МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

МЕХАНИКО-МАТЕМАТИЧЕСКИЙ ФАКУЛЬТЕТ

Кафедра дифференциальных уравнений и системного анализа

АГЕНТНЫЕ МОДЕЛИ С ЭЛЕМЕНТАМИ ПАМЯТИ И ОБУЧЕНИЯ С ПОДКРЕПЛЕНИЕМ

Курсовая работа

Ключник Карины
Владимировны

студентки 3 курса,
специальность 1-31 03 09
Компьютерная математика
и системный анализ

Научный руководитель:
кандидат физ.-мат. наук,
доцент О. А. Лаврова

Минск, 2021

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Как известно, моделирование – это общенаучный метод познания окружающего мира, при нем могут применяться все остальные методы познания (эмпирические: наблюдение, эксперимент, измерение; теоретические: абстрагирование, идеализация, формализация, индукция, дедукция; общенаучные: анализ, синтез, аналогия).

Курсовая работа будет неотрывно связана с моделированием, поэтому необходимо показать, что оно действительно необходимо и значимо. Что дает моделирование? Во-первых, это экономичность (сбережение ресурсов для проведения реальных экспериментов, т.е. отсутствие необходимости вкладывать деньги в построение настоящих осязаемых моделей). Во-вторых, безопасность, т.к. постановка экспериментов и наблюдений за объектами в реальных условиях могут быть затруднены/опасны (например, могут случаться аварийные ситуации; астрофизические явления также затрудняют эксперименты). Следующее преимущество моделирования – это универсальность (различные объекты могут описываться одними и теми же математическими моделями). Еще стоит отметить, что во время моделирования можно контролировать (ускорять и замедлять) время изучаемого процесса. Таким образом, курсовая работа актуальна, т.к. актуальна сама тема моделирования.

Сейчас так же большой популярностью пользуется машинное обучение, т.е. построение моделей, которые каким-то образом «обучаются», а потом начинают сами, на основании того, чему научились, предсказывать результаты. Все это актуально, и те, кто занимается машинным обучениям, на данный момент самые высокооплачиваемые работники сферы информационных технологий.

В последнее время особой популярностью пользуется интеграция моделирования и машинного обучения. В данной работе на примере сахарной модели попробуем понять, насколько уместной является такая интеграция.

Курсовая работа носит исследовательский характер. Целью работы является изучение внедрения машинного обучения в агентные модели.

Основными задачами курсовой работы является разработка и реализация модели Sugarscape в Anylogic без обучения агентов, разработка и реализация модели Sugarscape с обучением на основе Q-learning алгоритма, а также сравнительный анализ двух моделей.

ГЛАВА 1
ОБЩИЕ ПОЛОЖЕНИЯ

1.1 Понятие об агентном моделировании

Имитационное моделирование – это метод, который позволяет строить модели, описывающие процессы так, как они бы происходили в действительности.

Имитационное моделирование применяют для систем, которым свойственно нелинейное взаимодействие, «память», неочевидные зависимости между переменными, причинно-следственные связи, неопределенность и большое количество параметров. С задачами такого рода аналитическое (т.е. основанное на формулах) решение крайне сложно найти и тем более построить ментальную модель. [1]

Обозначим три вида имитационного моделирования:

- системная динамика (построение диаграмм причинно-следственных связей и влияний одних параметров на другие, на основе этих диаграмм модель имитируется на компьютере);
- дискретно-событийное моделирование (рассматриваются только основные события моделируемой системы, несущественные детали опускаются, т.е. происходит абстрагирование от непрерывной природы);
- агентное моделирование (исходя из предложений о поведении конкретного объекта, моделируется поведение группы таких объектов).

Агентное моделирование – это один из видов имитационного моделирования. Основным его преимуществом является то, что можно не знать поведение системы в целом, необходимы лишь сведения о том, как ведут себя отдельные объекты. Часто нужно видеть, как агенты ведут себя при взаимодействии друг с другом, со средой, которая может иметь собственную динамику. Таким образом, глобальное поведение системы формируется из многих процессов, протекающих параллельно, т.е. индивидуальное поведение каждого агента формирует глобальное поведение системы.

Агентная модель состоит из агентов, отношений и среды. Не существует единого соглашения об определении агента в агентном моделировании. Одно из определений агента определено далее.

Агент – это элемент модели, который моделирует некую сущность, и обладает определенными свойствами. Агент обладает свойством автономности. Агент

взаимодействует (локально) с другими агентами в некоторой среде и/или со средой. Взаимодействие может определяться простыми правилами или абстрактными моделями (нейронные сети и генетические алгоритмы). Агент имеет состояния, которые определяют поведение агента. Агент имеет границу, может обладать ресурсами и памятью. Агент может быть целенаправленным, адаптироваться и обучаться.

Необходимо отметить несколько фактов, касающихся агентов:

- агенты – это не обязательно люди; агентом может быть все, что угодно: человек, транспортное средство, земельные участки, проект, оборудование, организация или даже идея;
- агентные модели – это не клеточные автоматы и, вообще говоря, они не обязательно обитают в дискретном пространстве (во многих агентных моделях пространства вообще нет, либо же оно непрерывное);
- агентом может быть даже тот объект, который кажется пассивным;
- агенты могут вообще не взаимодействовать друг с другом.

Итак, при разработке агентной модели создаются агенты, задается их поведение и далее агентов помещают в некую среду, устанавливают возможные связи между агентами, так модель готова к запуску.

Агентное моделирование является инструментом, при помощи которого возможно успешное моделирование сложных адаптивных систем. Оно нашло применение в различных областях знаний: от моделирования процессов фондовых бирж до предсказания распространения инфекционных заболеваний. [2]

1.2 Программное обеспечение для имитационного моделирования – AnyLogic

AnyLogic – это современная среда разработки моделей на языке Java с русскоязычным графическим интерфейсом и тщательно продуманной контекстной справочной системой. [3] AnyLogic содержит большую библиотеку визуальных компонентов. Разработчик также может создавать и добавлять в среду собственные компоненты. AnyLogic используется для разработки имитационных исполняемых моделей и последующего их прогона для анализа. Разработка модели выполняется в графическом редакторе AnyLogic с использованием многочисленных средств поддержки, упрощающих работу. Модель конструируется с помощью встроенного компилятора и запускается на выполнение. В процессе выполнения параметры моделей можно регулировать, тем самым проводить различного рода эксперименты.

AnyLogic разработан на основе новых идей в области информационных технологий, теории параллельных взаимодействующих процессов и гибких гибридных систем. Программный продукт работает на основе объектно-ориентированной концепции. Другой базовой концепцией является представление модели как набора взаимодействующих, параллельно функционирующих, активностей. [4]

AnyLogic является поистине уникальным продуктом, разработанным в России. История данной среды для моделирования берет свое начало в 1990-х годах, тогда в компьютерной науке наблюдался большой интерес к построению описания взаимодействующих параллельных процессов, которое можно трактовать математически. Так группа ученых из Санкт-Петербургского Политехнического университета разработала новое программное обеспечение COVERS, созданное для анализа корректности систем. Анализируемая система определялась графически, с помощью описания ее структуры и поведения отдельных компонент. Данное программное обеспечение использовалось в исследовательских проектах hp, это и вдохновило в 1998 г. организовать уже коммерческую компанию для создания нового программного обеспечения для имитационного моделирования. Данный продукт был выпущен в 2000 г., он был основан на последних преимуществах информационных технологий: объектно-ориентированный подход, язык программирования Java, элементы стандарта UML, современный GUI и т.д. [10]

Продукт получил название AnyLogic, т.к. поддерживал все три подхода моделирования (о них было упомянуто в предыдущем разделе):

- системная динамика;
- дискретно-событийное моделирование;
- агентное моделирование.

На данный момент AnyLogic используют следующие мировые компании: Google, NASA, IBM, intel, McDonald’s, hp, Ford, Toyota, Volvo, BMW, Coca-Cola и многие другие.

Компании признают, что AnyLogic – это продукт, способный оптимизировать процессы и вывести использование ресурсов на недостижимый для большинства уровень. Таким образом, это не только теоретически полезный продукт, но и признанный мировыми компаниями.

ГЛАВА 2
САХАРНАЯ МОДЕЛЬ

2.1 Описание базовой модели Sugarscape

Для данной курсовой работы была выбрана модель Sugarscape (сахарная модель), она является одним из базовых примеров клеточных автоматов.

Sugarscape – это модель искусственного общества. Она описана в книге под названием «Growing Artificial Societies» (Джошуа Эпштейн и Роберт Акстелл, 1996 г.). В данном труде описывается вычислительная модель, в которой неоднородная популяция агентов конкурирует за возобновляемые ресурсы, неравномерно распределенные в двумерной среде. Агенты в модели автономны в том смысле, что ими не управляет какой-либо центральный орган, и они неоднородны в том смысле, что они различаются по своим генетическим атрибутам и исходным условиям окружающей среды (например, их первоначальное местоположение и богатство). Данная модель использовалась для демонстрации миграции, торговли, неравенства ресурсов (богатства), процессов болезней и т.д. В данной курсовой работе мы сосредоточимся на боевых действиях (т.е. на борьбе за ресурс).

Базовая модель Sugarscape, представленная Дж. Эпштейном и Р. Акстеллом, представляет собой сетку 50×50, где каждая клетка содержит определенное количество сахара, также может содержать агента. На каждом шаге агенты ищут соседнюю клетку с наибольшим содержанием сахара, перемещаются туда и съедают весь сахар в той клетке. Они могут оставлять загрязнение, умирать, размножаться, передавать информацию, обмениваться сахаром, передавать болезни – все это зависит от конкретного сценария и переменных, определенных при настройке модели. Так, с помощью этой модели, можно показать реальные социальные отношения.

В данной работе базовые правила будут модифицированы под конкретные социальные отношения, но сейчас более подробно опишем базовые правила поведения модели.

Среда представляет собой сетку 50×50, которая образует тор. Каждая ячейка имеет емкость и уровень сахара. Уровень сахара – это количество единиц сахара в клетке, а емкость сахара – это максимальное значение, которое может принимать уровень сахара в этой клетке. Емкость сахара фиксирована для каждой отдельной ячейки и может быть разной для разных ячеек. Пространственное распределение сахарных емкостей состоит из двух самых насыщенных областей, где емкость равна

4, эти области окружены сахарной долиной, где содержание сахара колеблется от 1 до 3, а вокруг этой долины расположены области совсем без сахара. Распределение сахарных емкостей можно увидеть на рисунке 2.1 ниже.

Рисунок 2.1 Распределение сахарных емкостей.

На каждом шаге модельного времени в каждой ячейке сахар снова растет со скоростью α единиц, такой рост продолжается до достижения емкости сахара в клетке (т.е. если значение емкости клетки 4, то сахара в ней не может стать 5).

Цель агентов в данной модели: бродить по сахарному миру, собирая как можно больше сахара. При этом каждый агент наделен индивидуальными (пожизненными) характеристиками, которые определяют его навыки и возможности для выживания в сахарном мире. Это следующие индивидуальные атрибуты:

- Видение/поле зрения (vision), которое представляет собой максимальное количество ячеек, которое агент может видеть в каждом из четырех основных направлений: север, юг, запад и восток.
- скорость метаболизма (metabolism), этот атрибут описывает количество сахара, сжигаемое агентом на каждом шаге, т.е. употребляемое им на поддержание жизнедеятельности.
- возраст (age) – количество модельных шагов, которые агент существует.

Также агенты имеют возможность накапливать сахарное богатство. Двум агентам запрещено занимать одну и ту же ячейку в сетке.

Правила, определяющие поведение агентов, следующие:

1. На каждом шаге агент осматривают территорию в пределах своего видения (vision), он рассматривает только незанятые ячейки и определяет ячейку с наибольшим содержанием сахара. Далее агент перемещается в эту ячейку (если ячеек с наибольшим количеством сахара несколько, то выбирается любая из них с равной вероятностью). Заняв новую ячейку, агент собирает весь имеющийся в ней сахар, тем самым увеличивая собственное сахарное богатство за счет собранного сахара и уменьшая его за счет метаболизма

(metabolism). Если сахарное богатство агента в этот момент становится не большим нуля, то агент умирает.

2. После голодной смерти агента, его заменяет новый возраста (age) 0, помещенный в случайно выбранную незанятую ячейку, имеющий случайные атрибуты видения (vision), метаболизма (metabolism) и начального богатства. Все случайные числа приведены в таблице 2.1 ниже.

Длина решетки L	50
Скорость роста сахара α	1
Количество агентов	250
Начальное богатство агентов	От 5 до 25 единиц
Метаболизм (metabolism)	От 1 до 4 единиц
Поле зрения агентов (vision)	От 1 до 6 единиц

Таблица 2.1. Параметры модели

Первоначально каждая ячейка модели содержит уровень сахара, равный ее емкости, а также 250 агентов создаются в случайном незанятом начальном месте со случайными (равномерное распределение) атрибутами, представленными в таблице 2.1 выше. Начальное состояние для базовой модели продемонстрировано на рисунке 2.2.

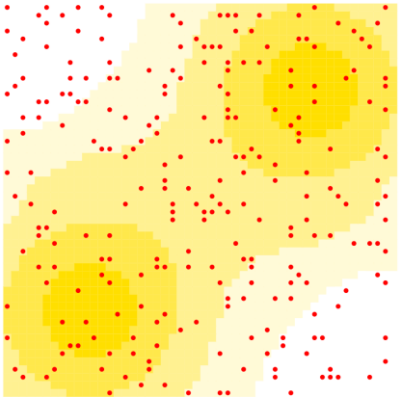

Рисунок 2.2 Начальное состояние базовой модели.

Стоит отметить, что для каждой ячейки сетки определяется множество ячеек, которое является окрестностью данной клетки (так называемое соседство). Например, окрестностью могут являться все ячейки, расположенные выше, ниже, слева, справа и по диагоналям от самой клетки (такой тип носит название Мурово соседство), если соседями считаются те же ячейки, только без диагональных, то соседство называется Евклидовым. Для каждой конкретной задачи определяется

свой тип окрестностей. На рисунке 2.3 представлены два типа соседства, описанных выше.

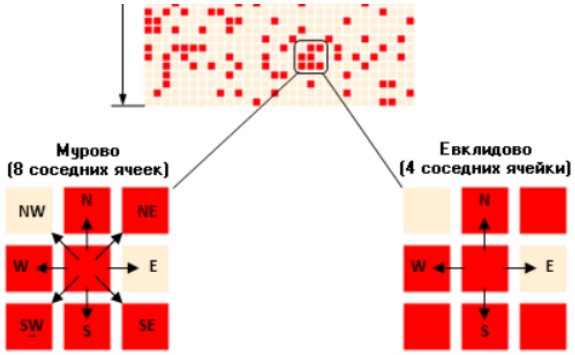

Рисунок 2.3 Типы соседства в дискретном пространстве. [6]

В рамках данной модели будем использовать Евклидово соседство, т.е. соседними клетками будем считать те, которые расположены, либо выше/ниже, либо справа/слева.

2.2 Описание усовершенствованной модели Sugarscape

Как было отмечено в предыдущем подразделе, в рамках курсовой работы хочется показать борьбу за ресурс. Но, действуя по правилом базовой модели, ни о какой борьбе и речи быть не может. Там мирное сосуществование агентов, где каждый сам за себя, каждый для себя собирает сахар и старается выжить. В усовершенствованной модели появятся некоторые новые атрибуты у агентов, а также структура сахарного общества поменяется.

Данная усовершенствованная модель основана на описании,которое приведено в ODD-протоколе для описанных агентных моделей, представленных в [9].

Для начала разобьем всех агентов модели на две враждующие группы: красные (группа А) и синие (группа Б). Данные группы будут соревноваться друг с другом и стараться накопить как можно больше сахара.

Из параметров базовой модели сохраняются следующие: поле зрения (vision), возраст (age), метаболизм (metabolism), количество сахара в клетке и сахарная емкость.

Что касается восстановления сахарной среды, то здесь никаких изменений не будет, т.е. на каждом шаге сахар растет со скоростью α единиц, такой рост продолжается до достижения емкости сахара в клетке.

Теперь опишем изменения в поведении агентов. На каждом шаге моделирования агент теперь будет не просто перемещаться на незанятую ячейку с наибольшим содержанием сахара, но изучать свое окружение, т.е. оценивать количество вражеских и дружественных агентов в поле его зрения. Изучив свое окружение, агент переходит в одно из трех состояний:

- нет контакта (в поле его зрения нет ни одного вражеского агента);
- слабое состояние (вражеских агентов в поле зрения больше чем дружеских, либо столько же);
- сильное состояние (в поле зрения больше друзей, чем врагов).

Далее в зависимости от состояния, в котором находится агент, он будет решать какое действие он будет выполнять.

Существует четыре типа действий: атака (attack), отступление (retreat), прыжок (jump) и действие, при котором агент остается на своем месте (stay). Опишем каждое из этих действий.

Первое действие – это атака (attack). Агент выбирает ближайшего врага (т.е. принадлежащего другой группе) в поле своего зрения, который занимает клетку с наибольшим содержанием сахара (если таких несколько, то выбор происходит с помощью случайного равномерного распределения), после выбора агент перемещается на клетку этого врага, убивает его и собирает весь накопленный сахар врага себе, а также весь сахар клетки.

Следующее действие – это отступление (retreat). Его суть состоит в том, что агент перемещается в любую клетку, где его группа (например, группа А) была инициализирована (где это место, будет описано ниже). Выбирается случайная свободная клетка, если таких несколько, то выбирается любая с помощью равномерного случайного распределения.

Третье действие – это прыжок (jump). Это то, что описано в базовой модели, то есть, агент выбирает ближайшую незанятую ячейку с наибольшим количеством сахара и перемещается туда. Если таких клеток несколько, то, опять же, выбирается любая с помощью равномерного случайного распределения.

Остаться на месте – последнее действие. Агент никуда не перемещается из своей ячейки и питается там сахаром, который восстановится в клетке за счет α.

Опишем начальное расположение агентов на сахарном пейзаже. Оно будет кардинально отличаться от расположения в базовой модели, в данном случае оно не будет случайным, также агентов будет не 250, а 882.

Группа А (красные агенты) первоначально будет расположена в левом нижнем углу (юго-запад), она займет поле размером 21×21, т.е. всего агентов группы А будет 441. Группа Б (синие агенты) займет правый верхний угол (северо-восток),

также поле размером 21×21. Начальное состояние усовершенствованной модели можно увидеть на рисунке 2.4 ниже.

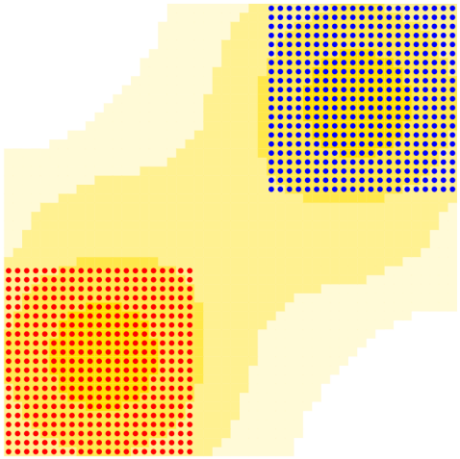

Рисунок 2.4 Начальное состояние усовершенствованной модели.

Агента усовершенствованной модели будут действовать по следующему алгоритму:

1. Определить состояние агента (обращаем внимание, что для данной реализации будет существовать только два состояния: нет контакта и сильное, другие состояния, описанные выше, будут использованы в другой реализации модели).
2. Если состояние сильное (strong; друзей больше, чем врагов), то агент атакует.
3. Если состояние, отличное от сильного, то агент совершает прыжок (как в базовой модели).

Все, что делает агент на каждом шаге приведено на рисунке 2.5.

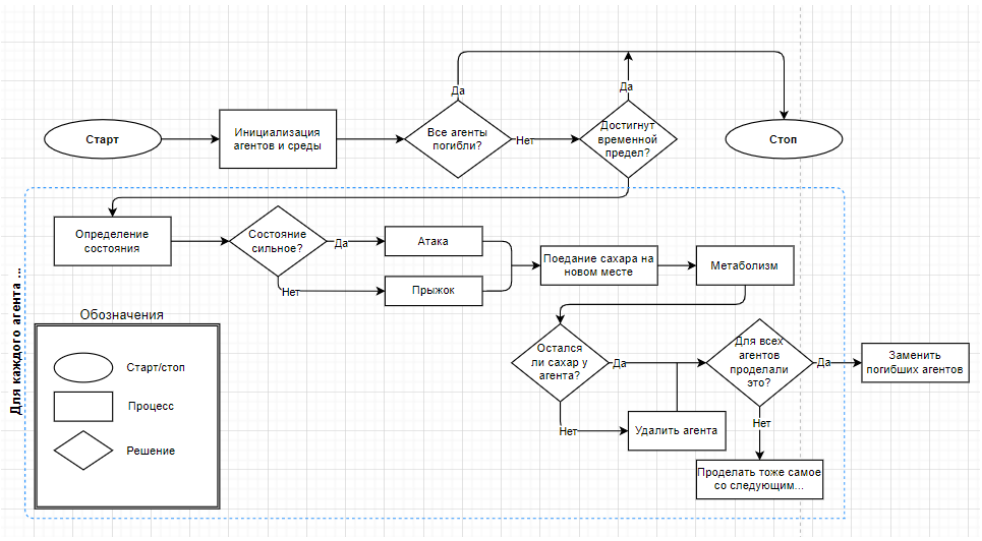

Рисунок 2.5 Схема функционирования усовершенствованной модели.

Отличия двух моделей приведены в таблице 2.2.

Параметр	**Базовая модель**	**Усовершенствованная модель**
Группы (breeds)	Нет	Группы А и Б
Начальное количество агентов	250	882
Состояния	Нет	Нет контактов/слабое/сильное
Действия	Прыжок	Остаться/прыжок/отступление/атака

Таблица 2.2 Отличия базовой модели от усовершенствованной

2.3 Реализация усовершенствованной модели Sugarscape в AnyLogic

Воспользуемся возможностями AnyLogic для построения модели, описанной в предыдущем разделе.

Для начала отметим, что создание агентной модели в AnyLogic заключается в создании агентов двух типов. В данной реализации это тип агента MyField для описания высокоуровневого объекта, где содержатся все агенты, и тип агента Cell для описания агента. Каждый тип будет подтипом базового типа Agent. Число агентов будет включено в тип MyField как дубликаты типа Cell.

Так как данная модель представляет собой, вообще говоря, клеточный автомат, то в рамках данной работы будет использоваться дискретное пространство, т.к. клеточные автоматы представляют собой именно его. А так в AnyLogic есть возможность использовать и непрерывное и ГИС пространства для моделирования.

Дискретное пространство представляет собой прямоугольный массив ячеек, который полностью или частично заполнен агентами, в одной ячейке может находится не более одного агента. Ранее упоминалось о типах соседства, в AnyLogic можно выбрать из двух самых распространенных: Мурово (8 соседних ячеек) или Евклидово (4 соседние ячейки). Здесь это делается просто выбором типа соседства в настройках свойств модели.

Создадим среду, в которой будут обитать агенты, размерностью 50×50.

Итак, есть тип агента, который называется MyField, он будет содержать в себе массив агентов Cell. Тогда Cell – это агент, содержащий параметры, события,

переменные, описывающие поведение агента. В MyField будут прописаны функции, относящиеся в поведению всей популяции.

Опишем создание Cell и MyField.

Создадим нового агента и назовем его Cell. В окно открывшегося графического редактора агента Cell поместим прямоугольник (rectangle) в начало координат, а также круг. Прямоугольник будет отображать состояние сахарной среды (т.е. цветом показывать количество сахара в ячейке), а круг будет отображать есть в данной ячейке агент, принадлежащий какой-либо группе, или нет (красного либо синего цвета). Далее агенту Cell задаются следующие параметры:

1. c – емкость ячейки (т.е. максимально возможное содержание сахара).
2. psugar – уровень сахара в ячейке.
3. age – возраст агента.
4. breed – группа, к которой принадлежит агент (принимает значение 1, если принадлежит группе А, 2 – Б, 0 – ячейка в которой нет агента).
5. metabolism – скорость метаболизма.
6. vision – поле зрения агента (для упрощения реализации установим это значение равным единице).
7. sugar – это накопленное агентом сахарное богатство.
8. state – состояние, в котором находится агент (нет контакта – 0, слабое – 1, сильное – 2).
9. action – действие, которое совершает агент (0 – остаться на месте, 1 – прыжок, 2 – отступление, 3 – атака).
10. fr_num – количество дружелюбных соседей (т.е. принадлежащих той же группе, с одним и тем же значением breed).
11. en_num – количество вражеских соседей.
12. empty – количество незанятых ячеек вокруг клетки.
13. starve_agent_A_die – количество умерших от голода агентов группы А в данной ячейке.
14. starve_agent_B_die – количество умерших от голода агентов группы Б в данной ячейке.
15. combat_agent_A_die – количество погибших в бою агентов группы А в данной ячейке.
16. combat_agent_B_die – количество погибших в бою агентов группы Б в данной ячейке.

Также модель имеет ряд вспомогательных параметров, которые необходимы для реализации усовершенствованной базовой модели.

Еще агенту Cell задаются события, которые изменяют параметры агента в соответствии правилам системы, данные события вызываются на каждом шаге моделирования:

1. get_away – удаляет погибших агентов из среды.
2. eda – реализовывает логику, где агент при перемещении на новую клетку съедает весь сахар.
3. minus_metabolism – агент тратит энергию на свои нужды.
4. get_my_friends – событие, которое подсчитывает вражеских/дружеских, а также незанятых ячеек вокруг агента, далее устанавливает состояние (state) агента. Также данное событие реализовывает логику перемещения на новое место.

Цвет прямоугольника, который отображает количество сахара с клетке, задается в поле «Цвет заливки» выражением:

```
psugar == 0 ? white : (psugar <= 1 ? new Color(255, 223, 0, 40) :
                      (psugar <= 2 ? new Color(255, 223, 0, 110) :
                      (psugar <= 3 ? new Color(255, 223, 0, 180) :
                                     new Color(255, 223, 0)))).
```

Цвет круга, который отображает агент какой группы находится в ячейке, задается следующим выражением:

```
breed == 1 ? red : (breed == 2 ? blue : new Color(255, 255, 255, 0)).
```

Так создана одна ячейка, теперь определим всю среду существования модели, таких ячеек будет 250.

Создадим нового агента MyField. Из окна структуры перетащим элемент класса Cell и зададим количество клеток. В настройках свойства класса MyField выберем тип пространства – дискретное, размерность 50×50 и, соответственно, ширину и высоту – 500 и 500, тип соседства – Евклидово.

Для начального распределения сахара по пейзажу воспользуемся файлом symmetric-sugar-map.txt, загрузим его в AnyLogic в режиме чтения и назовем его SugarMap. В AnyLogic есть возможность прочитать данные из файла, для этого проверим, есть ли что-то в файле, с помощью функции SugarMap.canReadMore(), далее будем читать из файла с помощью функции SugarMap.readDouble(). Так, загрузив данные из файла, зададим начальное состояние.

На каждом шаге моделирования необходимо оценивать окружения данного агента, для этого воспользуемся функцией getAgentNextToMe(CellDirection dir), которая возвращает агента по-соседству в направлении, которое указано в скобках (dir может принимать такие значения: NORTH, SOUTH, EAST, WEST). С помощью этой функции прописали логику, которая будет подсчитывать количество друзей/врагов и пустых клеток, а потом перемещать агента в другую ячейку в зависимости от состояния.

Также объявим переменную tick, которая будет увеличиваться на каждом шаге моделирования на единицу, тем самым отсчитывать модельное время. Всего возможно 20000 шагов.

Далее можно запускать модель. На рисунке 2.6, 2.7, 2.8 и 2.9 можно видеть, как выглядит среда с агентами на определенных шагах моделирования.

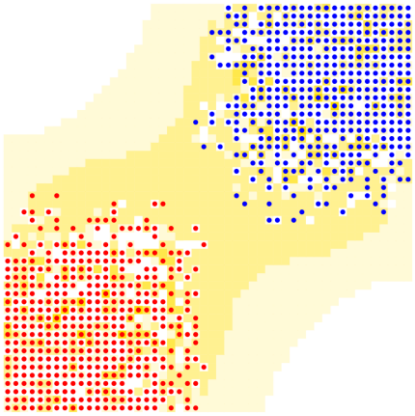

Рисунок 2.6 Состояние модели на 10 шаге.

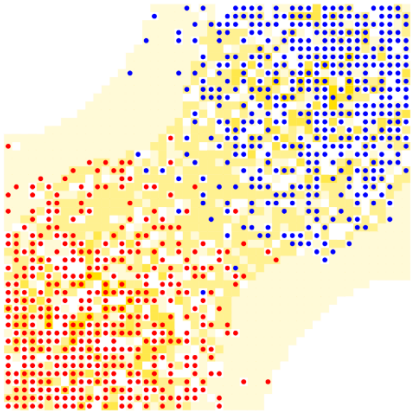

Рисунок 2.7 Состояние модели на 63 шаге.

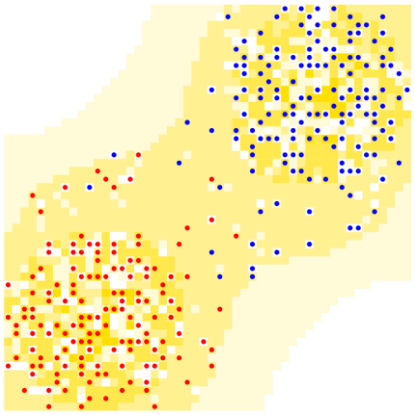

Рисунок 2.8 Состояние модели на 2763 шаге.

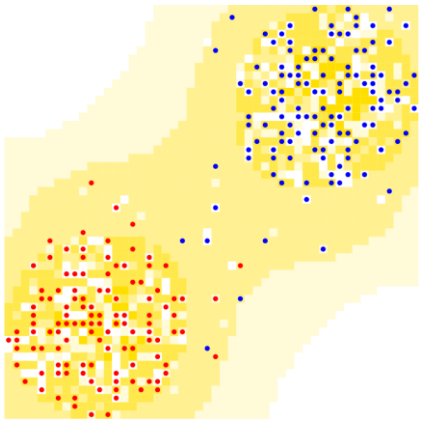

Рисунок 2.9 Состояние модели на 4427 шаге.

Таким образом, с течением времени мы видим, что агенты каждой группы возвращаются на «свою» территорию и обитают там, где больше всего сахара, не нападая на другую группу. Существуют отдельные индивидуумы, которые «ищут лучшей жизни» в зоне противника, но, поскольку основная масса сосредоточивается в месте большого содержания сахара, поход этих отдельных агентов в зону противника не может увенчаться успехом, его быстро одолеет группа, на которую он решил напасть в одиночку.

Введем вспомогательные переменные, которые помогут отслеживать за изменением определенных параметров. Например, цель каждой группы – собрать наибольшее количество сахара. Поэтому введем переменные all_sugar_A и all_sugar_B, в которых будет храниться весь сахар, собранный группой А и Б соответственно. Также введем переменные, которые будут хранить число погибших от голода агентов: starve_A и starve_B; и число погибших в бою: combat_A, combat_B.

На рисунках 2.10, 2.11 и 2.12 изображены графики, на которых отображается изменение накопленного богатства, погибших от голода и погибших в сражении.

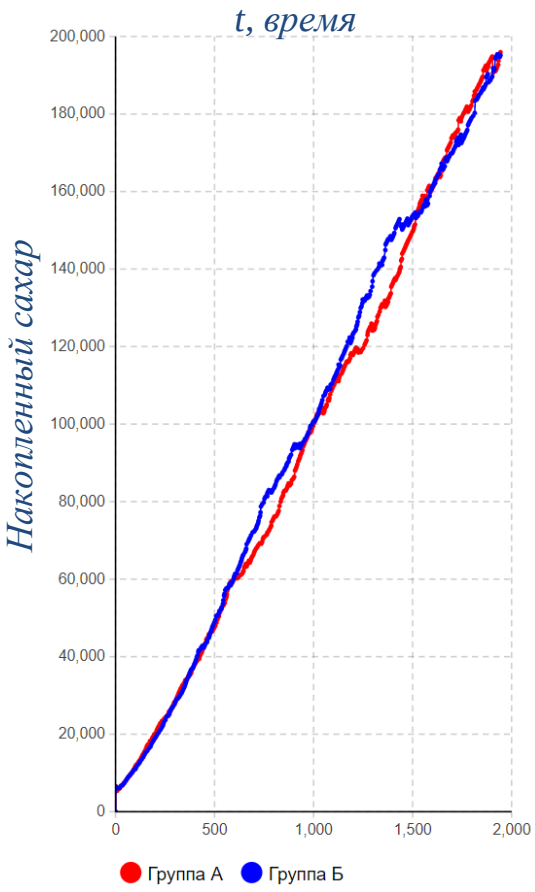

Рисунок 2.10 График изменения накопленного богатства во времени.

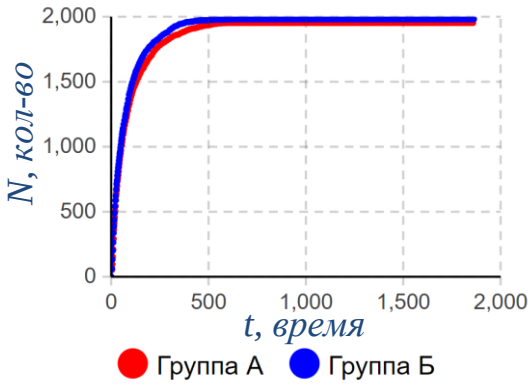

Рисунок 2.11 График количества умерших от голода.

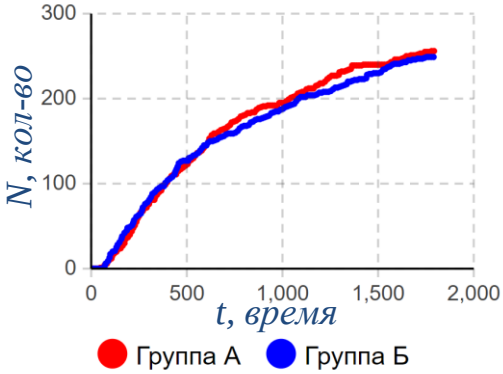

Рисунок 2.12 График количества погибших в бою.

Как видно на графиках, показатели у одной и другой группы почти одинаковые, это все из-за того, что они действуют по одним и тем же правилам. Еще интересно отметить, что в начале моделирования от голода умирает очень много агентов, со временем эта характеристика меняется уже не так сильно. Первая жертва, павшая в бою происходит приблизительно на сороковом шаге моделирования, это тот момент, когда две враждующие группы встречаются.

Введем также переменные num_A и num_B, в которых будет храниться общее число представителей групп А и Б соответственно.

Тогда построим график для среднего показателя накопленного богатства. Он приведен на рисунке 2.13.

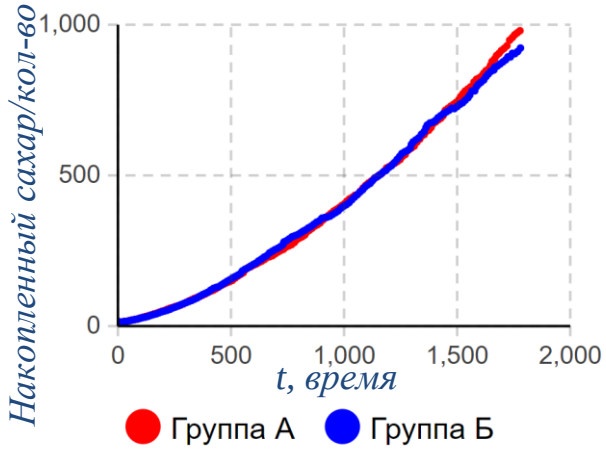

Рисунок 2.13 График среднего показателя накопленного богатства.

Если проецировать на реальный мир, то это очень похоже на поведение социальных групп: кажется, что где-то лучше, чем у тебя, поэтому сначала наблюдается активная атака, со временем же понимается, что и твое место обитания не плохое, что лучше там, где обитают те, кто может тебя защитить, а не враждебно настроенные агенты. Так они возвращаются назад, туда где много сахара, только потеряв часть своей группы в бою за ресурс.

ГЛАВА 3
ОБУЧЕНИЕ С ПОДКРЕПЛЕНИЕМ

3.1 Обучение с подкреплением: общие сведения

Существует много способов обучать модели с помощью машинного обучения. Когда имеется набор размеченных данных, то можно воспользоваться обучением с учителем, чтобы научить модель правильно предсказывать маркировку для каждого образца. Когда нет возможности предложить какую-то обратную связь, можно использовать обучение без учителя, т.е. позволить модели каким-то образом самой принимать решение, что делать. [7]

Иногда мы находимся где-то посередине. Возможно, мы знаем что-то о том, чему хотим научить, но у нас нет четко промаркированных данных. Возможно, все, что мы знаем, это то, как отличить хорошее от плохого. Например, в какой-то ситуации на основании логики такое действие абсолютно неудачное, а другое уже более удачное. Так мы можем сообщить модели, что мы думаем об ее определенных действиях. Более понятный пример описан ниже, он взят из книги «Глубокое обучение без математики» Эндрю Гласснера. Например, мы можем пытаться научить новый тип робота-гуманоида ходить на двух ногах. Но точно не знаем, как он должен двигаться, как ему сохранять баланс, но знаем, что хотим, чтобы он сохранял вертикальное положение и не падал. Если робот пытается ползти на животе или прыгать на одной ноге/руке, то мы говорим ему, что это неправильный способ передвижения. Однако если он стоит на обеих ногах и использует их для передвижения, то можем сказать ему, что он уже на правильном пути, и ему нужно продолжать изучать такие типы поведения.

Стратегия того, что мы определяем как прогресс, и называется обучением с подкреплением. [8]

Общая идея обучения с подкреплением состоит в том, что некий субъект совершает какое-то действие в окружающей среде. Эта среда затем посылает ответную реакцию на действие субъекта, которая говорит, насколько уместными, как мы полагаем, были его действия в его состоянии. После этого мы сообщаем новое состояние окружающей среды субъекту, а он, на основании этих знаний, как-то корректирует модель своего поведения.

Таким образом, путем проб и ошибок субъект может понять, какие действия в данной ситуации лучше, чем другие, и в следующий раз, когда наступит подобная ситуация, выбрать самую выгодную модель поведения.

В рамках данной курсовой работы будет использоваться один из алгоритмов обучения с подкреплением, а именно, Q-обучение (Q-learning). Его суть будет описана в следующем разделе.

3.2 Q-обучение и сахарная модель

А что если применить машинное обучение к сахарной модели? Может быть так агенту смогут накапливать больше сахара? Попробуем применить Q-обучение (Q-learning), где Q – «quality» (качество), – метод, применяемый в машинном обучении.

Субъекты, которые обучаются с помощью Q-обучения, индивидуально, накапливая неявные знания, сохраняют свой опыт в матрице Q-значений (Q-values). Первоначально эти субъекты исследуют мир случайным образом, обновляя свой опыт. С возрастом они все чаще принимают решения исходя из своего собственного опыта. Т.е. по мере того, как субъекты становятся старше, они меньше исследуют и больше следуют своим накопленным знаниям.

Итак, опишем подробнее процесс Q-обучения сахарной модели. Агенты Q-обучения используют обучение с подкреплением описанное в книге [8]. Начнем с построения таблицы, которая измеряет насколько хорошо будет выполнить определенное действие в любом состоянии. В нашем случае данная таблица будет иметь размерность 3×4 (т.к. для сахарной модели определены 3 состояния и 4 действия). Значения, хранящиеся в построенной таблице называются Q-значения (Q-values), а сама таблица называется Q-таблица (Q-table). Для начала инициализируем Q-таблицу случайными значениями (заполним ее нулями, как показано на рисунке 3.1).

	Остаться (Stay)	Прыжок (Jump)	Отступление (Retreat)	Атака (Attack)
Нет контактов	0	0	0	0
Слабое (Weak)	0	0	0	0
Сильное (Strong)	0	0	0	0

Рисунок 3.1 Инициализация Q-таблицы.

Агенты, использующие Q-обучение, принимают решения, совершают действия и обновляют свой опыт, пользуясь следующим алгоритмом:

1. Определить свое текущее состояние (нет контактов, сильное/слабое).
2. Воспользоваться эпсилон-жадным процессом для того, чтобы выбрать действие, которое совершит агент. Что это за процесс, будет описано ниже. Таким образом, на данном шаге будет выбрано действие.
3. Совершить выбранное с предыдущем пункте действие.

4. Определить награду, которую получит агент за совершенное действие, в соответствии с состоянием, в котором он находился. Награды приведены в таблице 3.1. Награда будет зависеть от емкости клетки, в которую перебрался агент, а также от содержания сахара в ней.
5. Определить новое состояние агента.
6. С помощью Q-таблицы найти действие, совершить которое в данной ситуации выгоднее всего (это действие с самым большим Q-значением в таблице при текущем состоянии).
7. Обновить значения в матрице Q-table.
   - $QV_{future} = Reward + QV_{max} \times \gamma$, где QV_{max} – это максимальное Q-значение для данного состояния, $Reward$ – награда, полученная на шаге 4, а γ – гипер-параметр, который позволяет нивелировать вклад последующих наград.
   - Посчитать разницу (ошибку) между QV_{future} и текущим значением Q-value ($QV_{current}$).
     $E = QV_{future} - QV_{current}$, где QV_{future} посчитано в прошлом подпункте, а $QV_{current}$ – Q-значение, соответствующее изначальному состоянию и совершенному действию.
   - Обновить Q-значение.
     $QV_{current} = QV_{current} + \lambda \times E$, где λ – скорость обучения (насколько резко меняем Q-значения), $QV_{current}$ – Q-значение, соответствующее изначальному состоянию и совершенному действию.
8. Установить новое состояние агента.

Награды		**Действие**			
		Остаться	**Прыжок**	**Отступление**	**Атака**
Состояние	**Нет контактов**	psugar	psugar	-100	-100
	Слабое	psugar - c	psugar + c	psugar + 10×c	psugar - 2×c
	Сильное	psugar	psugar + c	psugar - 2×c	psugar + 10×c

Таблица 3.1. Таблица наград (psugar – количество сахара, содержащееся в ячейке, куда перешел агент, c – емкость ячейки).

Опишем эпсилон-жадный процесс, который упоминается во втором пункте алгоритма Q-обучения. Для начала зададим эпсилон (ξ):

$$\xi = \frac{1}{1 + e^{\frac{t-5000}{1000}}}$$

где t – возраст агента.

По этой формуле видно, что $0 < \xi < 1$. Теперь необходимо сгенерировать число $0 \leq m < 1$. Суть ξ-жадного подхода в том, что если число $m > \xi$, то действие выбирается с помощью Q-таблицы. Из таблицы выбираем строку, соответствующую текущему состоянию, из этой строки выбираем самое большое

значение и совершается действие, соответствующее этому самому большому значению.

Если же $m \leq \xi$, то любое действие из четырех возможных выбирается случайно.

Таким образом мы будем обычно делать наиболее многообещающий выбор, хотя иногда выберем одно из других действий и посмотрим, куда оно приведет. Вероятность того, что мы будем выбирать Q-таблицу может быть максимум 95%, т.е. мы в любом случае оставляем минимум 5% для случайного действия.

3.3 Реализация Q-обучения в AnyLogic

Реализуем описанный в прошлом разделе алгоритм. Для этого понадобится ввести новые параметры для агента Cell:

1. rewards – награда за совершенное действие.
2. q_values – матрица, состоящая из Q-значений.
3. epsilon – параметр для эпсилон-жадного процесса.
4. rule_or_Q – параметр, который хранит по какому принципу поступает агент: обучается или действует по правилам.
5. gamma – параметр, который говорит, насколько важны будущие награды.
6. lambda – скорость обучения.

Gamma и lambda – это константы, значение которых установим 0.95.

В агент MyField добавим параметры method_A и method_B, в которых будет храниться закон, по которому поступает группа агентов (т.е. правила или Q-обучение), данные параметры нужны, чтобы с помощью выпадающего списка определить модель поведения группы. Это показано на рисунке 3.2.

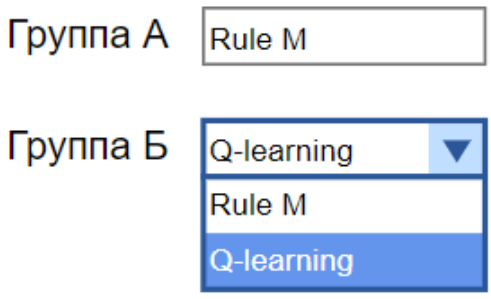

Рисунок 3.2 Выбор правил поведения для групп.

Так как для Q-обучения важно не только текущее состояние, но и будущее, метод для определения соседей кардинально изменился. Теперь понадобится текущего местоположение агента, для этого будем использовать следующие функции:

int getR() – возвращает строку ячейки текущего расположения агента.

int getC() – возвращает столбец ячейки текущего расположения агента.

Agent getAgentAtCell(int r, int c) – возвращает агента, который расположен в строке r и столбце c.

Также пропишем отдельный метод get_state(Agent a), который с помощью вышеупомянутых функций находит состояние агента.

Далее с помощью всего вышеперечисленного реализуем Q-обучение.

На рисунках 3.3, 3.4, 3.5 и 3.6 ниже приведены состояния модели на определенных шагах моделирования, где и группа А, и группа Б обучается по алгоритму Q-learning.

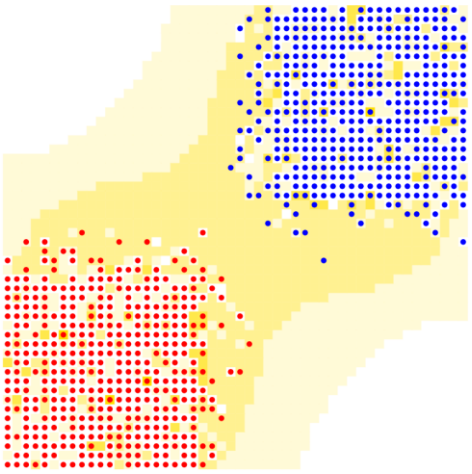

Рисунок 3.3 Состояние модели на 10 шаге.

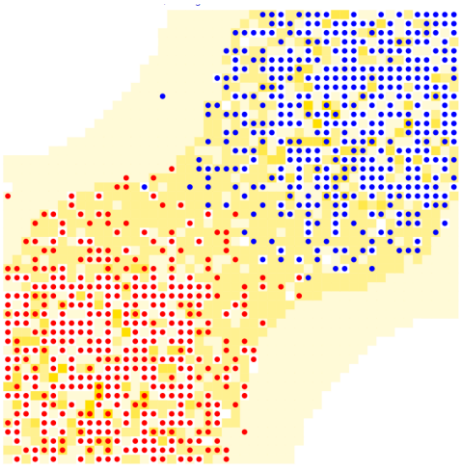

Рисунок 3.4 Состояние модели на 63 шаге.

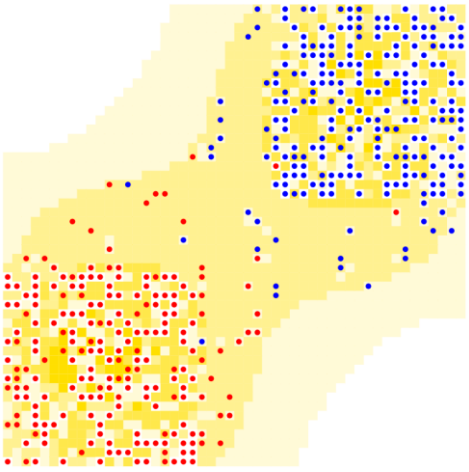

Рисунок 3.5 Состояние модели на 2803 шаге.

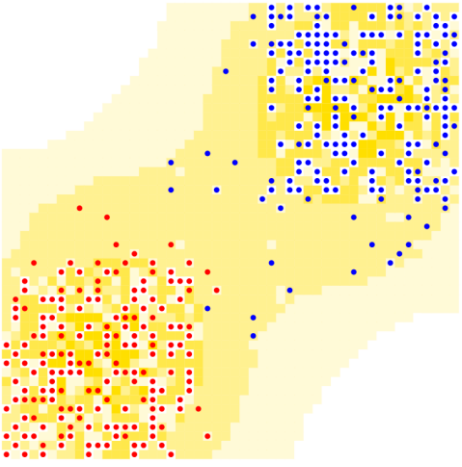

Рисунок 3.6 Состояние модели на 4404 шаге.

Также, проанализировав графики, получим, что первая смерть в бою наступает ближе к 60 шагу, это на 20 шагов позже, чем при действии по правилам. Получается, что обучающиеся агенты чуть более осторожны и дольше идут навстречу друг другу. Здесь наблюдается следующая тенденция: агенты покидают привычное место обитания в поисках сахара, но по истечению времени возвращаются обратно. Изредка кто-то пробует еще раз отправится на вражескую территорию, но большинство сосредоточено на «своей территории».

3.4 Сравнительный анализ поведения обучаемых и необучаемых групп агентов в рамках одной агентной модели

В данном разделе установим группе А одну модель поведения, а группе Б другую. Пусть группа Б выбирает действия на основании правил усовершенствованной модели, а агенты группы А будут использовать Q-обучение.

Тогда построим графики, которые покажут как отличаются различные характеристики у обучаемых и необучаемых групп агентов.

На рисунке 3.6 приведено изменение накопленного сахара во времени. На первых 2000 шагах обучающиеся агенты накапливают значительно больше сахара, но из-за того, как мы выбрали эпсилон, приблизительно до 5000 шага действия выбираются случайным образом, так что не стоит говорить о том, что модель хорошо обучена, пока это случайное поведение.

Что касается других характеристик, то количество голодных смертей в обучаемой группе агентов меньше, т.к. эта характеристика не сильно увеличивается со временем, можно утверждать, что обучаемые агенты ведут себя лучше относительно этой характеристики. Это отображено на рисунке 3.7.

Смертей в бою также у обучаемой группы меньше, значит и по этой характеристике обучаемые агенты выигрывают. Это показано на рисунке 3.8.

Важно показать, что ближе к 6000 шагу моделирования обучаемые агенты проигрывают в накопленном богатстве необучаемым. Это можно наблюдать на рисунке 3.9 и 3.10. Как уже отмечалась, агенты обучаемой группы до 5000 шага еще поступают случайным образом (так задавалось эпсилон), а далее уже поступают осознанно в соответствии с тем, чему они научились. Так вот, получилось, что результат лучше, когда у них была воля поступать случайным образом. По количеству жертв обучаемые модели лучше, т.к. количество жертв меньше, т.к. они более «нерешительные», могут оставаться на своих местах либо отступать, тогда когда у необучаемых агентов возможности отступить нет.

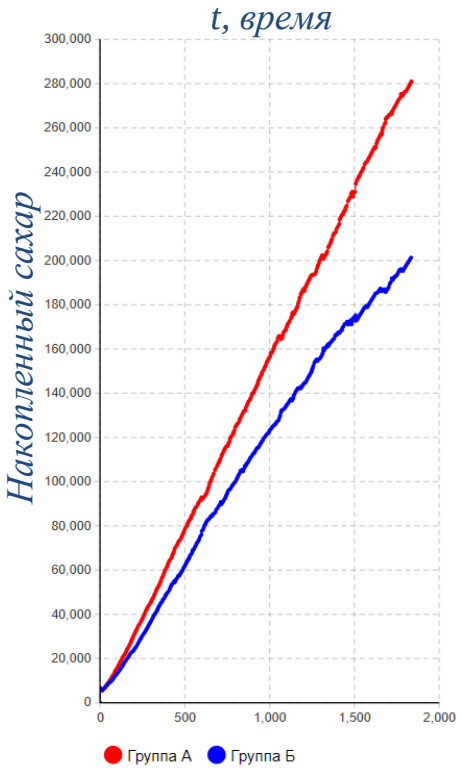

Рисунок 3.6 График изменения накопленного богатства во времени.

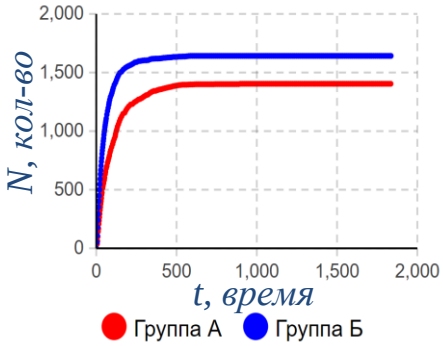

Рисунок 3.7 График количества умерших от голода.

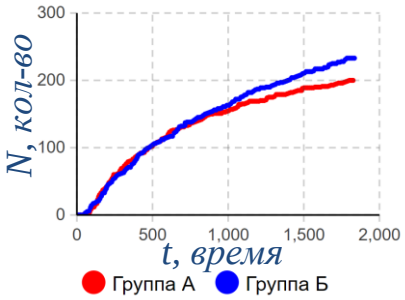

Рисунок 3.8 График количества погибших в бою.

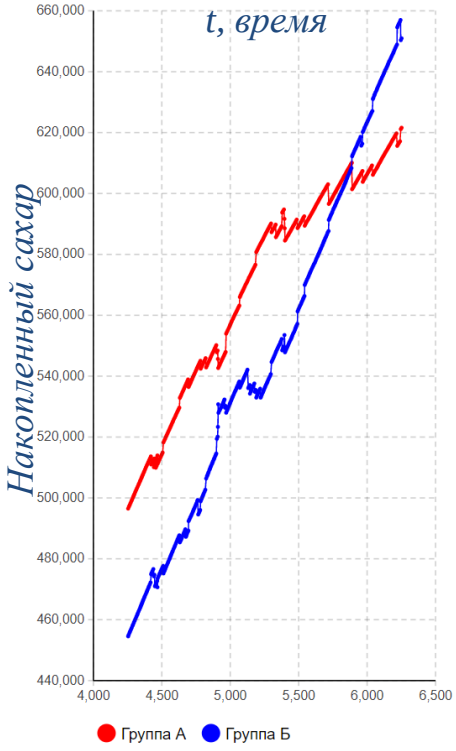

Рисунок 3.9 Изменение накопленного богатства от 4000 до 6500 шага.

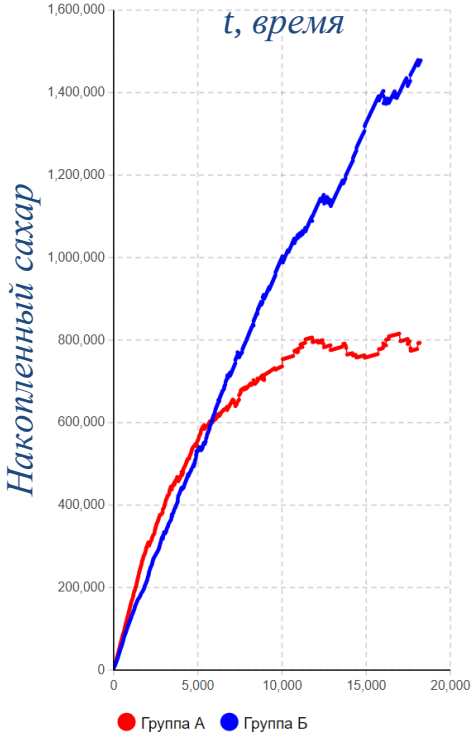

Рисунок 3.10 Изменение накопленного богатства от 0 до 20000 шага.

Получили, что использование обучения с подкреплением в рамках данной модели не принесло желаемого результата: накопление большего количества сахара. Однако, благодаря тому, что AnyLogic очень гибкая среда, писать методы машинного обучения здесь не составило особого труда. Следовательно, для того, чтобы пытаться улучшать агентные модели можно экспериментировать и соединять методы машинного обучения с агентными моделями.

ЗАКЛЮЧЕНИЕ

В данной курсовой работе была изучена, описана и реализована в среде AnyLogic сахарная модель, которая описывает взаимодействие социальных групп.

Далее было введено предположение, что накопить большее количество сахара можно с помощью алгоритмов машинного обучения. К поведению агентов применили Q-обучение. На основе проведенных экспериментальных исследований можно сделать вывод, что использование машинного обучения не всегда приводит в лучшим результатам. В ходе построения сахарной модели, где поведение ее агентов определялось либо элементарными правилами, либо машинным обучением выяснилось, что необученная модель накапливает больше сахара, данный результат соотносится с результатом статьи [9], в которой описывается, что машинное обучение действительно проигрывает базовым правилам.

За последнее десятилетие наблюдается рост интереса к агентному моделирования, также наблюдается повышенный интерес с интеграции машинного обучения и агентного моделирования. Удивительно, конечно, что агенты, действующие по базовым правилам, накопили больше сахара, однако это наталкивает на то, чтобы продолжать исследования в этой области и найти метод, который соберет больше сахара, чем базовые правила.

С учетом результатов, полученных в данной курсовой работе, следует всегда хорошо подумать, прежде чем добавлять методы машинного обучения в агентные модели. Хотя метод, который использовался в этой работе, хорошо зарекомендовал себя в машинном обучении, для такой модели социального поведения нужно разработать что-то другое.

Таким образом, с учетом всего, написанного выше, среда для агентного моделирования AnyLogic достаточно дружелюбна для реализации в ней машинного обучения, поэтому все, кто исследует агентные модели обязаны экспериментировать с добавлением обучения в модели, ведь это в конечном итоге может привести к лучшим результатам.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Григорьев И. AnyLogic за три дня. Практическое пособие по имитационному моделированию, Интернет издание, 2017.
2. Киселев И. Н. Агентное моделирование сердечно-сосудистых систем человека, 2012.
3. Куприяшкин А. Г. Основы моделирования систем: учеб. пособие //Норильск: НИИ. – 2015.
4. Федотова В. С. Технологии имитационного моделирования в системе AnyLogic //Царскосельские чтения. – 2013.
5. Epstein J., Axtell R. Growing Artificial Societies: Social Science From the Bottom Up. Washington, D.C.: MIT Press / Brookings Institution, 1996.
6. Справочная система AnyLogic. [Электронный ресурс] – Режим доступа: https://help.anylogic.ru/index.jsp.
7. Глубокое обучение без математики. Т. 2: Практика / пер. с англ. В. А. Яроцкого. – М.: ДМК Пресс, 2020. – 610 с.
8. Richard S. Sutton and Andrew G. Baro. Reinforcement Learning: An Introduction. A Bradford Book, NIT Press, 2017.
9. Brearcliffe, D., Crooks, A., 2020, Creating Intelligent Agents: Combining Agent-Based Modeling with MachineLearning, EasyChair Preprint.
10. Википедия – свободная энциклопедия [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/AnyLogic.

ПРИЛОЖЕНИЕ А.

Код программы

https://github.com/KarinaKey/MyCourseWork.git