким образом происходит наследование, то есть как свойства потомка зависят от свойств родителей.

Основной закон наследования интуитивно понятен каждому - он состоит в том, что потомки похожи на родителей. В частности, потомки более приспособленных родителей будут, скорее всего, одними из наиболее приспособленных в своем поколении. Чтобы понять, на чем основано это сходство, нужно немного углубиться в построение естественной клетки - в мир генов и хромосом.

Почти в каждой клетке любой особи есть набор хромосом, несущих информацию про эту особь. Основная часть хромосомы - нить ДНК,

определяющая, какие химические реакции будут происходить в данной клетке, как она будет развиваться и какие функции выполнять.

Ген - это отрезок цепи ДНК, ответственный за определенное свойство особи, например за цвет глаз, тип волос, цвет кожи и т.д. Вся совокупность генетических признаков человека кодируется с помощью приблизительно 60 тыс. генов, длина которых составляет более 90 млн. нуклеотидов.

Различают два вида клеток: половые (такие, как сперматозоид и яйцеклетка) и соматические. В каждой соматической клетке человека содержится 46 хромосом. Эти 46 хромосом - на самом деле 23 пары, причем в каждой паре одна из хромосом получена от отца, а вторая - от матери. Парные хромосомы отвечают за одинаковые признаки - например, родительская хромосома может содержать ген черного цвета глаз, а парная ей материнская - ген голубого цвета. Существуют определенные законы, управляющие участием тех или иных генов в развитии особи. В частности, в нашем примере потомок будет черноглазым, поскольку ген голубых глаз является "слабым" и подавляется геном любого другого цвета.

В половых клетках хромосом только 23, и они непарные. При оплодотворении происходит слияние мужской и женской половых клеток и получается клетка зародыша, содержащая 46 хромосом. Какие свойства потомок получит от отца, а какие - от матери? Это зависит от того, какие именно половые клетки принимали участие в оплодотворении. Процесс выработки половых клеток (так называемый мейоз) в организме подвергается случайностям, благодаря которым потомки во многом отличаются от своих родителей. При мейозе, происходит следующее: парные хромосомы соматической клетки сближаются вплотную, потом их нити ДНК разрываются в нескольких случайных местах и хромосомы обмениваются своими частями (рис. 3).

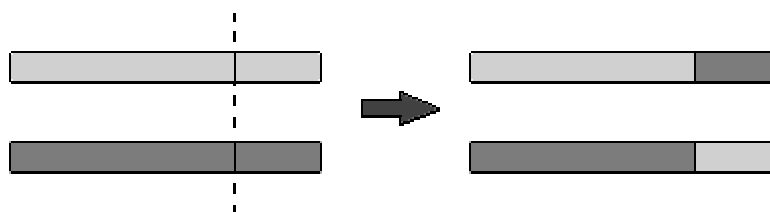


Рис. 3 Условная схема кроссинговера

Этот процесс обеспечивает появление новых вариантов хромосом и называется "кроссинговер". Каждая из вновь появившихся хромосом окажется затем внутри одной из половых клеток, и ее генетическая информация может реализоваться в потомках данной особи.

Второй важный фактор, влияющий на наследственность, - это мутации, которые выражаются в изменении некоторых участков ДНК. Мутации также случайны и могут быть вызваны различными внешними факторами, такими, как радиоактивное облучение. Если мутация произошла в половой клетке, то измененный ген может передаться потомку и проявиться в виде наследственной болезни либо в других новых свойствах потомка. Считается, что именно мутации являются причиной появления новых биологических видов, а кроссинговер определяет уже изменчивость внутри вида (например, генетические различия между людьми).

Как уже были отмечено выше, эволюция - это процесс постоянной оптимизации биологических видов. Естественный отбор гарантирует, что наиболее приспособленные особи дадут большое количество потомков, а благодаря генетическому наследованию, часть потомков не только сохранит высокую приспособленность родителей, но будет иметь и новые свойства. Если эти новые свойства оказываются полезными, то с большой вероятностью они перейдут и в следующее поколение. Таким образом, происходит накопление полезных качеств и постепенное повышение приспособленности биологического вида в целом. Зная, как решается задача оптимизации видов в природе, применим похожий метод для решения разных реальных задач.

Задачи оптимизации - наиболее распространенный и важный для практики класс задач. Их приходится решать любому из нас или в быту, распределяя свое время между разными делами, или на работе, добиваясь максимальной скорости работы программы или максимальной прибыльности компании - в зависимости от должности. Среди этих задач есть решаемые простым путем, но есть и такие, точное решение которых найти практически невозможно.

Введем обозначения и приведем несколько классических примеров. Как правило, в задаче оптимизации мы можем руководить несколькими параметрами (обозначим их значения через x_1, x_2, \dots, x_n , целью является максимизация (или минимизация) некоторой целевой функции, $f(x_1, x_2, \dots, x_n)$, зависящей от этих параметров. Например, если нужно максимизировать целевую функцию "доход компании", тогда управляемыми параметрами будут число сотрудников компании, объем производства, затраты на рекламу, цены на конечные продукты и т.д. Эти параметры связаны между собой - в частности, при уменьшении числа сотрудников скорее всего упадет и объем производства.

Генетический алгоритм - новейший, но не единственно возможный способ решения задач оптимизации. Известно два основных пути решения таких задач - метод перебора или градиентный метод. Рассмотрим классическую задачу коммивояжера. Суть задачи состоит в нахождении кратчайшего пути прохождения всех городов.

Переборный метод проще. Для поиска оптимального решения (максимум целевой функции) нужно последовательно вычислить значения функции во всех точках. Недостатком является большое количество вычислений.

Другой способ - градиентный спуск. Выбираем случайные значения параметров, далее значения постепенно изменяются, достигая наибольшей скорости роста целевой функции. Алгоритм может остановиться, достигнув локального максимума. Градиентные методы быстрые, но не гарантируют оптимального решения (поскольку целевая функция имеет несколько максимумов).

Генетический алгоритм представляет собой комбинацию переборного и градиентного методов. Механизмы кроссинговера (скрещивания) и мутации реализуют переборную часть, а отбор лучших решений - градиентный спуск. То есть, если на некотором множестве задана сложная функция от нескольких переменных, тогда генетический алгоритм является программой, которая за допустимое время находит точку, где значение функции находится довольно

близко к максимально возможному значению. Выбирая приемлемое время расчета, получаем лучшие решения, которые можно получить за это время.

Представим искусственный мир, населенный множеством существ (особей), причем каждая особь - это некоторое решение задачи. Будем считать особь более приспособленной, чем лучше соответствующее решение (чем больше значение целевой функции оно дает). Тогда задача максимизации целевой функции сводится к поиску более приспособленной особи. Конечно, мы не можем поселить в наш виртуальный мир все особи сразу, так как их очень много. Вместо этого будем рассматривать множество поколений, которые сменяют друг друга. Теперь, если мы сумеем задействовать естественный отбор и генетическое наследование, тогда полученная среда будет подчиняться законам эволюции. Целью этой искусственной эволюции будет создание наилучших решений. Очевидно, эволюция - бесконечный процесс, в ходе которого приспособленность особей постепенно повышается. Принудительно остановив этот процесс через длительное время после его начала и выбрав наиболее приспособленную особь в текущем поколении, получим не абсолютно точный, но близкий к оптимальному ответ. Такова идея генетического алгоритма. Перейдем теперь к точным определениям и опишем работу генетического алгоритма детальней.

Для того чтобы говорить о генетическом наследовании, нужно наделить наши особи хромосомами. В генетическом алгоритме хромосома - это некоторый числовой вектор, который отвечает подбираемому параметру, а набор хромосом данной особи определяет решение задачи. Какие именно векторы следует рассматривать в конкретной задаче, решает сам пользователь. Каждая из позиций вектора хромосомы называется геном.

Простой генетический алгоритм случайным образом генерирует начальную популяцию. Работа генетического алгоритма представляет итерационный процесс, который продолжается до тех пор, пока не выполнится заданное число поколений или любой другой критерий остановки. В каждом поколении генетического алгоритма реализуется отбор пропорционально приспособленности, одноточечный кроссинговер и мутация. Сначала, пропорциональный отбор назначает каждой структуре вероятность

$P_s(i)$ равную отношению ее приспособленности к суммарной приспособленности популяции:

$$P_s(i) = \frac{f(i)}{\sum_{i=1}^n f(i)}. \quad (8)$$

Затем происходит отбор (с замещением) всех n особей для дальнейшей генетической обработки, соответственно величине $P_s(i)$.

При таком отборе члены популяции с высокой приспособленностью с большей вероятностью будут выбираться чаще, чем особи с низкой приспособленностью. После отбора, n избранных особей случайным образом разбиваются на $n/2$ пары. Для каждой пары с вероятностью P_s может применяться кроссинговер. Соответственно, с вероятностью $1-P_s$ кроссинговер не происходит и неизмененные особи переходят на стадию мутации. Если кроссинговер происходит, полученные потомки заменяют родителей и переходят к мутации.

Определим понятия, отвечающие мутации и кроссинговеру в генетическом алгоритме.

Мутация - это преобразование хромосомы, которое случайно изменяет одну или несколько ее позиций (генов). Наиболее распространенный вид мутаций - случайное изменение только одного из генов хромосомы.

Кроссинговер (в литературе по генетическим алгоритмам также употребляется название кроссовер или скрещивание) - это операция, при которой из двух хромосом порождается одна или несколько новых хромосом. Одноточечный кроссинговер работает следующим образом. Сначала, случайным образом выбирается одна из $l-1$ точек разрыва. (Точка разрыва - участок между соседними битами в строке.) Обе родительские структуры в этой точке разрываются на два сегмента. Потом, соответствующие сегменты разных родителей склеиваются и выходят два генотипа потомков.

После того как заканчивается стадия кроссинговера, выполняются операторы мутации. В строке, к которой применяется мутация, каждый бит с вероятностью P_m изменяется на противоположный. Популяция, полученная после мутации записывает поверх старой и цикл одного поколения

завершается. Следующие поколения обрабатываются подобным образом: отбор, кроссинговер и мутация.

Элитные методы отбора гарантируют, что при отборе обязательно будут выживать лучший или лучшие члены популяции совокупности. Наиболее распространена процедура обязательного сохранения только одной лучшей особи, если она не прошла как другие через процесс отбора, кроссинговера и мутации. Элитизм может быть введен практически в любой стандартный метод отбора.

Двухточечный кроссинговер и равномерный кроссинговер - достойные альтернативы одноточечному оператору. В двухточечном кроссинговере выбираются две точки разрыва, и родительские хромосомы обмениваются сегментом, находящемся между этими точками. В равномерном кроссинговере, каждый бит первого родителя наследуется первым потомком с заданной вероятностью, в противном случае этот бит передается второму потомку. И наоборот.

Блок-схема генетического алгоритма изображена на рис. 4. Сначала генерируется начальная популяция особей (индивидуумов), то есть некоторый набор решений задачи. Как правило, это делается случайным образом. Потом моделируется размножение внутри популяции. Для этого, случайно отбирается несколько паров индивидуумов, происходит скрещивание между хромосомами в каждой паре, а полученные новые хромосомы переходят в популяцию нового поколения. В генетическом алгоритме сохраняется основной принцип естественного отбора - чем приспособленней индивидуум (чем больше соответствующее ему значение целевой функции), тем с большей вероятностью он будет брать участие в скрещивании. Теперь моделируются мутации - в нескольких случайно избранных особях нового поколения изменяются некоторые гены. Старая популяция частично или целиком уничтожается и переходим к рассмотрению следующего поколения. Популяция следующего поколения в большинстве реализаций генетических алгоритмов содержит столько же особей, сколько начальная, но в силу отбора приспособленность в ней в среднем выше. Теперь описанные процессы отбора, скрещивания и мутации повторяются уже для этой популяции и т.д.

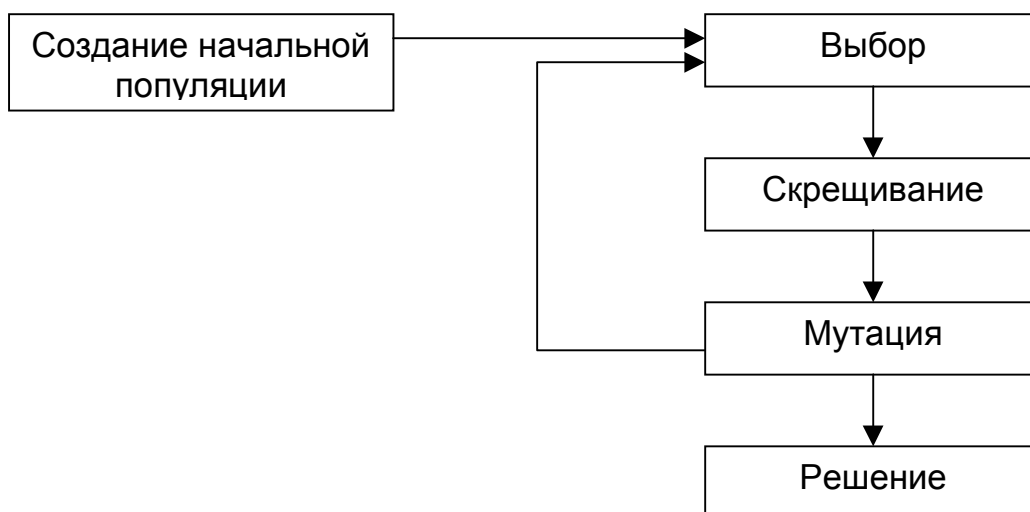


Рис. 4 Структурная схема генетического алгоритма

В каждом следующем поколении наблюдается возникновение новых решений задачи. Среди них будут как плохие, так и хорошие, но благодаря отбору, число приемлемых решений будет возрастать. Заметим, что в природе не бывает абсолютных гарантий, и приспособленный тигр может погибнуть от ружейного выстрела, не оставив потомков. Имитируя эволюцию на компьютере, можно избежать подобных нежелательных событий и всегда сохранять жизнь лучшему из индивидуумов текущего поколения - такая методика называется "стратегией элитизма".

2.4.2. Адаптация генетического алгоритма

Классический генетический алгоритм традиционно имеет критерий останова. После того, как будет найдено оптимальное решение, алгоритм останавливает свою работу. Поскольку мы ищем последовательности элементарных ходов, обладающих требуемой полезностью, изменим генетический алгоритм для наших нужд. На рис. 4 приведена схема такого алгоритма.

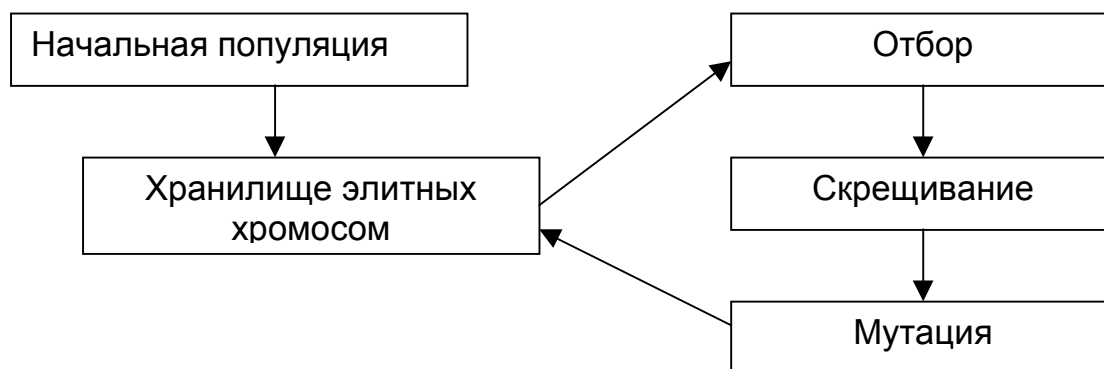


Рис. 5 Структурная схема адаптированного алгоритма

2.4.3. Выбор способа кодирования решений

Решением будем считать некоторую последовательность команд, записанную в виде списка. Таким образом, хромосома для нашей задачи может быть представлена в виде некоторой последовательности команд, отвечающих следующему требованию: так как правила игры допускают сокращение активного набора альтернатив, каждое последующее действие в цепочке должно присутствовать в активном наборе. На рис. 6 приведено описание такой хромосомы.

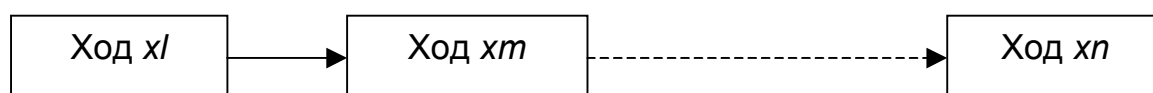


Рис. 6 Структура хромосомы

Чтобы использовать хромосому, игрок должен иметь в множестве возможных ходов ходы x_1 , x_m , ..., x_n , кроме того, для того, чтобы начать реализацию хромосомы, игрок должен иметь в активном наборе ход x_1 .

Так как в работе используется кодирование хромосом переменной длины, для борьбы с возможным "распуханием" решения введем параметр L , характеризующий максимально возможную длину решения.

Для обозначения возможности выбора цепочки введем понятие *начальный ключ* – $Ks = \{x_l; S\}$. Данная запись означает, что множество ходов $S = \{x_l, \dots, x_n\}$, используемых в хромосоме, является подмножеством множества всех возможных ходов X_i (см. математическую модель), а x_l входит в активный набор X_i .

Для обозначения возможности продолжения цепочки введем понятие *конечный ключ* – $Ke = \{X_i\}$, который характеризует активный набор ходов на момент окончания цепочки.

Будем говорить, что для игрока i с полным набором стратегий X_i ключи Ks и Ke совпадают, если $x_l \in X_i$ и $S \subset X_i$.

2.4.4. Формулировка правил скрещивания и мутации.

Новый ген формируется из двух родителей путем разрезания хромосом на две части, причем для новых хромосом формируются новые ключи Ks и Ke . На рис. 7 приведен пример скрещивания двух хромосом.

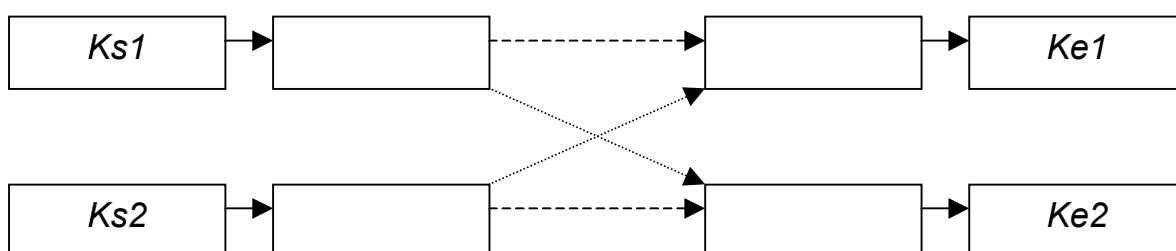


Рис. 7 Скрещивание хромосом

В результате скрещивания образуются новые хромосомы, причем конечные ключи просто обмениваются, а для начальных ключей вычисляется новое значение множеств S , путем объединения всех используемых ходов в гене, соответственно дублирующиеся ходы учитываются один раз.

Мутация является частным случаем скрещивания с хромосомой длиной в один ход.

2.4.5. Правило отбора

Поскольку возможны различные исходы игры, будем использовать следующие правила отбора:

- Приоритетное
- Использование решения, полученного с помощью модели
- Безусловное

Приоритетное правило работает для игрока, выигравшего игру. Если этот игрок – компьютерная программа, то считаем, что нам известны все хромосомы, использованные в игре. Используем их для дальнейшего развития.

Если выигравший игрок – человек, то используем для скрещивания следующие хромосомы, полученные с помощью математической модели:

- 1) Запоминаем первый ход игрока.
- 2) На каждом шаге игры вычисляем лучший из возможных ходов игрока по функции полезности.
- 3) Записываем последовательность ходов до тех пор, пока игрок не сделает на каком-то шаге лучший ход. Тогда записываем последовательность ходов в виде хромосомы и переходим к шагу 1. Если последовательность ходов превышает максимальную, то переходим к шагу 1.

Безусловное скрещивание используется в случае, если выиграл противник, но не удалось выделить последовательности ходов при помощи математической модели. Тогда используется собственный набор хромосом в комплексе с повышенной вероятностью мутации.

2.5. Описание принципа работы полученного алгоритма

При первом запуске алгоритма (хранилище элитных хромосом пусто, см. рис. 5) генерируется набор случайных хромосом, которые помещаются в хранилище. Это противоречит его назначению, но в течение некоторого времени эти хромосомы будут замещены, в случае их негодности. После чего виртуальный игрок извлекает из хранилища хромосомы с совпадающими ключами, имея при этом ключ Ke . В случае если игрок не может выполнить очередной ход из хромосомы, он помечает хромосому как потенциально ошибочную (счетчик) и выбирает следующую хромосому из хранилища. При превышении некоторого порога неудачных попыток использования хромосома удаляется. Если игрок не может совершить найти хромосому с совпадающим ключом – он совершает случайный ход. В конце игры производится операция отбора, скрещивания и мутации, результат помещается в хранилище, если в отборе участвовали хромосомы из хранилища – они замещаются новыми хромосомами. В дальнейшем процессе развития используются хромосомы из хранилища. Блок – схема алгоритма представлена на рис. 8.

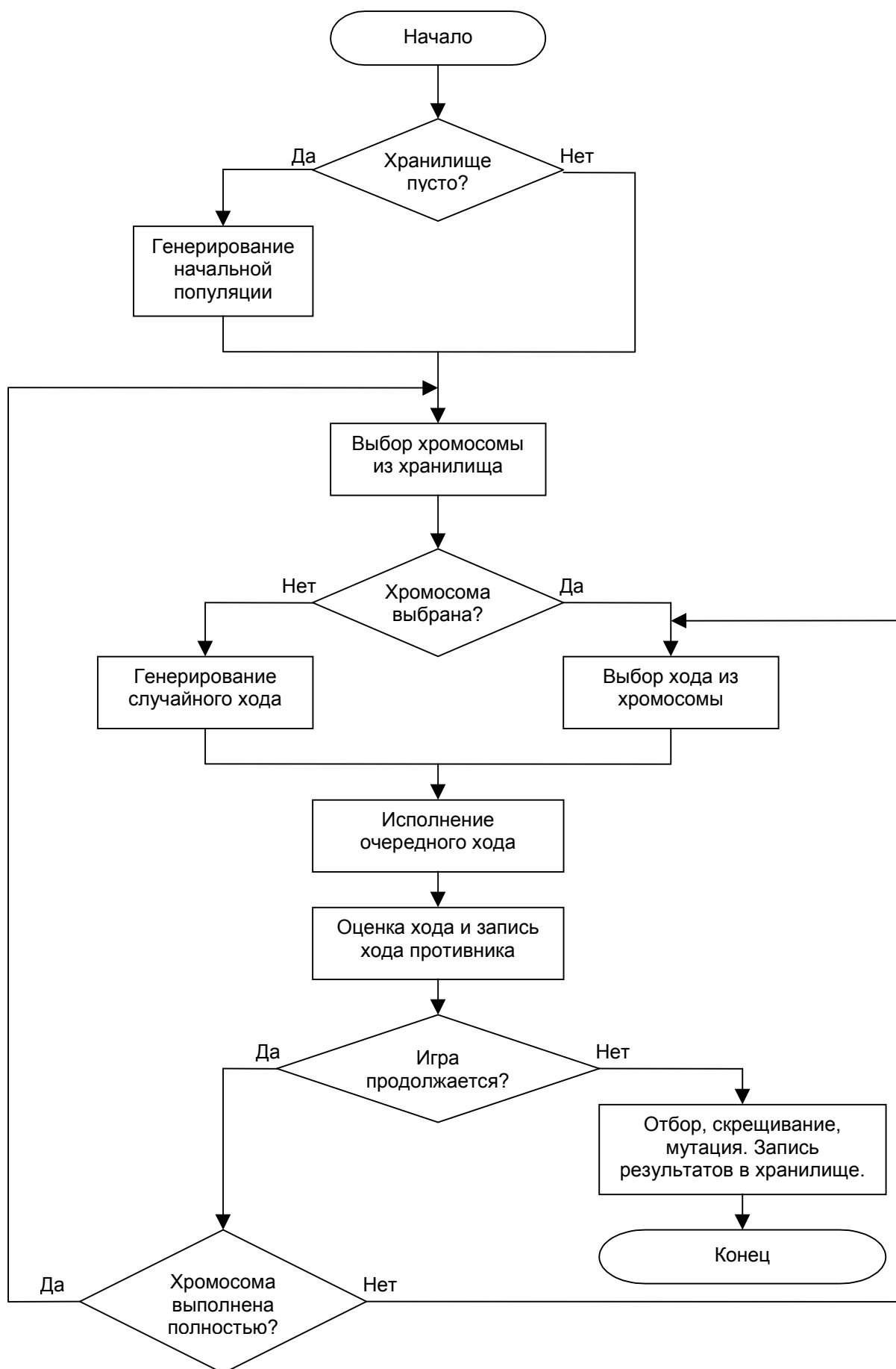


Рис. 8 Блок – схема разработанного алгоритма

2.6. Выводы

Разработанный алгоритм является не поиском в пространстве параметров модели, а поиском в пространстве решений задачи. Из чего можно сделать вывод, что с его помощью можно найти более эффективные решения задачи.

На каждом из шагов игры вычисляется только функция полезности, дерево решений не просчитывается. Следовательно, алгоритм не является требовательным к вычислительным мощностям компьютера. Это утверждение будет являться верным при условии, что функция полезности не будет сложной. Таким образом, поиск решения будет растянут во времени, с сохранением промежуточных результатов.

К недостаткам можно отнести использование допущений при построении модели, что отрицательно сказывается на ее точности.

Заключение

В данной работе был рассмотрен подход использования знаний в компьютерных играх. В данном случае под знаниями понималось "умение" виртуального игрока играть в некоторую игру.

В ходе работы была выбрана, детально описана и классифицирована ролевая сетевая игра, составлена ее математическая модель. Разработан алгоритм, позволяющий извлекать знания из действий игрока – человека, с использованием математической модели.

Для оценки полученных знаний был разработан генетический алгоритм, для которого были сформулированы правила скрещивания и мутации в соответствии с особенностями задачи. Конечным результатом работы алгоритма является набор стратегий. Данный набор используется как набор "элементарных" действий компьютерного оппонента. К достоинствам этого метода можно отнести то, что эволюция не останавливается, постоянно идет поиск новых решений. Такая особенность обладает полезным качеством – при изменении правил игры алгоритм через некоторое время вновь найдет некоторые оптимальные решения. К недостаткам алгоритма можно отнести то, что для оценки используется грубая модель, которая может влиять как на качество решения, так и на время поиска. Учитывая то, что пространство решений очень велико, а эволюция идет достаточно медленно – неточность модели может сильно сказываться на получении результатов.

В ходе работы были самостоятельно изучены элементы теории игр и генетических алгоритмов в объеме, необходимом для проведения исследования.

В данной работе не было рассмотрено использование нейронных сетей для обработки знаний в играх. Поскольку нейронные сети представляют собой достаточно обширную область искусственного интеллекта, можно рассматривать такое исследование как самостоятельное или являющееся продолжением данного.

Список литературы:

1. Webster's New Collegiate Dictionary, Merriam Co., Springfield, Mass., 1956,- 1239.
2. International Game Developers Association - <http://www.igda.org/review.php>
3. Winograd T. A Procedural Model of Language Understanding // Computer Models of Thought and Language / Schank R. C., Colby K. M. (eds.), 1973, pp. 152 - 86. San Francisco: W. H. Freeman.
4. Holland J.H. Adaptation in Natural and Artificial Systems. Ann Arbor: The University of Michigan Press, 1975, - 371.
5. Goldberg D.E. Genetic Algorithms in Search, Optimization, and Machine learning. Addison - Wesley, 1989, - 603.
6. Sean Luke. Issues in Scaling Genetic Programming: Breeding Strategies, Tree Generation, and Code Bloat, 2000. – 178.
7. Koza, J. R. Genetic Programming: On the Programming of Computers by Means of Natural Selection, 1992. – 519.
8. А. Тьюринг, Может ли машина мыслить?, М.: Физматиздат, 1960. – 235 с.
9. Льюис Р.Д., Райфа Х. Игры и решения. - М.: ИЛ, 1961. – 423 с.
10. Шикин Е.В. От игр к играм. Математическое введение. - М.: Эдиториал УРСС, 1998. – 483 с.
11. Вилкас Э.Й. Оптимальность в играх и решениях. – М.: Наука. Гл. ред. физ. - мат. лит., 1990. – 256 с.

Листинг программы