Документ подписан простой электронной подписью
Информация о владельце:
ФИО: Минцаев Магомед Шавалович
Должность: Ректор
Дата подписания: 23.11.2023 14:58:02
Уникальный программный ключ:
236bcc35c296f119d6aafdc[illegible]

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

**«УФИМСКИЙ ГОСУДАРСТВЕННЫЙ НЕФТЯНОЙ
ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

М.Г. Баширов

МУЛЬТИАГЕНТНЫЕ СИСТЕМЫ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

Учебно-методическое пособие по выполнению практических и лабораторных работ

Уфа

Издательство УГНТУ

2021

Рецензенты:
Вильданов Р.Г., докт. техн. наук, профессор

Чурагулов Д.Г., старший преподаватель

Методическое пособие по выполнению практических и лабораторных работ по дисциплине " Мультиагентные системы искусственного интеллекта" для студентов направления подготовки 13.04.02 Электроэнергетика и электротехника: методические указания / УГНТУ; сост. М.Г. Баширов. - Уфа: УГНТУ, 2021. - 1,4 Мб. - Текст: электронный

© ФГБОУ ВО «Уфимский государственный нефтяной технический университет, 2021
© Баширов М.Г., 2021

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ…….……4

1 ПРАКТИЧЕСКИЕ ЗАНЯТИЯ……5

1.1 Инструментальные средства Agent Builder, BeeGent, JADE. Назначение агентных платформ и средств, предоставляемых разработчику агентных систем ……....5

1.2 Архитектура агентной платформы JADE. Контейнеры и платформы ……..……...7

1.3 Агенты AMS и DF …….8

1.4 Среда JADE для управления агентным Приложением……..9

1.5 Пример построения мультиагентной системы ……..11

1.6 Изучение методов мультиагентной оптимизации ……...12

1.7 Проектирование онтологии для взаимодействия агентов …….19

2 ЛАБОРАТОРНОЕ ЗАНЯТИЕ. Проектирование и реализации коммуникации агентов, изучение и программирование различных типов поведения агентов……22

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ……....26

ВВЕДЕНИЕ

В настоящее время наблюдается развитие агентных систем, возникших в результате технической эволюции информационных и программно-аппаратных средств современной инфосферы. Для эффективного использования агентных приложений необходимо освоить методологию и инструменты их проектирования и эксплуатации.

Целями преподавания дисциплины «Мультиагентные системы» являются изучение методов, моделей, средств и технологий компьютерной обработки информации и автоматизированного управления на основе теории искусственных агентов и мультиагентных систем, получение базовых знаний в области проектирования и программирования мультиагентных систем.

Учебно-методическое пособие предназначено для использования при проведении практических и лабораторных занятий по дисциплине «Мультиагентные системы искусственного интеллекта» магистерской программы «Интеллектуальные средства и системы управления, защиты и диагностики электроэнергетических комплексов», направление подготовки (специальность): 13.04.02 Электроэнергетика и электротехника.

1 ПРАКТИЧЕСКИЕ ЗАНЯТИЯ

1.1 Инструментальные средства Agent Builder, BeeGent, JADE. Назначение агентных платформ и средств, предоставляемых разработчику агентных систем

Агент – это программный объект, способный воспринимать ситуацию,принимать решения и коммуницировать с подобными себе объектами,динамически устанавливая с ними связи. Под *мультиагентной системой* (МАС) будем понимать множество программных агентов, организованных водно или несколько сообществ, и предназначенных для решения определенной задачи. Средой разработки и существования МАС являются *агентные платформы*. Было разработано множество программных реализаций агентных платформ, каждая из которых имеет свои особенности, достоинства и недостатки. Вот лишь небольшой список из более чем ста доступных платформ, публикуемых на сайте организации *AgentLink* (*European Coordination Action for Agent-based Computing*): JADE, FIPA-OS, AOS, ZEUS, KADOMA, NOMADS, ARA, AGLETS, GRASSHOPPER, TRACY, AJANTA, LEAP, JACK, SEMOA. Многие из них успешно существуют в виде коммерческих проектов (таких как JACK) или проектов, позиционируемых как проекты с открытым исходным кодом (JADE, ZEUS и др.). В 90-х годах возникла необходимость создания единых стандартов на разработку агентных систем. В этот период были основаны две организации *MASIF* (*Mobile Agent System Interoperability Facility*) и *FIPA* (*Foundation of Physical Intelligent Agents*). В результате их работы появились стандарт MASIF и стандарт FIPA, дающие рекомендации по созданию систем мобильных агентов и систем интеллектуальных агентов. Одной из наиболее популярных агентных платформ в настоящее время является платформа *JADE (Java Agent DEvelopment Framework)*. Проект JADE разрабатывается компанией Telecom Italia Lab с 2000 г. Агентная платформа

JADE является типичным *Middleware*, т.е. программным обеспечением (ПО) среднего уровня, представляющим собой набор средств для создания и управления системой с множеством агентов.

Платформа разработки мультиагентных систем JADE включает в себя динамическую среду, где могут «жить» JADE агенты; библиотеку классов, которую программисты могут использовать для разработки собственных агентов; набор графических инструментов, позволяющих управлять активностью запущенных агентов. JADE предоставляет программисту – разработчику агентных систем следующий набор средств:

- *FIPA-compliant Agent Platform* – агентную платформу, основанную на стандарте FIPA и включающую обязательные типы системных агентов, которые автоматически активируются при запуске платформы.

- *Distributed Agent Platform* – распределенную агентную платформу, которая может использовать несколько компьютеров (узлов), причем на каждом узле запускается только одна Java Virtual Machine. Агенты исполняются как Java-потоки. Для доставки сообщений между агентами, в зависимости от их местонахождения, используется соответствующий транспортный механизм – *Multiple Domains support* – множество основанных на FIPA-спецификациях специализированных агентов, которые могут объединяться в федерацию, реализуя таким образом мультидоменную агентную среду.

- *Multithreaded execution environment with two-level scheduling*. Каждый JADE-агент имеет собственный поток управления, но он также способен работать в многопотоковом режиме. *Java Virtual Machine* проводит планирование задач, исполняемых агентами или одним из них.

- *Object-oriented programming environment*. Большинство концепций, свойственных FIPA-спецификации, представляются Java-классами, формирующими интерфейс пользователя. ▪ *Library of interaction protocols*. Использование стандартных интерактивных протоколов *fipa-request* и *fipa-contract-net*. Для того чтобы создать агента, который мог бы действовать согласно таким протоколам, разработчикам прикладных программ нужно только

имплементировать специфические доменные действия, в то время как вся независимая от прикладной программы протокольная логика будет осуществляться системой JADE.

- *Administration GUI*. Простые операции управления платформой могут исполняться через графический интерфейс, отображающий активных агентов и контейнеры агентов. Используя GUI, администраторы платформы могут создавать, уничтожать, прерывать и возобновлять действия агентов, создавать иерархии доменов и мультиагентные федерации. Платформа JADE написана на языке программирования Java с использованием Java RMI, Java CORBA IDL, Java Serialization и Java Reflection API. Она упрощает разработку мультиагентных систем благодаря использованию FIPA-спецификаций и инструментов (tools), которые поддерживают фазы исправления ошибок (debugging) и развертывания (deployment) системы. Эта агентная платформа может распространяться среди компьютеров с разными операционными системами, и ее можно конфигурировать через удаленный GUI-интерфейс. Процесс конфигурирования этой платформы достаточно гибкий. Единственным требованием такой системы является установка на компьютере Java Run Time требуемой версии.

1.2 Архитектура агентной платформы JADE. Контейнеры и платформы

Платформа JADE является распределенной и представляет собой набор *контейнеров*. *Контейнером* называется динамическая среда исполнения мультиагентных приложений, в которой находятся агенты. Каждый контейнер может содержать несколько агентов. Набор активных контейнеров называется *платформой*. Один из контейнеров всегда является главным (*Main container*), все остальные контейнеры связываются с ним и регистрируются в момент запуска. Поэтому первым контейнером при старте платформы должен быть главный, а все остальные контейнеры должны быть «обыкновенными» (т.е.

неглавными) контейнерами и должны заранее «знать», как найти главный контейнер, на котором они будут регистрироваться, т.е. должны иметь данные о хосте и порте. Другой главный контейнер, запущенный где-либо в сети, представляет собой другую платформу, на которой могут зарегистрироваться новые обычные контейнеры. Концепция, показывающая две JADE-платформы, состоящие из трех и одного контейнера соответственно. JADE-агенты определяются с уникальными именами. При условии, что они знают имена других агентов, они могут общаться, независимо от их фактического местонахождения: в общем контейнере (агенты A2 и A3), в разных контейнерах на одной платформе (агенты A1 и A2), или вообще на разных платформах (A4 и A5). Пользователю не обязательно знать, как работает динамическая среда JADE, но необходимо запускать её перед началом выполнения своих агентов.

1.3 Агенты AMS и DF

Кроме возможности приёма регистраций от других контейнеров, главный контейнер отличается от обычного контейнера тем, что содержит два специальных агента, автоматически запускаемых одновременно с контейнером:

- *AMS* (*Agent Management System* – система управления агентами) обеспечивает службу управления агентами, которая позволяет создавать и удалять агентов, а также содержит в себе пространство имен агентов. Имя агента является уникальным и имеет следующий формат: <nickname>@<platform-name>. Зная имена друг друга, агенты могут обмениваться сообщениями как внутри контейнера и платформы, так и между различными платформами.

- *DF* (*Directory Facilitator* – менеджер директорий) представляет собой службу «желтых страниц» (*yellow pages*), где агенты могут публиковать информацию о предоставляемых ими сервисах. С помощью DF агент может находить агентов, предоставляющих необходимые ему сервисы, и вступать с ними в переговоры. Внутри одной платформы может существовать несколько

DF, предоставляющих информацию о различных группах сервисов или о сервисах различных групп агентов. Использование DF описано в Сервис «желтых страниц».

1.4 Среда JADE для управления агентным приложением

JADE предоставляет набор инструментов, выполняющих управление агентным приложением и его отладку.

Запуск среды JADE. Для того чтобы запустить среду JADE, необходимо:

- установить на компьютер J2EE не ниже версии 1.5;

- установить на компьютер среду разработки Java-приложений IntelliJ IDEA;

- установить на компьютер среду разработки JADE доступной версии, например, версии 3.6.1. Для этого необходимо скопировать\загрузить файлы среды JADE на свой компьютер, например, в папку C:\Jade;

- изменить конфигурационный параметр CLASSPATH следующим образом: в режиме командной строки ввести prompt> set JADE_HOME=c:\jade prompt> set CLASSPATH=%JADE_HOME%\lib\jade.jar; %JADE_HOME%\lib\jadeTools.jar; %JADE_HOME%\lib\http.jar; %JADE_HOME%\lib\iiop.jar; %JADE_HOME%\lib\commons-codec\commons-codec-1.3.jar; %JADE_HOME%\classes;

- для запуска *Main Container* (с графическим интерфейсом) необходимо ввести prompt> java jade.Boot –gui. Т.к. переменную CLASSPATH необходимо устанавливать каждый раз при перезапуске окна с командной строкой, целесообразно создать файл RunJade.bat, содержащий приведенные выше команды. Достаточно запустить файл RunJade.bat, чтоб открыть графический интерфейс RMA. prompt> set JADE_HOME=c:\jade prompt> set CLASSPATH=%JADE_HOME%\lib\jade.jar; %JADE_HOME%\lib\jadeTools.jar; %JADE_HOME%\lib\http.jar; %JADE_HOME%\lib\iiop.jar; %JADE_HOME%\lib\commons-codec\commons-codec-1.3.jar; %JADE_HOME%\classes java jade.Boot –gui;

- скопировать на компьютер папку \demo. Эта папка содержит приложения *HelloWorldAgent, Order&Resource*.

Remote management agent. *Remote management agent* представляет собой графическую консоль для управления мультиагентным приложением. Позволяет создавать новые контейнеры, управлять агентами, создавать сообщения и запускать средства отладки.

Dummy agent. *Dummy agent* является графической утилитой, которая позволяет посылать и получать сообщения от имени определенного агента, а также сохранять и загружать очередь его сообщений (отправленных и полученных).

Sniffer agent. *Sniffer agent* – это графическая утилита для просмотра потока сообщений между избранными агентами. Представляет обмен сообщениями в виде диаграмм последовательностей. Позволяет сохранять/загружать поток сообщений между агентами.

Introspector agent. *Introspector agent* – графическая утилита для просмотра внутреннего состояния агента. Позволяет контролировать жизненный цикл агента, просматривать очередь его сообщений, активные и выполненные поведения, а также запускать исполнение агента с задержками между операциями или по шагам. При этом «шагом» поведения агента считается исполнение метода *action()*, а не команда в коде языка Java.

Log manager agent. *Log Manager agent* – графическая утилита для отображения лога сообщений в процеесе работы агентного приложения.

DF GUI. *DF GUI* – графическая утилита для визуализации желтых страниц. Позволяет регистрировать и удалять сервисы агентов, а также осуществлять поиск сервисов.

1.5 Пример построения мультиагентной системы

Рассмотрим примеры простых агентных приложений в среде JADE.

Агент HelloWorldAgent. Создадим и запустим агента HelloWorldAgent. Он является расширением типа Agent в JADE. Ниже приведен код класса агента HelloWorldAgent.java.

Этот класс необходимо скомпилировать в jar-файл с таким же названием. Соответствующий файл находится в папке \demo\ HelloWorldAgent. /* **Код класса HelloWorldAgent.java** */ import jade.core.Agent; public class HelloWorldAgent extends Agent { protected void setup() { System.out.println("Hello World! My name is "+getLocalName()); } }

Созданый агент необходимо откомпилировать следующим образом:

```
javac –classpath <JADE-classes> HelloWorldAgent.java.
```

Для того чтобы запустить откомпилированный агент на выполнение, необходимо запустить среду JADE и указать имя агента:

```
java –classpath <JADE-classes>;. jade.Boot Peter:HelloWorldAgent.
```

Для создания агентного приложения используется стандартный инструмент, входящий в состав JADE, – *Remote Agent Management GUI (RAM)*. Прежде всего, необходимо выбрать контейнер с агентами (на рисунке по умолчанию выбран главный контейнер *Main-Container*).

Перед созданием агентов обязательно надо указать контейнер, иначе агенты не будут созданы.

Создадим экземпляр агента типа *HelloWorldAgent* с именем *Peter*. Для этого выполним команды *Actions -> Start New Agent*. Необходимо указать имя экземпляра агента (*Peter*), а также тип создаваемого агента, который должен совпадать с именем класса агента (*HelloWorldAgent*). После нажатия кнопки <OK> агент создается и регистрируется в системе, затем выполняются действия, описанные в методе *Setup()* агента. В данном случае агент выводит сообщение со своим именем на экран и остается в системе.

Результат действий агента отображается в окне командной строки. После выполнения действий агент остаетсяся в системе в активном состоянии: он присутствует в списке агентов контейнера *Main-Container*.

Для того чтобы удалить агента из системы (а также из списка агентов контейнера), в код класса агента необходимо добавить вызов метода *doDelete()*:

public class HelloWorldAgent extends Agent { protected void setup() { System.out.println("Hello World! My name is "+ getLocalName()); // Make this agent terminate doDelete() } }.

Контрольные вопросы

1. Какие функции выполняет агентная платформа JADE?

2. Что такое платформа и контейнер?

3. Для чего предназначены агенты AMS и DF?

4. Какие инструменты предоставляет JADE для управления агентными приложениями? Каковы их функции?

5. Какова последовательность работы с агентом в среде JADE?

6. Какие поля включает сообщение агенту согласно международному стандарту FIPA взаимодействия агентов?

7. Какие типы сообщений, пересылаемых между агентами, Вам известны?

8. Как работают методы посылки/получения сообщений?

9. Как реализовать ожидание сообщения агентом?

10. Как реализовать выбор сообщения с указанными характеристиками из очереди сообщений агента?

11. Как реализовать сложные переговоры между агентами?

12. Какие функции выполняет сервис «желтых страниц»?

13. Как опубликовать сервис?

14. Как найти необходимый для агента сервис?

1.6 Изучение методов мультиагентной оптимизации

Теоретические сведения. Одним из основополагающих методов решения таких задач является метод ветвей и границ, но в случае модели с большим количеством переменных метод неприменим, т.к. потребует значительного количества временных затрат.

В данной практической работе будем использовать методы мультиагентной оптимизации (Multi-agent optimization system или Ant System или AS). AS алгоритмы – алгоритмы поиска, основанные на эвристике и применяемые для решения многих комбинаторных задач оптимизации.

Модель и структура алгоритма основана на кооперативном поведении реальных насекомых – муравьев. Муравьи решают проблемы поиска путей с помощью химической регуляции. Каждый муравей оставляет за собой на земле дорожку особых веществ – феромонов. Другой муравей, почуяв след на земле, устремляется по нему. Чем больше по одному пути прошло муравьев – тем явнее след, а чем явнее след – тем большее «желание» пойти в ту же сторону возникает у муравьев. Поскольку муравьи, нашедшие самый короткий путь к «кормушке», тратят меньше времени на путь туда и обратно, их след быстро становится самым заметным. Он привлекает большее число муравьев, и круг замыкается. Остальные пути – менее используемые – потихоньку пропадают.

Алгоритм решения задачи. Предположим, что окружающая среда для муравьев представляет собой полный неориентированный граф. Каждое ребро имеет вес, который обозначается как расстояние между двумя вершинами, соединенными им. Граф двунаправленный, поэтому муравей может путешествовать по грани в любом направлении (рисунок 1).

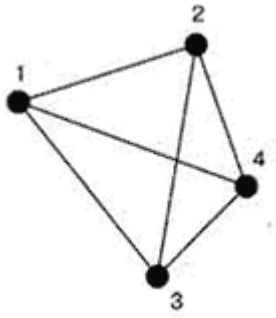

Рисунок 1 – Пример графа

Вероятность включения ребра в маршрут отдельного муравья пропорциональна количеству феромонов на этом ребре, а количество откладываемого феромона пропорционально длине маршрута. Чем короче

маршрут, тем больше феромона будет отложено на его ребрах, следовательно, большее количество муравьев будет включать его в синтез собственных маршрутов. Моделирование такого подхода, использующего только положительную обратную связь, приводит к преждевременной сходимости – большинство муравьев двигается по локально-оптимальному маршруту. Избежать этого можно, моделируя отрицательно обратную связь в виде испарения феромона. Причем, если феромон испаряется быстро, то это приводит к потере памяти колонии и забыванию хороших решений, с другой стороны, большое время испарений может привести к получению устойчивого локального оптимального решения.

Муравей – это программный агент, который является членом большой колонии и используется для решения какой-либо проблемы. Муравей снабжается набором простых правил, которые позволяют ему выбирать путь в графе. Он поддерживает список узлов, которые он уже посетил. Таким образом, муравей должен проходить через каждый узел только один раз. Путь между двумя узлами графа, по которому муравей посетил каждый узел только один раз, называется путем Гамильтона.

Узлы в списке «текущего путешествия» располагаются в том порядке, в котором муравей посещал их. Позже список используется для определения протяженности пути между узлами. Настоящий муравей во время перемещения по пути будет оставлять за собой феромоны. В алгоритме муравья агент оставляет феромоны на ребрах графа после завершения путешествия.

Стартовая точка, куда помещается муравей, зависит от ограничений, накладываемых условиями задачи, так как для каждой задачи способ размещения муравьев является определяющим. Либо все они помещаются в одну точку, либо в разные с повторениями, либо без повторений.

На этом же этапе задается начальный уровень феромона. Он инициализируется небольшим положительным числом для того, чтобы на начальном шаге вероятности перехода в следующую вершину не были нулевыми.

Движение муравья основывается на одном и очень простом вероятностном уравнении. Если муравей еще не закончил путь, то есть не посетил все узлы сети, для определения следующего ребра пути используется уравнение:

$$P = \frac{\tau(r,u)^{\alpha}}{\sum_k \tau(r,u)^{\alpha}\eta(r,u)^{\beta}}$$

Здесь $\tau(r,u)$ – интенсивность фермента на ребре между узлами r и u, $\eta(r,u)$ – функция, которая представляет измерение обратного расстояния для грани, α – вес фермента, а β – коэффициент эвристики. Параметры α и β определяют относительную значимость двух параметров, а также их влияние на уравнение. Так как муравей путешествует только по узлам, которые еще не были посещены (как указано списком табу), вероятность рассчитывается только для ребер, которые ведут к еще не посещенным узлам. Переменная k представляет эти ребра.

Пройденный муравьем путь отображается, когда муравей посетит все узлы графа. Циклы запрещены, поскольку в алгоритм включен список табу. После завершения длина пути может быть подсчитана – она равна сумме длин всех ребер, по которым путешествовал муравей. Уравнение показывает количество феромона, который был оставлен на каждом ребре пути для муравья k. Переменная Q является константой.

$$\Delta\tau_{i,j}^{k}(t) = \frac{Q}{L^{k}(t)}$$

Результат уравнения является средством измерения пути, - короткий путь характеризуется высокой концентрацией феромонов, а более длинный путь – более низкой. Затем полученный результат используется в уравнении, чтобы увеличить количество феромона вдоль каждого ребра пройденного муравьем пути:

$$\tau_{i,j}(t) = \Delta\tau_{ij}(t) + (\tau_{ij}^{k}(t)\rho.$$

Важно, что данное уравнение применяется ко всему пути, при этом каждое ребро помечается феромоном пропорционально длине пути. Поэтому следует дождаться, пока муравей закончит путешествие и только потом обновить уровни

феромона, в противном случае истинная длина пути останется неизвестной. Константа p – значение между 0 и 1.

В начале пути у каждого ребра есть шанс быть выбранным. Чтобы постепенно удалить ребра, которые входят в худшие пути графа, ко всем ребрам применяется процедура испарения феромона. Используя константу p из последнего уравнения, мы получаем уравнение:

$$\tau_{i,j}(t) = \Delta\tau_{ij}(t)(1-\rho).$$

Поэтому для испарения феромона используется обратный коэффициент обновления пути.

После того, как путь муравья завершен, ребра обновлены в соответствии с длиной пути, и произошло испарение феромона на всех ребрах, алгоритм запускается повторно. Список табу очищается, и длина пути обнуляется. Муравьям разрешается перемещаться по графу, основывая выбор ребра на соответствующем уравнении. Этот процесс может выполняться для постоянного количества путей или до момента, когда на протяжении нескольких запусков не было отмечено повторных изменений. Затем определяется лучший путь, который и является решением.

Выбор исходных данных. В качестве графа необходимо взять конкретный полигон (использовать карту железных дорог РФ). Необходимо найти кратчайший путь от места зарождения грузопотока до места перевалки, то есть морского терминала. Учитываем расстояние (стоимость перевозки, время) между узлами (станциями), таким образом, ненаправленный граф будет состоять из конечного множества вершин – станций и ребер-расстояний.

Реализация алгоритма.

1. Задать входную матрицу, расчет коэффициентов видимости, каждой дуге присвоить начальное количество феромона, равное минимальной константе. Задать количество прогонов. Установить счетчик прогонов равным 1.

2. Установить табу-список каждого агента.

3. В момент времени t=0 агенты помещаются в первую и конечную вершины графа.

4.первый элемент табу-списка каждого агента назначается в соответствии с его местом в графе первым и конечным узлом.

5.Для каждого агента рассчитываются значения вероятностей переходов в следующую вершину по формуле.

6. Случайным образом для каждого агента с учетом полученных вероятностей выбирается следующая вершина.

7. В табу-список k-го агента помещается выбранная вершина. Агент перемещается в выбранную вершину.

8. После прохождения из начальной точки в конечную всех агентов, необходимо уточнить значения «следа феромона» для всех ребер.

9. Увеличить значение счетчика прогонов.

10. Если счетчик прогонов меньше максимального, перейти к пункту 3.

11. Определить лучший путь и узлы, входящие в него.

Пример расчета. Разберем функционирование рассмотренного выше алгоритма на простом примере, чтобы увидеть, как работают уравнения. Возьмем простой сценарий с двумя агентами и с двумя ребрами между двумя узлами (V_0 и V_1). Каждое ребро инициализируется и имеет одинаковые шансы на то, чтобы быть выбранным.

Два агента находятся в узле V_0 помечаются как A_0 и A_1. Так как вероятность выбора любого пути одинакова, в этом цикле мы проигнорируем уравнение выбора пути. Данные для задачи:

- число пройденных шагов: для A_0 – 20, для A_1 – 10;
- уровень феромона (Q/пройденное расстояние): для A_0 – 0.5, A_1 – 1.0;
- $\rho = 0.5$;
- $\alpha = 0.3$;
- $\beta = 1.0$.

Каждый агент выбирает свой путь A_0 идет по верхнему пути, а A_1 – по нижнему). A_0 сделал 20 шагов, а A_1, - только 10. По уравнению рассчитываем количество феромонов, которое должно быть «нанесено».

Примечание: Работу алгоритма можно изменить, переопределив его параметры (например, α, β или p), например, придав больший вес феромонам или расстоянию между узлами.

Далее по уравнению рассчитывается количество феромона, которое будет применено. Для A_0 результат составляет: 0,1+(0,5*0,6)=0,4 A_1 результат составляет: 0,1+(1,0*0,6)=0,7. Далее с помощью уравнения определяется, какая часть феромонов испарится и, соответственно, сколько останется. Результаты (для каждого пути соответственно) составляют:

0,4*(1,0-0,6)=0,16

0,7*(1,0-0,6)=0,28

Эти значения представляют новое количество феромонов для каждого пути (верхнего и нижнего, соответственно). Теперь переместим агентов обратно в узел V_0 воспользуемся вероятностным уравнением выбора пути, чтобы определить, какой путь должны выбрать агенты. Вероятность того, что агент выберет верхний путь (представленный количеством феромона 0,16), составляет:

$$\frac{(0{,}16)^{3{,}0} * (0{,}5)^{1{,}0}}{(0{,}16)^{3{,}0} * (0{,}5)^{1{,}0}} + (0{,}28)^{3{,}0} * 1{,}0^{1{,}0} = \frac{0{,}002048}{0{,}024} = \mathrm{P}(0{,}085)$$

Вероятность того, что агент выберет нижний путь (представленный количеством феромона 0,28) составляет:

$$\frac{(0{,}28)^{3{,}0} * (1{,}0)^{1{,}0}}{(0{,}16)^{3{,}0} * (0{,}5)^{1{,}0}} + (0{,}28)^{3{,}0} * 1{,}0^{1{,}0} = \frac{0{,}021952}{0{,}024} = \mathrm{P}(0{,}915)$$

При сопоставлении двух вероятностей оба агента выберут нижний путь, который является наиболее оптимальным.

Задания к выполнению практической работы.

1. Изучите особенности AS-алгоритма.

2. Выполните представленную последовательность действий.

3. Сформируйте отчет по заданию преподавателя.

Контрольные вопросы

1. Поясните AS-алгоритм.
2. Чем отличаются искусственные агенты от реальных муравьев?
3. Что понимается под «следом феромона»?
4. Что понимается под коэффициентом видимости?
5. Охарактеризуйте параметры α, β и ρ.
6. Поясните ваш выбор среды моделирования.

1.7 Проектирование онтологии для взаимодействия агентов

Онтологией называют схему, состоящую из классов связанных между собой посредством различных отношений и правил. Это своеобразная форма представления некоторой области знаний в формальном виде. В настоящее время онтологии широко используются в программировании, обучении, различного рода исследованиях.

Допустим необходимо разработать онтологию, выполняющую задачу классификации. Воспользуемся программо Protege Для примера возьмем простейшую классификацию электронагревателей. В рамках предметной области можно выделим несколько основных классов, а именно:

Электронагреватель – главный класс, содержащий 3 класса-наследника – малой мощности, средней мощности и большой мощности;

Владелец – класс, содержащий информацию о человеке-владельце;

Регион – класс, содержащий информацию о месте обитания владельца.

Далее можно переходить к созданию проекта. Первым шагом запускаем программу Protege и создаем новый проект. В окне настроек выбираем Protege Files. На экране появляется рабочее окно, в котором нам и предстоит работать. Первым делом при создании онтологии необходимо создать классы. Все спроектированные нами классы будут отображаться в окне Class Browser. Для

создания нового можно щелкнуть на иконку «Create Class» или правой кнопкой мыши на поле браузера классов с указанием действия создания класса. После создания класса, можно переименовать его в поле Name. Для создания подкласса щелкаем по классу-родителю, нажимаем правую кнопку мыши и выбираем команду "Create Subclass”. При желании класс можно сделать абстрактным, выбрав соответствующий пункт в выпадающем списке Role. После создания классов необходимо прописать в них поля – свойства. К примеру, у класса Регион будет свойство «Имя», которое будет содержать название региона, в котором проживают владельцы электронагревателей. Для добавления свойства в класс необходимо щелкнуть правой кнопкой мыши в окне «Template Slots» и указать команду «Create Slot». При создании слота ему можно задать название, тип, значение по умолчанию, временное значение, описание и т.п. Стоит отметить, что в качестве типа слота может выступать объект другого класса. Таким способом в программе Protégé устанавливается взаимосвязь между 2 классами. Если ранее какой-либо слот, например, «имя» уже создавался, то его можно просто добавить в класс (при условии, что он подходит), нажав на кнопку в виде прямоугольника с плюсом в правом верхнем углу окна Template Slots. После создания слотов и классов можно приступать к созданию экземпляров или Instances. Для этого сверху щелкните на одноименную вкладку и посередине увидите окно под названием Instance Browser. Для того, чтобы создать экземпляр какого-либо класса, щелкните в окне Class Browser на нужный класс, а затем в окне Instance Browser нажмите на иконку добавления сущности Create Instance. После нажатия в окне Instance Browser появится строчка с вновь созданной сущностью, а справа в окне Instance Editor поля, соответствующие слотам класса, который необходимо заполнить. Чтобы в окне Instance Browser отображать объекты по какому-либо признаку, щелкните на треугольник справа на панели и выберите свойство, которое отображать в браузере. В данном примере укажем свойство «Имя». После создания сущностей можно приступать к формированию запросов. Формы для них находятся во вкладке запросы. Чтобы создать необходимый запрос, нужно выбрать класс, в котором будет производиться

поиск, свойство, по которому будет производится поиск, а также указать признак. Под признаком может пониматься как строчка, так и условие is, is not, contains, begins with и так далее. Если запрос необходимо сделать составным, то есть содержащим 2 и более условий, в окне следует нажать кнопку more и ввести данные в соответствующие поля. Созданный запрос можно сохранить и оставить в библиотеке, введя внизу в поле имя и нажав кнопку «Add to Query Library».

Контрольные вопросы

1. Что такое онтология для искусственного интеллекта?
2. Какие уровни онтологий известны?
3. Какие программные среды для построения онтологий известны?
4. Что такое дескриптивные логики?
5. В чем назначение OWL?

2 ЛАБОРАТОРНОЕ ЗАНЯТИЕ

Лабораторная работа. Проектирование и реализации коммуникации агентов, изучение и программирование различных типов поведения агентов

Цель работы: ознакомление с приложениями теории агентов и многоагентных систем.

Основные теоретические положения. Агент — лицо или организация. наделенное юридическими полномочиями представлять другое лицо или организацию и вести их дела. Из этого определения понятна важная роль агентов в современных науках как промежуточного звена между субъектом и объектом (рис. 1). Если двигаться от субъектного полюса, то агент может пониматься как «квазисубъект», способный в некотором смысле замещать другого субъекта (своего «хозяина»), имеющий определенные обязательства перед ним и действующий по его поручению. Напротив, при движении от объектного полюса агентом считается «активный объект» или метаобъект, способный манипулировать другими объектами, а в более широком смысле, формировать собственные программы действий, которые вызваны некоторыми потребностями и направлены на достижение определенных целей. Отсюда становятся понятными основные пути построения искусственных агентов и их базовые характеристики. По сути, понятие «агента» по-новому иллюстрирует знаменитое изречение В. Гейзенберга о том, что разрыв между субъектом и объектом невозможен.

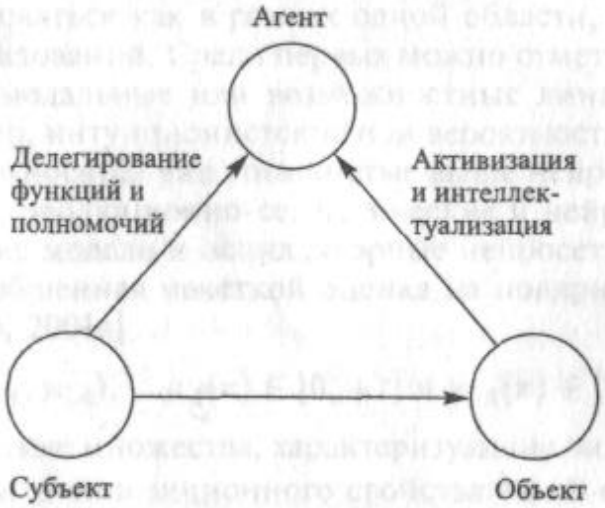

Рис. 1. Агент – промежуточное звено между субъектом и объектом

Любой агент представляет собой открытую систему, помещенную в некоторую среду, причем эта система обладает собственным поведением, удовлетворяющим некоторым экстремальным принципам. Таким образом, агент считается способным воспринимать информацию из внешней среды с ограниченным разрешением, обрабатывать ее на основе собственных ресурсов, взаимодействовать с другими агентами и действовать на среду в течение некоторого времени, преследуя свои собственные цели.

Это значит, что при построении искусственного агента минимальный набор базовых характеристик включает такие свойства как:

а) активность, способность к организации и реализации действий;

б) реактивность, способность воспринимать состояние среды;

в) автономность, относительная независимость от окружающей среды или наличие некоторой «свободы воли», обусловливающей собственное поведение, которое должно иметь хорошее ресурсное обеспечение;

г) общительность, вытекающая из необходимости решать свои задачи совместно с другими агентами и обеспечиваемая развитыми протоколами коммуникации;

д) целенаправленность, предполагающая наличие собственных источников мотивации, а в более широком плане, особых интенциональных характеристик.

Необходимыми условиями реализации искусственным агентом некоторого поведения выступают специальные устройства, непосредственно воспринимающие воздействия внешней среды (рецепторы) и исполнительные органы, воздействующие на среду (эффекторы), а также процессор — блок переработки информации и память. Под памятью здесь понимается способность агента хранить информацию о своем состоянии и состоянии среды. Таким образом, исходное представление о простейшем агенте сводится к модели «организм-среда» (рис. 2).

Рецепторы образуют систему восприятия агента, обеспечивая прием и первичную обработку информации, которая поступает к нему из среды (как внешней, так и внутренней), а затем отправляется в память. Система восприятия может контролировать действия путем определения различий между текущими и ожидаемыми состояниями. В памяти агента должны иметься сведения о типовых реакциях на информационные сигналы от рецепторов, а также информация о состоянии эффекторов и о располагаемых ресурсах. Кроме того, в памяти должны храниться программы переработки входной информации в управляющие сигналы, подаваемые на эффекторы, и обязательно результаты реакций на ту или иную внешнюю ситуацию.

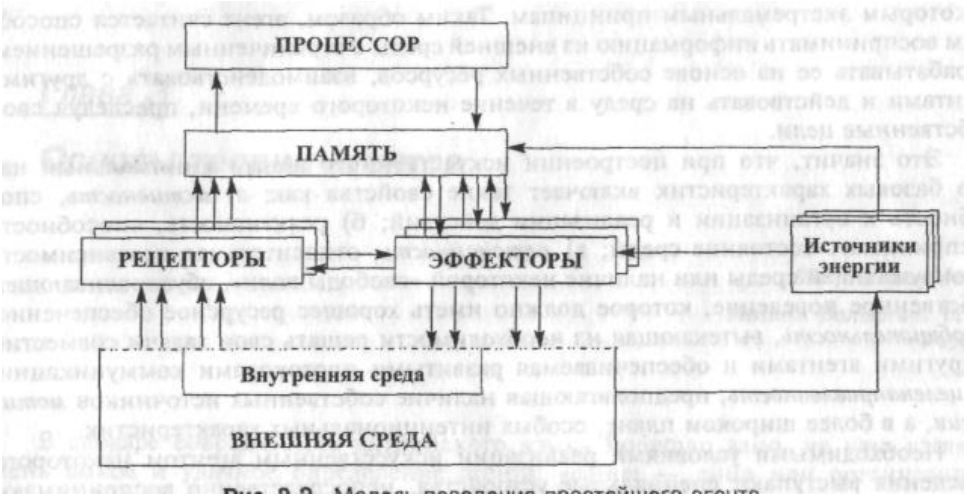

Рис.2. Модель поведения простейшего агента

Блок памяти обычно включает три основных компонента: систему фильтров, обеспечивающих выделение наиболее значимой для агента информации, а также внутреннюю модель внешнего мира и модель самого агента. В конечном счете, именно объем памяти, количество и разнообразие хранимых в ней знаний и программ, степень развития внутренней модели внешнего мира и возможности рефлексии определяют сложность и характер поведения агента, уровень его автономности и интеллектуальности.

Процессор (система процессоров) обеспечивает объединение и переработку разнородных данных, выработку соответствующих реакций на информацию о состоянии среды, принятие решений о выполнении тех или иных действий. Выбор соответствующих действий при заданных ограничениях — одна из ключевых способностей любых агентов.

Функция эффекторов состоит в воздействиях на среду, например, в перемещении объектов внешней среды, выдаче информации в символьной форме, поддержании равновесия внутренней среды (т. е. желаемого состояния самого агента) и т. д.

Источники ресурсов, например, энергопитания, обеспечивают все необходимые условия для поддержания (и, при необходимости, воспроизведения) жизненного цикла агента.

Задание и методические указания к выполнению лабораторной работы

Задание на лабораторную работу.

Разработать модель функционирования агента в предметной области, описанной в конкретном задании. Представить работу модели в виде алгоритма, готового к реализации в виде программы.

Вариант задания выдается на группу из двух человек. При защите лабораторной работы каждый из них должен отчитаться по выполненной работе. Выполненные задания оцениваются по следующим критериям:

Умение автора модели объяснить работу своей части алгоритма.

Степень сложности модели и ее функциональность.

Умение автора объяснить приложение теории многоагентных систем к выбранной предметной области.

По каждому критерию выставляется оценка по пятибальной шкале. Общая оценка за выполненное задание является суммой оценок по каждому критерию. Оценка за выполненное задание влияет на сдачу экзамена по дисциплине. Работа считается не сданной, если оценка за один из трех критериев составляет ниже трех баллов.

Форма отчетности. Отчет по лабораторной работе, который включает в себя следующие разделы:

постановка задачи,

краткое описание предметной области,

описание разработанной модели,

алгоритм работы модели.

Отчет сдается преподавателю в печатном и электронном виде.

Устная беседа по выполненной работе, в ходе которой задаются практические вопросы по алгоритму работы модели и теоретические вопросы потеме лабораторной работы.

3.3.3. Методические указания к выполнению работы. Для выполнения работы необходимо:

Собрать материал и изучить предметную область, описанную в задании.

На основе теоретического материала и предыдущих лабораторных работ по курсу «Многоагентные системы» разработать модель функционирования агента в предметной области.

Описать алгоритм работы модели.

Составить отчет о проделанной работе в текстовом редакторе Microsoft Word.

Контрольные вопросы

Дать определение следующим понятиям:

Агент.

Искусственный агент (ИА).

Характеристики ИА:

Активность.

Реактивность.

Автономность.

Общительность.

Целенаправленность.

Эффектор ИА.

Рецептор ИА.

Блок памяти ИА.

Процессор ИА.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Методическое пособие по дисциплине «Многоагентные системы» и указания к выполнению лабораторных работ Специальность 230102 «Автоматизированные системы обработки информации и управления» /Составители: Тарасов В.Б., Новиков С.О. – М: Изд-во МАТИ, 2006.

2 Мезенцев, К. Н. Мультиагентное моделирование в среде NetLogo: учебное пособие / К. Н. Мезенцев. — Санкт-Петербург : Лань, 2021. — 176 с. — ISBN 978-5-8114-1933-3. — Текст: электронный // Лань: электронно-библиотечная система. — URL: https://e.lanbook.com/book/168871.

3 Андрейчиков, А. В. Интеллектуальные цифровые технологии концептуального проектирования инженерных решений: учебник / А.В.

Андрейчиков, О.Н. Андрейчикова. — Москва: ИНФРА-М, 2021. — 511 с. - ISBN 978-5-16-014884-7. - Текст: электронный. - URL: https://znanium.com/catalog/product/1241808.

4 Использование платформы JADE для разработки мультиагентных приложений: [Электронный ресурс]: метод. указания / Е.В. Симонова. – Самара: Изд-во Самарского университета, 2017. – 62 с.

5 Мультиагентные системы. Учебно-методическое издание. Составитель канд. техн. наук Е. М. Борчик. Могилев: Изд-во Белорусско-Российского университета, 2020. – 44 с.

6 Основы теории агентов и мультиагентных систем // lms2.sseu.ru.