Федеральный исследовательский центр «Информатика и управление»

Российской академии наук

На правах рукописи

Горященко Алексей Сергеевич

Оптимизация задач маршрутизации на основе взаимодействующих интеллектуальных транспортных агентов

2.3.1 Системный анализ, управление и обработка информации, статистика

Диссертация на соискание ученой степени кандидата технических наук

Научный руководитель:

д.т.н., профессор В.М.Хачумов

Москва, 2022

Содержание

Введение

Актуальность темы исследования. Агентное моделирование широко применяется для изучения свойств сложных динамических систем. Интеллектуальный агент – программная сущность, обладающая свойствами автономности, общественного поведения, реактивности и про-активности. В настоящей работе используется термин интеллектуальный транспортный агент – это интеллектуальный агент, применяемый для решения задач маршрутизации, дополнительно обладающий возможностью перемещения и атрибутами скорости и грузоподъемности.

Под задачами маршрутизации в настоящей работе понимаются транспортная задача (задача Монжа-Канторовича, transportation problem, TP) и мультитранспортный вариант задачи маршрутизации (задача Данцига-Рамсера, split delivery vehicle routing problem, SDVRP), особенности решения которых в предлагаемых постановках заключаются в том, что в первом случае используются коалиции транспортных агентов, а во втором – взаимодействующие агенты, не образующие коалиции. В дальнейшем эти задачи будут называться классическими.

Коалиции агентов – набор или группа агентов, возможно обладающих различными свойствами, действующие сообща для достижения поставленной общей цели, в условиях, когда индивидуальные агенты этой цели достичь не могут.

Решение задач формирования групп или коалиций для совместного достижения целей изучалось на примере автоматов в работах М.Л.Цетлина, В.Л.Стефанюка, В.И.Варшавского. Многоагентные системы и их применение анализируются в работах В.И.Городецкого и П.О.Скобелева. Подходы к формированию коалиций агентов и роботов рассматриваются в работах А.В.Смирнова. Изучение совместного поведения агентов и интеллектуальных роботов продолжено в работах В.Б.Тарасова, В.Э.Карпова, И.А.Каляева, А.А.Кулинича. Теория активных систем, элементы которых обладают собственными интересами и предпочтениями, рассмотрена в работах В.Н.Буркова и Д.А.Новикова. Различные подходы к решению оптимизационных задач, в том числе задач маршрутизации, представлены в работах В.М.Курейчика. Различные

варианты задач маршрутизации изучаются в работах C.Archetti, J.-F.Cordeau, J.Desrosiers, G.Laporte, A.Mingozzi, Е.М.Бронштейна.

Представляют значительный практический интерес задачи оптимизации маршрутизации, в которых состав элементов и/или их свойства могут изменяться в процессе решения. В этом случае использование ранее предложенных подходов, основанных на точных переборных алгоритмах, становится очень ресурсоемким и поэтому нецелесообразным. Это определяет внимание к развитию распределенных приближенных алгоритмов.

В России проф. Г.С.Осиповым был предложен подход к разработке мультиагентных интеллектуальных систем, в котором используется идея улучшения «способностей» агентов на основе оценки результатов их деятельности и последующего изменения оценки успешности.

Широкое внедрение в практику распределенных приближенных алгоритмов предполагает изучение близости получаемых результатов к результатам оптимальных алгоритмов. Настоящая работа посвящена изучению возможностей интеллектуальных агентов, обладающих механизмом оценки успешности, в рамках мультиагентой системы как способу повышения качества решений задач маршрутизации, в том числе транспортной задачи и мультитранспортного варианта задачи маршрутизации (SDVRP).

Цель и задачи исследования. Целью работы является повышение качества решения задач маршрутизации большой размерности на основе использования коалиций интеллектуальных транспортных агентов с изменяющимися в процессе деятельности характеристиками и оценками успешности действий.

Достижение поставленной цели предполагает решение следующих **задач**:

1. Формирование на основе анализа научных публикаций критериев выбора мультиагентной платформы и ее нагрузочное тестирование.

2. Разработка и реализация централизованного и распределенного алгоритмов формирования коалиций агентов с возможностью изменения их характеристик в процессе моделирования, изучение его свойств.

3. Разработка распределенного алгоритма решения задач маршрутизации на

основе коалиций агентов, в том числе обладающих способностью к изменению оценок успешности, в мультиагентной системе и его программная реализация.

4. Проведение серии вычислительных экспериментов с целью получения сравнительных характеристик качества разработанных алгоритмов.

Работа соответствует следующим пунктам паспорта специальности 2.3.1 «Системный анализ, управление и обработка информации, статистика»:

п.4. «Разработка методов и алгоритмов решения задач системного анализа, оптимизации, управления, принятия решений и обработки информации»;

п.5. «Разработка специального математического и алгоритмического обеспечения систем анализа, оптимизации, управления, принятия решений и обработки информации»;

п.10. «Методы и алгоритмы интеллектуальной поддержки при принятии управленческих решений в технических системах».

Научная новизна заключается в разработке и исследовании метода решения задач маршрутизации на основе интеллектуальных транспортных агентов, обладающих способностью к изменению оценок успешности, в условиях, когда каждый агент не обладает достаточными ресурсами для выполнения заданий, и характеристики агентов в процессе решения задачи могут меняться. Метод включает несколько новых алгоритмов и оценки их вычислительной сложности, в том числе:

1. Централизованный и распределенный алгоритмы формирования коалиций агентов с учетом случайного изменения их характеристик, существенно расширяющие класс решаемых задач маршрутизации, в которых состав элементов и их свойства могут изменяться в процессе решения;

2. Распределенный алгоритм решения задач маршрутизации в рамках мультиагентных систем, в которых агенты используют оценки успешности для улучшения своих способностей, что позволяет уменьшать их количество при прочих равных условиях;

3. Теоретическую и экспериментальную оценки вычислительной сложности созданных алгоритмов, из которых следует применимость разработанного метода

для решения задач большой размерности и в условиях изменения характеристик агентов.

Положения, выносимые на защиту:

1. Коалиционный метод решения задач маршрутизации в рамках агентного подхода в условиях случайного изменения характеристик и состава агентов, что существенно расширяет класс решаемых задач.

2. Алгоритмы формирования коалиций и решения задач маршрутизации в рамках мультиагентных систем, в которых агенты используют оценки успешности для улучшения своих способностей в результате деятельности.

3. Сравнительные оценки вычислительной сложности алгоритмов формирования коалиций агентов; оценки близости результатов решения задач маршрутизации к оптимальным в случае, когда оптимальные решения возможны.

4. Результаты применения программной реализации предложенного метода для решения модельных задач маршрутизации в условиях изменения характеристик и состава агентов.

Практическая значимость заключается в том, что применение предложенного подхода, основанного на коллективном поведении агентов, делает возможным практическое решение задач маршрутизации большой размерности в условиях изменения параметров задачи в процессе нахождения решения. Программные реализации разработанных алгоритмов оформлены в виде свидетельства о регистрации программы для ЭВМ «Программный комплекс для решения распределительных задач с использованием коалиций интеллектуальных агентов», регистрационный N 2021617131 от 11.05.2021. Программный комплекс позволяет проводить прототипирование и отладку алгоритмов поведения автономных интеллектуальных технических систем.

Методы исследования. В работе использованы методы системного анализа, линейного программирования, методы оценки вычислительной сложности алгоритмов, методы машинного обучения, агентное моделирование как метод имитационного моделирования и теория интеллектуальных динамических систем.

Достоверность полученных в диссертационной работе результатов

подтверждается вычислительными экспериментами, успешной апробацией на научных конференциях и семинарах, а также реализацией предложенных алгоритмов с использованием мультиагентной системы и их применением для решения прикладных задач.

Апробация результатов исследования. Основные результаты работы докладывались диссертантом и обсуждались на 5 всероссийских и международных научно-технических конференциях, в том числе:

- Intelligent Systems Conference (IntelliSys) 2021 (2-3 сентября 2021, Amsterdam, The Netherlands).

- Future of Information and Communication Conference FICC-2019 (14-15 марта 2019, San Francisco, USA);

- Пятой Всероссийской научной конференции молодых ученых с международным участием «Информатика, Управление и Системный Анализ» ИУСА-2018 (6-8 июня 2018, г.Ростов-на-Дону, Россия);

- 7-й Международной конференции "Системный анализ и информационные технологии" САИТ-2017 (13-18 июня 2017, г. Светлогорск, Россия);

- IV Всероссийской научной конференции молодых ученых с международным участием ИУСА-2016 (8-11 июня 2016, г.Тверь, Россия);

- научных семинарах ФИЦ ИУ РАН, РУДН, ИПС им. А.К.Айламазяна РАН.

Внедрение. Разработанные алгоритмы внедрены в учебный процесс кафедры информационных технологий факультета ФМиЕН РУДН, программные реализации используются в деятельности ООО «РИТЕХ», получены соответствующие Акты.

Публикации. Основные результаты по теме диссертации изложены в 8 печатных работах, 4 из которых опубликованы в рецензируемых журналах из списка ВАК или индексируемых в базах данных Scopus, 3 – в материалах всероссийских и международных конференций и 1 – свидетельство о регистрации программы для ЭВМ.

Объем и структура работы. Диссертация состоит из введения, четырех глав, заключения и списка литературы. Текст работы изложен на 138 страницах,

содержит 17 рисунков и 10 таблиц. Список литературы содержит 121 наименование.

Во **введении** обосновывается актуальность применения агентов и их коалиций в мультиагентных системах для решения задач маршрутизации, указывается цель работы, отмечается новизна результатов исследования, их практическая значимость, приводятся данные о структуре и объеме работы.

Первая глава посвящена рассмотрению вопросов развития интеллектуальных программных систем – от систем, основанных на правилах, до мультиагентных. Описана структура одной из типовых современных мультиагентных платформ, рассмотрены ее основные свойства, области применения, классы задач, которые можно решать с ее помощью. На основе анализа научных публикаций сформулированы критерии, которым должна отвечать используемая в настоящей работе мультиагентная платформа: соответствие спецификации *FIPA*, возможность работы с несколькими сотнями агентов, зависимость времени работы от количества агентов не хуже, чем линейная.

Описаны основные свойства известных к настоящему времени алгоритмов формирования коалиций агентов, основанных на использовании характеристической функции: точных переборных или приближенных. Преимущество приближенных алгоритмов по сравнению с точными заключается в возможности работы с сотнями и тысячами агентов, например, в задачах балансировки «умных электрических сетей».

Кратко рассмотрен представленный в научных публикациях знаковый подход к планированию и распределению ролей агентов. Обсуждены возможности алгоритма выбора предпочитаемых действий в коалиции интеллектуальных агентов. Алгоритм состоит в том, что наиболее предпочтительный агент-исполнитель выбирается на основе интегральной оценки успешности выполнения им предыдущих действий.

Приведены примеры использования агентного моделирования, в том числе коалиций агентов, для решения различных задач.

На основе рассмотренных научных публикаций сделан вывод о том, что подход к решению задач маршрутизации с использованием агентного моделирования является актуальным. При этом особый интерес представляют приближенные распределенные алгоритмы, позволяющие в полной мере использовать преимущества мультиагентных систем и современных коммуникационных сетей. Показано, что недостаточно изучены возможности интеллектуальных агентов, в том числе со способностью к изменению оценок успешности, в рамках агентного моделирования. Применение таких агентов позволит повысить качество решения, получаемого приближенным алгоритмом.

Во **второй главе** рассмотрена модельная задача маршрутизации. В зависимости от количества агентов – источников (пунктов отправления), пунктов назначения и соотношения потребностей пунктов назначения, а также грузоподъемности источников имеют место либо классическая транспортная задача, либо мультитранспортный вариант задачи маршрутизации (SDVRP)

Описаны предложенные алгоритмы: формирования коалиций с использованием характеристической функции; распределения шагов заранее созданного плана по исполнителям, которые способны изменять оценки успешности; решения целевой задачи маршрутизации на основе формирования коалиций интеллектуальных агентов. Алгоритмы предложены в централизованном и распределенном вариантах, изучены их свойства.

Третья глава содержит описание структуры агентов, цикла изменения оценки успешности, форматов и типов сообщений и программной реализации предложенных алгоритмов на языке программирования *Python*. Подробно рассматриваются используемые агенты, их взаимодействие между собой. При реализации каждый объект (источник и пункт назначения) представляется одним агентом. Агенты получают информацию друг от друга и от среды путем передачи и

получения сообщений. В состав программной реализации входят агенты-исполнители и управляющий агент.

Четвертая глава содержит описание и результаты экспериментальной проверки качества работы Алгоритмов 4 и 6, примеры использования созданного программного комплекса для решения различных практических задач.

Автор выражает искреннюю благодарность и признательность
д.ф.-м.н., проф. Г.С. Осипову.

Глава 1. Задачи маршрутизации, представление знаний в мультиагентных системах, подходы к формированию коалиций и распределению ролей агентов

Агентное моделирование является одним из подходов к моделированию и изучению свойств сложных динамических систем. Оно основано на изучении, как изменения свойств элементов системы – агентов – влияют на поведение системы в целом. Агенты способны к взаимодействию с системой и между собой, они реагируют на различные события, например, на получение сообщений. Моделирование происходит «снизу вверх» - у агентов задаются свойства и правила взаимодействия между ними, после чего выполняется наблюдение за динамикой изучаемой системы. Такой подход очень удобен и часто применяется при анализе сложных динамических систем, в которых имеется большое число действующих элементов и поведение этих элементов легко описать, например, при моделировании экономических, социальных задач, при моделировании «роевого интеллекта», взаимодействия роботов и так далее.

Согласно определениям, данным в работах Д.А.Поспелова, В.Б.Тарасова, В.И.Городецкого [1-3], агенты – некоторая программная сущность, обладающая свойствами автономности, целенаправленности, реактивности, способностью к взаимодействию с другими агентами. Важной отличительной чертой агентов является наличие у них целей и мотивов достижения этих целей. Кроме того, наличие кооперации между агентами в результате их взаимодействия позволяет им совместно достичь таких целей, которые недостижимы каждым агентов индивидуально. Предшественником этого подхода являются начатые в 1960х годах М.Л.Цетлиным, В.Ю.Крыловым, В.И.Варшавским, Д.А.Поспеловым и В.Л.Стефанюком исследования математических автоматных моделей. Были получены результаты по поведению одиночного автомата с различными свойствами и по коллективному поведению автоматов [4-7]. Агентный подход можно применять при моделировании поведения роботов [8], что позволяет сначала изучить все требуемые аспекты поведения и их взаимодействия и лишь затем приступить к их воплощению.

К настоящему времени предложено несколько определений термина «агент», рассматривающие разные аспекты этого понятия. В отличие от объекта в смысле объектно-ориентированного подхода, который представляет собой оболочку методов и атрибутов некоторого программного модуля, программный агент является абстракцией более высокого уровня. Он характеризуется не набором атрибутов и методов, а набором действий, для выполнения которых он предназначен. Программный агент является автономным, он может действовать как независимый процесс и выполнять действия без участия пользователя. Основной характеристикой агента является его возможность делать независимые суждения в процессах решения задачи, разрешения конфликтов и принятия решений [9]. Агенты по своему поведению более автономны и проактивны, чем объекты. Согласно принятому в 2020 году Национальному стандарту РФ, термин «агент» определяется как «физический/программный объект, который оценивает собственное состояние, состояние других объектов и окружающей среды для выполнения своих действий, включая прогнозирование и планирование, которые максимизируют успешность, в том числе при неожиданном изменении оцениваемых состояний, достижения своих целей» [10].

Агенты могут быть различной степени сложности, начиная от тех, набор действий которых определяется набором условий вида "если <условие> то <действие>", и заканчивая когнитивными агентами, созданными на основе знаковой картины мира, моделирующими когнитивные функции сознания человека [11, 12].

Специализированный программный комплекс, позволяющий создавать агентов с нужным набором правил поведения, модифицировать их, прекращать выполнение конкретного агента или типа агентов и берущий на себя обеспечение передачи сообщений между агентами, предоставление информации о присутствии агента с нужными параметрами, другие служебные функции, называется мультиагентной системой. В рамках одной мультиагентой системы могут быть созданы агенты различных типов с различными свойствами и поведением, в том числе и когнитивные агенты.

В работах В.И.Городецкого с соавторами мультиагентная система (МАС) определяется как «сеть слабо связанных решателей частных проблем (агентов), которые существуют в общей среде и взаимодействуют между собой для достижения тех или иных целей системы. Взаимодействие может осуществляться агентами либо прямым образом – путем обмена сообщениями, либо некоторым косвенным образом, когда одни агенты воспринимают присутствие других агентов через изменения во внешней среде, с которой они взаимодействуют. МАС может содержать несколько однотипных или разнотипных агентов, которые могут иметь общие и/или различные цели, могут быть распределенными по компьютерной сети, могут быть написаны на различных языках программирования и работать на различных операционных платформах» [13].

Целью исследования в настоящей диссертационной работе является разработка метода решения задач маршрутизации на основе интеллектуальных транспортных агентов и их коалиций в условиях, когда каждый агент не обладает достаточными ресурсами для выполнения заданий и характеристики агентов в процессе моделирования могут меняться; каждый из транспортных агентов обладает способностью к изменению своих оценок успешности, определяемых в результате его целенаправленной деятельности, предназначенных для улучшения качества (оптимизации) получаемого решения.

1.1. Различные задачи маршрутизации

В современных научных исследованиях много внимания уделяется таким представляющим значительный практический интерес задачам, как транспортная задача (задача Монжа-Канторовича) и задача маршрутизации (задача Данцига-Рамсера, *VRP)*. В общем случае эти задачи связаны с доставкой некоторой продукции из нескольких пунктов одного типа (источников) в пункты другого типа (пункты назначения). Требуется доставить определенный объем продукции с минимальными затратами. В качестве затрат могут выступать как физические величины – расстояние, время, так и затраты в экономическом смысле. К настоящему времени изучены различные варианты задач маршрутизации,

связанные с дополнительными ограничениями на вид продукции, свойства транспортных средств, временные ограничения и т.д. Например, в работе [14] рассматривается классификация таких задач на основе 16 различных параметров. Некоторые сочетания параметров приводят к таким широко известным задачам, как: транспортная задача, задача о минимальном паросочетании, задача о загрузке рюкзака, задача коммивояжера.

Классическая транспортная задача

Если количества имеющейся и требуемой продукции равны (сбалансированы), то задача является задачей линейного программирования. В противном случае вводится дополнительный источник или пункт назначения для балансировки. Известно несколько алгоритмов оптимального решения задачи, минимизирующих суммарную стоимость перевозок. Например, для итерационного алгоритма сначала определяется начальный опорный план методом северно-западного угла. Затем последовательно применяется метод потенциалов. В графовой постановке решается задача нахождения максимального потока минимальной стоимости. К настоящему времени известно несколько алгоритмов решения, например, со сложностью $O(m \cdot n^2 \cdot (\log m + n \cdot \log n))$ [15].

Известна динамическая постановка задачи, когда значения потребности пунктов назначения и доступное количество продукции у источников могут меняться со временем [16, 17]. В этом случае переход к классической постановке возможен, если количество тактов изменения потребностей и доступного количества является конечным и все значения потребностей и доступного количества заданы заранее. Кроме того, в результате сведения задачи в динамической постановке к классической может получиться задача очень большой размерности [16, 17].

Классическая задача маршрутизации

Первоначальный вариант задачи (*VRP, Vehicle Routing Problem*) был сформулирован следующим образом [14]. Пусть имеется один источник

однородной продукции (склад, депо) и несколько пунктов назначения. Продукция доставляется несколькими транспортными средствами (ТС) с одинаковой грузоподъемностью. ТС выезжают из депо с грузом продукции, могут посетить несколько пунктов назначения и после исчерпания у них запаса продукции возвращаются в депо. В каждый пункт назначения продукция в требуемом количестве доставляется один раз и одним ТС. Требуется минимизировать суммарный пробег всех ТС.

Представляет интерес мультитранспортный вариант задачи маршрутизации (*Split Delivery VRP, SDVRP*). В этом варианте допускается, чтобы потребность пункта назначения была удовлетворена более чем одним ТС, не обязательно прибывающими в этот пункт одновременно. В таком варианте по сравнению с классической постановкой в зависимости от соотношения потребностей пунктов назначения и грузоподъемности ТС, их суммарный пробег может уменьшиться в несколько раз, но задача остается NP-полной [18]. Для решения этой задачи предложены точные переборные алгоритмы [19, 20]. Кроме того, известны более чем 80 стандартных наборов входных данных, для некоторых из них (для большинства наборов входных данных с количеством до 50 агентов, а также для одного набора с 75 агентами и одного набора с 100 агентами) установлены результаты оптимального решения [21].

В настоящее время благодаря применению современных систем высокоскоростной связи и точного позиционирования стало возможным отслеживать как потребности Пунктов назначения, так и положение и состояние Источников – транспортных средств в режиме реального времени. Поэтому возникает потребность в разработке метода решения задач маршрутизации, позволяющего находить решение в условиях изменения следующих параметров:

- количества пунктов назначения и источников;

- значений потребности пунктов назначения и доступного количества продукции у источников;

- характеристик источников – оценок успешности, определяемых в

результате целенаправленной деятельности.

Для моделирования возникающих взаимодействий в такой изменчивой среде широко используются мультиагентные системы. Еще одним известным и хорошо зарекомендовавшим себя подходом являются системы, основанные на правилах. В большинстве известных публикаций системы, основанные на правилах, использовались в качестве экспертных систем. Для решения задач маршрутизации системы, основанные на правилах, применяются не очень широко, обычно в сочетании с использованием нечеткой логики [22]. Использование системы правил для реализации агентов позволяет повысить гибкость в поведении агентов, легко их модифицировать и дает возможность отделить программный код от описания действий, определяющих поведение. Совместное применение мультиагентных систем и агентов, основанных на правилах, позволяет использовать преимущества обоих подходов [23].

Благодаря тому, что у задач маршрутизации известны стандартные наборы входных данных, для которых получены оптимальные решения, становится возможным оценить близость полученных решений к оптимальным [24]. Для того, чтобы приблизить получаемые решения к оптимальным, применяются различные подходы к улучшению характеристик агентов [11].

Далее в этой главе будут подробно системы, основанные на правилах, мультиагентные системы, различные способы модификации агентов и их поведения.

1.2. Интеллектуальные программные системы

Спектр применения интеллектуальных программных систем очень широк: с их помощью решаются задачи поддержки принятия решений, моделирования бизнес-процессов, управления группами независимых автономных устройств [25].

Продукционные системы или системы, основанные на правилах, являются одним из видов интеллектуальных систем. Продукционное представление знаний и в настоящее время используется в различных отраслях искусственного интеллекта,

от экспертных и консультационных систем до управляющих в режиме реального времени и динамических систем. В рамках этого подхода изучаются способы хранения данных, методы обеспечения полноты и правильности данных в базе знаний, механизмы наполнения или модификации базы знаний, в том числе, в удобном для пользователя графическом виде.

Другим направлением развития стало создание мультиагентных систем. Важными техническими характеристиками оболочки мультиагентых систем является ее масштабируемость и производительность. Важными свойствами агентов можно назвать их внутреннее устройство, применяемые при их создании методы хранения информации и обработки входящих сообщений и степень их интеллектуальности.

1.2.1. Системы, основанные на правилах

Согласно определению, система, основанная на правилах, «состоит из рабочей памяти, множества правил, выполняющих различный действия (во внешней среде и в рабочей памяти) и некоторой стратегии управления» [25].

Правило — упорядоченная тройка множеств:

r = <Con, Add, Del>, (1)

где *Con* – условие правила; *Add* – множество фактов, добавляемых правилом *r*; *Del* – множество фактов, удаляемых правилом r. Множества добавляемых и удаляемых фактов состоят из атомарных формул исчисления предикатов первого порядка. В состав формул входят свободные переменные, значения которых заменяются на текущие из рабочей памяти, таким образом формулы превращаются в факты. Правило применимо, если условие правила истинно после подстановки текущих значений из рабочей памяти. В результате применения правил состояние рабочей памяти изменяется. Стратегия управления из всех применимых правил определяет одно и применяет его. Процесс останавливается, если либо состояние рабочей памяти соответствует целевому, либо все правила оказываются неприменимыми [25].

К основным достоинствам систем, основанных на правилах, относятся

простота механизма логического вывода и модификации системы правил. Недостатками являются сложность оценки целостного образа знаний, описываемых системой правил, и необходимость проверки применимости всех правил на каждом шаге логического вывода.

Наполнение и редактирование базы знаний

Приобретение знаний интеллектуальными системами возможно как с использованием автоматизированных методов, так и от эксперта-специалиста [26]. Для эффективной работы эксперта ему необходимо предоставить удобный механизм задания правил для описания предметной области. Одним из способов является задание правил в наглядном графическом представлении, используемом для языка *UML*. В [27] рассмотрена надстройка над оболочкой для создания продукционных систем *JESS* [28, 29], позволяющая визуально редактировать правила и базу знаний. Система *JESS* для организации базы знаний использует фреймы. Поскольку фреймы могут наследоваться и быть вложенными, то наполнение базы знаний вручную является сложной задачей. Поэтому для упрощения работы экспертов авторы разработали систему визуального задания правил. Система позволяет задавать классы, объекты, правила и функции. При указании классов возможно наследование. Объекты — это экземпляры классов с уже заданными значениями атрибутов. Можно задать специфичные для данной системы пользовательские функции – они могут использовать другие пользовательские функции или стандартные функции языка *Java*. В окне задания правил указывается условие и список действий, при этом из подготовленного списка можно выбирать заданные ранее классы, объекты и использовать функции. После задания всей требуемой информации, пользовательские описания переводятся в формат языка *Java* и запускается их выполнение. В качестве планируемого расширения авторы предлагают включить системы проверки синтаксиса и отладки [27].

В статье [30] описывается подход к визуальному представлению и редактированию правил. Для визуального редактирования правил используется

система *Palantir Government* [31], которая предназначена для анализа и визуализации данных большого объема. В разработанной системе пользователь может импортировать данные, создавать и редактировать данные, осуществлять логический вывод на основе указанных данных и получать результаты. В основе подхода лежит создание двух графов, которые потом станут условием и результирующей частью соответствующего правила. Есть два типа условий: наличие/отсутствие в базе факта с заданными атрибутами и отношение между атрибутами существующих фактов. В результирующей части может быть изменение/удаление факта из базы или добавление нового факта в базу. При создании правила используется программное средство *Palantir Government:* пользователь создает граф, содержащий тело правила, его условия, объекты и отношения между ними. После этого задаются свойства объектов, переменные и ограничения. Затем создается другой граф, модификация первого, который содержит заголовок правила. Пользователь добавляет/удаляет/изменяет объекты или отношения на этом графе. Заключения — изменения в базе знаний после применения правила — могут быть представлены как разница между двумя графами. Для сравнения разных графов в системе нет встроенного графического средства, поэтому авторами была разработана панель визуального создания правила. Созданные в таком виде правила переводятся в формат представления системы *JESS*, подаются на вход системы и полученный результат выдается пользователю в виде второго графа для сравнения.

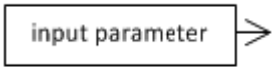
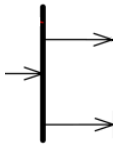
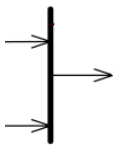
В формате представления *Jess* объекты и отношения на графе представляются тройками субъект-предикат-объект. Отношение (ребро) графа переводится в предикат, начальная точка — в субъект, конечная точка — в объект. Двунаправленная программа перевода информации из формата *Palantir* в формат *JESS* создана с применением *XML*-схемы соответствия. С помощью разработанного подхода применен анализ налогообложения компаний в Европе. Правила сохраняются в *XML* и для использования конвертируются в формат системы *JESS* [30].

Для визуального задания правил получило распространение использование

редакторов языка *UML* [32]. *UML* является языком графического описания для моделирования при разработке программного обеспечения, системного проектирования, моделирования бизнес-процессов. Этот язык обладает большими возможностями, *UML* диаграммы довольно просты и наглядны, для работы с ними разработано большое количество программ-редакторов.

UML представление правил используется в работе [33]. Внутреннее представление правил является гибридным: таблицы решений связаны в древовидную структуру. Таким образом достигается модульность представления базы знаний. На нижнем уровне правила, действующие в одном контексте, представляются единой таблицей решений. На верхнем уровне хранится полная база знаний в виде связанных таблиц решений. Для представления таблиц решений используется механизм диаграмм активности из представления *UML*. Установленные соответствия приведены в Табл. 1.

Таблица 1. Соответствие элементов диаграмм активности и таблиц решений

Элемент диаграмм активности UML	Графическое представление	Смысл в предлагаемом подходе
Activity Parameter		Параметр хранит XTT2 атрибут
Action		Устанавливает значение выходного параметра
Decision Node		Проверят что входные параметры различны
Merge Node		Логическое «или» для потоков
Fork Node		Разделение потоков
Join Node		Логическое «и» для потоков

Каждая таблица нижнего уровня имеет соответствующий *Action*, который вызывает поведение, описанное на нижнем уровне. *Actions* соединяются так же, как и на нижнем уровне модели. Для нижнего и верхнего уровней созданы метамодели, описывающие как нужно строить описанные модели. Таким образом, с использованием метамоделей можно строить трансформации этих моделей. На основе спецификации *XML Metadata Interchange* описаны методы трансформации схем для передачи информации между различными средствами, работающими с *UML*. После того, как средствами *UML* редактора были созданы правила, их требуется перевести в представление, содержащее двухуровневые таблицы решений. Сначала переводятся таблицы нижнего уровня, затем строятся связи между полученными таблицами. Создан алгоритм перехода обратно от древовидных таблиц решений в *UML* диаграммы. Поскольку для представления таблиц применяется формат *XML*, то можно использовать *XSLT* для конвертации, для этого разработано много сторонних программ [33].

В статье [34] авторы используют надстройку над диаграммами классов *UML* для представления правил. Есть много редакторов для представления *UML*, которые могут их отображать. Для представления правил был разработан специальный язык *URML*, который расширяет нотацию *UML* и определяет графическое отображение правил. С целью обеспечения возможности обмена правилами между разными системами был разработан язык разметки правил *R2ML*. Правила могут быть двух видов: вывода и продукции. Поскольку это расширение *UML*, то стандартные средства использовать нельзя [34].

Проверка полученной базы знаний

Важным этапом после наполнения базы знаний является проверка полноты и правильности полученной системы правил.

Полнота базы знаний системы означает, что система позволит ответить на все возможные вопросы, которые могут быть заданы, с учетом области знаний, в которых будет применяться система и планируемых сценариев использования.

Правильность означает, что в наборе правил в базе знаний системы

отсутствуют все неопределенности, несоответствия, расхождения и избыточности.

С ростом сложности решаемых задач и, соответственно, ростом количества правил вывода, хранящихся в базе знаний продукционных систем, автоматическая проверка полноты и правильности системы правил становится все более важной. Если раньше это удавалось проверить на простых примерах, то теперь, ввиду сложности и комплексности моделируемой предметной области, требуется разработка автоматизированного подхода к решению этой задачи.

Например, статья [35] посвящена разработке системы проверки полноты и правильности базы данных правил для продукционной системы. Для решения поставленной задачи предлагается сначала задавать информацию от эксперта в виде таблиц решений, которые позволяют представлять информацию в виде, удобном для анализа и проверки. Таблица состоит из строк-условий в верхней части и строк-действий в нижней. Столбцы задают правила: на пересечении данного столбца со строками-условиями заполняются условия (пусто, истина, ложь), на пересечении столбца со строками-действиями галочками отмечаются те действия, которые необходимо предпринять при выполнении заданных условий. Такое представление позволяет наглядно проверить избыточность и полноту. При необходимости информацию легко добавлять. Возможно создание иерархических таблиц, когда вместо действий ячейка ссылается на другую, подчиненную таблицу. Таблица решений естественным образом переводится в систему правил.

Описываемая в работе программа загружает все правила из базы данных системы в единую таблицу-решение, таким образом, для каждого правила в таблице предназначен один столбец, для каждого уникального условия и каждого уникального действия — строка.

После этого правила разделяются на логически несвязанные — все правила, которые имеют хотя бы одно общее правило с хотя бы одним правилом из подгруппы, объединяются в подмножество. Процесс повторяется, пока к подгруппе больше нельзя добавить ни одного правила. Затем каждая подгруппа проверяется на избыточность, полноту и отсутствие нужных правил. Если избыточность можно убрать автоматически, то она убирается. Полнота или

отсутствие нужных правил предъявляется пользователю — инженеру по знаниям для принятия решения о необходимости добавления правил с целью устранения найденных проблем [35].

Управляющие системы, основанные на правилах

Помимо хранения фактов-данных и вывода на их основе новых данных продукционные системы могут быть использованы для управления различными процессами или агентами. Динамические системы, основанные на правилах, предложены давно и хорошо изучены [36]. В этом типе систем каждое из множеств в Формуле 1 (*Con*, *Add*, *Del*) разбивается на два подмножества – в формулах первого из них обязательно участвует переменная, соответствующая дискретному времени t, в формулах второго такой переменной нет. Первое подмножество соответствует изменению состояния системы (то есть, действиям), эти правила являются правилами переходов. Второму подмножеству соответствуют правила замыкания, которые пополняют множество фактов о состоянии системы. Описаны стратегии применения правил замыкания и правил переходов [25].

Для моделирования сложного поведения возможно применение нескольких уровней правил. Первый (активный) уровень отвечает за базовые способности агента, например, перемещение, простые действия. Более высокий (делиберативный) уровень отвечает за более сложное поведение, например, глобальное планирование траектории перемещения, согласование действий с другими агентами, объединение агентов в группы для решения сложной задачи и т.д. [25, 37].

Для коллективного решения некоторой задачи группой агентов был разработан подход с использованием механизма общих намерений (*joint intentions*). Должен существовать план действий группы агентов, который потом может быть уточнен для каждого отдельного агента, где будут отражены вклад и ответственность каждого агента за достижение поставленной перед группой общей задачи [38].

Модель BDI [39] представляет собой процесс рассуждений, который

помогает принять решение о выборе подходящего действия для достижения некоторой цели. Она состоит из трех частей: убеждений — знаний агента о своем окружении, себе и других агентах; желаний — желаемых состояний, которых он хочет достичь; намерений — последовательности шагов, которые нужно предпринять для достижения желания. Эти части представляют соответственно информационное, мотивационное и обосновывающее состояния агента. Процесс рассуждений работает следующим образом: функция проверки убеждений получает входную информацию из среды (например, от датчиков) и отвечает за обновление базы убеждений. На основе обновленной базы убеждений и базы намерений могут быть сформулированы новые желания. Далее обновляется база намерений с учетом ее предыдущего состояния и текущего состояния баз убеждений и баз желаний. В результате выбирается и выполняется одно из намерений агента.

Расширяя BDI модель, вводятся следующие понятия [40].

Агент A имеет слабую цель P, если:

1. A убежден, что P в настоящее время ложно и имеет целью сделать P истинным (т.е., имеет P как обычную цель),

либо

2. A убежден, что P истинно, недостижимо или потеряло смысл и имеет целью убедить в этом всех остальных агентов в группе.

Группа агентов имеет общую цель, если каждый агент в группе имеет собственное намерение достичь эту цель.

Для выполнения общей цели P требуется следующее:

1. все агенты в группе убеждены, что P в настоящее время ложно

2. все агенты в группе уверены, что они намерены сделать P истинным

3. все агенты в группе до тех пор, пока не будет известно всем агентам, что P достигнуто, недостижимо или потеряло смысл, будут иметь в своем плане P в качестве слабой цели.

В начале при добавлении общей цели P каждый агент в группе имеет P как обычную цель. Однако, с течением времени агенты могут рассматривать цель P

только как слабую цель. Это связано с тем, что какой-то из агентов мог установить, что P достигнуто, недостижимо или потеряло смысл, и в это время находится в процессе сообщения об этом остальным агентам. Наличие общей слабой цели в группе гарантирует, даже если то, что цель P достигнута, недостижима или потеряла смысл могут определить только некоторые агенты из группы, то, и в этом случае об изменении статуса P будут уведомлены все агенты в группе.

Для выполнения общей цели агентам необходимо обмениваться сообщениями между собой о прогрессе в выполнении плана. Однако, в некоторых случаях обмен сообщениями может не требоваться, если прогресс выполнения очевиден всем агентам – например, группе агентов требуется добраться до точки сбора с известными координатами и все агенты прибыли на место сбора.

Например, в работе [41] рассматривается пример группы ударных вертолетов, имеющих задачу сообща нанести ракетный удар по общей цели. Сначала группа должны прибыть в точку сбора, ожидать там, пока разведчик не обнаружит противника, нанести удар и вернуться на базу. На Рис. 1 показана иерархия действий агентов. Групповые действия обозначены прямоугольниками, частные действия индивидуальных агентов – обычным текстом. Структура групповых и частных действий одинакова и состоит из трех частей – правил применимости, основная часть и правил завершения. Подцелью группового действия может быть либо групповое, либо частное действие.

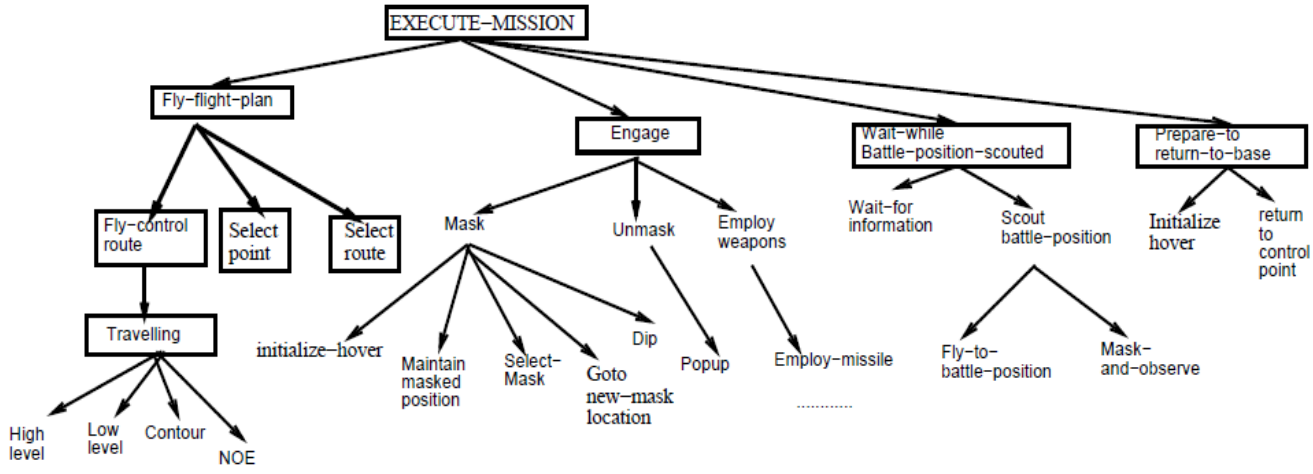


Рис. 1. Иерархия действий, выполняемых индивидуальным агентом [41]

Для моделирования групповых действий агентов-вертолетов была использована система из примерно 1000 правил. В результате моделирования было установлено, что система способна к выполнению большинства групповых действий. Если при выполнении плана один или несколько агентов не могут выполнить назначенные им действия (например, они потерпели крушение), то предусмотрен механизм замены этих агентов на других, которые активны и способны выполнить требуемые действия [41].

1.2.2. Мультиагентные системы

Мультиагентые системы состоят из частей (агентов), взаимодействующих между собой, при этом поведение каждого агента обычно определяется набором правил. Основа взаимодействия агентов между собой — передача сообщений. Таким образом, мультиагентная система — это специализированный программный комплекс, позволяющий создавать агентов с нужным набором правил поведения, модифицировать их, прекращать выполнение конкретного агента или типа агентов; обеспечивающий передачу сообщений между агентами; выполняющий служебные функции, например, оповещение об активности конкретного агента или типа агентов, отладочную информацию и т.д.

Согласно работе [42], отличительной особенностью мультиагентного подхода является то, что автономные части системы (агенты) координируют между собой свои знания, цели и планы для совместных действий или взаимодействия для решения некоторой задачи. В мультиагентной системе агенты совместно могут решать одну общую задачу, или несколько различных задач, которые пересекаются между собой. Более того, цели, которые агенты планируют достичь, могут конфликтовать между собой. Агенты должны делиться между собой информацией о задачах и путях их решения. Кроме того, они должны быть способны к координации между собой в условиях открытого мира, когда нет глобального управления, глобальных систем хранения знаний, глобальных целей и критериев достижения цели.

В рамках мультиагентного подхода изучается множество задач, например,

формирование коалиций агентов в результате переговоров [43, 44], процедура голосования, определения лидера и распределения ролей в однородной группе [45]; модели социального поведения в группах [46]. Важным применением мультиагентных технологий является построение на их основе интеллектуальных систем управления ресурсами предприятий в реальном времени [47].

Ранние мультиагентные системы

В настоящем разделе рассмотрены несколько типичных ранних мультиагентных систем [48].

Система *Activation Framework (AF)* – основа для создания систем искусственного интеллекта на нескольких компьютерах, соединенных между собой, она была создана на основе системы *Hearsay2* [49] и работает в парадигме "сообщество экспертов".

Несколько агентов системы (*AFO, Activation Framework Object*) общаются путем передачи сообщений. Каждый агент (как эксперт) имеет предметную область, в которой он работает, и набор предположений и связанных с ними наборами процедурного программного кода. У каждого агента есть очередь входящих и исходящих сообщений. Он получает первое сообщение из входящих, обрабатывает его по своим правилам и, возможно, помещает ответное сообщение в исходящие. Агенты работают асинхронно, то есть не ожидают того, как сообщение будет отправлено.

У предположений есть порог активации (*Activation level*) – значение между -1 и 1, которое имеет смысл уверенности в истинности предположения, где -1 соответствует ложности, а 1 соответствует истинности предположения. Таким образом, может быть установлена истинность выводов, сделанных на основе этих предположений.

Порог активации есть у самих агентов и каждого сообщения. Отправляющий агент назначает сообщению порог активации в зависимости от важности сообщения. Порог активации объекта есть сумма порогов активации всех входящих сообщений для этого объекта. Если это значение превышает некоторое

значение, то запускается агент с наибольшим значением порога активации. Входящие сообщения упорядочиваются по времени поступления и по значениям порога активации.

Каждое сообщение содержит информацию об агенте-отправителе и агенте-получателе, о типе сообщения и строку с сообщением.

Агенты входят в состав оболочек (*Framework*) – сущностей, которые отвечают за активизацию входящих в них агентов и передачу сообщений между ними. Они также отвечают за пересылку сообщений средствами операционной системы агентам из других оболочек, выполняющимся на удаленных компьютерах. Оболочки могут содержать агенты, написанные на разных языках программирования.

MACE [50] - еще один пример ранней мультиагентой системы. Агенты в ней выполняются параллельно, общаются путем передачи сообщений и имеют возможность представления знаний. Основные части системы:

- язык описания агентов *ADL*;

- база данных с описаниями агентов;

- ядро системы (отвечает за передачу сообщений, ввод-вывод, распределение агентов по вычислительным процессорам);

- системные агенты (интерпретатор команд, генератор агентов, монитор агентов);

- базовые общие части, которые могут использоваться всеми агентами (стандартные сообщения, стандартные ошибки).

Агенты хранят знания в виде значений атрибутов, получают информацию извне из входящих сообщений и событий системы и могут выполнять следующие действия:

- изменение внутреннего состояния (изменение значений атрибутов);

- отправка сообщений.

Для выполнения перечисленных действий агенты имеют навыки – локально определенные функции, реализующие процедурные знания. Навыки являются функциями, написанными на языке программирования *LISP*. К навыкам нельзя

обратиться извне агента, но их можно передать внутри сообщения – так реализована возможность обмена опытом между агентами. Навыки вызываются ядром агента – его единственной действующей частью. Ядро определяет действия агента, обрабатывает поступившие сообщения и манипулирует значениями атрибутов. Ядро начинает работу при получении входящего сообщения и переходит в режим ожидания после окончания обработки сообщения. Агент имеет специальный служебный атрибут «статус» со значениями: «новый», «активный», «неактивный», «ожидание», «остановлен».

Получение информации может быть пассивным (получение входящих сообщений) или активным (отправка запроса и получение ответной информации). Сообщения упорядочиваются по времени поступления, имеют информацию об отправителе – его имя, класс, компьютер, на котором он выполняется. Содержание сообщения может быть любым. Сообщения могут быть адресованы индивидуальному агенту, группе агентов или классу агентов. События, возникающие в результате действий агентов (изменение статуса, создание или уничтожение агента), отслеживаются специальным механизмом – демонами. При возникновении таких событий демоны отправляют сообщения тем агентам, которые подписались на информирование о событиях, произошедших с выбранными агентами. Агенты хранят знания об окружающем их мире как значения своих внутренних атрибутов. Имеется специальный атрибут "знакомые". Знакомые являются отражением внешнего мира агента и хранят имена и адреса, класс, роли, умения, цели, планы тех агентов, с которыми он взаимодействует.

Язык описания агентов *ADL* – процедурный язык для описания действий с базой данных, необходимых для создания новых описаний агентов. Описания агентов хранятся в базе данных и используются для генерации исполняемого кода агентов. В нем напрямую не используется объектно-ориентированный подход, но есть возможность выборочно импортировать из описаний других типов агентов их атрибуты и функции. Можно скопировать имеющееся описание и, например, добавить, удалить или указать новые атрибуты. Имеется возможность задания специальных действий при инициализации создания исполняемого кода объектов.

Структура современных мультиагентых систем

К настоящему времени сформировалась типовая структура мультиагентой системы, включающая: агентов, сообщения, саму оболочку системы с базовыми функциями и ее служебные подсистемы.

Агенты — основная часть системы. Агенты могут взаимодействовать друг с другом и с системой в целом. Взаимодействие происходит путем передачи сообщений от других агентов или от системы. Агенты автономны, активны и, как правило, не обладают полной информацией об окружающем их мире. Агенты могут быть различной степени сложности, начиная от тех, набор действий которых определяются множеством условий вида "если <условие> то <действие>" и конечных автоматов (*Finite State Machine*), и заканчивая когнитивными агентами, способными к планированию целенаправленного поведения и изменению своих характеристик в результате деятельности.

Оболочка мультиагентной системы обеспечивает ее базовые функции – создание, запуск, остановку агентов и передачу сообщений между ними. Часто в состав оболочки бывают встроены средства для отладки – получение информации о состоянии каждого агента, список переданных сообщений и тд. В качестве дополнительных средств оболочки можно рассматривать службы поиска нужных агентов по критериям, получения от агентов информации об их реализуемых функциях, справочной информации, которую агенты сами предоставляют о себе и т.д.

Сообщения — способ передачи и получения информации агентами. В первых мультиагентных системах формат сообщений не был определен, как правило, в состав сообщения входили данные об отправителе, получателе, иногда класс или тема сообщения. Затем была предпринята попытка стандартизации сообщений, так, чтобы разные мультиагентные системы могли общаться между собой. При этом важным вопросом является взаимодействие агентов между собой на различных уровнях [51].

Первый уровень – информационное содержание передаваемых сообщений.

На этом уровне передаются факты. Язык *Knowledge Interchange Format (KIF)* [52] определяет логическую структуру языка передачи информации и концептуальные словари или онтологии для передачи информации между различными программами или агентами.

Второй уровень описывает намерения агентов, т.е. их отношения к настоящим фактам. Для этого предназначен язык *Knowledge Query and Manipulation Language (KQML)* [53] – для выражения намерений так, чтобы все остальные могли однозначно их понять. Общение происходит при помощи специальных лингвистических конструкций – перформативов. Перформативы – допустимые «речевые акты» агентов, на основе которых строятся модели взаимодействия агентов более высокого уровня. Примерами перформативов могут служить предупреждения, приказания или обещания. Базовыми перформативами языка *KQML* являются: *evaluate, ask-if* (запросы), *reply, sorry* (ответы), *standby, ready* (управление последовательностью запросов), register, broadcast (управление режимами взаимодействия).

Третий уровень – уровень соглашений, т.е. возможность координировать свои цели и действия путем переговоров. В ответ на предложение может быть направлено согласие, отказ или контрпредложение. На этом уровне сначала использовался язык *COOL* [51], в нем агенты представляются конечными автоматами. Имеются начальные (агент может начать беседу) и конечные (текущая беседа заканчивается) состояния. Набор правил общения определяет, как агент в текущем состоянии реагирует на поступление сообщений определенного типа – меняет собственное состояние, обновляет внутренние значения атрибутов, отвечает другим сообщением. К стандартным перформативам языка *KQML* были добавлены следующие: *propose, counter-propose, accept, reject, satisfy, fail*. Правила общения представляют собой шаблоны сообщений заданного типа с переменными, на место которых требуется подставить соответствующие значения. Дополнительно имеются правила для временной приостановки и возобновления переговоров, например, когда одному из участников нужно уточнить информацию от третьего агента.

В результате проведения исследований по созданию языков общения агентов и обобщения полученных сведений были выработаны стандарты *FIPA Agent Communication Language* [54] и *FIPA Communicative Act Library* [55]. Эти стандарты описывают протоколы взаимодействия между агентами и состав и формат сообщений между агентами.

Протоколы взаимодействия определяют состав и последовательность применения различных перформативов для типичных ситуаций общения.

Самые применяемые перформативы: *propose, call for proposal, accept proposal, reject proposal, request, confirm, refuse;* составные: *inform, subscribe;* условные: *inform if, request when*.

Согласно рекомендации *FIPA Agent Communication Language* [55], формат сообщения включает несколько групп полей, представленных в Табл. 2.

Таблица 2. Группы и составляющие их поля сообщений.

Группа	Название поля	Значение
Перформатив	performative	Перформатив, акт общения
Адреса	Sender	Отправитель
	Receiver	Получатель
	reply-to	Ответить другому получателю
Передаваемое сообщение	content	Содержимое сообщения
Параметры сообщения	language	Язык
	encoding	Кодировка
	ontology	Онтология
Управление ходом общения	Protocol	Используемый протокол
	conversation-id	Идентификатор беседы
	in-reply-to	В ответ на идентификатор другой беседы
	reply-by	Отправить ответ до указанного времени

К настоящему времени с использованием описанных выше подходов и требований было создано значительное количество различных мультиагентных систем. Наиболее известными и широко применяемыми современными мультиагентными системами являются *SPADE* [56] (язык программирования - *Python*) и *JADE* [57], (язык программирования – *Java*).

Далее рассматривается структура мультиагентной системы *SPADE* и ее основные свойства.

Структура мультиагентной системы SPADE

Система разработана согласно предложениям *FIPA* для мультиагентных систем [56]. У нее есть встроенные базовые сервисы (Система управления агентами и Справочный координатор), реализованные как компоненты сервера *Jabber* [58]. На коммуникационном уровне используется протокол обмена Jabber или возможно применение протокола *HTTP*. Схема приведена на Рис. 2.

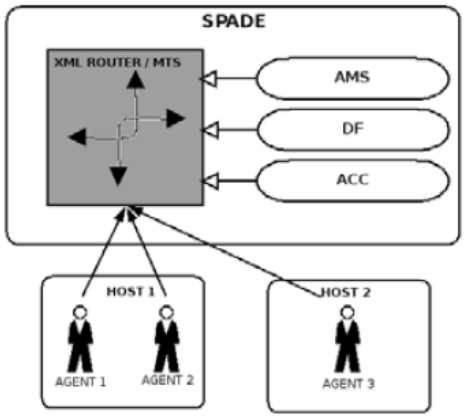


Рис. 2. Структура мультиагентной системы *SPADE* [56]

Центральной частью системы является *XML* роутер, к нему присоединяются все сервисы мультиагентной системы и агенты. Роутер реализован как стандартный *XMPP* сервер и обеспечивает передачу сообщений от отправителя всем указанным получателям, таким образом играя роль системы передачи сообщений внутри системы. К мультиагентной платформе могут присоединяться другие платформы, и, если они используют сообщения формата *XMPP*, то

сообщения между платформами могут передаваться без дополнительных преобразований и обработки.

Еще одним сервисом системы является Канал связи между агентами, обеспечивающий обработку всех сообщений в системе. Он получает и обрабатывает сообщения в стандартизованном *FIPA* формате сообщений *FIPA-ACL*. Получателями сообщений могут быть агенты или другие встроенные компоненты мультиагентной системы.

Безопасность системы поддерживается на нескольких уровнях:

1. Для доступа и присоединения к *XML* роутеру требуются логин и пароль. Таким образом, только сервисы и платформы, авторизованные администратором системы, могут получить доступ к системе.

2. *XML* роутер поддерживает шифрование сообщений по алгоритму *SSL*, что обеспечивает надежность и конфиденциальность при передаче сообщений.

3. *XML* роутер не допускает подмену значения поля "от" сообщения отправителем, поэтому невозможно появление нежелательных сообщений типа «спам».

Каждый компонент системы является агентом. Агенты – это элементы системы, присоединяющиеся к системе передачи сообщений, которая обеспечивает передачу сообщений внутри одной системы или между разными мультиагентыми системами. Каждый агент имеет механизм связи с платформой, диспетчера сообщений и множество различных задач, которым диспетчер перенаправляет на обработку поступившие агенту сообщения. Для регистрации на платформе каждому агенту требуется уникальный идентификатор *JID* и пароль.

Агент одновременно может выполнять несколько задач, при этом количество задач не ограничивается. Задачи являются программными процессами, в системе *SPADE* есть несколько встроенных типов задач: Циклический, Одноразовый, Периодический, По тайм-ауту, Конечный автомат. Встроенные типы позволяют создавать агентов с разнообразной и гибкой реакцией на входящие сообщения.

Когда агенту поступает входящее сообщение с определенными характеристиками (тип, отправитель, содержание и т.д.) диспетчер сообщений перенаправляет сообщение той задаче агента, которая отвечает за обработку сообщений этого типа. Схема работы агента представлена на Рис. 3.

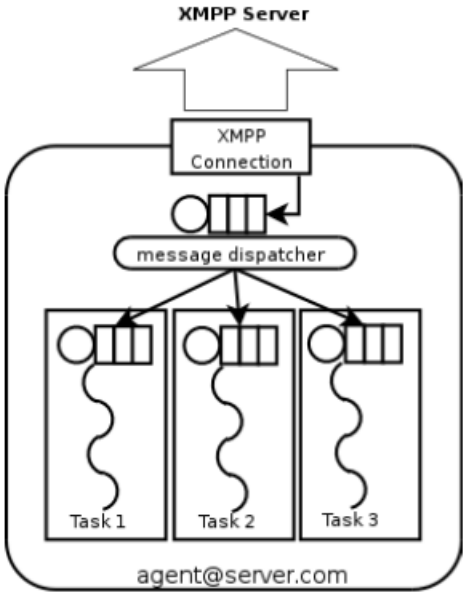


Рис. 3. Схема работы агента в мультиагентной системе *SPADE* [56]

Таким образом, мультиагентная система *SPADE* обладает следующими возможностями и отличительными чертами:

- соответствие спецификациям *FIPA*, реализованы сервисы Система управления агентами и Справочный координатор;

- поддерживает два протокола передачи сообщений: *HTTP* и *XMPP*, благодаря чему можно передавать сообщения между мультиагентной системой *SPADE* и другими мультиагентными системами;

- поддерживаются два формата передачи содержимого в сообщениях *FIPA-SL* и *RDF*, используемые в подавляющем большинстве современных мультиагентных систем;

- каждый агент может иметь несколько выполняемых задач, имеется несколько встроенных типов задач;

- имеется специальный агент, отображающий пользовательский интерфейс

системы. Этот агент позволяет пользователю запускать и останавливать работу агентов, искать агентов по параметрам, посылать сообщения любому агенту системы;

- система создана на языке программирования *Python,* что позволяет запускать систему на различных аппаратных платформах и в различных операционных системах. Кроме того, возможно применение специализированных средств *Python* для увеличения производительности системы;

- есть специализированный механизм Информации о присутствии. Это позволяет системе отслеживать состояние каждого компонента системы в реальном времени, а пользовательским агентам получать информацию о статусе тех агентов, на получение информации от которых они подписаны;

- возможность отправки сообщений агентами некоторой группе агентов. Это реализуется с помощью механизма многопользовательских конференций *Jabber*;

- для передачи сообщений используется протокол *Jabber*, предназначенный для передачи сообщений в условиях, когда количество сообщений и пользователей очень велико. Благодаря этому, мультиагентная система характеризуется хорошими масштабируемостью и производительностью;

- для регистрации каждого агента в системе требуется логин и пароль, передаваемые сообщения можно шифровать, что улучшает безопасность и обеспечивает конфиденциальность.

Масштабируемость и производительность

В результате экспериментов установлено, что производительность мультиагентной системы *SPADE* выше, чем у системы *JADE* при использовании не более 100 агентов [59].

Статья [60] посвящена моделированию «умных электрических сетей» («*smart grid*»). Поставщики и потребители электроэнергии сообщают о своих текущей и прогнозной потребностях, при этом стоимость электроэнергии может меняться в течение дня. Требуется в результате переговоров о цене обеспечить всех потребителей электроэнергией, минимизируя ее суммарную стоимость. Каждый

потребитель представлен одним агентом, в моделировании участвовали до 16 агентов. В работах [61, 62] рассмотрено применение генетического алгоритма с целью оптимизации нагрузки системы *smart grid,* основанной на солнечных батареях, и объединение нескольких локальных *smart grid* в единую систему для динамического распределения нагрузки и повышения надежности системы в целом. В работе [63] рассмотрена задача прогнозирования потребления электроэнергии городом средних размеров. Каждое домовладение представлено одним агентом. При моделировании использовались реальные данные о застройке, плотности и составе населения. Мультиагентная система *SPADE* моделировала поведение до 600 агентов-домохозяйств.

Таким образом, мультиагентная система *SPADE* является надежным и распространенным средством решения задач на основе агентного моделирования.

1.3. Задача формирования коалиций

Согласно определению, формирование коалиций — «процесс объединения агентов в результате их взаимодействия для того, чтобы они совместно могли решать задачи, которые не могут решить индивидуальные агенты» [64].

Постановка задачи

Проблема формирования коалиций определена следующим образом. Пусть A - конечное множество агентов, f(A) - характеристическая функция, которая каждому подмножеству агентов из A (коалиции) ставит в соответствие числовое значение. Требуется найти такое разбиение множества агентов A на коалиции A_1, A_2, .. A_m, чтобы суммарное значение для выбранных коалиций было максимальным, то есть $\sum f(A_i) \rightarrow \max$ [65].

В значительном количестве опубликованных работ каждой из возможных коалиций агентов в соответствие ставится числовое значение. В работе [66] рассматривалась задача уменьшения корпоративных налогов для компаний из США, поэтому в качестве числовых значений характеристической функции были использованы реальные данные.

В последующих работах для получения числовых значений были использованы случайные величины с равномерным или нормальным распределением. Например, в работах [67, 68] использовалось равномерное распределение, были исследованы два варианта получения значений весов коалиций:

$f(A_i) \sim U(a,b)$, где a=0, b=1

или

$f(A_i) \sim |A_i| \cdot U(a,b)$, где a=0, b=1, $|A_i|$ соответствует количеству элементов в коалиции A_i.

В работе [69] в дополнение к приведенным выше было применено нормальное распределение:

$f(A_i) \sim |A_i| \cdot N(\mu, \sigma^2)$, μ =1, σ=0,1,

и было доказано утверждение, что если числовые значения коалиций заданы по приведенным выше формулам, то в результате работы алгоритма более вероятно формирование коалиций, содержащих небольшое количество агентов. Для преодоления указанного недостатка в [69] было предложено использовать нормальное распределение с другими параметрами:

$f(A_i) \sim N(\mu, \sigma^2)$, $\mu = |A_i|$, $\sigma = \sqrt{|A_i|}$

Для некоторых случаев возможно построение характеристической функции специального вида, состоящей из двух слагаемых – супераддитивного, т.е. $f(A_j \cup A_k) \geq f(A_j) + f(A_k)$ и субаддитивного, т.е. $f(A_j \cup A_k) \leq f(A_j) + f(A_k)$ для всех A_j и A_k.

Например, в работе [65] для представления потребляемой агентами электроэнергии, закупаемой на форвардном (определенный объем закупается заранее, стоимость ниже) и спотовом (недостающий объем закупается в момент потребления, стоимость выше) рынках, применяется класс функций вида $f(A) = f^+(A) + f^-(A)$, состоящих из супераддитивной (f^+) и субаддитивной (f^-) частей. Таким образом, на некотором этапе процесса формирования коалиций выгода от объединения коалиций агентов в одну становится отрицательной, что ограничивает количество агентов в каждой коалиции. В качестве решения требуется найти соотношение количества электроэнергии, закупленной из этих

двух источников, для минимизации платы конечных потребителей. Доступны большие наборы реальных данных по объемам потребления электроэнергии домохозяйствами различных стран [70].

Для решения поставленной задачи разработаны подходы для точного решения на основе полного перебора и приближенные подходы. В последнее время активно развивается подход, в котором используется идея улучшения «способностей» агентов на основе оценки результатов их деятельности и последующего изменения оценки успешности [11, 12]. Подход основан на психологической модели когнитивной деятельности человека. Таким образом, коалиция агентов может быть образована на основе других критериев, например, с учетом общности интересов агентов; действий, которые они предпочитают выполнять или выполняют лучше, чем другие агенты.

Подходы на основе полного перебора

Оптимальное решение проблемы формирования коалиций агентов является NP-трудной задачей [71]. В результате исследований были созданы алгоритмы для решения поставленной задачи со следующими свойствами: централизованные или распределенные; выдающие результат только после окончания работы или итерационные (*anytime*), которые даже будучи остановленными до окончания работы, выдают результат, более близкий к конечному решению.

Решение задачи оптимального формирования коалиций агентов методом динамического программирования было предложено в работе [66]. Алгоритм очень затратен по памяти, но обеспечивает время работы $O(3^n)$, где n – количество агентов, что является приемлемым результатом. Работа алгоритма была проверена для количества агентов от 4 до 13.

В статье [72] описан итерационный алгоритм решения задачи формирования коалиций агентов - он быстро дает приближенный результат и потом его последовательно улучшает. Вводится понятие соотношение производительности R. Точное решение для каждой подкоалиции получают методом динамического программирования. Установлено, чем больше значение R, тем быстрее получается

конечный результат. Для каждой подкоалиции агентов используется фаза предобработки, во время которой методом динамического программирования определяется коалиция с наибольшим весом: для всех коалиций с числом агентов от 2 до n проверяется, не увеличится ли вес коалиции, если из нее удалить некоторого агента. Если увеличится, то эта коалиция запоминается. Во второй фазе определяется оптимальная коалиция для всех подпроблем, содержащих не более n/r агентов. Работа алгоритма была проверена для 25 агентов.

Семейство алгоритмов с различными свойствами и комбинации этих алгоритмов были предложены группой исследователей из университета Саутгемптона [68, 69, 73-81]. Первым из предложенных является алгоритм *IDP* [73]. В нем используется многоуровневый граф, представляющий все допустимые коалиции и связи между ними. В узлах графа находятся коалиции, а ребрами соединяются узлы, в которых коалиции отличаются только одним элементом. В работе доказано утверждение, что если из графа удалить ребра, оставив только ровно одно, ведущее к каждой вершине, то оптимальное решение все равно будет найдено (Рис.4).

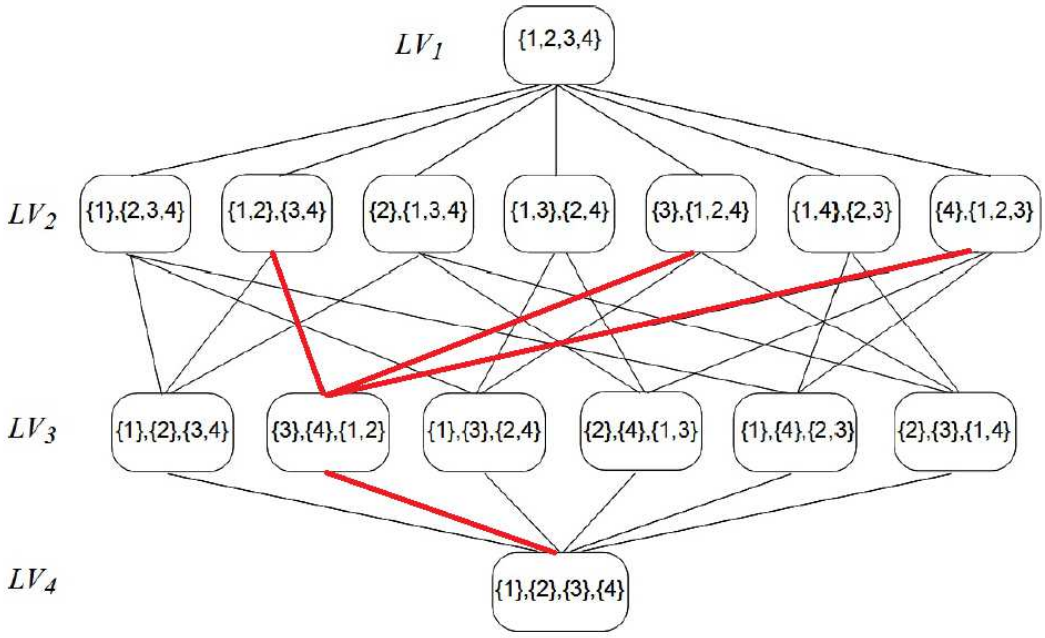


Рис.4. Представление коалиций в алгоритме *IDP* [73]. Выделены ребра, ведущие к одной из вершин.

Поэтому можно ограничиться одним ребром (т.е. одним переходом к соседней коалиции), не вычисляя при этом все возможные состояния. Таким образом, для получения оптимального решения количество вычислений по сравнению с работой [66] составляет около 40 %, нужный объем памяти - 33 %. Работа алгоритма была проверена для 25 агентов. Время работы алгоритма оценивается как $O(3^n)$. Использование ресурсов графического процессора видеокарты компьютера позволяет уменьшить время работы алгоритма в несколько десятков раз, но из-за ограничений по памяти количество агентов не может превышать 30 [75].

Алгоритм *IP* [68, 69] использует представление пространства поиска коалиций в двоичном виде. Пространство поиска разбивается на подпространства по размеру содержащихся в них коалиций. Полученные предварительные оценки верхней и нижней границы значений характеристической функции используются для отсечения подпространств, в которых оптимальное решение недостижимо. Затем алгоритм анализирует оставшиеся подпространства поиска методом ветвей и границ. Алгоритм является итерационным (*anytime*), но время работы алгоритма в самом плохом случае оценивается как $O(n^n)$.

Дальнейшим развитием стало применение алгоритма *IP* на нескольких вычислительных узлах [77]. Благодаря тому, что в алгоритме *IP* коалиции хранятся в двоичном представлении, и вычислительные узлы имеют уникальные идентификаторы, каждый вычислительный узел может однозначно выделить подмножество коалиций, которое он должен обработать. Далее у каждого вычислительного узла начинает работать алгоритм *IP*. В своем подмножестве коалиций, которые нужно рассмотреть, агент вычисляет предварительные оценки. Те подмножества, которые не смогут увеличить значение характеристической функции, отбрасываются. Оставшиеся упорядочиваются по убыванию оценок и сначала рассматриваются самые перспективные, используя поиск в глубину. Узлы пересылают друг другу полученные значения, на их основе вычисляются общее среднее и максимальное значения весов коалиций (Рис.5).

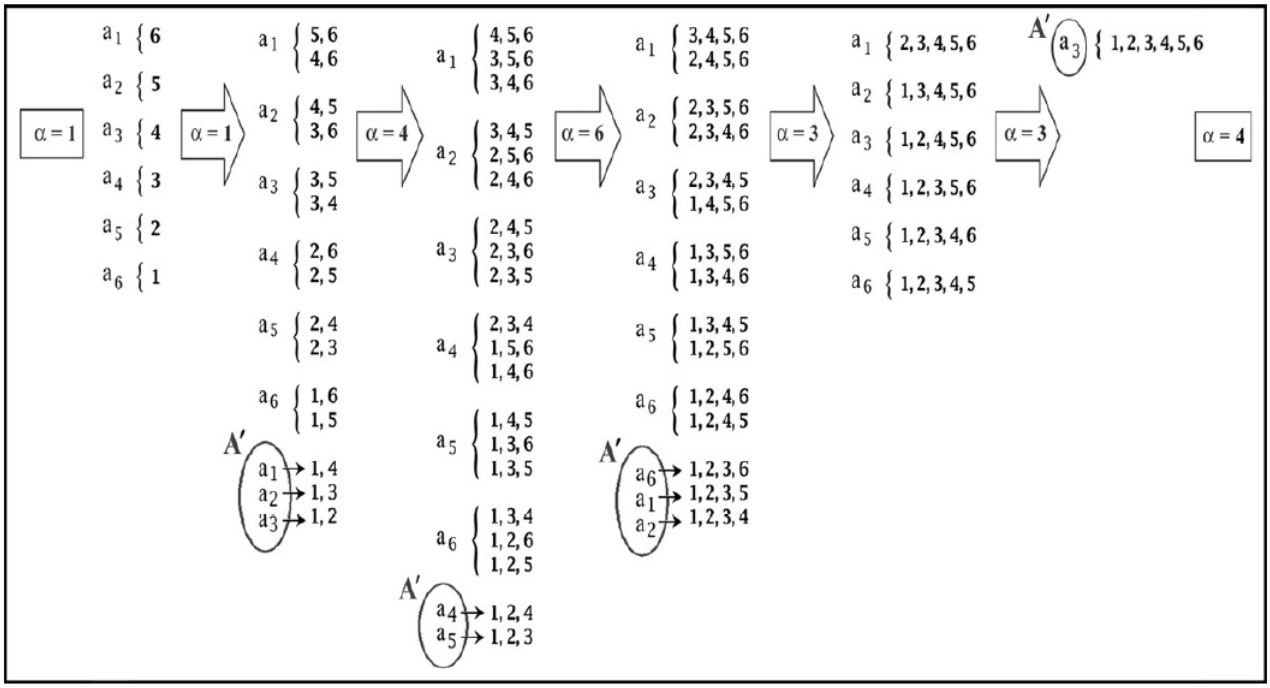


Рис.5. Распределение весов коалиций между 6 вычислительными узлами [77]

Каждый вычислительный узел отсекает согласно полученным ранее оценкам все бесперспективные коалиции. Все оставшиеся коалиции распределяются между всеми агентами для детального изучения. Эксперименты были проведены для 28 агентов.

Поскольку алгоритмы *IDP* (хорошее время работы, но не итерационный) и *IP* (итерационный, но в самом плохом случае требуется полный перебор) взаимно дополняют друг друга, то была предпринята попытка их совместить [76]. В предложенном гибридном алгоритме сначала работает алгоритм *IDP* для отсечения тупиковых подмножеств, а оставшиеся обрабатываются алгоритмом *IP*. Скорость нахождения решения у объединенного алгоритма *IDP-IP* существенно выше, чем у алгоритмов по отдельности: для получения решения алгоритму *IDP-IP* требуется 28% времени работы алгоритма *IP* и 0.3% времени алгоритма *IDP* на тех же входных данных [76].

Исследования в этом направлении были продолжены, в результате чего был создан алгоритм *ODP-IP* – один из быстрейших оптимальных алгоритмов формирования групп агентов [80, 81]. Авторы улучшили алгоритм динамического

программирования *DP* [66], удалив все избыточные вычисления. На его основе был создан алгоритм *ODP*, который для получения решения должен выполнить только одну треть вычислений по сравнению с алгоритмом *DP*. Было модифицировано представление разбиения пространства поиска, которое использует алгоритм *IP*, благодаря чему стала возможной совместная одновременная работа этих алгоритмов. Результаты, получаемые алгоритмом *ODP,* используются в работе алгоритма *IP* для более быстрой оценки и отсечения тех ветвей графового представления пространства перебора, в которых не может быть получено оптимальное решение. Полученный алгоритм имеет время работы порядка $O(3^n)$, является итерационным. Работа алгоритма была проверена для 25 агентов.

Все рассмотренные ранее алгоритмы используют значения весов, заданные для всех коалиций в явном виде, поэтому затраты по памяти только на хранение этих значений можно оценить как $O(2^n)$. Одним из способов уменьшения как требуемого объема памяти для хранения весов коалиций, так и пространства перебора является использование синергетических ядер коалиций [82, 83].

Синергетическими ядрами называют некоторое подмножество возможных коалиций, для которых возникает синергетический эффект, т.е. значение характеристической функции для ядра больше, чем сумма значений для всех элементов этой коалиции. Синергетические ядра неделимы (все коалиции-подмножества ядер по отдельности не рассматриваются), поэтому значения характеристической функции можно задавать для небольшого подмножества всех возможных коалиций, что существенно уменьшает перебор и увеличивает скорость нахождения решения. Кроме того, такое представление характеристической функции позволяет представлять многие классы функций, например несупераддитивные или даже немонотонные [83].

В работе [84] используется понятие синергетического графа. Дуги в графе представляют некоторую связь между агентами, например, возможность к общению, доверие, физические ограничения. Если между некоторыми агентами нет связи, то коалиция с их участием не рассматривается. Благодаря такому

подходу алгоритм способен к работе с 50 агентами [84]. Была изучена зависимость вычислительной сложности алгоритма формирования групп от топологии синергетических графов: линейной, циклической, дерева, дерева с ограничением по ширине, полносвязной. Установлено, что для линейных и циклических графов вычислительная сложность ниже [85].

Приближенные подходы

Для решения задачи формирования коалиций из несколько сотен или тысяч агентов могут быть использованы только приближенные алгоритмы.

В большинстве случаев агенты должны действовать максимально быстро, нет времени для того, чтобы рассмотреть все возможные коалиции агентов, поэтому должно быть выбрано некоторое их перспективное подмножество. Описан приближенный алгоритм, который за заранее установленное время либо находит решение, либо, при превышении времени работы останавливается, получив близкое к оптимальному решение. Метод основан на последовательном объединении полученных коалиций агентов и использовании стохастического локального поиска. Случайный поиск позволяет избежать попадания в локальный максимум, однако, нахождение глобального оптимального решения не гарантируется. Скорость нахождения решения для 27 агентов оказалась на 6 порядков выше, чем у алгоритма *IDP* [86].

Алгоритм *C-Link* [65] применяет функцию взаимозависимости, которая определяет увеличение выгоды при объединении двух выбранных коалиций в одну. На первом шаге работы алгоритма каждый из агентов представляет собой коалицию из одного элемента. Затем на каждом шаге выбираются две коалиции, для которых значение функции взаимозависимости будет максимальным. Происходит объединение коалиций и переход на следующий шаг. Работа алгоритма прекращается, когда невозможно найти пару коалиций – кандидатов на объединение (Рис. 6).

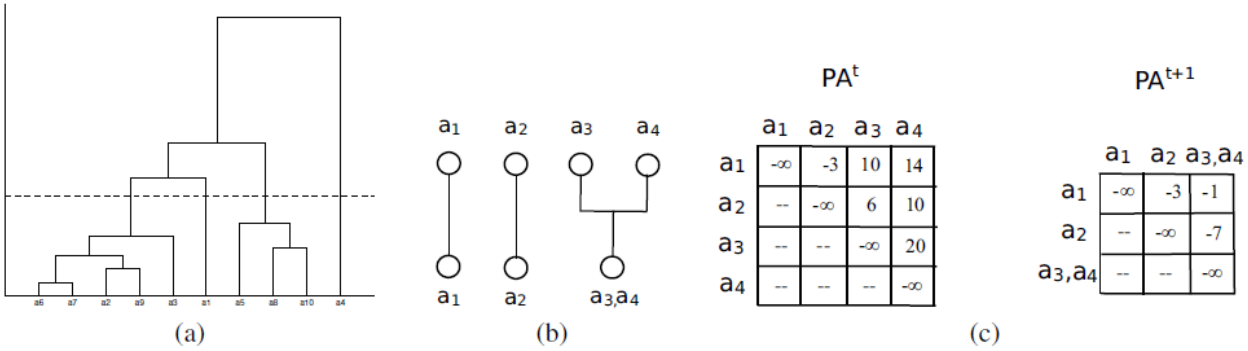


Рис.6. Схема работы алгоритма *C-Link* [65]. Последовательность объединения коалиций (a, b), пример расчета матрицы выгод (c).

Установлено, что для количества агентов не более 25 (которые может обработать оптимальный алгоритм *IDP-IP* [76]), близость полученных решений к оптимальным составляет около 90%. Предложенный алгоритм способен получить результат для 2732 агентов за несколько минут.

В работе [64] рассматривается алгоритм формирования коалиций агентов для решения задачи доставки посылок. Агенты неоднородны, они имеют различные значения таких атрибутов как честность, память, скорость, доверие. Был рассмотрен процесс объединения агентов, которые, возможно, имеют несовпадающие цели, в коалиции для улучшения их коллективных возможностей. Агенты при встрече принимают решение, будут ли они взаимодействовать с другими, и, если будут, то в рамках существующей или новой коалиции.

Задача *Coalitional Skill Games* является обобщением задачи формирования коалиций. В ней рассматриваются агенты $A=\{a_1,..., a_n\}$, задания $T=\{t_1,.., t_m\}$ и навыки $S=\{s_1,..., s_r\}$

Каждое задание $t_j \in T$ требует набора навыков $S(t_j) \subset S$, где $S(t_j) \neq \varnothing$.

Каждый агент $a_i \in A$ имеет набор навыков $S_i \subset S$.

Коалиция C состоит из агентов $C \subseteq A$ и $C \neq \varnothing$. Набор навыков у C - S(C), объединение навыков всех участников коалиции $S(C)=\cup S_i$.

Коалиция C может выполнить задание, если только обладает всеми нужными

навыками $S(t_j) \subseteq S(C)$.

Характеристическая функция v(C) – количество заданий, которые может выполнить коалиция C.

В статье [87] авторы рассматривают два способа решения такой задачи формирования коалиций агентов — на основе использования двоичного представления коалиций и последующего применения либо роевого интеллекта либо генетического алгоритма. Время работы предложенных алгоритмов для 20 агентов на несколько порядков меньше, чем время работы алгоритма *IDP*. В то же время анализ оптимальности полученного решения авторами не проводился.

Авторы [88] решали задачу *Coalitional Skill Game* для нескольких заданий. Каждое задание требует от агента наличие от одного до трех навыков. Каждый агент имеет от одного до трех навыков, поэтому агентам необходимо формировать коалиции. Коалиции формируются жадным образом: сначала выбираются самые близко расположенные агенты для выполнения текущего задания. Авторы оценивали количество сообщений, переданных при работе алгоритма. Анализ оптимальности полученного решения авторами не проводился. Работа алгоритма была проверена для 25 агентов. Для реализации предложенного алгоритма авторы использовали мультиагентную систему *JADE*.

Сравнение результатов точных и приближенных алгоритмов

Алгоритмы, основанные на полном переборе и выдающие гарантированно оптимальное решение, способны работать с наборами, не превышающими 30 агентов. Для оценки качества результатов, полученных приближенными алгоритмами, предлагается использовать следующие метрики [70].

Значение общего выигрыша. $\mathcal{G}^m = \frac{\sum_{C \in CS^m} v(C) - \sum_{a \in A} v(a)}{\sum_{a \in A} v(a)}$, где CS^m – коалиции, сформированные в результате работы алгоритма. Эта метрика позволяет оценить выигрыш от фомирования коалиций по сравнению с индивидуальными коалициями (каждая состоит из единственного агента). Преимущество этой метрики состоит в том, что нет необходимости получать результат работы

оптимального алгоритма для сравнения, что важно для количеств агентов выше 30.

Усредненное значение выигрыша. G^m / G^{opt}. Это соотношение общего выигрыша, полученного изучаемым алгоритмом, к общему выигрышу, полученному оптимальным алгоритмом. Эта метрика может применяться только для количеств агентов, не превышающих возможности оптимальных алгоритмов.

Усредненное соотношение с оптимальным. $V(CS^m) / V(CS^{opt})$. Вычисляет, на сколько значения сформированных изучаемым алгоритмом коалиций далеки от значений коалиций, полученных оптимальным алгоритмом. Это может применяться для не более 30 агентов.

Созданные к настоящему времени оптимальные алгоритмы способны к нахождению решения для коалиций, количество агентов в которых не превышает нескольких десятков. Основным недостатком приближенных методов решения задачи формирования коалиций агентов является то, что нет возможности оценить близость полученного решения к оптимальному без получения решения при помощи оптимального алгоритма. Кроме того, время нахождения решения при использовании некоторых алгоритмов, например, генетического, может очень сильно отличаться при различных начальных условиях.

1.4. Задача распределения ролей

Согласно работе [89], использование ролей агентов позволяет кластеризировать типы поведения агентов. В статье рассматриваются агенты, стремящиеся достичь общей цели. Понятие роли удобно использовать, поскольку они описывают более высокий уровень абстракции, позволяющий избежать рассмотрения низкоуровневых аспектов реализации и взаимодействия агентов.

Важность применения ролей определяется следующими соображениями.

Высокий уровень взаимодействия. Количество взаимодействий между агентами может быть уменьшено благодаря введению некоторой структуры группы, которая будет определять, какими ресурсами могут пользоваться агенты, какой информацией они должны обмениваться, какие цели должны быть достигнуты каждым из агентов для достижения общей групповой цели. Введение

ролей предоставит требуемый уровень абстракции для решения описанных выше аспектов.

Динамика окружающей среды. Для того, чтобы планировать достижение целей в условиях постоянно меняющейся обстановки, агенты должны абстрагироваться от низкоуровневых подробностей. Например, некоторый агент из группы может столкнуться с невозможностью выполнить назначенную ему цель из-за внезапного недостатка ресурсов. Поэтому агенты должны обладать исчерпывающей информацией, какую роль выполняет в группе каждый из агентов, и при каких условиях данному агенту может быть назначена некоторая роль. Более того, необходим механизм динамического присвоения ролей агентам. Попытка исправления сложившейся ситуации может включать изменения способа решения поставленной задачи или изменение состава и переконфигурация группы.

Распределенность. Для достижения поставленной задачи агенты должны сформировать группу с общими целями, взаимодействовать между собой для получения обобщенного видения окружающей обстановки и интегрального видения состояния и возможностей всех агентов, входящих в группу. Полная спецификация ролей, их отношений и взаимосвязи между собой позволит иметь исчерпывающую информацию для группы агентов о состоянии всех участников группы и выполняемых ими действиях. Требуется, чтобы агенты могли эффективно управлять ролями, определять количество и типы ролей, которые нужно использовать в рамках группы, количество агентов, которые будут играть каждую роль. При использовании ролей необходимо обеспечить динамическое назначение ролей агентам, при этом сами роли в процессе работы меняться не могут, но они могут назначаться и открепляться от конкретных агентов.

Авторы отмечают, что на момент написания работы [89] методологии мультиагентных систем (а именно, *AAII* [90], *Gaia* [91] and *MaSE* [92]) не имели прямой реализации понятия ролей, а их применение планировалось на интуитивно понятном уровне.

В системе *MaSE* роли использовались на этапе анализа, и заменялись на классы агентов на уровне разработки.

В системе *Gaia* роль определяется 4 атрибутами: ответственность, разрешения, действия и протоколы. Роли определяются соответствующими схемами, но в момент развертывания системы они уже не участвуют.

В методологии *AAII* используется модель *BDI*. Роли не используются в финальной версии создаваемой системы, но участвуют в проектировании, при описании ролей и их взаимодействия между собой.

AALAADIN является метамоделью, для описания наборов агентов она использует понятия агента, группы и роли. Группа состоит из конечного множества ролей, которые обрабатываются некоторыми агентами. Группа создается любым агентом, который автоматически принимает роль менеджера группы. Менеджер группы обрабатывает запросы других агентов на вступление в группу или на назначение определенной роли. В этой модели роль — абстрактное представление функции, службы или идентификации агента внутри группы. У агента может быть несколько ролей, в каждой роли могут участвовать несколько агентов. Отличительной особенностью системы является то, что агент может создавать новые группы с ролями, таким образом, структура системы может изменяться динамически.

В работе [93] описан подход к контролю поведения агентов, причем в случае несоответствия нормам агенты могут быть подвергнуты санкциям. Он основан на указании прав, ответственности и санкций в виде, который позволяет переназначать их динамически во время выполнения программы. В мультиагентой системе указывается нормативная база, которая реагирует на некоторые действия агента или на некоторое событие.

Агенты делятся на внутренних, которые уже ограничены в правах, и внешних, которые могут быть созданы в другой системе и не знают о нормах этой системы. Внешним агентам при регистрации назначается базовая роль. В результате действий внешнего агента его роль может быть изменена системой (например, ограничен доступ к некоторому ресурсу).

Роли внешних агентов изменяются: при входе и выходе их из системы; при выполнении агентом некоторого действия; при наступлении некоторого события;

при срабатывании ограничения по времени.

Для динамического назначения ролей системы выполняет следующие шаги:

1. Нормативная база данных задается разработчиками системы. Она описывает все права, обязанности и ответственность для каждой из ролей;

2. Система присваивает каждому вошедшему в систему агенту некоторую роль на основе его начальных действий. Например, для участника аукциона может быть применена роль продавец или покупатель;

3. Система изменяет роль у агента, если выполняется некое предусловие. Если агент выполняет действие, которое может изменить его роль в системе, система незамедлительно меняет его роль;

4. Сразу после выполнения действия или наступления события и в случае, если выполнены условия нормы, которая будет соответствовать роли, то конкретные права и обязанности, связанные с нормой, применяются ко всем агентам, имеющим эту роль.

Система, реализующая этот подход, состоит из следующих модулей: переводчик норм, обработчик событий и действий, запись событий, анализатор, репортер прав и обязанностей, активатор, деактиватор, таймер.

Ниже рассмотрен процесс динамического присвоения ролей.

Перед запуском создается нормативная база ролей. Описание ролей задаются в синтаксисе, используемом в системе *Jess*. Если синтаксис задания ролей отличается, то запускается модуль-переводчик норм, переводящий из пользовательского представления в синтаксис *Jess*. После запуска системы в нее начинают добавляться внешние агенты. За их действиями следит обработчик событий и действий, который фиксирует все действия. Затем обработчик событий на основе базы фактов и базы правил сохраненных в формате *Jess* классифицирует событие и выдает нормативные команды анализатору. Затем нормативная база фактов с использованием активатора, деактиватора и таймера дополняется.

Динамическое присвоение ролей происходит в следующих случаях:

1. Вход или выход агентов из системы. Агенту присваивается новая роль, происходит динамическое присвоение прав и обязанностей и нормы, связанные с

новой ролью, применяются к агенту;

2. При совершении агентом определенного действия. Например, агент делает ставку на аукционе, таким образом, он получает право на получение ответа от аукциониста;

3. Возникновение определенного события. Например, агент, участвующий в аукционе, получает право купить вещь, если он предложил более высокую цену по сравнения со всеми остальными агентами;

4. Наступление заранее установленного времени. Например, наступление конца аукциона.

В статье [94] предпринята попытка описать метамодель системы для работы с ролями. Роль — пожелания, как должен действовать агент, но не четкие указания. У агента может быть несколько ролей. Построена система диаграмм классов, описаны свойства объектов и соотношения между ними. Поскольку назначение ролей должно происходить динамически на этапе выполнения программы, то требуются дополнительные механизмы для реализации этого. Авторы называют их: административные свойства (как идентифицировать роли и отличить одну от другой), ответственность (роль содержит список действий, которые может выполнять агент, принявший на себя эту роль) и потребности (список способностей, разрешений, ресурсов и сервисов, которые являются необходимыми для того, чтобы некоторый агент мог достичь поставленной цели данной роли). Описан механизм репозитория ролей, позволяющий использовать роли повторно и процесс принятия агентом выбранной роли. Модель хорошо проработана, но программного применения ее нет.

В работе [95] описывает создание агентов на *Java* и использование для работы с ролями изменения байт-кода программы. Роль можно рассматривать как интерфейс для взаимодействия, предоставляющего базовые функции. Для создания ролей используется объектно-ориентированный подход, возможно наследование ролей. Каждая роль представляет собой набор поведений и способностей, которые агент может использовать для достижения поставленный целей в выбранном контексте. Роли нужно менять динамически во время

выполнения программы, таким образом принимая на себя набор поведений, способностей и знаний, необходимых для выполнения поставленной задачи. При этом другие агенты должны видеть изменение роли текущего агента и использовать полученную информацию, например, блокируя некоторые действия агента, роль которого не допускает выполнение этих действий. У каждого агента может быть несколько ролей. Выбран язык программирования *Java*, который позволяет создавать агенты единообразно для нескольких вычислительных платформ, просто обеспечивать миграцию агентов между различными платформами. Требуется обеспечить множественное наследование (одновременно от нескольких базовых классов), но такую возможность язык программирования *Java* не предоставляет. Поэтому предложено определить роль, как состоящую из двух компонентов — класса роли, который задает поведение роли и интерфейса роли, который позволяет получать информацию от других агентов. Когда агент принимает на себя роль, ему необходимо реализовать ее интерфейс и в последующем сменить его в случае смены роли. Поскольку в языке программирования *Java* такого механизма нет, то авторы реализовали это путем программного изменения байт-кода класса соответствующего агента.

Дальнейшим развитием подхода является статья [96], способ реализации такой же – изменение байт-кода, полученного после работы интерпретатора программы. В работе вводятся роли двух типов: видимая роль (остальные агенты только видят) и публичная роль, в ней агент может предоставлять услуги другим агентам. Агенты могут получить информацию о ролях других агентов тремя способами: запросом к интересующему агенту, широковещательной рассылкой агента остальным о смене роли и прямым наблюдением того, экземпляром какого класса роли этот агент является.

1.5. Развитие способностей в результате деятельности

Один из подходов, изучающих развитие способностей агентов в результате деятельности на базе психологических соображений, основан на использовании знаковой картины мира. Подход заключается в представлении знаний об объектах,

отношениях и действиях с ними в виде знаков – именованных структур, хранящих декларативные и процедурные знания об объекте или процессе. В состав знака входят: образ (существенные признаки, характеризующие описываемый объект или процесс), значение (обобщенные знания о внешней среде и о возможных методах использования объектов) и личностный смысл (информация об индивидуальном опыте взаимодействия с объектом, как удачном, так и неудачном). С использованием знаковой картины мира разработаны механизмы приобретения знаний об объектах и процессах, на основе этих механизмов решены задачи целеполагания и планирования поведения. Далее при описании этого подхода используется терминология и обозначения из [97].

Формально, знак представляет собой четверку $s = \langle n, p, m, a \rangle$, такую, что:

$n \in N$, где N – множество слов конечной длины в некотором алфавите, множество имен;

$p \in P$, где P - множество замкнутых атомарных формул языка исчисления предикатов первого порядка, множество свойств;

$m \in M$, M - множество значений;

$a \in A$, A – множество личностных смыслов.

Множества значений и множество личностных смыслов состоят из действий, каждое из которых представляется в виде правила (Формула 1). В множестве значений хранятся общие возможные действия с объектами внешнего мира, в множестве личностных смыслов — действия с этими объектами, характерные для конкретного субъекта, полученные в результате личного опыта взаимодействия с внешними объектами.

На множествах образов, значений и личностных смыслов определяются следующие отношения и операции (функции).

На множестве образов: отношения эквивалентности, включения, сходства и противопоставления; операция обобщения.

На множестве личностных смыслов: отношения поглощения, противопоставления; операция агглютинации (присоединения).

На множестве значений: отношения эквивалентности, сходства и сценарное.

На множестве элементарных знаков определяется R – семейство бинарных отношений, F – множество функций.

Тогда алгебраическая система $W = < S^+ , R , F >$ - картина мира субъекта I, если:

$S = \{<n, p, m, a>\}$, где $s = <n, p, m, a>$ - элементарный знак,

$R= \{R_1, R_2,\ldots,R_8\}$ и для каждого $i=1,2,\ldots, 8$, $R_i \subseteq S\times S$; $(i\neq j)\rightarrow (R_i \cap R_j) =\varnothing$;

$F= \{F_1, F_2, F_3\}$, где F_1 и F_2 – бинарные операции на S, F_3 – унарная операция на S^+, такие что:

F_1: $S \times S\rightarrow S$ так, что $F_1(s_1, s_2)\rightarrow s_3$, $(s_1, s_2, s_3) \in S$;

F_2: $S \times S\rightarrow S$ так, что $F_2(s_1, s_2)\rightarrow s_3$, $(s_1, s_2, s_3) \in S$;

F_3: $S^+\rightarrow S^+$ так, что $F_3(s_1)\rightarrow s_2$, $(s_1, s_2) \in S^+$.

Функция $F_3^*(s) = F_3(F_3...(F_3(s))\ldots)$, где $s\in S$, - замыкание на множестве знаков S^+, порожденный ею знак – сценарий, а знак s – опорный знак сценария.

Элементы множества S – элементарные знаки. Каждый знак имеет имя n, образ p, значение m, и личностный смысл a.

На множестве знаков определены отношения из семейства R, порожденные компонентами знаков. Образами знаков порождены отношения R_1 - R_4 (отношения эквивалентности, включения, сходства и противопоставления, соответственно); значениями - отношения R_5 (сценарное) и личностными смыслами – отношения R_7, R_8 (поглощения и противопоставления, соответственно) [98].

Для представления фактов и действий в знаковом подходе используется специализированная структура — каузальная матрица. Она представляет собой матрицу, в которой строки задают набор признаков, а единицы, проставленные в столбцах, отмечают те признаки, совокупность которых позволяет распознать и идентифицировать характерные признаки некоторого объекта или процесса. Признаки могут быть как простыми (представляющие статические объекты), так и процедурными (представляющие некоторые процессы или действия), в этом случае порядок столбцов в каузальной матрице обозначает

причинно-следственные связи, последовательность событий при выполнении определенного действия.

На основе знакового подхода было реализовано построение плана действий субъекта поведения [99]. Был предложен иерархический алгоритм *MAP*, строящий план от целевого состояния к начальному. Работа алгоритма на каждом шаге построения плана состоит из четырех стадий.

На первой, S-стадии, находятся прецеденты для выполнения действий в текущей ситуации. Просматриваются личностные смыслы всех знаков. Если текущие условия удовлетворяют каузальной матрице некоторого личностного смысла выбранного знака, то список прецедентов пополняется личностным смыслом этого знака.

На второй, M-стадии, происходит поиск у всех знаков, входящих в ситуацию, применимых действий на сети значений. Компоненты полученного множества каузальных матриц используются как начальные точки для распространения активности по сети значений.

На третьей, A-стадии, выполняется создание личностных смыслов для найденных значений и создание конкретного действия на их основе с использованием эвристики. Распространение активности по каузальной сети значений приводит к активации множества значений знаков, связанных с процедурной матрицей. Затем создаются новые процедурные каузальные матрицы на сети личностных смыслов, в которые копируются значения исходных, с подстановкой конкретных знаков вместо абстрактных ссылок. Затем из созданных процедурных матриц выбираются только применимые в текущей ситуации (эффекты которых включены в ситуацию). Запускается эвристическая проверка, которая выбирает действия, быстрее всего ведущие к достижению целевых условий.

На заключительной, P-стадии, строится новая ситуация, с использованием предусловий от найденных действий. Затем, если начальная (целевая) ситуация планирования является подмножеством полученной новой ситуации, то план построен и алгоритм заканчивает работу.

В результате работы алгоритма строится план длиной n, состоящий из пар $<z_{si}, z_{pi}>$, где z_{si} – каузальная матрица некоторого узла на сети личностных смыслов, соответствующего i-й ситуации планирования, z_{pi} – каузальная матрица некоторого личностного смысла, представляющего действие, которое предлагается предпринять в ситуации z_{si}. Ситуация z_{si+1} в этом случае – результат предпринятого действия z_{pi}.

Схема работы алгоритма *MAP* приведена на Рис. 7.

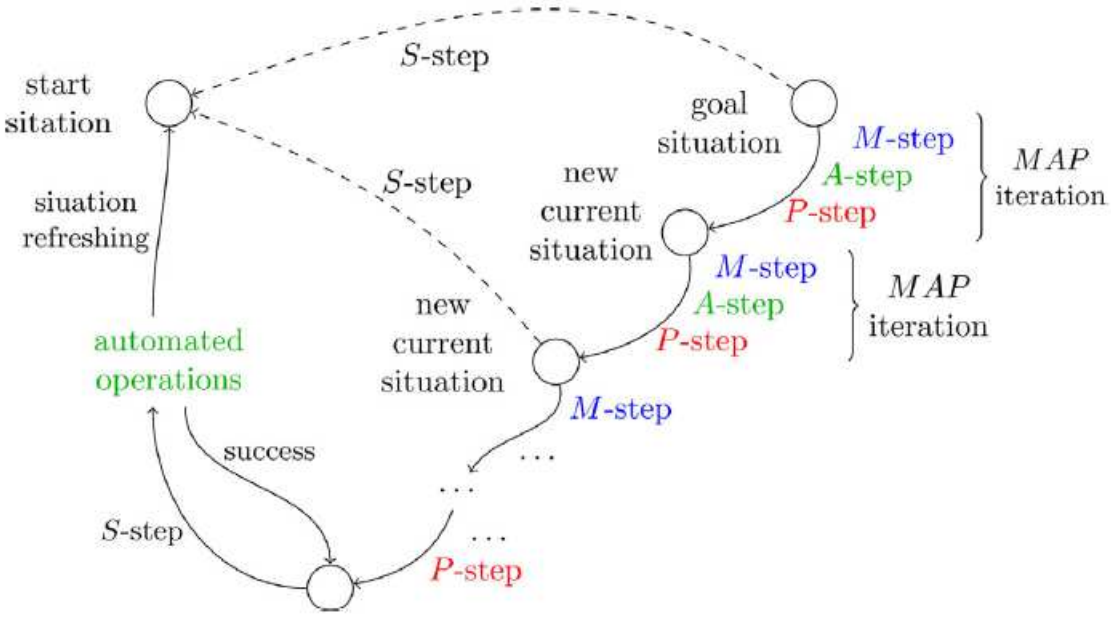


Рис. 7. Схема процесса планирования с использованием алгоритма MAP [99]

Предложенный алгоритм был использован для решения задачи построения планов поведения в классических доменах «*Block world*» и «*Logistics*».

Задача распределения ролей в коалиции когнитивных агентов

Рассмотренный знаковый подход был применен для решения задачи распределения ролей в коалиции когнитивных агентов [100]. В статье описывается применение знакового подхода к планированию действий нескольких агентов, обладающих знаковой картиной мира, с использованием описанного алгоритма *MAP*. В дополнение к обычным знакам у каждого агента используются

специализированные знаки «Я» и «Они». Первый представляет собственные знания и умения (предпочитаемые действия, т.е. личностные смыслы) агента, второй — информацию о знаниях и умениях всех остальных агентов, полученную от них. На каждом шаге планирования каждый из агентов принимает решение, будет ли он выполнять действие сам или будет делегировать выполнение этого действия другим агентам, которые справятся с поставленной задачей лучше него (оценивая действия и сравнивая результаты знаков «Я» и «Они»). После того, как каждый агент построил собственный план, выделенный агент-координатор проводит согласование планов агентов и построение единого плана действий на основе выбора каждого шага плана на аукционе между агентами-исполнителями.

В работе [98] предложен еще один алгоритм динамического распределения ролей в коалиции когнитивных агентов. В этом случае используется уже построенный план по достижению целевого состояния из начального. Для выполнения каждого шага плана требуется выбрать наиболее предпочтительного агента-исполнителя, то есть такого, который справится с этим лучше других агентов. Выбор наиболее предпочтительного агента осуществляется с использованием специализированных знаков «Я» и «Они» в заранее сформированной картине мира каждого агента. В работе предложено использовать интегральную оценку успешности выполнения выбранного действия некоторым агентом, которая является произведением значения оценки успешности при выполнении этим агентом выбранного действия и значения успешности или надежности самого агента, присвоенная ему по результатам его предыдущих действий. Более подробно этот алгоритм и его программная реализация рассмотрены в п.2.3.3 настоящей работы. В дальнейшем, в настоящей работе подход на основе знаковой картиной мира не применяется, а лишь наследуется его идея об изменении характеристик агентов – в процессе выполнения действий агент увеличивает степень успешности выполнения этих действий, изменяя соответствующие оценки успешности.

1.6. Выводы

В настоящей главе рассмотрены различные варианты задач маршрутизации, известные способы их решения и стандартные наборы входных данных задачи *SDVRP*, для которых найдены оптимальные решения. Решение задач маршрутизации представляет значительный практический интерес.

Недостатками известных подходов к решению задач маршрутизации являются: невозможность практического моделирования систем высокой размерности, затрудненность или невозможность их применения в условиях, когда значения потребности пунктов назначения и доступное количество продукции у источников могут меняться со временем.

Рассмотрены свойства и способы создания интеллектуальных программных систем, основанных на правилах, и мультиагентных систем. Описана структура типовой современной мультиагентной системы *SPADE*, рассмотрены ее области применения.

Приведено описание и основные свойства известных алгоритмов формирования групп агентов и распределения их ролей. Рассмотрены подходы к улучшению характеристик и распределению ролей агентов: они обладают большими возможностями для создания интеллектуальных агентов, способных решать задачи целеполагания и планирования поведения.

Мультиагентный подход характеризуется децентрализацией, возможностью изменения количества и свойств участвующих в моделировании агентов. С его помощью решаются задачи динамического распределения нагрузки в системах «умных» электрических сетей, прогнозирования развития населенных пунктов, другие задачи. Предложенное использование систем, основанных на правилах, для реализации агентов позволяет повысить гибкость в поведении агентов, легко их модифицировать и дает возможность отделить программный код от описания действий, определяющих поведение. Подход, основанный на улучшении характеристик агентов в результате выполнения агентами действий, может быть применен для того, чтобы результаты получаемых приближенных решений стали ближе к оптимальным.

Перспективным является применение комбинации мультиагентного подхода, использования системы правил с целью описания действий агентов и улучшения характеристик агентов в результате деятельности для решения задач маршрутизации в условиях изменения количества и характеристик агентов в процессе моделирования. Благодаря тому, что у задач маршрутизации известны стандартные наборы входных данных, для которых установлены результаты оптимального решения, становится возможным оценить близость полученных решений к оптимальным.

Глава 2. Разработка алгоритмов для решения задач маршрутизации

В главе предложены и изучены алгоритм формирования коалиций агентов, алгоритм распределения предпочитаемых действий агентов, обладающих способностью к изменению оценок успешности, и алгоритм формирования коалиций агентов с учетом этих оценок. Описан предлагаемый способ представления правил и механизм задания правил и ввода начальных данных для агентов.

2.1. Используемые в работе определения

В настоящей работе используются следующие определения.

Агент – программная сущность, обладающая свойствами автономности, реактивности и целенаправленности. Основа взаимодействия агентов между собой – передача сообщений и реакция на полученные сообщения.

Мультиагентная система – специализированное программное средство, позволяющее создавать агентов с требуемым набором правил поведения, модифицировать их, завершать выполнение конкретного агента или типа агентов; обеспечивающее передачу сообщений между агентами; выполняющее служебные функции, например, оповещение о активности конкретного агента или типа агентов, отладочная информация и т.д.

Коалиции агентов – набор или группа агентов, возможно обладающих различными свойствами, действующие сообща для достижения поставленной общей цели. Задается характеристическая функция, которая каждой коалиции ставит в соответствие числовое значение – вес данной коалиции.

Интеллектуальный агент – программная сущность, обладающая свойствами автономности, общественного поведения, реактивности, про-активности и способная изменять свои свойства или приобретать новые в процессе деятельности. В частности, обладающая способностью к изменению оценок успешности.

Интеллектуальный транспортный агент – интеллектуальный агент,

применяемый для решения задач маршрутизации, дополнительно обладающий возможностью перемещения и атрибутами скорости и грузоподъемности.

Формирование коалиций агентов – процесс объединения нескольких индивидуальных агентов в группу таким образом, чтобы они сообща были способны к выполнению некоторого задания, которое не может быть выполнено каждым агентом по отдельности. В том случае, если индивидуальные агенты или их коалиции обладают несколькими компетенциями, то есть могут выполнять различные задания с разной эффективностью, то возникает задача распределения предпочитаемых действий – какому агенту или какой коалиции агентов поручить выполнение выбранного задания для обеспечения наибольшей эффективности (например, с наименьшими затратами ресурсов, наименьшим временем выполнения, наибольшей вероятностью успеха и т.д.).

Решение задач формирования коалиций агентов и распределения предпочитаемых действий представляет значительный практический интерес и основано на использовании точных переборных или приближенных алгоритмов.

2.2. Общая постановка модельной задачи

В качестве модельной рассматривается задача маршрутизации. При этом в зависимости от количества транспортных агентов – источников (пунктов отправления) и пунктов назначения и соотношения потребностей пунктов назначения и грузоподъемности источников имеют место либо классическая транспортная задача (задача Монжа-Канторовича), либо мультитранспортный вариант классической задачи маршрутизации (*SDVRP*).

Пусть заданы:

1. Источники A_i с количеством продукции (грузоподъемностью) a_i, где a_i – целые числа, i=1..m. *M** - множество всех источников. Продукция является однородной.

2. Пункты назначения B_j с потребностью в продукции b_j, где b_j – целые числа, j=1..n. *N** - множество всех пунктов назначения.

3. Матрица стоимости $c_{ij}>0$ перевозки из пункта i в пункт j. Не ограничивая общности, считается, что скорости передвижения транспортных агентов - источников одинаковы. Таким образом, стоимость перевозки пропорциональна расстоянию между пунктами.

Требуется найти количество продукции $x_{ij}\geq 0$, перевозимое из пункта i в пункт j, такое что: $\Sigma_i\Sigma_j\, c_{ij}\cdot x_{ij} \rightarrow \min$ (минимизация общей стоимости перевозки).

В ***случае I***, если:

$m \geq n,$ (I.4)

$a_i < b_j$, для всех i и j, (I.5)

$\Sigma_i\, a_i \leq \Sigma_j\, b_j,$ (I.6)

то имеет место классическая транспортная задача. В классической транспортной задаче положение всех источников и пунктов назначения фиксировано и не меняется с течением времени. В случае, если количество имеющейся и требуемой продукции равны (сбалансированы), транспортная задача является задачей линейного программирования. Известно несколько алгоритмов оптимального решения транспортной задачи, минимизирующих суммарную стоимость перевозок [101]. Таким образом, значение суммарной стоимости перевозок для каждых начальных данных, полученное оптимальным алгоритмом, может служить эталоном при сравнении качества решений, полученных другими алгоритмами для этих же начальных данных. Подобный подход оценки свойств изучаемых алгоритмов широко применяется, например, в работах [70, 102].

Поскольку количество продукции у каждого источника меньше потребности любого пункта назначения (условие в п. I.5), то возникает задача формирования коалиции источников для удовлетворения потребности каждого пункта назначения. Источник A_i с наибольшим количеством продукции a_i выбирает пункт назначения B_j, для которого стоимость перевозки c_{ij} минимальна. Этот источник становится ведущим, он упорядочивает список остальных источников по стоимости перевозки в этот же пункт назначения B_j. Из этого списка ведущий выбирает k первых элементов, сумма количества продукции у которых достаточна для удовлетворения

потребности пункта назначения B_j - они являются коалицией для пункта назначения B_j. Среди оставшихся источников снова выбирается ведущий и процесс продолжается. Алгоритм останавливается, если удовлетворены потребности всех пунктов назначения, или сумма количества продукции у оставшихся источников недостаточна для удовлетворения потребности ни одного пункта назначения.

В случае, если значения a_i и b_j изменяются с течением времени, то модельная задача является NP-полной [103].

В ***случае II***, если:

$m < n$, (II.4)

a_i одинаковы и равны Q для всех i, $Q > b_j$ для всех j, (II.5)

$\Sigma_i a_i \geq \Sigma_j b_j$, (II.6)

все источники в начальный момент времени находятся в одном пункте

и после исчерпания продукции должны вернуться в этот пункт, (II.7)

то имеет место мультитранспортный вариант классической задачи маршрутизации (*SDVRP*). Эта задача является NP-полной [18]. Для этой задачи известны более чем 80 стандартных наборов входных данных, для некоторых из них (для большинства наборов входных данных с количеством до 50 агентов, а также для одного набора с 75 агентами и одного набора с 100 агентами) установлены результаты оптимального решения [21]. В этом случае для сравнения качества решений, полученных предложенными в настоящей работе алгоритмами, с результатами, полученными оптимальным алгоритмом для этих же начальных данных, достаточно сравнить полученное значение суммарной стоимости перевозок с установленным значением.

2.3. Метод решения переборных задач. Лингвистическая геометрия

Лингвистическая геометрия – метод решения переборных задач большой размерности, который первоначально был предложен для анализа и планирования решения шахматных задач [104, 105]. Основная идея метода - сведение сложной системы к иерархии подзадач-ступеней. Каждая ступень моделируется своим

формальным языком, при этом цепочка языка некоторого уровня определяет терминальный символ языка более высокого уровня.

В ранних работах [104, 105] использовались три уровня языков: язык траекторий, семейство траекторий и язык зон.

В общем виде сложная система определяется как: точки, элементы, отношение достижимости, начальное и конечное состояние системы и описание операторов перехода из одного состояния в другое.

Правила перехода на каждом уровне языка задаются недетерминированной формально-параметрической грамматикой. В процессе вывода к текущей цепочке применяется каждая из множества допустимых продукций. В результате применения образуется новая цепочка и новое множество допустимых продукций. Вывод для каждой из цепочек, полученных из данной, происходит независимо, таким образом происходит построение дерева решений. Процесс вывода останавливается, если допустимые продукции применить нельзя. В случае, если полученная цепочка состоит только из терминальных символов, то она поступает во множество результатов вывода, в противном случае – отбрасывается.

В более поздних работах [106] определяется понятие контролируемой грамматики как следующей структуры:

$G = (V_T, V_N, V_{PR}, E, H, Parm, L, R)$,

где

V_T – алфавит терминальных символов;

V_N – алфавит нетерминальных символов;

V_{PR} – алфавит исчисления предикатов первого порядка, $V_{PR} = Truth \cup Con \cup Var \cup Func \cup Pred \cup \{Символы\ логических\ операций\}$,

Здесь

Truth – зарезервированные символы T (истина) и F (ложь);

Con – символы констант;

Var – символы переменных;

Func – функциональные символы, $Func = F_{con} \cup F_{var}$. Терм может быть константой, переменной или функцинальным выражением.

Функциональное выражение – имя функции арности k и набор из k термов, заключенных в скобки и разделяемых запятой;

Pred – предикатные символы. Атом – предикат арности k и набор из k термов, заключенных в скобки и разделяемых запятой. Символы T (истина) и F (ложь) – также атомы. Формулы – атомы и последовательности атомов, связанные логическими операциями.

E – множество, описывающее предметную область;

H – отображение V_{PR} на множество E;

Parm – отображение $V_T \cup V_N$ на 2^{Var}, связывающее с каждым символом из алфавита $V_T \cup V_N$ множество формальных параметров, Parm(S)=Var;

L – конечное множество меток;

R – конечное множество продукций, каждая из которых имеет вид:

$(l, Q, A \rightarrow B, \pi_k, \pi_n, F_T, F_F)$

где

l – метка продукции из множества L, разные продукции имеют разные метки;

Q – формула предикатов из V_{PR}, условие применимости продукции;

$A \rightarrow B$ – ядро продукции, A из V_N, B из $(V_T \cup V_N)$;

π_k – последовательность функциональных формул, соответствующих каждому вхождению символов из $V_T \cup V_N$ в строки A и B;

π_n – последовательность функциональных формул, соответствующих каждому функциональному символу из Fvar;

F_T – набор меток, допустимых для следующего шага вывода, если Q=True;

F_F – набор меток, допустимых для следующего шага вывода, если Q=False.

Вывод происходит следующим образом. Символ S является начальным символом для вывода, в качестве допустимого набора продукций берется все множество L. К текущей строке применяются все продукции из множества допустимых на этом шаге, в которые входит символ A. В результате применения

продукции формируется новая строка и новое допустимое множество. Дальнейший вывод для каждой полученной строки происходит независимо от других.

В случае, если ни одна из продукций из множества допустимых не может быть применена, то вывод для данной строки прекращается. Если полученная строка состоит только из терминальных символов, то она помещается во множество полученных результатов, в противном случае строка более не рассматривается. Множество полученных результатов образует дерево решения.

Продукции применяются следующим образом. Выбирается самое первое вхождение символа A в строке. Вычисляется значение предиката Q. Если Q = False, то допустимым множеством становится F_F и применение продукции прекращается. Если Q = True, то символ A заменяется на строку B, происходит вычисление значений всех формул из π_k, соответствующих параметрам символов, и параметры принимают полученные значения. Допустимым множеством становится F_T, применение продукции прекращается.

В качестве примера грамматика языка траекторий [106] применена для построения пути агента-источника из модельной задачи. Входными параметрами являются пары координат, представляющие текущее и целевое положения агента, а также количество тактов дискретного времени, за которое агент должен достичь целевое положение. В результате применения грамматики языка траекторий к такому агенту в качестве вывода могут быть сформированы несколько путей.

L	Q	Ядро	F_T	F_F
1	Q_1	S(p,p_goal,v) → A(p,p_goal,v)	two	Ø
2_i	Q_2	A(p,p_goal,v) → a(p) A($next_i$(p,v), p_goal, f(v))	two	3
3	Q_3	A(p,p_goal,v) → a(p_goal)	Ø	Ø

В этом случае:

V_T = {a};

V_N = {S, A};

V_{PR}

Pred = { Q_1, Q_2, Q_3};

Var = {p, p_goal, v};

F = F_{con} ⋃ F_{var};

F_{con} = {f, $next_1$, .. $next_n$} n=|X|

f(v) = v-1, D(f) – положительные целые числа

$next_i$ – специальная функция, которая на основе текущего положения агента, допустимого для него характера движения и возможной недостижимости некоторых пар координат (таким образом описаны препятствия) вычисляет его допустимое положение на следующем шаге

F_{var} = {p_0, v_0};

E = Z_+ ⋃ X ⋃ P;

Parm: S→Var, F→Var, a→{x};

L = {1, 3} ⋃ two, two={2_1, 2_2, .. 2_n}.

Начальное состояние: p=p_0, v=v_0, p_0 из X, v_0 из Z_+.

Для количественной оценки степени ветвления получаемого дерева решений в [104] используется числовое значение показателя ветвления B, задаваемое формулой: $B+B^2+..+B^L=T$, где L – длина максимального поддерева, T – общее количество узлов в дереве решений. Предложенный метод решения переборных задач строит узкие направленные деревья решений – например, при анализе различных шахматных партий значения показателя ветвления B составляли от 1,34 до 1,05, что свидетельствует о высокой эффективности рассмотренного метода [107].

2.4. Алгоритмы формирования коалиций и распределения предпочитаемых действий агентов, способных к изменению оценок успешности

В настоящей работе автором предложена новая модель решения

транспортной задачи и задачи маршрутизации, имеющих большие размерности, основанная на применении интеллектуальных агентов, как не использующих, так и использующих оценки успешности. Задача решается путем интеграции нескольких алгоритмов.

Алгоритм решения задачи формирования коалиций с использованием характеристической функции (Алгоритм 1) предложен автором. Алгоритм распределения шагов заранее созданного плана по исполнителям, которые способны изменять оценки успешности, т.е. улучшать свои способности в результате деятельности, (Алгоритм 2) схематично рассмотрен в работе [98]. Алгоритм 2 реализован в ходе диссертационного исследования. Алгоритм 3 является объединением Алгоритмов 1 и 2, в котором при формировании коалиций исполнители изменяют оценки успешности. Алгоритмы 1-3 являются централизованными.

Алгоритм формирования коалиций из агентов, не использующих оценки успешности, (Алгоритм 4) применим для решения задач маршрутизации. Алгоритм распределения шагов плана по исполнителям – интеллектуальным агентам в распределенной среде (Алгоритм 5) наследует идею оценок успешности. Алгоритм 6 решает целевую транспортную задачу на основе формирования коалиций интеллектуальных агентов, интегрируя Алгоритмы 4 и 5 с учетом применения оценок успешности. Оценки изменяются в результате многократного решения целевой задачи. Алгоритмы 4-6, их свойства и особенности реализации рассматриваются в главе 3. Алгоритмы 4-6 предложены автором, функционируют в распределенной среде и в настоящей работе реализованы с использованием мультиагентной платформы.

Взаимосвязь алгоритмов представлена на Рис. 8.

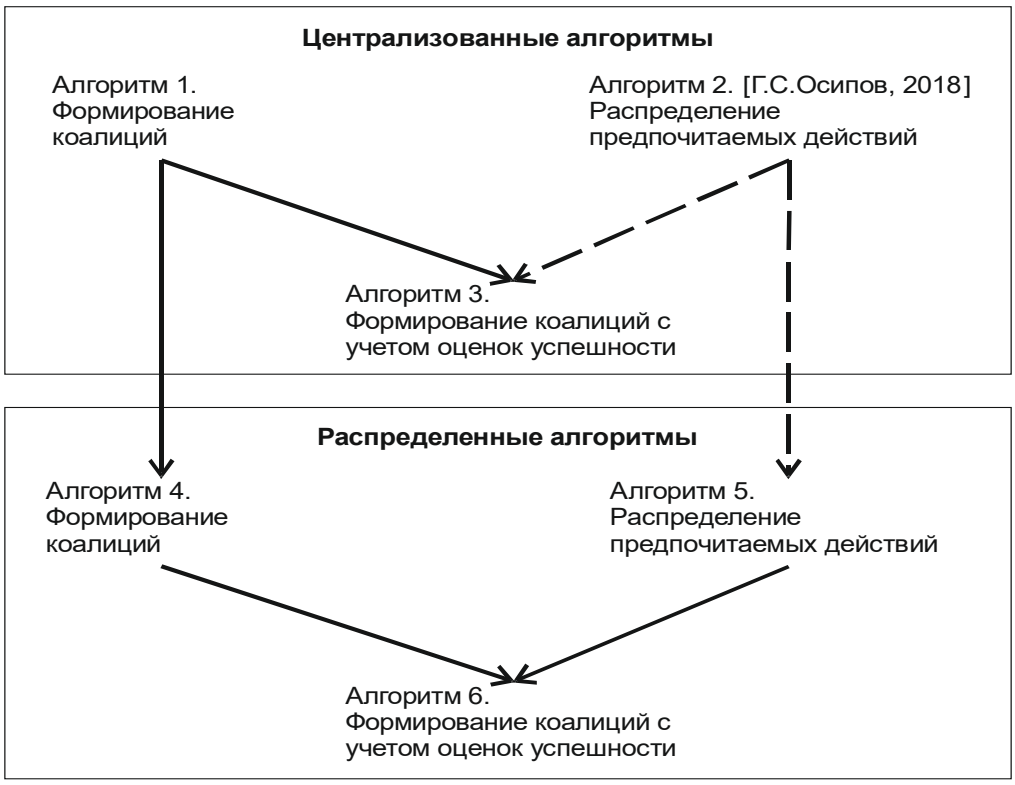


Рис. 8. Взаимосвязь предлагаемых в работе алгоритмов.

Алгоритмы, решая задачу маршрутизации, наследуют ее терминологию, а именно: оперируют понятиями «Пункт назначения» (*Factory*), «Источник» (*Truck*), «продукция», «стоимость перевозки». Кроме того, в Алгоритмах 2 и 5 используются понятия «Действие» (*Action*) и «Полезность» (*ActionValue*). Далее рассмотрен предлагаемый механизм добавления и модификации правил, задающих поведение агентов.

2.4.1. Механизм задания правил и начальных данных агентов

Для работы с системой правил, определяющих поведение агентов, требуется удобный механизм создания, изменения и удаления правил, задания начальных данных.

Модуль задания правил [108]

Предметная область представляется набором объектов. Фактом называется несколько свойств объекта, позволяющих однозначно его охарактеризовать для данной предметной области.

Информация о предметной области представляется структурой, состоящей из трех частей: шаблоны для описания фактов, указание фактов и указание правил.

Шаблон для описания фактов включает название и перечисление слотов (т.е. свойств), характерных для данного типа фактов. Наследование шаблонов задается ключевым словом «extends» и названием родительского шаблона. Предложенная система описаний шаблонов применяется при проверке правильности указания фактов.

```
<ШаблонФакта> ::= <ИмяШаблона> [extends <ИмяШаблона>] <СписокСлотов>
<СписокСлотов> ::= <Слот> | <Слот><СписокСлотов>
<Слот> ::= slot <ИмяСлота> [type=<ТипСлота>]
                                [default=<ЗначениеПоУмолчанию>]
<ТипСлота> ::= integer | float | string
```

Листинг 1. БНФ шаблона для описания фактов

Факт состоит из: названия, имени шаблона для описания факта, набора пар «название слота»-«значение слота для этого факта».

```
<Факт> ::= <ИмяФакта> <ИмяШаблонаФакта> <СписокЗнСлотов>
<СписокЗнСлотов> ::= <ЗначениеСлота> | <ЗначениеСлота> ; <СписокЗнСлотов>
<ЗначениеСлота> ::= <ИмяСлота> = <ЗначениеДанногоСлота>
```

Листинг 2. БНФ описания факта

Правило включает название, перечисление применяемых шаблонов фактов и основную часть: применимость правила (секция “where”) и набор действий (секция “do”). Подобное представление используется в экспертных системах [28].

```
<Правило> ::= <ИмяПравила> <СписокШаблонов> where <Условие>
                        do <Действия>
```

Листинг 3. БНФ описания правила

Секция <Условие> – логическое выражение, а <Действия> – добавление, удаление или изменения в свойствах фактов. Правило применимо только к фактам, шаблоны которых входят в <СписокШаблонов>. Описание предметной области транслируется в программу на языке *Python*. Указанный язык программирования был выбран исходя из его гибкости в описании данных и исполнении кода; кроме того, он был успешно применен при разработке экспертной системы [109].

2.4.2. Алгоритм формирования коалиций агентов

В алгоритме используются агенты двух типов – Пункты назначения и Источники. Для каждого Пункта назначения, имеющего потребность в продукции, формируется коалиция Источников, значение суммарного веса перевозимой продукции которых позволяет удовлетворить указанную потребность.

Исходными данными алгоритма являются:

Набор объектов B_j типа Пункт назначения (*Factories*) с требуемым весом продукции ($b_j>0$), j=1..n.

Набор объектов A_i типа Источник (*Trucks*) со значением веса перевозимой продукции ($a_i\geq0$), i=1..m.

Матрица стоимости $c_{ij}>0$ перевозки из пункта i в пункт j.

Получаемый результат:

Для каждого объекта типа Пункт назначения формируется коалиция объектов типа Источник или сообщение о невозможности создания коалиции.

Алгоритм 1. Алгоритм формирования групп агентов, централизованный.

Для каждого пункта назначения F выполняется следующее:

1. Для выбранного пункта назначения F и каждого источника T вычисляется расстояние по формуле $\sqrt{(pos_x(F)-pos_x(T))^2+(pos_y(F)-pos_y(T))^2}$ и добавляется в список вместе с весом продукции у источника.

2. Полученный список сортируется по увеличению расстояния. Для одинаковых расстояний элементы сортируются по уменьшению веса продукции у

источников.

3. К списку создаваемой коалиции добавляется первый элемент из отсортированного списка.

Поскольку по условию задачи требуемый вес продукции для каждого пункта назначения превосходит вес продукции, имеющейся у каждого источника, далее к списку создаваемой коалиции добавляются элементы из отсортированного списка до того момента, как суммарный вес источников в коалиции равен или превышает вес текущего пункта назначения, или элементов отсортированного списка не осталось.

4. В первом случае сообщается об успехе, во втором случает – о неуспехе.

5. Источники, попавшие в список, уменьшают количество имеющейся у них продукции (фактически, только у последнего могут остаться излишки, все предыдущие источники расходуют свою продукцию полностью).

Листинг 4. Псевдокод централизованного алгоритма формирования групп агентов

Исходные данные:
Набор объектов B_j типа Пункт назначения с требуемым весом продукции ($b_j>0$), j=1..n.
Набор объектов A_i типа Источник с перевозимым весом продукции ($a_i\geq0$), i=1..m.
Матрица стоимости $c_{ij}>0$ перевозки из пункта i в пункт j.
Получаемый результат: Для каждого Пункта назначения формируется коалиция объектов типа Источник или сообщение о невозможности создания коалиции.

В этом и следующих алгоритмах используется вспомогательная функция:

Function *MakeCoalition1 (C, CW, CC, T, MaxWeight):*
Input: C – список включенных в коалицию агентов, CW – вес коалиции,
CC – количество агентов в коалиции, T – указатель на текущий агент типа Источник, MaxWeight – минимально необходимый вес для коалиции
Output: C – обновленный список включенных в коалицию агентов,
CW – обновленный вес коалиции
while (*CW<MaxWeight*) **and** (*T<>Ø*) :
C.append(*T*) // текущий агент добавляется в список
$CW=CW+T_a$ // к весу коалиции добавляется вес этого агента
CC=CC+1 // размер коалиции увеличивается
$T=T_{Next}$ // переход на следующего агента
return *C, CW*

Основная часть Алгоритма 1:

for all $F \in N^*$:

// расчет расстояния от текущего Пункта назначения до всех Источников

for all $T \in M^*$:

$DistFT$=Distance(F,T)

// упорядочение по росту расстояния и уменьшению веса продукции

$SortedDist$=**Sort**($DistFT$, T_a)

// первый в списке агент – наиболее близкий с наибольшим весом

$Leader=SortedDist[0]$

$Coalition$.Append($Leader$)

$CoalitionWeight=Leader_a$

$Count=1$

// продолжаем формировать коалицию

MakeCoalition1($Coalition,CoalitionWeight,Count,Leader_{Next}$, F_b)

// проверка, сформирована ли коалиция

if ($CoalitionWeight \geq F_b$) **then print** "Коалиция есть "+$Coalition$

else print "Коалиция не найдена"

Для описанного алгоритма верно следующее.

Утверждение 1. Сложность Алгоритма 1 составляет O(n · (2m+m log m))

Действительно, для каждого пункта назначения выполняется следующее:

1. Расчет расстояний от каждого источника до текущего пункта назначения, сложность O(m)

2. Сортировка полученного списка методом стандартной сортировки Тима [110]. На больших списках она работает, как сортировка слиянием, а на маленьких фрагментах — как сортировка вставками. Сложность этого алгоритма составляет O(m log m) [111].

3. Проход по полученному списку, в самом худшем варианте сложность O(m)

Таким образом, для каждого пункта назначения O(m·(2+log m))

В целом для задачи O(n·m·(2+log m))

2.4.3. Алгоритм распределения предпочитаемых действий агентов

В настоящем разделе рассмотрено построение плана для коалиции агентов, каждый из которых способностью к изменению оценок успешности. Схема описываемого алгоритма представлена в работе [98] и кратко приведена ниже.

Коалицией агентов называется такая совокупность агентов, в которой: а) действия и их результаты доступны всем агентам; б) существуют общие цели поведения агентов; в) в достижении каждой цели участвует более одного агента [98].

В начале работы алгоритма выделенный агент-координатор или один из агентов опрашивает всех агентов и получает список действий, доступных для каждого агента. На основе полученной информации доступные действия всех агентов упорядочиваются по предпочтительности этих действий.

Затем агент-координатор строит план решения поставленной задачи, например, с использованием алгоритма *MAP*, описанного в п. 1.4. Если такой план удалось построить, то происходит переход к реализации плана. Каждый агент для каждого действия имеет некоторое число, сообщающее о предпочтительности этого действия. Кроме того, агент-координатор ведет реестр доступных агентов и хранит числовое значение, характеризующее их надежность или успешность при выполненни заданий.

В дополнение к описанной выше транспортной задаче у каждого источника $A_{i,}$ i=1..m заданы:

- значение «успешности агента» r_i, которое задает оценку успешности источника A_i при выполнении назначенных ему заданий. Оно может увеличиваться после участия в успешно сформированной коалиции.

- вектор пар <t:p>$_i$ вида <название задания>:<успешность при выполнении этого задания>, задающий, какие задания может выполнять источник A_i и с какой вероятностью успеха. Значение «успешности действия» может увеличиваться в результате успешного выполнения задания.

Исходные данные:

Требуемое действие (*Action*)

Минимально необходимое значение «успешности» (*ActionValue*)

Набор объектов A_i, i=1..m типа Источник (*Trucks*) с данными:

- значение «успешности агента» источника при формировании коалиций

(вещественное $r_i>0$)

- вектор пар (*ActionList*) <название задания>:<успешность при выполнении этого задания>, где «название» – строчная переменная, а «успешность действия» – вещественное *ActionVal*>0.

Получаемый результат: коалиция объектов типа Источник или сообщение о невозможности создания коалиции.

Алгоритм 2. Алгоритм распределения ролей агентов, централизованный.

При формировании коалиции, предназначенной для выполнения некоторого действия с определенным весом, происходит следующее.

Для выбранного действия и каждого источника T, способного к выполнению требуемого действия, предпринимаются шаги:

1. Создается список из всех таких источников вместе со значением успешности выполнения конкретным источником выбранного действия, которое вычисляется как $N_{t(xi)} = r(x_i)\bullet p(x_i)$.

2. Полученный список сортируется по убыванию.

3. Из полученного отсортированного списка последовательно выбираются его элементы, начиная с первого, до того момента, пока суммарная успешность источников в коалиции не станет равна или превысит требуемую успешность для выбранного действия, или элементов отсортированного списка не останется.

4. В первом случае сообщается об успехе, во втором случае – о неуспехе.

Листинг 5. Централизованный алгоритм распределения предпочитаемых действий агентов

Исходные данные:
Требуемое действие (Action)
Минимально необходимое значение «полезности» (ActionValue)
Набор объектов A_i, i=1..m типа Источник (Trucks) с данными:
- значение «полезности» источника при формировании коалиций (вещественное $r_i>0$)
- вектор пар (ActionList) <название задания>:<способность к выполнению этого задания>, где «название» – строчная переменная, а «способность к выполнению» – вещественное ActionVal>0.

Получаемый результат: коалиция объектов типа Источник или сообщение о невозможности создания коалиции.

В этом и следующих алгоритмах используется вспомогательная функция:

Function *MakeCoalition2 (C, CW, CC, T, MaxWeight):*

Input: C – список включенных в коалицию агентов, CW – вес коалиции,
CC – количество агентов в коалиции, T – указатель на текущий агент типа Источник, MaxWeight – минимально необходимый вес для коалиции

Output: C – обновленный список включенных в коалицию агентов,
CW – обновленный вес коалиции

while (*CW<MaxWeight*) **and** (*T<>Ø*) :
C.append(*T*) // текущий агент добавляется в список
$CW=CW+T_R*T_{ActionVal}$ // к суммарной «полезности» коалиции добавляется «полезность» текущего агента
$CC=CC+1$ // размер коалиции увеличивается
$T=T_{Next}$ // переход на следующего агента
return *C, CW, CC*

Основная часть Алгоритма 2:

for all $T \in M^*$:
if (*Action* **in** $T_{ActionList}$) **then** *ValuesList*.Add($T_R*T_{ActionVal}$)
// упорядочение по уменьшению «полезности»
SortedValuesList=**Sort**(*ValuesList*)
// первый в списке агент – наиболее «полезный»
Leader= SortedValuesList[0]
Coalition.Append(*Leader*)
C=1
$CoalValue= Leader_R*Leader_{ActionVal}$
// продолжаем формировать коалицию
MakeCoalition2(*Coalition,CoalValue,C,*$Leader_{Next}$*,ActValue*)
// проверка, сформирована ли коалиция
if (*CoalValue*≥*ActValue*) **then print** "Коалиция есть "+*Coalition*
else print "Коалиция не найдена"

Для описанного алгоритма верно следующее.

Утверждение 2. Сложность алгоритма 2 составляет O(m·(2 + log m))

Действительно:

1. Сложность формирования списка источников с оценками O(m).

2. Сложность сортировки полученного списка - O(m log m)

3. В самом худшем варианте для формирования коалиции или отказа от нее

сложность составляет O(m).

Таким образом всего сложность O(m·(2 + log m)).

2.4.4. Алгоритм формирования коалиций агентов, способных к изменению оценок успешности

Для моделирования улучшения способностей агентов в процессе деятельности так, чтобы поведение агентов-источников стало более близким к реальности, к агентам добавлен механизм изменения оценок успешности. В результате успешного выполнения действий это значение у агентов увеличивается.

Исходные данные:

Коэффициент изменения оценки (вещественное $coeff>0$)

Набор объектов B_j типа Пункт назначения (*Factories*) с данными как в Алгоритме 1.

Набор объектов A_i, i=1..m типа Источник (*Trucks*) с данными:

- значение «полезности» источника при формировании коалиций (вещественное $Trust_i>0$)

- вес перевозимой продукции ($a_i \geq 0$)

Получаемый результат:

Для каждого объекта типа Пункт назначения формируется коалиция объектов типа Источник или сообщение о невозможности создания коалиции. Источники в коалиции умножают значение «полезности» на величину *coeff*.

В алгоритме используются агенты типа Пункт назначения (как в п.2.3.2) и типа Источник с возможностью изменения оценок успешности (как в п.2.3.3), но они в отличие от п.2.3.3 могут выполнять только одно действие.

Алгоритм 3. Алгоритм формирования групп агентов с учетом изменения оценок успешности, централизованный.

Для каждого пункта назначения F выполняется следующее:

1. Вычисляется расстояние между выбранным пунктом назначения F и каждым источником T с неотрицательным количеством продукции по формуле $\sqrt{(pos_x(F) - pos_x(T))^2 + (pos_y(F) - pos_y(T))^2}$ и добавляется в список вместе с общим весом продукции у источника, которое равно произведению веса имеющейся продукции на значение оценки успешности $N_{xi} = r_i \bullet a_i$.

2. Полученный список сортируется по увеличению расстояния. Для одинаковых расстояний элементы сортируются по уменьшению общего веса продукции у источников.

3. К списку создаваемой коалиции добавляется первый элемент из отсортированного списка. Далее последовательно добавляются элементы из отсортированного списка до того момента, как суммарный общий вес продукции у источников в коалиции равен или превышает вес текущего пункта назначения, или элементов отсортированного списка не осталось.

4. В первом случае сообщается об успехе, во втором случае – о неуспехе.

5. Источники, попавшие в список, уменьшают количество имеющейся у них продукции и увеличивают значение собственной оценки успешности $r(x_i)$ на коэффициент увеличения оценки.

Листинг 6. Централизованный алгоритм формирования групп агентов с учетом опыта.

Исходные данные:

Коэффициент накопления опыта (вещественное *coeff*>0)

Набор объектов B_j, j=1..n типа Пункт назначения (*Factories*) с данными как в Алгоритме 1.

Набор объектов A_i, i=1..m типа Источник (*Trucks*) с данными:

- значение «полезности» источника при формировании коалиций (вещественное $Trust_i$>0);
- перевозимый вес продукции ($a_i \geq 0$).

Получаемый результат:

Для каждого объекта типа Пункт назначения формируется коалиция объектов типа Источник или сообщение о невозможности создания коалиции. Источники в коалиции умножают значение «полезности» на величину *coeff*.

Основная часть Алгоритма 3:

for all $\in N^*$:

// расчет расстояния от текущего Пункта назначения до всех Источников

for all $T \in M^*$:

$DistFT$=Distance(F,T)

// упорядочение по росту расстояния и уменьшению веса продукции

$SortedDist$=**Sort**($DistFT$, T_a)

// первый в списке агент – наиболее близкий с наибольшим весом

$Leader=SortedDist[0]$

$Coalition$.Append($Leader$)

$CoalitionWeight=Leader_a*Leader_{Trust}$

$NextT=Leader_{Next}$

// продолжаем формировать коалицию

while ($CoalitionWeight < F_b$) and ($NextT <> Ø$) :

$Coalition$.Append($NextT$) // текущий агент добавляется в список

$CoalitionWeight=CoalitionWeight+NextT_a*NextT_{Trust}$ // к «полезности» коалиции добавляется «полезность» этого агента

$NextT=NextT_{Next}$ // переход на следующего агента

// проверка, сформирована ли коалиция

if ($CoalitionWeight \geq F_b$) **then**

for all $C \in Coalition$: $C_{Trust}= C_{Trust}*coeff$

print “Коалиция есть ”+$Coalition$

else print “Коалиция не найдена”

Для описанного алгоритма верно следующее утверждение.

Утверждение 3. Сложность Алгоритма 3 составляет O(n · (3m+m log m))

Количество действий аналогично Утверждению 1, добавляется только умножение оценки успешности источников на заданный множитель, в самом худшем варианте сложность O(m).

В целом для задачи O(n·m·(3+log m))

2.4.5. Динамический алгоритм решения транспортной задачи

Установлено, что для того, чтобы жадный алгоритм решения транспортной задачи возвращал оптимальное значение, необходимо и достаточно выполнение условия Хоффмана [112] – матрица перевозок (матрица стоимости перевозки между всеми источниками и пунктами назначения) должна быть «матрицей Монжа». Это означает, что для любой подматрицы размером 2x2 матрицы сумма

чисел на главной диагонали должна быть меньше или равна, чем сумма чисел на дополнительной диагонали.

Были предложены алгоритмы, которые проверяют, удовлетворяет ли данная матрица условию Хоффмана. Первый алгоритм [113] является конструктивным, т.е. в процессе работы он строит нужную последовательность. Если алгоритм завершился успешно, то он возвращает требуемую последовательность. Если он завершился, обработав не все вершины, то полученная подпоследовательность все равно может быть обработана жадно, а к необработанной части матрицы можно применить стандартный оптимальный алгоритм. Таким образом, все равно будет достигнуто улучшение, поскольку размер подматрицы, которую придется обработать стандартным алгоритмом окажется меньше исходной, возможно, намного.

Второй алгоритм [114] является расширением предыдущего, с дополнительной возможностью обработки матрицы перевозок, в которой некоторые маршруты перевозок между источниками и пунктами назначения являются запрещенными.

Описанные выше алгоритмы применимы только в том случае, когда матрица перевозок определена и неизменна.

Однако, в реальности гораздо более частой является ситуация, когда количество продукции, имеющейся у источников или требуемого пунктом назначения, может меняться со временем. Кроме того, некоторые источники и пункты назначения могут появляться и пропадать в случайные промежутки времени. Авторы статьи [115], в которой рассматривается такая постановка задачи, назвали ее динамической стохастической транспортной задачей. Время в этой задаче является дискретным. Известны первоначальные количество продукции у источников, потребности у пунктов назначения и какие маршруты перевозок являются разрешенными или запрещенными. В каждый следующий момент времени количество доступной или требуемой продукции для каждого источника или пункта назначения может увеличиваться на случайную величину. Кроме того, некоторые разрешенные маршруты могут стать запрещенными и наоборот. В

каждый момент времени требуется определить, какое количество продукции необходимо отправить и по какому разрешенному маршруту. На каждом шаге часть доступной источникам продукции может быть задержана на источниках и распределена между пунктами назначения на следующих шагах. Необходимо найти такие значения количества перевозимой продукции на каждом шаге, чтобы суммарная стоимость перевозки всей продукции за все время моделирования была минимальной.

Для решения поставленной задачи авторы предлагают структуру, которую называют «звездой Монжа». Она состоит из одного источника и нескольких пунктов назначения. Доказано утверждение, что если на некотором шаге присутствует такая структура, то безусловно выгоднее распределить между пунктами назначения всю имеющуюся продукцию у источника на этом шаге, не перенося распределение на следующие шаги.

В качестве модельной авторы рассматривают задачу работы аэропорта в условиях тумана, когда имеется очередь самолетов, ожидающих взлета, но при этом в каждую единицу времени может взлететь только один самолет. Некоторые самолеты выполняют стыковочные рейсы, поэтому их отклонение от утвержденного расписания должно быть минимальным. Самолеты, не выполняющие стыковочных рейсов, могут быть задержаны на большее время. Требуется обеспечить взлет всех рейсов, имеющих стыковки, минимизируя общее время задержки всех самолетов на земле.

2.5. Выводы

В главе рассмотрена постановка модельной задачи, предложены алгоритм формирования коалиций агентов, алгоритм распределения предпочитаемых действий агентов, обладающих способностью к изменению оценок успешности, и алгоритм формирования коалиций агентов с учетом этих оценок. Установлены вычислительные сложности предложенных алгоритмов.

Глава 3. Программные реализации алгоритмов для решения задач маршрутизации

В настоящем разделе рассмотрены программные реализации с использованием мультиагентной системы распределенного алгоритма формирования коалиций агентов; распределенного алгоритма распределения предпочитаемых действий агентов, обладающих способностью к изменению оценок успешности; распределенного алгоритма формирования коалиций агентов, обладающих способностью к изменению оценок успешности; схема задания правил и начальных данных для агентов. Дается методика оценки производительности, масштабируемости мультиагентой системы и полученные результаты.

3.1. Оценка параметров мультиагентной системы

Производительность и масштабируемость при выборе мультиагентной системы являются важными параметрами. Согласно [59], производительность мультиагентной системы *SPADE* выше, чем производительность системы *JADE*, поэтому *SPADE* представляется более перспективной для использования.

В то же время недостаточны данные о масштабируемости *SPADE* при количестве агентов порядка нескольких сотен. Поэтому с целью дальнейшего изучения возможностей мультиагентной системы *SPADE* и ее применимости к решению поставленной задачи, в рамках данной работы была проведена экспериментальная проверка ее возможностей.

Количественная оценка параметров системы агентов выполнялась с использованием среднего времени возврата сообщения (*round trip time, RTT*) [116]. Время возврата сообщения определялось как разница между временем отправки сообщения получателю и временем получения ответного сообщения от него. Для системы с несколькими отправителями среднее время возврата сообщения – сумма времен возврата сообщения для каждого отправителя, поделенное на количество отправителей. Каждый эксперимент проводился с размещением отправителей и получателей на одном компьютере (Intel Core i5 - 2,6 GHz, 4 Gb RAM, Win7).

С целью изучения масштабируемости была реализована следующая конфигурация агентов. Участвуют несколько агентов типа «Отправитель», один агент типа «Получатель» и управляющий агент. Каждый Отправитель после получения команды от управляющего агента создает запрос и отправляет его Получателю. После поступления запроса Получатель добавляет к сообщению метку о приеме и возвращает модифицированное сообщение Отправителю. Отправитель замеряет время возврата сообщения. Число агентов-Отправителей увеличивается и замеряется среднее время возвратов сообщений. Было обнаружено, что с ростом количества агентов-Отправителей с 2 до 250 рост *RTT* составляет с 0,130 до 0,460 секунд. Таким образом, при росте количества агентов в 125 раз RTT увеличивается в 3,5 раза.

Схема проведения эксперимента и график зависимости приведены на Рис. 9.

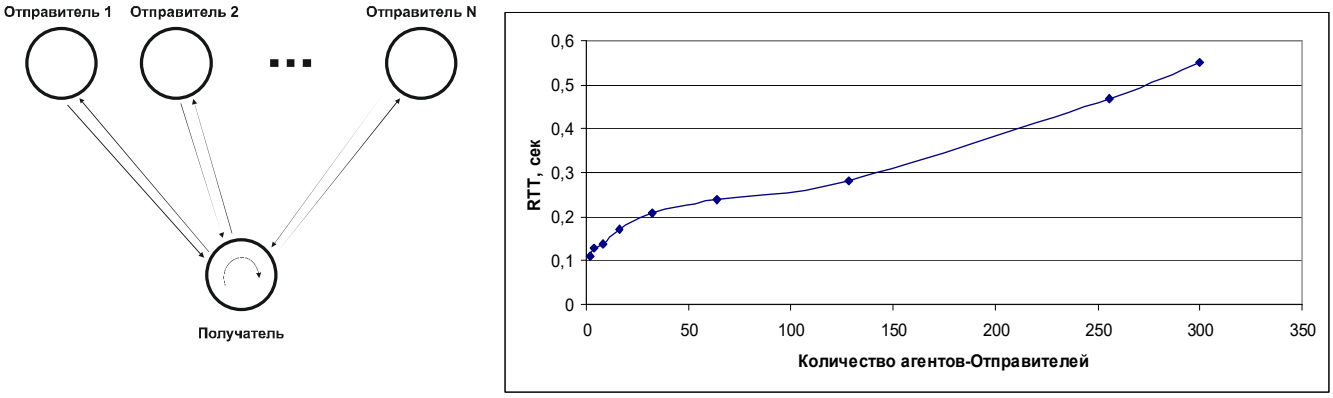


Рис. 9. Схема проведения эксперимента и зависимость *RTT* от количества агентов-отправителей

Для получения данных о производительности использовалась следующая конфигурация агентов. В ней участвуют несколько пар агентов двух типов «Отправитель» и «Получатель», а также управляющий агент. Отправитель после получения команды от управляющего агента создает запрос и отправляет его соответствующему Получателю. После поступления запроса Получатель выполняет вычисления, занимающие время и вычислительные мощности. В качестве решаемой ресурсоемкой задачи было использовано нахождение перебором всех простые числа в диапазоне от 1 до 20000. После завершения

вычислений Получатель возвращает ответ. Отправитель замеряет время возврата сообщения. Число пар Отправитель-Получатель увеличивается и на каждом шаге замеряется среднее время возвратов сообщений. В результате проведения эксперимента найдено, что при увеличении количества пар Отправитель-Получатель с 2 до 250 рост *RTT* составляет с 0,350 до 18 секунд.

Схема проведения эксперимента и график зависимости приведены на Рис. 10.

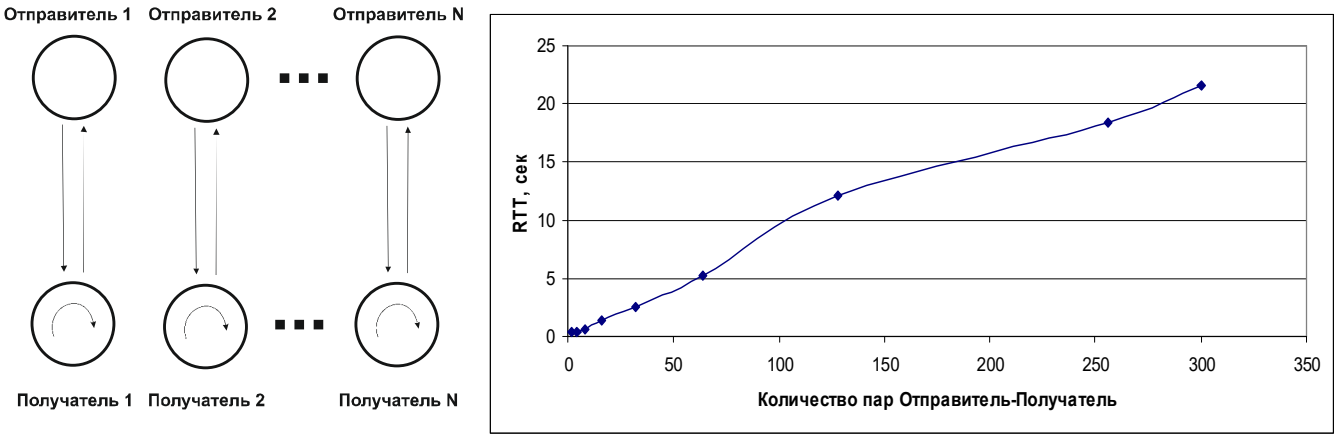


Рис. 10. Схема проведения эксперимента и зависимость *RTT* от количества пар агентов Отправитель-Получатель

Таким образом, в результате проведенных экспериментов получены данные, свидетельствующие о том, что мультиагентная система *SPADE* обладает нужными характеристиками для решения поставленной задачи.

3.2. Механизм задания правил и начальных данных агентов

Проверка системы на примере модельной задачи

Для проверки работы системы была использована описанная в п. 2.2 модельная задача взаимодействия двух групп агентов в двумерном мире.

В данной задаче фактами являются совокупности свойств объектов – скорость перемещения, потребность или имеющийся запас продукции. Правила задают поведение объектов – перемещение. Результатом работы модели является

прогноз развития ситуации.

Далее приведен пример заполнения базы знаний системы.

```
object
  slot name
  slot curX type=integer
  slot curY type=integer
  slot material type=integer
truck extends object
  "Грузовики"
  slot speedX type=integer
  slot speedY type=integer
factory extends object
  "Фабрики"
  slot active type=integer
```

В этом примере задается базовый шаблон со свойствами, общими для обоих типов объектов – координатами и количеством продукции (имеющейся или требуемой). На основе базового шаблона создаются шаблоны для описания фактов об объектах первой (*truck*) и второй (*factory*) групп путем добавления полей, требуемых для описания соответствующего типа фактов.

Далее задаются факты о конкретных объектах.

```
fact1
truck
name=ГАЗ-51; curX=10; curY=95; speedX=6; speedY=6; material=10
fact2
truck
name=ЗИЛ-130; curX=80; curY=95; speedX=4; speedY=4; material=15
fact3
factory
name=Фабрика-1; curX=30; curY=5; material=8; active=0
```

Для каждого объекта задаются конкретные значения: собственное имя, координаты текущего местоположения, скорости перемещения по двум осям и т.д.

Затем создается правило, например, перемещения объекта типа Грузовик на каждом шаге:

```
stepTruck
truck
where
 ((truck.material>0)and(truck.curY>0))
do
truck.curY=truck.curY-truck.speedY
```

Все элементы (задание шаблонов фактов, задание фактов на основе шаблонов, задание правил) проходят синтаксическую проверку и переводятся в текст программы на языке *Python*. Факты переводятся в массив словарей, правила – в функции. Создается главная функция, которая управляет последовательностью вызова функций-правил. Работа главной функции заканчивается, если достигается целевое состояние системы (успех), или применение ни одной из функций-правил невозможно (неуспех).

Для описания двумерной среды с препятствиями используется матрица проходимости – бинарная матрица, каждому элементу которой соответствует проходимая («0»), либо непроходимая («1») область на плоскости. Такая модель представляется МТ-графом [117]. Поиск кратчайшего пути осуществляется с использованием алгоритма A* [118].

Была создана специализированная графическая система для удобного задания и изменения начальных данных агентов и правил, определяющих их поведение, а также отображения результатов моделирования.

Двухуровневые правила

Правила в продукционной системе разделены на два уровня с целью моделирования планирования совместных действий коалиции интеллектуальных

агентов. Первый отвечает за базовые действия агентов – перемещение по кратчайшему пути в указанную точку с учетом обхода препятствий. Второй уровень правил моделирует взаимодействие агентов, если, например, для запуска пункта назначения требуется прибытие нескольких источников одновременно. В этом случае необходимо некоторым образом выбрать несколько объектов-источников для запуска выбранного пункта назначения так, чтобы время прибытия всех источников к цели было одинаковым и суммарное количество доставленной продукции было достаточно для запуска пункта назначения.

3.3. Программный комплекс для формирования коалиций агентов в мультиагентной системе

Возможности программного комплекса на основе мультиагентой системы *SPADE* проверены на примере модельной задачи, описанной в п.2.2.

Каждый объект (источник и пункт назначения) представляется одним агентом. Агенты в мультиагентных системах получают информацию друг от друга и от среды путем передачи и получения сообщений. В мультиагентой системе *SPADE* каждое сообщение обязательно имеет следующие части:

- имя агента-отправителя;
- имя или имена агентов-получателей;
- тип (группу) сообщения;
- тело сообщения (любая текстовая информация).

В мультиагентой системе *SPADE* сообщения могут быть личными (один или несколько получателей) и широковещательными (всем получателям определенного класса). Вне зависимости от того, сообщение личное или широковещательное, агент-отправитель создает одно сообщение и в мультиагентой системе оно считается за одно.

Кроме того, в мультиагентой системе *SPADE* имеется механизм доски объявлений (*Directory Facilitator, DF*). Каждый агент может опубликовать любую

информацию в виде набора пар «ключ»:»значение». Все агенты при обращении к *DF* конкретного агента могут считать опубликованную информацию.

Каждый агент при получении сообщения в зависимости от типа и содержимого тела сообщения выполняет соответствующую подпрограмму-обработчик. В результате работы подпрограммы могут быть изменены внутренние переменные и состояние агента, отправлены сообщения другим агентам и тд.

Для проверки возможностей программного комплекса на основе мультиагентой системы *SPADE* предложенный в п.2.3.1 централизованный Алгоритм 1 был модифицирован для реализации с использованием мультиагентной системы.

В предлагаемой реализации время является дискретным, поэтому в дополнение к двух основным типам агентов – агенты-пункты назначения (*F*), агенты-источники (*T*) требуется вспомогательный агент-таймер (*Timer*).

Алгоритм 4. Алгоритм формирования групп агентов, распределенный.

1. Агент-таймер сообщает агентам-пунктам назначения о начале очередной итерации. В ответ агенты-пункты назначения, которые еще не обеспечены продукцией ($F_{акт}$), сообщают свои координаты и нужный объем продукции.

2. Агент-таймер пересылает всем агентам-источникам, имеющим продукцию ($T_{акт}$), список активных агентов-пунктов назначения и информацию об их местоположении и потребности в продукции.

3. Каждый агент-источник сообщает всем агентам из $T_{акт}$ вес имеющейся у него продукции и количество шагов, за которое он может достигнуть каждого агента-пункта назначения из $F_{акт}$.

Затем агенты-источники начинают формирование групп согласно следующему алгоритму. Источник с наибольшим весом (он становится ведущим, $t_{акт,N}^{0}$) находит пункт назначения из $F_{акт}$, до которого он может добраться за минимальное число шагов (N). Затем он просматривает список $T_{акт}$ и формирует

список ($T_{акт,N}$) всех агентов-источников, которые могут добраться до выбранного пункта назначения за это же число шагов. Источники в $T_{акт,N}$ упорядочиваются по убыванию веса перевозимой продукции, ведущий источник $t_{акт,N}^{0}$ всегда первый в списке.

4. Если общий вес перевозимой источниками из $T_{акт,N}$ продукции превышает вес продукции, необходимой для удовлетворения потребности выбранного пункта назначения, то ведущий источник $t_{акт,N}^{0}$ сообщает источникам $t_{акт,N}^{1}$, $t_{акт,N}^{2}$, .. $t_{акт,N}^{K}$ о том, что они включаются в группу под его управлением, и все они начинаются двигаться к выбранному пункту назначения. Задача ведущего источника выполнена, он освобождает роль ведущего.

Если общий вес продукции, перевозимой источниками из $T_{акт,N}$, недостаточен, то ведущий источник увеличивает N и снова находит подмножество источников $T_{акт,N+1}$.

5. В случае, если общий вес продукции у источников из $T_{акт,N+1}$ достаточен, то ведущий источник сообщает нужным источникам из $T_{акт,N+1}$ о включении их в его группу и отдает распоряжение о движении к выбранному пункту назначения. Задача ведущего источника выполнена, он освобождает роль ведущего.

В противном случае ведущий источник пробует удовлетворить потребность другого пункта назначения из $F_{акт}$, которого может достигнуть за число шагов, превышающее N. После этого ведущий источник освобождает роль ведущего.

6. Источники из $T_{акт}$, не получившие распоряжений, снова выбирают ведущего и цикл повторяется.

Когда агенты-источники прибывают в пункт назначения с достаточным суммарным объемом продукции, то потребность пункта назначения удовлетворена. Возможные излишки продукции, оставшиеся у прибывших источников, могут быть использованы для других пунктов назначения.

Процесс моделирования останавливается, когда суммарное количество продукции, имеющееся у всех источников, недостаточно для удовлетворения потребности ни одного пункта назначения.

Действия, которые выполняет агент-источник, приведены в Листинге 7.

Листинг 7. Алгоритм 4, действия агента-источника.

```
Case Message.Text of:
"Publish":
    DF.Add(My.ID, C_ij, My.Weight) //публикуем расстояние и вес
"SearchAll":
    AllTrucks=DF.GetAll() //получаем все опубликованные данные
"FormGroup":
    SortedDist=Sort(AllTrucks, DistFT, T_a) // упорядочение по росту расстояния и уменьшению веса продукции
    Leader= SortedDist[0]
    if My_ID==Leader_ID then //текущий агент – лидер
        Coalition.Append(Leader)
        CoalitionWeight=Leader_a
        CoalitionCount=1
        MakeCoalition1(Coalition, CoalitionWeight, CoalitionCount,
                       Leader_Next, F_b)
        if ( CoalitionWeight>=F_b ) then
            for all C in Coalition:
                sendMessage(C_ID,"OfferToJoin")
            else  print "Коалиция не сформирована "
"OfferToJoin":
    SendResponse(Sender_ID)
"AcceptToJoin":
    CoalitionCount=CoalitionCount-1
      if CoalitionCount==0 then print "Коалиция сформирована "+Coalition
"DeclineToJoin":
    // отказавшийся агент исключается из коалиции
    CoalitionWeight=CoalitionWeight-Sender_Weight
    CoalitionCount=CoalitionCount-1
    NextTruck= NextTruck_Next
    // продолжается формирование коалиции
    MakeCoalition1(Coalition, CoalitionWeight,
                   CoalitionCount, NextTruck, F_b)
    if ( CoalitionWeight>=F_b ) then
            for all C in Coalition:
                sendMessage(C_ID,"OfferToJoin")
          else print "Коалиция не сформирована"
```

Для применения Алгоритма 4 для решения задачи II достаточно поменять местами Источники и Пункты назначения, изменения в самом алгоритме не требуются.

Типы сообщений, передаваемые агентами в процессе моделирования

Типы сообщений у агента-таймера. Агент-таймер является координатором взаимодействия всех остальных агентов. Очередной цикл работы остальных агентов начинается после получения сообщения от агента-таймера. Он отправляет следующие управляющие сообщения:

Для пунктов назначения

Тип: *NextTickFactory*

Сообщение уведомляет агенты-пункты назначения о начале следующей итерации. В ответ пункты назначения с неудовлетворенной потребностью присылают информацию о себе.

Подробно описано в разделе пунктов назначения.

Для источников

Тип: *NextTick*

Сообщение уведомляет агенты-источники о начале следующей итерации. Подробно описано в разделе источников.

Агент-таймер получает следующие сообщения:

1. Тип: *FactoryData*

Отправитель: агент-пункт назначения

Тип: личное

Тело сообщения: четверка <имя пункта назначения>,<координата X>,<координата Y>,<требуемый вес продукции для пункта назначения>

2. Тип: *GroupReady*

Отправитель: агент-источник

Тип: личное

Тело сообщения: *GroupReady*

Сообщение от ведущего агента-источника о том, что коалиция для выбранного пункта назначения успешно сформирована

3. Тип: *GroupNotFormed*

Отправитель: агент-источник

Тип: личное

Тело сообщения: *GroupNotFormed*

Сообщение от ведущего агента-источника о том, что коалицию для выбранного пункта назначения сформировать не удалось

Типы сообщений у агентов-пунктов назначения. Каждый из таких агентов обрабатывает сообщение:

1. Тип: *NextTickFactory*

Отправитель: *Timer*

Тип: широковещательное

Тело сообщения: номер итерации

Действия при получении: если пункт назначения имеет неудовлетворенную потребность, то он отправляет агенту *Timer* сообщение *FactoryData* со своими названием, координатами и значением требуемого веса продукции.

Типы сообщений у агентов-источников. Каждый из таких агентов обрабатывает следующие сообщения:

1. Тип: *NextTick*

Отправитель: *Timer*

Тип: широковещательное

Тело сообщения: название, координаты и требуемый вес продукции для очередного пункта назначения.

Действия при получении: если вес продукции у агента неотрицательный, то он опубликовывает на своей доске объявлений название, вес имеющейся продукции и расстояние до указанного пункта назначения.

2. Тип: *CreateGroup*

Отправитель: *Timer*

Тип: широковещательное

Тело сообщения: *SearchAll*

Действия при получении: агент сохраняет себе информацию о всех агентах-источниках, которую они опубликовали на предыдущем шаге

3. Тип: *CreateGroup*

Отправитель: *Timer*

Тип: широковещательное

Тело сообщения: *FormGroup* и название пункта назначения, для которого формируется коалиция

Действия при получении: Агент сравнивает свои данные с полученной на предыдущем шаге информацией обо всех агентах. В случаях, если:

1. он самый близкий к выбранному пункту назначения,

2. есть несколько агентов, самых близких к выбранному пункту назначения, и агент имеет самый большой вес,

то этот агент продолжает обработку сообщения и начинает выполнять роль ведущего. В противном случае агент прекращает обработку сообщения.

Если агент – ведущий и продолжил свою работу, то он формирует список всех источников, упорядоченных по увеличению расстояния до целевого пункта назначения. Источники на одинаковом расстоянии упорядочиваются по уменьшению веса. Очевидно, что ведущий агент всегда находится в списке на первой позиции. Пусть длина полученного списка равна L. Из полученного списка ведущий агент последовательно добавляет k первых агентов в коалицию до тех пор, пока:

1. k=L и суммарный вес продукции у всех агентов из коалиции меньше требуемого для удовлетворения потребности выбранного пункта назначения. В этом случае коалиция не сформирована, ведущий агент отправляет сообщение

GroupNotFormed агенту *Timer* и прекращает обработку сообщения.

2. k<=L и суммарный вес продукции у всех агентов из коалиции равен или больше требуемого для удовлетворения потребности выбранного пункта назначения. Тогда коалиция успешно сформирована, ведущий агент отправляет всем агентам из этого подсписка сообщение *OfferToJoin* с указанием нужного от этого агента веса продукции. Агенты с 1 по k-1 расходуют всю имеющуюся у них продукцию. У k-го агента может остаться излишек продукции, равный разнице между суммарным весом продукции у всех k агентов и потребностью в продукции у выбранного пункта назначения. После этого ведущий агент прекращает обработку сообщения.

Таким образом, при обработке сообщения происходит попытка формирования коалиции с характеристической функцией, равной суммарному весу входящих в коалицию агентов. При этом решается вспомогательная задача минимизации пробега источников.

4. Тип: *CreateGroup*

Отправитель: Агент-источник, ведущий

Тип: личное, всем агентам из подсписка на предыдущем шаге

Тело сообщения: *OfferToJoin* и требуемый вес от агента-получателя.

Действия при получении: агент уменьшает свой вес на указанную величину и отправляет *AcceptToJoin* ведущему.

5. Тип: *CreateGroup*

Отправитель: Агент-источник, участник коалиции

Тип: личное, получатель агент-ведущий

Тело сообщения: *AcceptToJoin*

Действия при получении: агент-ведущий при получении каждого такого сообщения проверяет, ответили ему все агенты из подсписка, полученного на шаге 3. Если ответили все, то посылает сообщение *GroupReady* агенту *Timer*, информируя его о том, что коалиция успешно сформирована.

Для Алгоритма 4 верны следующие утверждения.

Утверждение 7. Сложность Алгоритма 4 для каждого агента составляет не более, чем $O(n+m\cdot(2+\log m))$

Для доказательства рассмотрим все участвующие в работе типы агентов:

Для агента *Timer*

$O(n)$ – в худшем случае перебор по всем пунктам назнчения

Для агента типа пункт назначения

$O(1)$ – перебора нет, только отправка сообщения о себе по запросу от *Timer*

Для агента типа источник

Для всех агентов-источников

1. $O(n)$ – перебор по всем пунктам назначения для вычисления расстояния от них до конкретного источника.

2. $O(m)$ – сравнение каждого источника со всеми остальными для выявления самого загруженного (ведущего)

Только для одного источника-ведущего, дополнительно к предыдущему

3. $O(m \log m)$ – сортировка полученного списка источников по увеличению расстояния до пункта назначения

4. $O(m)$ – добавление источников из отсортированного списка в список участников коалиции

Для распределенных алгоритмов важной характеристикой является оценка количества сообщений, передаваемых в процессе их работы. Важно, что мультиагентная система *SPADE* может отправлять одно сообщение нескольким получателям одновременно.

Утверждение 8. Количество сообщений для Алгоритма 4 составляет O(n(m+1)), где n – количество пунктов назначения, m – количество источников.

Для доказательства рассмотрим последовательность передачи сообщений.

1. Агент Timer при отправке сообщений с типом *NextTick*, типом *CreateGroup* и текстом *SearchAll* или текстом *FormGroup* каждый раз отправляет 1 сообщение, возможно нескольким получателям.

2. Агенты типа источник отправляют каждый 1 сообщение с типом *CreateGroup* и текстом *OfferToJoin*, количество агентов-отправителей не более, чем m-1.

3. Ведущий агент-источник отправляет 1 сообщение (*GroupReady* или *GroupNotFormed*) агенту *Timer*.

Пункты 1-3 повторяются не более, чем для n пунктов назначения.

Таким образом, общее количество сообщений составляет O(n(m+1)).

Для целенаправленного поведения требуется построение плана действий, который приведет систему из начального состояния в заданное целевое.

3.4. Динамическое распределение предпочитаемых действий в коалиции агентов, способных к изменению оценок успешности

Пусть имеется множество исполнителей — m агентов. У каждого агента сформированы наборы действий t_i, упорядоченных по их предпочтительности для данного агента. С каждым из действий связано числовое значение p_i — количество продукции или «успешность действия» t у агента i. Это значение можно рассматривать как вероятность успешного выполнения выбранного действия конкретным агентом. Кроме того, агент-координатор для каждого агента-исполнителя имеет числовое значение r_i – «успешность агента» i при выполнении выданных ему заданий. Общая «Оценка успешности агента при выполнении действия» конкретным агентом выбранного действия t_i вычисляется как произведение этих оценок $N_{ti} = r_i \bullet p_i$.

Структура агента и цикл изменения оценки успешности приведены на Рис. 11. «Успешность действия» – число (0,1], «Успешность агента» – положительное число. Задаются коэффициенты изменения успешности действий и агентов C_R, C_A. После успешного выполнения действия агент умножает свои значения «Успешности агента» и «Успешности действия» на соответствующие коэффициенты. В случае отказа агента от назначенного действия, «Успешность агента» делится на C_R, «Успешность действия» не изменяется.

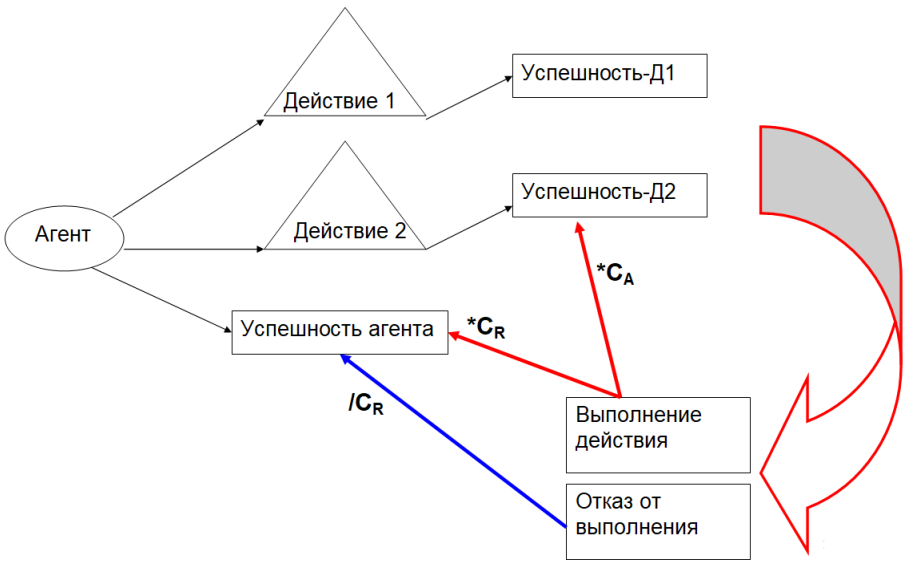


Рис. 11. Предложенная структура агента и цикл изменения оценки успешности

Предложенная модель позволяет гибко менять параметры взаимодействия агентов в процессе моделирования на основе учета их текущих характеристик, что положительно влияет на качество решений задач маршрутизации.

Пусть имеется некоторая последовательность действий длиной *n*, на каждом ее шаге необходимо найти одного или нескольких агентов, которые смогут сообща выполнить данное действие, причем суммарный вес продукции у коалиции должен быть не меньше, чем требуемый вес для выполнения этого действия.

Предлагаемая модель реализована с использованием мультиагентой системы *SPADE*. В состав модели входят агенты-исполнители и управляющий агент. Цикл создания коалиции представлен в Алгоритме 5.

Алгоритм 5. Распределенная версия Алгоритма 2.

1. Управляющий агент получает наименование действия на некотором шаге плана и числовое значение веса, который требуется набрать при создании коалиции.

2. Управляющий агент сообщает агентам-исполнителям наименование действия.

3. Каждый агент-исполнитель, в случае, если у него имеется возможность выполнить требуемое действие, опубликовывает в специальном механизме *Directory Facilitator* (*DF*, аналог доски объявлений) это действие и число - общую оценку количества продукции N_i. Агенты являются честными и не изменяют значения оценок самостоятельно.

4. Управляющий агент дает команду на формирование коалиции.

5. Каждый агент-исполнитель опрашивает *DF* всех активных агентов-исполнителей.

Агент, у которого по результатам опроса опубликованный вес имеющейся продукции является максимальным, назначается «ведущим», остальные агенты-исполнители могут получать от него указания. Если опубликованный вес максимален не у одного агента, то выбирается агент с наибольшим *UID* (автоматически присваивается каждому агенту в системе *SPADE* при его создании).

6. «Ведущий» агент сравнивает собственный вес с требуемым. Если требуемый вес меньше его веса, то он сортирует список весов остальных агентов по возрастанию, находит агента, с меньшим чем у старшего весом, но большим, чем требуемый, и сообщает координатору об успешном формировании коалиции и присылает список, состоящий только из найденного агента.

7. Если требуемый вес больше веса «ведущего» агента-исполнителя, то он сортирует список весов остальных агентов по возрастанию и последовательно добавляет их к коалиции. Если на некотором шаге вес коалиции превышает требуемый, то коалиция сформирована. «Ведущий» агент-исполнитель высылает координатору список агентов, входящих в коалицию, и сообщает каждому

агенту-исполнителю из списка, что они назначены в коалицию. Агенты-исполнители при получении сообщения от «ведущего» агента-исполнителя могут отказаться от вхождения в коалицию.

Если нужный вес не найден, то «ведущий» агент-исполнитель сообщает управляющему агенту о неуспехе. Все отказавшиеся агенты-исполнители получают штраф к добросовестности r_i. Значения успешности действий p_i каждого участника в этом случае не меняются.

8. Управляющий агент, получив успешно созданную коалицию, увеличивает оценку успешности действия каждого участника p_i, на часть общего веса коалиции, обратно пропорционального весу каждого агента. Таким образом, агент с наиболее высоким вкладом получает минимальное вознаграждение, а с самым малым — максимальное. Отказавшиеся агенты-исполнители получают штраф к значению r_i добросовестности и в состав коалиции не включаются.

Действия, которые выполняет агент-источник, приведены в Листинге 8.

Листинг 8. Алгоритм 5, действия агента-источника.

Входные данные: Action – требуемое действие; ActionValue – минимальная необходимая «общая успешность»

```
Case Message.Text of:
"Publish":
    if (Action in My_ActionList):
        DF.Add(My_ID, My_R*My_ActionVal)
"SearchAll":
    AllTrucks=DF.GetAll()
"FormGroup":
    SortedDist=Sort(AllTrucks)
    Leader= SortedDist[0]
    if My_ID==Leader_ID then
        Coalition.Append(Leader)
        CoalitionWeight=Leader_R*Leader_ActionVal
        CoalitionCount=1
        MakeCoalition2(Coalition, CoalitionWeight,
                CoalitionCount, Leader_Next, ActionValue)
        if ( CoalitionWeight>= ActionValue ) then
            for all C in Coalition:
                sendMessage(C_ID,"OfferToJoin")
```

```
            else print "Коалиция не сформирована"
    "OfferToJoin":
        SendResponse(Sender_ID)
    "AcceptToJoin":
        CoalitionCount=CoalitionCount-1
        if CoalitionCount==0 then
            for all C in Coalition:
                C_R= C_R*Coeff_R
                C_ActionVal= C_ActionVal*Coeff_A
            print "Коалиция сформирована "+Coalition
    "DeclineToJoin":
        // отказавшийся агент исключается из коалиции
        CoalitionWeight= CoalitionWeight - Sender_R*Sender_ActionVal
        CoalitionCount= CoalitionCount -1
        Sender_R= Sender_R/Coeff_R
        NextTruck= NextTruck_Next
        // продолжается формирование коалиции
        MakeCoalition2(Coalition, CoalitionWeight,
                    CoalitionCount, NextTruck, ActionValue)
        if ( CoalitionWeight>= ActionValue ) then
                for all C in Coalition:
                    sendMessage(C_ID,"OfferToJoin")
        else print "Коалиция не сформирована"
```

Для Алгоритма 5 верны следующие утверждения.

Утверждение 9. Сложность Алгоритма 6 составляет $O(n\cdot(m^2 + m\cdot(1+\log m)))$

На каждом шаге плана выполняются действия:

Пункт 6 – сортировка агентов и в худшем случае перебор их всех для получения первого варианта коалиции O (m log m + m)

В самом худшем варианте все агенты могут отказаться от участия в коалиции, что приведет к необходимости дополнительного перебора, снова O(m). Количество дополнительных переборов не может быть больше количества агентов m. Т.е. общая оценка $O(m^2)$.

Таким образом на каждом шаге плана $O(m^2 + m\cdot(1+\log m))$

Количество шагов плана n, общая оценка всего алгоритма $O(n\cdot(m^2 + m\cdot(1+\log m)))$

Утверждение 10. Количество переданных сообщений при работе Алгоритма 5 составляет O(n·(3+2m))

Количество сообщений, передаваемых на каждом шаге алгоритма:

Шаг 2 – 1 сообщение

Шаг 4 – 1 сообщение

Шаг 6 – в худшем случае 1 сообщение каждому из m агентов и ответное сообщение от того, итого 2m

Шаг 7 – 1 сообщение (результат формирования коалиции)

Итого: O(3+2m) на каждый шаг плана.

Количество шагов плана n, общая оценка O(n·(3+2m)).

Типы сообщений, передаваемые в процессе моделирования

Каждый агент-исполнитель обрабатывает следующие сообщения:

1. Тип: *NextTick*

Отправитель: *Timer*

Тип: широковещательное

Тело сообщения: название действия

Действия при получении: если агент может выполнять такое действие, то он опубликовывает на своей доске объявлений название действия и оценку, равную произведению оценки собственной успешности и вероятности выполнения этого действия.

2. Тип: *CreateGroup*

Отправитель: *Timer*

Тип: широковещательное

Тело сообщения: *SearchAll*

Действия при получении: агент сохраняет себе информацию о всех агентах-исполнителях, которую они опубликовали на предыдущем шаге.

3. Тип: *CreateGroup*

Отправитель: *Timer*

Тип: широковещательное

Тело сообщения: *FormGroup* и суммарный вес, который нужно набрать

Действия при получении: Агент сравнивает свою оценку с полученной на предыдущем шаге информацией обо всех агентах. В случаях, если:

1. у него самая высокая оценка,

2. есть несколько агентов, с одинаковой самой высокой оценкой, и *UID* агента лексиграфически первое среди этих агентов,

то этот агент продолжает обработку этого сообщения и начинает выполнять роль «ведущего». В противном случае агент прекращает обработку сообщения.

Если агент – «ведущий» и продолжил свою работу, то он формирует список всех исполнителей, упорядоченных по уменьшению значений оценки. Пусть длина полученного списка равна L. Очевидно, что ведущий агент всегда находится в списке на первой позиции. Из полученного списка «ведущий» агент последовательно добавляет k первых агентов в коалицию до тех пор, пока:

(1). k=L и суммарная оценка меньше требуемой. В этом случае коалиция не сформирована, «ведущий» агент отправляет сообщение *GroupNotFormed* агенту *Timer* и прекращает обработку сообщения.

(2). k<=L и суммарная оценка равна или больше требуемой. Тогда «ведущий» агент отправляет всем агентам из этого подсписка сообщение *OfferToJoin* с указанием нужного от этого агента значения суммарной оценки. Затем «ведущий» агент прекращает обработку сообщения.

4. Тип: *CreateGroup*

Отправитель: Агент-исполнитель, «ведущий»

Тип: личное, всем агентам из подсписка на предыдущем шаге

Тело сообщения: *OfferToJoin* и требуемый вес от агента-получателя.

Действия при получении: если агент согласен на присоединение к коалиции, то он отправляет *AcceptToJoin* «ведущему» и увеличивает значение своей оценки

успешности.

Если агент не согласен на присоединение, то он отправляет сообщение *DeclineToJoin* «ведущему». Значение оценки успешности агента будет уменьшено.

5. Тип: *CreateGroup*

Отправитель: Агент-исполнитель, участник коалиции

Тип: личное, получатель агент-«ведущий»

Тело сообщения: *AcceptToJoin*

Действия при получении: агент-«ведущий» при получении каждого такого сообщения проверяет, ответили ему все агенты из подсписка, полученного на шаге 3. Если ответили все, то посылает сообщение *GroupReady* агенту *Timer*, информируя его о том, что коалиция успешно сформирована.

6. Тип: *CreateGroup*

Отправитель: Агент-исполнитель, участник коалиции

Тип: личное, получатель агент-«ведущий»

Тело сообщения: *DeclineToJoin*

Действия при получении: агент-«ведущий» при получении такого сообщения возобновляет проход по отсортированному списку агентов – возможных исполнителей (см. п.3) для нахождения замены отказавшемуся агенту. Далее аналогично п.3: если такие агенты были найдены, то им отправляется сообщение *OfferToJoin* с указанием нужного от каждого агента значения суммарной оценки.

Если агенты для замены отказавшегося агента найдены не были, то в этом случае коалиция не сформирована, «ведущий» агент отправляет сообщение *GroupNotFormed* агенту *Timer* и прекращает обработку сообщения.

3.5. Программный комплекс для формирования коалиций агентов, способных к изменению оценок успешности, решение задач маршрутизации

В пункте 3.3 показано, что для частного случая сбалансированной

транспортной задачи отклонение результатов, полученных предложенным жадным алгоритмом, от оптимального составляет 4-30%. Для того, чтобы уменьшить это отклонение, к распределенному Алгоритму 4 был добавлена способность агентов к изменению оценок успешности - агент в процессе своей деятельности приобретает опыт и справляется с последующими заданиями более эффективно.

Далее представлен Алгоритм 6, являющийся Алгоритмом 4 с добавлением механизма изменения оценок успешности.

Алгоритм 6. Алгоритм формирования групп агентов с учетом изменения оценок успешности, распределенный.

1. Агент-таймер сообщает агентам-пунктам назначения о начале очередной итерации. В ответ агенты-пункты назначения, которые еще не обеспечены продукцией ($F_{акт}$), сообщают свои координаты и требуемый объем продукции.

2. Агент-таймер пересылает всем агентам-источникам, имеющим продукцию ($T_{акт}$), список активных агентов-пунктов назначения и информацию о их местоположении и потребности в продукции.

3. Каждый агент-источник сообщает всем агентам из $T_{акт}$ общий вес имеющегося у него продукции, вычисляемый для источника i как произведение веса имеющейся продукции на значение оценки успешности $N_{xi} = r_i \bullet a_i$, и количество шагов, за которое он может достигнуть каждого агента-пункта назначения из $F_{акт}$.

Затем агенты-источники начинают формирование групп согласно следующему алгоритму. Источник с наибольшим весом (он становится ведущим, $t_{акт,N}^{0}$) находит пункт назначения из $F_{акт}$, до которого он может добраться за минимальное число шагов (N). Затем он просматривает список $T_{акт}$ и формирует список ($T_{акт,N}$) всех агентов-источников, которые могут добраться до выбранного пункта назначения за это же число шагов. Источники в $T_{акт,N}$ упорядочиваются по убыванию веса перевозимой продукции, ведущий источник $t_{акт,N}^{0}$ всегда первый в списке.

4. Если общий вес перевозимой источниками из $T_{акт,N}$ продукции превышает

вес продукции, необходимой для удовлетворения потребности выбранного пункта назначения, то ведущий источник $t_{акт,N}^{0}$ сообщает источникам $t_{акт,N}^{1}$, $t_{акт,N}^{2}$, .. $t_{акт,N}^{K}$ о том, что они включаются в группу под его управлением.

Некоторые источники при получении запроса могут отказаться от включения в группу. В этом случае отказавшиеся источники получают штраф к Успешности агента: значение Успешности агента делится на коэффициент C_R. Таким образом моделируется возможность выхода из строя некоторых источников.

Источники, согласившиеся на включение в группу, если их общий вес продукции достаточен, начинаются двигаться к выбранному пункту назначения. Задача ведущего источника выполнена, он освобождает роль ведущего. После успешного удовлетворения потребности пункта назначения все источники, участвовавшие в коалиции, увеличивают значение оценки успешности (C_A) на некоторый множитель.

Если общий вес продукции, перевозимой источниками из $T_{акт,N}$, недостаточен, то ведущий источник увеличивает N и снова находит подмножество источников $T_{акт,N+1}$.

5. В случае, если общий вес продукции у источников из $T_{акт,N+1}$ достаточен, то ведущий источник сообщает нужным источникам из $T_{акт,N+1}$ о включении их в его группу и отдает распоряжение о движении к выбранному пункту назначения. Задача ведущего источника выполнена, он освобождает роль ведущего. После успешного удовлетворения потребности пункта назначения все источники, участвовавшие в коалиции, увеличивают значение оценки успешности (C_A) на некоторый множитель.

В противном случае ведущий источник пробует удовлетворить потребность другого пункта назначения из $F_{акт}$, которого может достигнуть за число шагов, превышающее N. После этого ведущий источник освобождает роль ведущего.

6. Источники из $T_{акт}$, не получившие распоряжений, снова выбирают ведущего и цикл повторяется. Источники, участвовавшие в коалиции, увеличивают значение собственной оценки успешности на некоторый множитель. Таким образом, если продукция у источника была израсходована не полностью, то

при последующем его участии в коалиции общий вес будет вычислен как произведение веса неизрасходованной продукции на новое, увеличенное значение оценки успешности (C_A).

Типы и состав сообщений, передаваемых агентами в процессе моделирования при работе программной реализации Алгоритма 6, аналогичны типам сообщений, приведенным в описании Алгоритма 4.

Действия, которые выполняет агент-источник, приведены в Листинге 9.

Листинг 9. Алгоритм 6, действия агента-источника.

```
Case Message.Text of:
"Publish":
   DF.Add(My_ID, C_ij, My_a*My_R) //публикуем расстояние и вес
"SearchAll":
   AllTrucks=DF.GetAll() //получаем все опубликованные данные
"FormGroup":
   SortedDist=Sort(AllTrucks, DistFT, T_a*T_R) // упорядочение по росту расстояния и уменьшению общего веса продукции
   Leader= SortedDist[0]
   if My_ID==Leader_ID then //текущий агент – лидер
      Coalition.Append(Leader)
      CoalitionWeight=Leader_a*Leader_R
      CoalitionCount=1
      T=Leader_Next
      while (CoalitionWeight<F_b) and (T<>Ø) :
         Coalition.append(T) // текущий агент добавляется в список
         CoalitionWeight= CoalitionWeight+T_a*T_R // добавляется текущий агент
         CoalitionCount=CoalitionCount+1 // размер коалиции увеличивается
         T=T_Next // переход на следующего агента
      if (CoalitionWeight>=F_b) then
         for all C in Coalition: sendMessage(C_ID,"OfferToJoin")
      else  print "Коалиция не сформирована "
"OfferToJoin":
         SendResponse(Sender_ID)
"AcceptToJoin":
   CoalitionCount=CoalitionCount-1
   if CoalitionCount==0 //все агенты подтвердили участие
      for all C in Coalition: C_R= C_R*Coeff_R  ; C_a= C_a*Coeff_A
      print "Коалиция сформирована "+Coalition
"DeclineToJoin":
   // отказавшийся агент исключается из коалиции
   CoalitionWeight=CoalitionWeight-Sender_a*Sender_R
```

CoalitionCount=CoalitionCount-1
$Sender_R = Sender_R/Coeff_R$
$T=NextTruck_{Next}$
while (*CoalitionWeight*<F_b) **and** (*T*<>Ø) : // продолжается формирование
Coalition.append(*T*) // текущий агент добавляется
$CoalitionWeight=CoalitionWeight+T_R*T_a$
CoalitionCount=CoalitionCount+1
$T=T_{Next}$ // переход на следующего агента
if (*CoalitionWeight*>=F_b) **then**
for all *C* **in** *Coalition*: sendMessage(C_{ID},"OfferToJoin")
else print "Коалиция не сформирована"

Для применения Алгоритма 6 для решения задачи II достаточно поменять местами Источники и Пункты назначения, изменения в самом алгоритме не требуются.

Свойства Алгоритма 6 аналогичны свойствам Алгоритма 4.

Утверждение 12. Сложность Алгоритма 6 для каждого агента составляет не более, чем O(n+m·(2+log m))

Утверждение 13. Количество сообщений для Алгоритма 6 составляет O(n(m+1)), где n – количество пунктов назначения, m – количество источников.

3.6. Выводы

В настоящем разделе приведена методика оценки производительности и масштабируемости мультиагентой системы, полученные результаты демонстрируют применимость системы *SPADE* для реализации предложенных алгоритмов. Рассмотрены программные реализации распределенного алгоритма формирования коалиций агентов; распределенного алгоритма распределения предпочитаемых действий агентов, обладающих способностью к изменению оценок успешности; распределенного алгоритма формирования коалиций агентов, обладающих способностью к изменению оценок успешности. Установлены характеристики программных реализаций – вычислительная сложность и количество передаваемых сообщений.

Глава 4. Практическое применение предложенных алгоритмов

В настоящей главе описываются эксперименты, проведенные с программными реализациями предложенных алгоритмов. Были изучены:

- производительность программной реализации, т.е. зависимость количества переданных сообщений и времени, затраченного на формирование всех групп, от количества агентов-источников и агентов-пунктов назначения;

- насколько близки результаты, полученные предложенными алгоритмами, к результатам известного оптимального алгоритма решения транспортной задачи;

- изучение увеличения оценок успешности источниками в процессе удовлетворения потребности одного или нескольких пунктов назначения. Во втором случае новые оценки успешности, полученные при удовлетворении потребности одного пункта назначения, применяются уже при удовлетворении потребности другого пункта назначения.

Рассматриваются практические задачи и способ их решения с применением предложенных программных реализаций.

4.1. Генерация случайных начальных данных

Работа программных реализаций предложенных алгоритмов была проверена с использованием наборов случайных данных, полученных с использованием следующего алгоритма. Входными параметрами являются:

- для агентов-пунктов назначения: количество и максимальная потребность в продукции *FactoryMaxWeight*;

- для агентов-источников: количество и максимальная грузоподъемность *TruckMaxWeight*.

На основе входных параметров создается требуемое количество агентов-пунктов назначения и агентов-источников со следующими свойствами.

У каждого агента-пункта назначения:

- положение по оси X – случайное значение в диапазоне [1, 100];

- положение по оси Y – случайное значение в диапазоне [1, 25];

- потребность в продукции – случайное значение в диапазоне [*TruckMaxWeight* + 1, *FactoryMaxWeight*].

У каждого агента-источника:

- положение по оси X – случайное значение в диапазоне [1, 100];

- положение по оси Y – случайное значение в диапазоне [75, 100];

- грузоподъемность – случайное значение в диапазоне [1, *TruckMaxWeight*].

Все эксперименты проводились на компьютере Intel Core i5 - 2,6 GHz, 4 Gb RAM, Win7.

4.2. Результаты экспериментальной проверки работы Алгоритма 4 для задачи I

Эксперимент был проведен для изучения быстродействия и масштабируемости программной реализации при увеличении количества агентов. Экспериментальные данные о зависимости количества переданных сообщений и времени, затраченного на формирование всех групп, от количества агентов-источников и агентов-пунктов назначения приведены в Таблице 3.

Таблица 3. Зависимость количества переданных сообщений и времени, затраченного на формирование всех групп, от количества агентов-источников и агентов-пунктов назначения

Количество агентов-пунктов назначения	Количество агентов-источников	Количество сообщений	Время, секунды
5	20	25	0.2
	50	84	4.6
	100	117	105
10	20	40	1.5
	50	144	40
	100	237	230
20	20	42	5
	50	61	86
	100	378	307

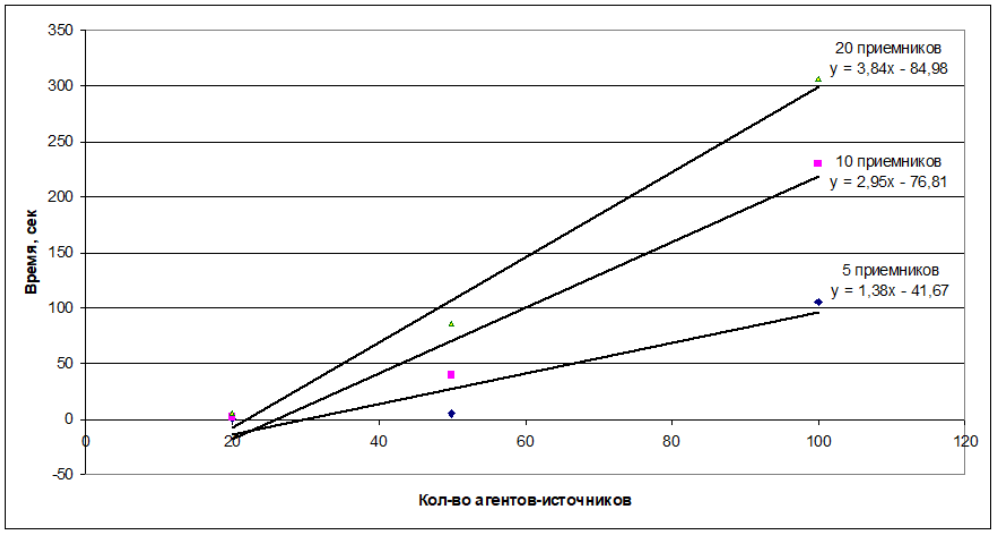


Рис. 12. Время работы программной реализации в зависимости от количества агентов

Из приведенных экспериментальных данных видно, что зависимость времени работы от количества агентов является близкой к линейной, при этом увеличение количества агентов-пунктов назначения в 4 раза приводит к увеличению наклона кривой менее, чем в 3 раза (Рис. 12). Полученный результат свидетельствует о хорошей масштабируемости реализации предложенного алгоритма, что согласуется с ранее полученными теоретическими оценками сложности алгоритмов.

4.3. Сравнение результатов с результатами оптимального алгоритма

Для оценки качества решений, получаемых предлагаемым алгоритмом, в работе сравниваются его результаты с результатами известного алгоритма решения транспортной задачи в тех условиях, когда он применим. Таким образом, значение общей стоимости перевозок для каждых начальных данных, полученное оптимальным алгоритмом, может служить эталоном при сравнении качества решений, полученных алгоритмами, предложенными в настоящей работе, для этих же начальных данных. Поэтому при получении случайных входных данных (п.1) требуется дополнительно суммировать и запоминать общую потребность в продукции для всех агентов-пунктов назначения и общую грузоподъемность для

всех источников. Если они не равны между собой, то значение грузоподъемности для некоторых источников изменяется так, чтобы общая грузоподъемность была равна общей потребности в продукции. Кроме того, для работы оптимального алгоритма требуется расчет матрицы транспортных расходов, представляющей стоимость доставки единицы продукта каждым источником в каждый пункт назначения. Стоимость вычисляется как евклидово расстояние от каждого источника до каждого пункта назначения.

Затем для каждого набора начальных данных вычисляются значения суммарных затрат, полученные оптимальным и предложенным алгоритмами. Значение суммарных затрат, полученных оптимальным алгоритмом, принимается за 100%, после чего вычисляется превышение значения суммарных затрат, полученных предложенным алгоритмом. Полученные значения приведены в Табл. 4. Значения превышения затрат являются средними значениями, полученными в результате пяти независимых испытаний.

Таблица 4. Сравнение значений суммарной стоимости перевозок, полученных предложенным и оптимальным алгоритмами.

Количество агентов-пунктов назначения	Количество агентов-источников	Разница, %
5	20	18
	50	5,9
	100	4,2
10	20	29
	50	17
	100	27
20	20	19
	50	31
	100	9

Таким образом, превышение суммарных затрат, полученных жадным алгоритмом по сравнению с суммарными затратами оптимального алгоритма, не превышают 31%. Полученное значение согласуется с данными, приведенными в научных публикациях, – в них отклонение от оптимального алгоритма составляет 12-70%. Преимуществом предложенного в настоящей работе алгоритма является возможность его работы в условиях, когда оптимальный алгоритм неприменим – если состав элементов и/или их свойства могут изменяться в процессе решения [119, 120].

4.4. Результаты экспериментальной проверки работы Алгоритма 6 для задачи I

Для изучения того, каким образом использование механизма увеличения оценок успешности влияет на эффективность решения модельной транспортной задачи, было изучено изменение оценок при повторном удовлетворении потребности одного и того же пункта назначения несколькими источниками.

В эксперименте рассматривались один пункт назначения и несколько источников. В результате успешного участия источников в коалициях у них увеличивалась оценка успешности и, следовательно, общая оценка имеющейся у них продукции, а состав и начальные положения источников не изменялось. После нескольких последовательных удовлетворений потребности характеристики источников улучшались. Была изучена зависимость скорости увеличения оценок успешности (количество итераций, потребовавшихся для уменьшения на 1 количества источников в коалиции для удовлетворения потребности неизменного пункта назначения) от значения множителя увеличения оценки успешности (Табл. 5). Результаты пяти независимых испытаний были усреднены.

При увеличении множителя, то есть при большей способности к обучению, количество требуемых повторов, то есть время, нужное для обучения, снижается. Приведенная зависимость позволяет оценить целесообразность дальнейшего улучшения характеристик агентов для минимизации затрат на удовлетворение потребности пункта назначения.

Таблица 5. Зависимость скорости увеличения оценок успешности от значения множителя ее увеличения.

Множитель для увеличения оценки успешности	Количество итераций
1,05	17
1,10	8
1,15	5
1,20	5
1,25	4

Влияние увеличения оценки полезности на суммарную стоимость перевозки при удовлетворении потребности набора пунктов назначения

Для изучения того, каким образом использование механизма увеличения оценок успешности влияет на эффективность решения модельной транспортной задачи, было изучено изменение оценок источниками в процессе удовлетворения потребности нескольких пунктов назначения. В данном случае опыт, полученный при удовлетворении потребности одного пункта назначения, применяется уже для следующего пункта назначения.

В эксперименте рассматривались несколько пунктов назначения и источников. Источники, успешно создавшие коалицию, перемещались в координаты расположения пункта назначения с удовлетворенной потребностью, количество имеющейся у них продукции уменьшалось на значение, которое было потрачено для этого пункта назначения. Для источников из коалиции их оценка полезности увеличивалась на некоторый множитель. В процессе работы вычислялась суммарная стоимость перевозки. Было показано уменьшение суммарной стоимости перевозки после запуска фиксированного набора пунктов потребления при увеличении множителя оценки полезности за успешно созданную коалицию (Табл. 6). Результаты пяти независимых испытаний были усреднены.

Таблица 6. Влияние изменения оценки полезности на суммарную стоимость перевозки

Множитель для увеличения оценки успешности	Суммарная стоимость перевозки
1,00	495
1,10	440
1,15	434
1,20	430
1,25	421

Таким образом, при росте множителя оценки успешности с 1,05 до 1,25 суммарная стоимость перевозки сокращается на 15 %. Это означает, что использование механизма оценок успешности для агентов позволяет без изменения схемы алгоритма получать результаты, более близкие к результатам оптимального алгоритма.

4.5. Результаты тестирования Алгоритма 4 для задачи II

Сравнение результатов применения Алгоритма 4 для решения мультитранспортного варианта задачи маршрутизации (случай II) с точно установленными в научных публикациях оптимальными результатами для этих же наборов входных данных приведено в Табл. 7.

Таким образом, превышение суммарной стоимости перевозки, полученное предложенным Алгоритмом 4 по сравнению с суммарной стоимостью оптимального алгоритма, составляет не более 23%. Полученное значение согласуется с данными, приведенными в научных публикациях, – отклонение от оптимального алгоритма в них составляет 15-45% [121, 24]. Существенным преимуществом предложенного в настоящей работе алгоритма является возможность его работы в условиях, когда оптимальный алгоритм неприменим – если состав элементов и/или их свойства могут изменяться в процессе решения.

Таблица 7. Сравнение значений суммарной стоимости перевозки, полученных предложенным и оптимальным алгоритмами

Название набора	Оптимальный результат [21]	Результат Алгоритма 4	Отклонение, %
eil30	510	588	16
eil33	835	961	15
p01_7090	2109	2451	16
SD1	228	257	13
SD2	708	873	23
SD3	432	492	14
SD4	630	693	10
SD5	1392	1593	14

4.6. Примеры модельных задач

Задача Самолеты-Пожары

Задача состоит в тушении группой беспилотных летательных аппаратов (БПЛА) лесных пожаров, состоящих из нескольких очагов. Каждый БПЛА имеет некоторый запас воды. Каждый пожар имеет порог объема воды, при превышении которого он будет потушен. Порог объема воды для тушения каждого пожара существенно превосходит запасы воды у каждого БПЛА. Группе БПЛА требуется потушить все пожары, минимизируя собственные перемещения.

При решении задачи с помощью предложенных в настоящей работе алгоритмов используются обозначения: БПЛА являются Источниками, пожары – Пунктами назначения, продукция – объем имеющейся воды, элементы матрицы стоимости – евклидовы расстояния между соответствующими БПЛА и пожарами.

Значения начальных входных данных были сгенерированы случайным образом, они представлены на Рис. 13.

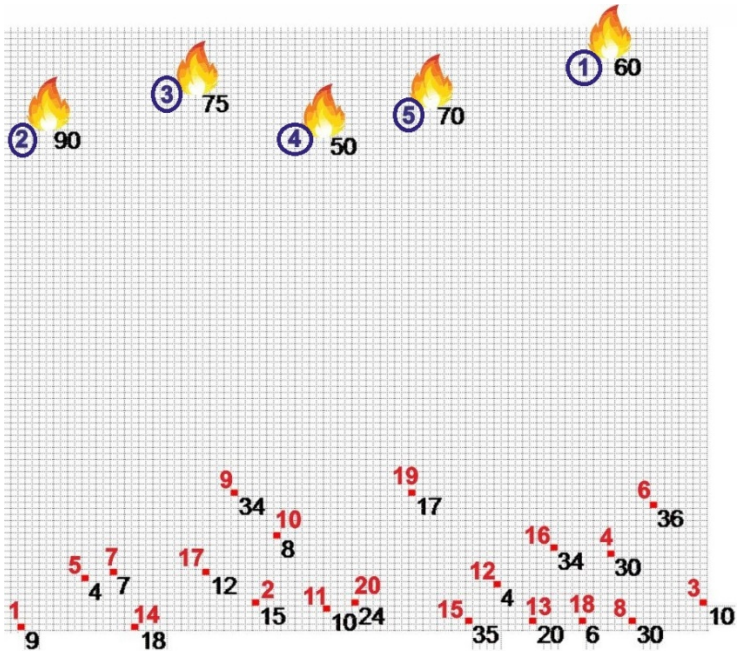
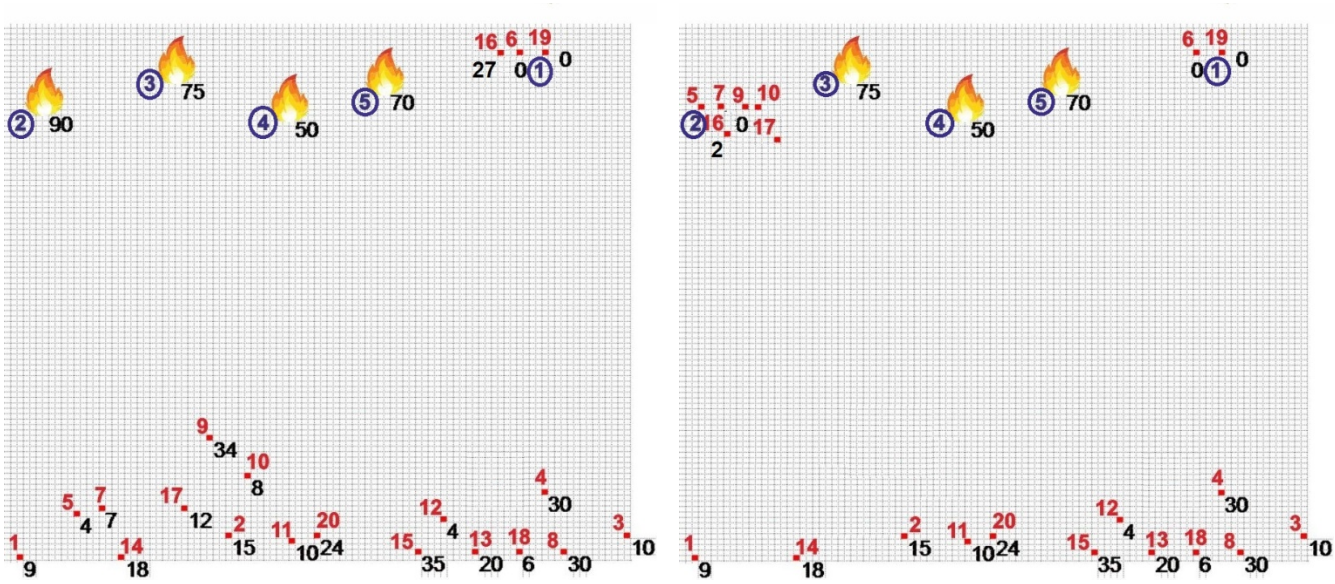


Рис. 13. Расположение и параметры Пожаров и БПЛА

Результаты решения без использования механизма увеличения оценок успешности (значения коэффициентов увеличения оценки успешности составляют C_R=1, C_A=1) представлены ниже на Рисунке 14 и в Таблице 8. В ней указаны номера БПЛА и количество единиц объема воды у БПЛА, которые задействованы в тушении каждого из пожаров. Значение суммарной стоимости перевозки составляет 1685.

Для иллюстрации хода решения задачи на Рисунке 14 показаны положение БПЛА и характеристики пожаров после тушения Пожара 1 (Рис.14,а) и после тушения Пожара 2 (Рис.14,б). На Рис.14,а видно, что после тушения пожара 1 у БПЛА № 6 и БПЛА № 19 запас воды полностью исчерпан, они участия в дальнейших действиях не принимают. У БПЛА № 16 остается еще 27 единиц объема воды. Таким образом, БПЛА № 16 может принять участие в ликвидации следующего пожара.

На Рис.14,б показано положение всех агентов после тушения второго пожара. БПЛА № 16 затратил 25 единиц объема воды на его тушение, у него остается еще 2 единицы объема воды. БПЛА № 5, 7, 9, 10, 17 запас воды при тушении этого пожара полностью исчерпали и участия в дальнейших действиях принять не могут.



а б

Рис. 14. Положение БПЛА и пожаров: а – после тушения пожара 1, б – после тушения пожара 2

Таблица 8. Полученное решение при C_R=1, C_A=1

Пожар	Требуемый объем воды	Номера и **объем воды** у БПЛА
1	60	16:**7**, 19:**17**, 6:**36**
2	90	10:**8**, 16:**25**, 17:**12**, 5:**4**, 7:**7**, 9:**34**
3	75	1:**2**, 11:**10**, 12:**4**, 14:**18**, 16:**2**, 2:**15**, 20:**24**
4	50	1:**7**, 15:**13**, 4:**30**
5	70	13:**20**, 15:**22**, 18:**6**, 8:**22**

Результаты решения с использованием механизма увеличения оценок успешности (значения коэффициентов увеличения оценки успешности составляют C_R=1, C_A=1.2) представлены в Табл. 9. Значение суммарной стоимости превозки составляет 1561. Видно, что состав коалиций изменился и общая суммарная стоимость уменьшилась примерно на 9 %. Таким образом, поставленная задача была решена более эффективно при неизменном составе участников и тех же начальных данных.

Таблица 9. Полученное решение при C_R=1, C_A=1.2

Пожар	Требуемый объем воды	Номера и **объем воды** у БПЛА
1	60	16:**7**, 19:**17**, 6:**36**
2	90	10:**8**, 16:**25**, 17:**12**, 5:**4**, 7:**7**, 9:**34**
3	75	11:**10**, 14:**18**, 16:**8**, 2:**15**, 20:**24**
4	50	12:**4**, 15:**16**, 4:**30**
5	70	13:**20**, 15:**22**, 18:**6**, 8:**22**

Задача Грузовики-Фабрики

Задача состоит в доставке некоторого однотипного исходного сырья грузовиками на фабрики. Каждый грузовик перевозит некоторое количество сырья. На пути грузовиков к фабрикам имеются препятствия. Для описания двумерного мира с препятствиями используется матрица проходимости – бинарная матрица, каждому элементу которой соответствует проходимая («0»), либо непроходимая («1») область на плоскости. Для представления такой модели используется МТ-граф [117]. Для запуска производства фабрике требуется получить сырье, при этом потребность каждой фабрики превосходит грузоподъемность каждого грузовика. Грузовикам требуется запустить все фабрики, доставив на них сырье, минимизируя суммарный пробег.

При решении задачи с помощью предложенных в настоящей работе алгоритмов используются обозначения: грузовики являются Источниками, фабрики – Пунктами назначения, продукция – вес перевозимого сырья, элементы матрицы стоимости – длины кратчайших путей между соответствующими грузовиками и фабриками, найденные при помощи алгоритма A* [118].

Значения начальных входных данных и матрицы проходимости были сгенерированы случайным образом, они представлены в Табл. 10 и на Рис. 15.

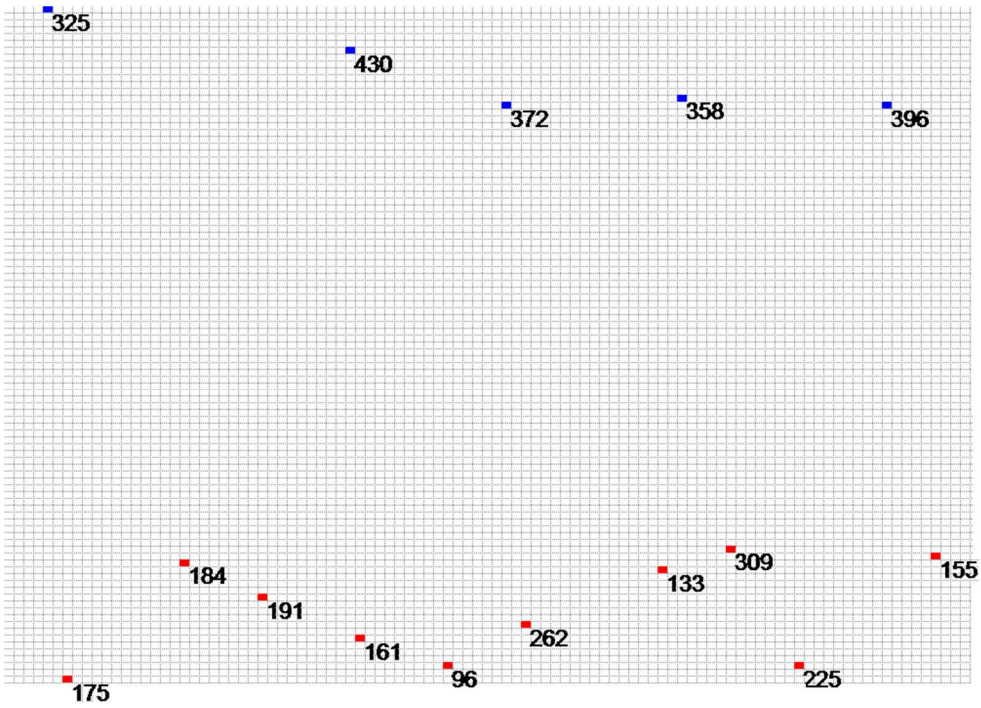


Рис. 15. Расположение и параметры Фабрик и Грузовиков

Пути, найденные при помощи алгоритма A* в мире без препятствий, представлены на Рис 16.

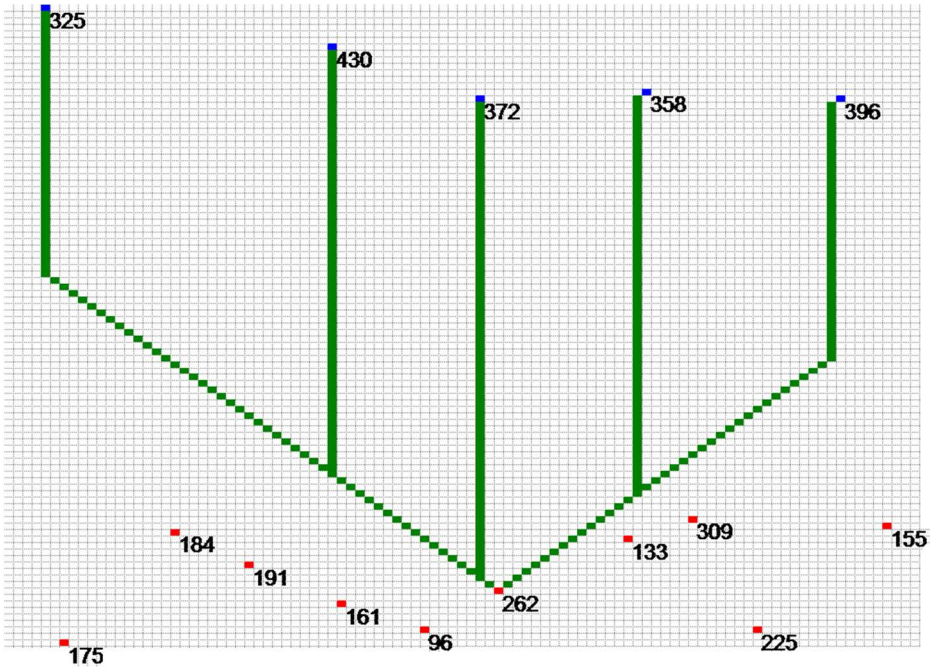


Рис. 16. Полученные пути в мире без препятствий

Пути, найденные при помощи алгоритма A* в мире с препятствиями, представлены на Рис. 17.

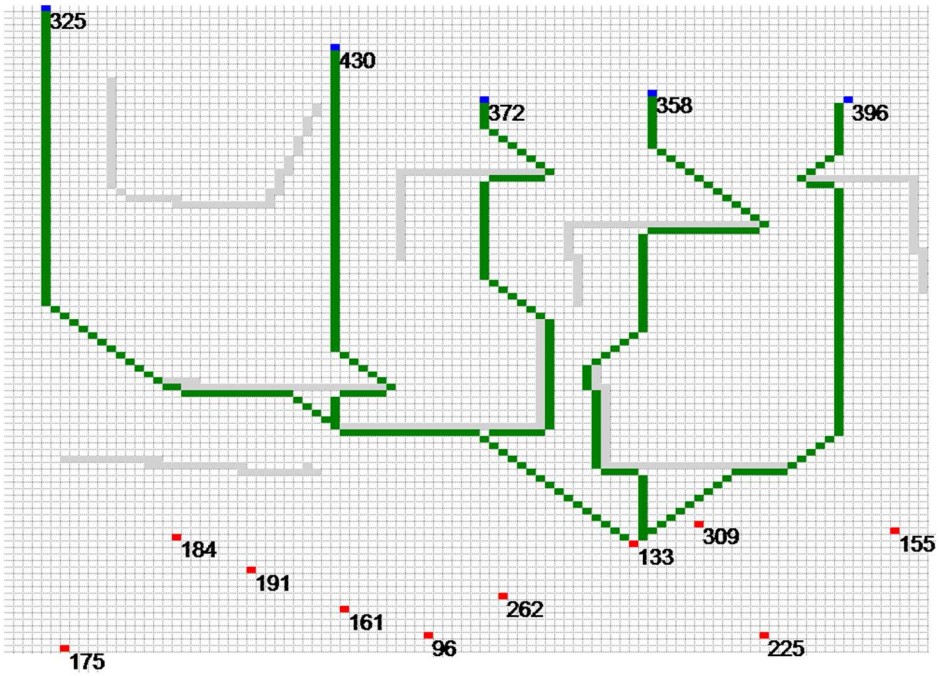


Рис. 17. Полученные пути в мире с препятствиями (препятствия обозначены серым цветом)

Таблица 10. Результаты решения.

Фабрика	Требуемый вес	Номера и **вес сырья** у грузовиков
1	430	1:**191**, 10:**55**, 3:**184**
2	358	10:**207**, 8:**151**
3	325	6:**167**, 8:**158**
4	372	4:**96**, 5:**133**, 6:**8**, 7:**135**
5	396	2:**215**, 7:**26**, 9:**155**

4.7. Выводы

В результате выполнения работы показано, что применение у интеллектуальных агентов способности к изменению оценок успешности позволяет улучшить результаты работы жадного алгоритма формирования групп агентов. Предложенный подход применен к решению транспортной задачи, при

условии когда каждый агент не обладает ресурсами для самостоятельного выполнения задания, и к решению мультитранспортного варианта задачи маршрутизации. Показано, что при увеличении скорости изменения оценок успешности с 1,05 до 1,25, суммарная стоимость перевозки сокращается на 15% - это позволяет получать результаты, более близкие к результатам оптимального алгоритма, без изменения схемы предложенного алгоритма. Важным свойством реализаций предложенных распределенных Алгоритмов 4-6 является возможность изменения количества и характеристик агентов в процессе моделирования. Это позволяет применять предложенные алгоритмы для моделирования задач, приближенных к реальным. Кроме того, вычислительная сложность алгоритмов для каждого агента ниже, чем в централизованном варианте алгоритма, что позволяет в полной мере использовать преимущества мультиагентных систем и современных коммуникационных сетей. Установлено, что время, затраченное на формирование всех коалиций при фиксированном количестве агентов-пунктов назначения и изменении количества агентов-источников, изменяется линейно, что экспериментально подтверждает ранее приведенные теоретические оценки сложности алгоритмов. Показано, что полученная предложенным алгоритмом суммарная стоимость перевозки превышает не более, чем на 31% стоимость перевозки у оптимального алгоритма на входных данных, для которых применим оптимальный алгоритм в случае классической транспортной задачи, и не более, чем на 23% в случае мультитранспортного варианта классической задачи маршрутизации.

Заключение

Решение задач маршрутизации представляет значительный практический интерес, однако основная часть описанных в литературе методов предназначена для статического варианта. Перспективным для решения задач маршрутизации в условиях изменения количества и характеристик агентов в процессе моделирования является применение комбинации мультиагентного подхода, использования системы правил для задания поведения агентов и улучшения характеристик агентов в результате выполнения ими действий.

В работе предложены алгоритм формирования коалиций агентов, алгоритм распределения предпочитаемых действий агентов, обладающих способностью к изменению оценок успешности, и алгоритм формирования коалиций агентов с учетом этих оценок, установлены вычислительные сложности предложенных алгоритмов. Рассмотрены программные реализации распределенного алгоритма формирования коалиций агентов; распределенного алгоритма распределения предпочитаемых действий агентов, обладающих способностью к изменению оценок успешности; распределенного алгоритма формирования коалиций агентов, обладающих способностью к изменению оценок успешности. Установлены характеристики программных реализаций – вычислительная сложность и количество передаваемых сообщений.

Созданный в результате выполнения работы программный комплекс позволяет моделировать поведение до 250 агентов и проводить прототипирование и отладку алгоритмов поведения автономных интеллектуальных технических систем. Отклонение результатов предложенного алгоритма решения задач маршрутизации от оптимального решения, если оно возможно, составляет 23-31% в зависимости от варианта задач маршрутизации. Использование оценок успешности действий агентов позволяет получать результаты, более близкие к результатам оптимального алгоритма, без изменения схемы предложенного алгоритма – при увеличении оценок успешности на 20% результаты решения улучшаются на 15%.

Результаты работы в виде алгоритмов и программ внедрены в учебный процесс кафедры информационных технологий факультета ФМиЕН РУДН и используются в деятельности ООО «РИТЕХ».

В диссертационной работе получены следующие основные результаты.

1. Разработан и исследован коалиционный метод решения задач маршрутизации (классической транспортной задачи и мультитранспортного варианта классической задачи маршрутизации) в рамках агентного подхода в условиях случайного изменения параметров и состава агентов.

2. Разработаны алгоритмы формирования коалиций и решения задач маршрутизации в рамках мультиагентных систем, в которых агенты используют способности к изменению оценок успешности для улучшения своих характеристик в результате деятельности.

3. На основе анализа алгоритмов и вычислительных экспериментов получены их сравнительные характеристики качества. Вычислительные эксперименты демонстрируют, что коалиционный метод позволяет получать практически значимые результаты для задач большой размерности и в условиях случайного изменения характеристик решаемой задачи.

4. Создан программный комплекс (свидетельство о регистрации программы для ЭВМ N 2021617131 от 11.05.2021) для решения задач маршрутизации, обладающий возможностью изменения числа и свойств участвующих в моделировании агентов, на основе мультиагентной платформы *SPADE* и языка программирования *Python*.

5. Показана возможность использования программного комплекса для решения модельных задач маршрутизации в условиях изменения характеристик и состава агентов.

Литература

1. Поспелов Д.А. От моделей коллективного поведения к многоагентным системам. //Программные продукты и системы. 2003. №. 2. С. 39-44.

2. Тарасов В.Б. От многоагентных систем к интеллектуальным организациям: философия, психология, информатика. — М.: Эдиториал УРСС, 2002. — 352 с.

3. Городецкий В.И., Карсаев, О.В., Самойлов В.В., Серебряков С.В. Прикладные многоагентные системы группового управления. //Искусственный интеллект и принятие решений. 2009. №. 2. С. 3-24.

4. Цетлин М.Л. Исследования по теории автоматов и моделированию биологических систем. М: Наука, 1969.

5. Крылов В.Ю., Цетлин М.Л. Об играх автоматов. //Автоматика и телемеханика. 1963. Т.24. № 7. С.975-987.

6. Стефанюк В.Л. Пример задачи на коллективное поведение двух автоматов. //Автоматика и телемеханика. 1963. Т. 24. №. 6. С.781-784.

7. Варшавский В.И. Коллективное поведение автоматов - М. Наука, 1973.

8. Kaminka G.A. Robots are Agents, Too!. // Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems (AAMAS '07). 2007. ACM, New York, NY, USA, Article 4.

9. Abar, S., Theodoropoulos, G. K., Lemarinier, P., O'Hare, G. M. Agent based modelling and simulation tools: a review of the state-of-art software //Computer Science Review. 2017. V. 24. P. 13-33.

10. Классификация систем искусственного интеллекта. Национальный стандарт Российской Федерации ГОСТ Р 59277-2020. – М. Стандартинформ, 2021

11. Осипов Г.С., Панов А.И., Чудова Н.В. Управление поведением как функция сознания. I. Картина мира и целеполагание // Известия Российской академии наук. Теория и системы управления. 2014. № 4. С. 49–62.

12. Осипов Г.С., Панов А.И., Чудова Н.В. Управление поведением как функция сознания. II. Синтез плана поведения // Известия Российской академии наук. Теория и системы управления. 2015. № 6. С. 47–61.

13. Городецкий В.И., Бухвалов О.Л., Скобелев П.О., Майоров И.В. Современное

состояние и перспективы индустриальных применений многоагентных систем. //УБС. 2017. №. 66. С.94-157.
14. Бронштейн Е.М., Заико Т.А. Детерминированные оптимизационные задачи транспортной логистики //Автоматика и телемеханика. 2010. №. 10. С. 133-147.
15. Brenner U. A faster polynomial algorithm for the unbalanced Hitchcock transportation problem //Operations Research Letters. 2008. V. 36. N. 4. P. 408-413.
16. Кривоножко В. Е., Пропой А. И. О методе решения динамических транспортных задач //Автоматика и телемеханика. 1979. №. 12. С. 133-145.
17. Блюмин С. Л., Козлов П. А., Миловидов С. П. Динамическая транспортная задача с задержками //Автоматика и телемеханика. 1984. №. 5. С. 158-161.
18. Dror M., Trudeau P. Split delivery routing //Naval Research Logistics (NRL). 1990. V.37. N.3. P.383-402.
19. Belenguer J. M., Martinez M. C., Mota E. A lower bound for the split delivery vehicle routing problem //Operations research. 2000. V.48. N.5. P.801-810.
20. Chen S., Golden B., Wasil E. The split delivery vehicle routing problem: Applications, algorithms, test problems, and computational results //Networks: An International Journal. 2007. V.49. N.4. P. 318-329.
21. Archetti C., Bianchessi N., Speranza M. G. Branch-and-cut algorithms for the split delivery vehicle routing problem //European Journal of Operational Research. 2014. V.238. N.3. P.685-698.
22. Ghadle K. P., Munot D. A. Recent Advances on Reliable Methods for Solving Transportation Problem and Fuzzy Transportation Problem //Systematic Review Article. 2019. V. 26. N. 2. P. 95-107.
23. Poggi A., Tomaiuolo M. Rule engines and agent-based systems //Machine learning: Concepts, methodologies, tools and applications. IGI Global, 2012. P. 211-218.
24. Doerner, K. F., Gronalt, M., Hartl, R. F., Kiechle, G., Reimann, M. Exact and heuristic algorithms for the vehicle routing problem with multiple interdependent time windows //Computers & Operations Research. 2008. V. 35. N. 9. P. 3034-3048.
25. Осипов Г.С. Методы искусственного интеллекта – М.: Физ-матлит, 2011. – 296 с.

26. Boose J. H. A survey of knowledge acquisition techniques and tools //Knowledge acquisition. 1989. V. 1. №. 1. P. 3-37.
27. Grissa-Touzi A., Ounelli H., Boulila A. VISUAL JESS: AN Expandable Visual Generator of Oriented Object Expert systems //WEC (5). 2005. P. 290-293.
28. Friedman-Hill E.J. Jess, the java expert system shell //Distributed Computing Systems, Sandia National Laboratories, USA. 1997.
29. Friedman-Hill E.J. Jess in Action. Manning Publications Co., 2003.
30. Nowak M., Bak J., Jedrzejek C. Graph-based Rule Editor //RuleML (2). 2012. Proceedings of the RuleML2012@ECAI Challenge and Doctoral Consortium at the 6th International Symposium on Rules, Montpellier, France, August 27th-29th.
31. Palantir Government Platform, http://palantir.com/government
32. Грейди Буч, Джеймс Рамбо, Айвар Джекобсон. Язык UML. Руководство пользователя = The Unified Modeling Language user guide. 2-е изд. М., СПб.: ДМК Пресс, Питер, 2004. 432 с.
33. Nalepa G. J., Kluza K. UML representation for rule-based application models with XTT2-based business rules //International Journal of Software Engineering and Knowledge Engineering. 2012. V. 22. №. 04. P. 485-524.
34. Lukichev S., Wagner G. Visual rules modeling //International Andrei Ershov Memorial Conference on Perspectives of System Informatics. Springer, Berlin, Heidelberg, 2006. P. 467-473.
35. Cragun B. J., Steudel H. J. A decision-table-based processor for checking completeness and consistency in rule-based expert systems //International Journal of Man-Machine Studies. 1987. V. 26. №. 5. P. 633-648.
36. Осипов Г. С. Динамические интеллектуальные системы //Искусственный интеллект и принятие решений. 2008. №. 1. С. 47-54.
37. Панов А. И., Яковлев К. С. Взаимодействие стратегического и тактического планирования поведения коалиции агентов в динамической среде //Искусственный интеллект и принятие решений. 2016. №. 4. С. 68-78.
38. Cohen P. R., Levesque H. J. Teamwork //Nous. 1991. V. 25. № 4. P. 487-512.
39. Rao, A.S. Georgeff, M. BDI Agents: From Theory to Practice. In Proceedings of the

First International Conference on Multiagent Systems (ICMAS), San Francisco, CA, USA, 12–14 June 1995; P. 312–319
40. Jennings N. R. Controlling cooperative problem solving in industrial multi-agent systems using joint intentions //Artificial intelligence. 1995. V. 75. № 2. P. 195-240.
41. Tambe M. Implementing agent teams in dynamic multiagent environments //Applied Artificial Intelligence. 1998. V. 12. № 2-3. P. 189-210.
42. Grant T. J. A review of multi-agent systems techniques, with application to Columbus user support organisation //Future generation computer systems. 1992. V. 7. №. 4. P. 413-437.
43. Смирнов А. В., Шереметов Л. Б. Модели формирования коалиций между кооперативными агентами: состояние и перспективы исследований //Искусственный интеллект и принятие решений. 2011. №. 1. С. 36-48.
44. Smirnov A. V., Sheremetov L., Teslya N. Fuzzy Cooperative Games Usage in Smart Contracts for Dynamic Robot Coalition Formation: Approach and Use Case Description // In Proceedings of the 21st International Conference on Enterprise Information Systems (ICEIS 2019). 2019. P. 361-370.
45. Карпов В. Э. Процедура голосования в однородных коллективах роботов //XIV национальная конференция по искусственному интеллекту с международным участием КИИ-2014 (24-27 октября 2014 г., Казань, Россия): Труды конференции. 2014. Т. 2. С. 159-167.
46. Карпов В. Э. Модели социального поведения в групповой робототехнике //Управление большими системами: сборник трудов. 2016. №. 59. С. 165-232.
47. Скобелев П.О. Интеллектуальные системы управления ресурсами в реальном времени: принципы разработки, опыт промышленных внедрений и перспективы развития // Приложение к теоретическому и прикладному научно-техническому журналу «Информационные технологии». 2013. №1. С. 1–32.
48. Burckert H. J., Muller J., Schupeta A. RATMAN and its relation to other multi-agent testbeds. 1991.
49. Lesser V. R., Erman L. D. A Retrospective View of the Hearsay-II Architecture //IJCAI. 1977. V. 5. P. 790-800.

50. Gasser L., Braganza C., Herman N. Implementing distributed AI systems using MACE //Readings in Distributed Artificial Intelligence. Morgan Kaufmann, 1988. P. 445-450.
51. Barbuceanu M., Fox M. S. COOL: A Language for Describing Coordination in Multi Agent Systems //ICMAS. 1995. P. 17-24.
52. Genesereth, M.R., Fikes, R.E. Knowledge Interchange Format, Version 3.0, Reference Manual, Computer Science Department, Stanford University, Technical Report Logic-92-1.
53. Finin, T., Fritzson, R., McKay, D. and McEntire, R., KQML - An Information and Knowledge Exchange Protocol, in Kazuhiro Fuchi and Toshio Yokoi, editors, Knowledge Building and Knowledge Sharing, Ohmsha and IOS Press, 1994.
54. FIPA ACL Message Structure Specification, http://www.fipa.org/specs/fipa00061/SC00061G.html
55. FIPA Communicative Act Library Specification, http://www.fipa.org/specs/fipa00037/SC00037J.html
56. Gregori M. E., Camara J. P., Bada G. A. A jabber-based multi-agent system platform //Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems. ACM. 2006. P. 1282-1284.
57. Bellifemine, F., Poggi, A., Rimassa, G. JADE–A FIPA-compliant agent framework // Proceedings of PAAM. 1999. P. 97-108.
58. Jabber Software Foundation. Extensible Messaging and Presence Protocol (XMPP): Core. Technical report, http://www.ietf.org/rfc/rfc3920.txt, October 2004
59. Amato, A., Di Martino, B., Scialdone, M., Venticinque, S. Design and evaluation of P2P overlays for energy negotiation in smart micro-grid // Computer Standards & Interfaces. 2016. № 44. P. 159-168.
60. Jacovella, S., Vingerhoets, P., Deconinck, G., Honeth, N., Nordstrom, L. Multi-Agent platform for Grid and communication impact analysis of rapidly deployed demand response algorithms // 2016 IEEE International Energy Conference. 2016. P. 1-6.
61. Monteiro, J., Eduardo, J., Cardoso, P. J., Semiao, J. A distributed load scheduling mechanism for micro grids // 2014 IEEE International Conference on Smart Grid

Communications. 2014. P. 278-283.
62. Jiang, S., Venticinque, S., Horn, G., Hallsteinsen, S., Noebels, M. A distributed agent-based system for coordinating smart solar-powered microgrids // 2016 SAI Computing Conference. 2016. P. 71-79.
63. Pijoan, A., Kamara-Esteban, O., Borges, C. E. Environment modelling for spatial load forecasting // Agent Environments for Multi-Agent Systems IV. 2015. P. 188-206.
64. Van De Vijsel, M., Anderson, J. Coalition formation in multi-agent systems under real-world conditions // Proceedings of association for the advancement of artificial intelligence. 2004. P. 54-60.
65. Farinelli, A., Bicego, M., Ramchurn, S.D., Zucchelli, M. C-Link: A Hierarchical Clustering Approach to Large-scale Near-optimal Coalition Formation // IJCAI. 2013. P. 106-112.
66. Yeh D. Y. A dynamic programming approach to the complete set partitioning problem //BIT Numerical Mathematics. 1986. V. 26. №. 4. P. 467-474.
67. Larson, K. S., Sandholm, T. W. Anytime coalition structure generation: an average case study //Journal of Experimental & Theoretical Artificial Intelligence. 2000. V. 12. N. 1. P. 23-42.
68. Rahwan, T., Ramchurn, S.D., Giovannucci, A., Dang, V.D., Jennings, N. R. Anytime optimal coalition structure generation. //AAAI. 2007. V. 7. P. 1184-1190.
69. Rahwan, T., Ramchurn, S. D., Jennings, N. R., Giovannucci, A. An anytime algorithm for optimal coalition structure generation //Journal of artificial intelligence research. 2009. V. 34. P. 521-567.
70. Farinelli, A., Bicego, M., Bistaffa, F., Ramchurn, S. D. A hierarchical clustering approach to large-scale near-optimal coalition formation with quality guarantees //Engineering Applications of Artificial Intelligence. 2017. V. 59. P. 170-185.
71. Sandholm, T., Larson, K., Andersson, M., Shehory, O., Tohme, F. Coalition structure generation with worst case guarantees //Artificial Intelligence. 1999. V. 111. №. 1. P. 209-238.
72. Adams, J., Service, T. Constant factor approximation algorithms for coalition structure generation //Autonomous Agents and Multi-Agent Systems. 2011. V. 23. №. 1.

P. 1-17.
73. Rahwan, T., Jennings, N. R. An algorithm for distributing coalitional value calculations among cooperating agents //Artificial Intelligence. 2007. V. 171. №. 8-9. P. 535-567.
74. Rahwan T., Jennings N. R. An improved dynamic programming algorithm for coalition structure generation //Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 3. International Foundation for Autonomous Agents and Multiagent Systems, 2008. P. 1417-1420.
75. Pawlowski, K., Kurach, K., Svensson, K., Ramchurn, S.D., Michalak, T.P., Rahwan, T. Coalition structure generation with the graphics processing unit // Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems. 2014. P. 293-300.
76. Rahwan, T., Jennings, N. Coalition structure generation: Dynamic programming meets anytime optimisation. // In Proceedings of the Twenty Third Conference on Artificial Intelligence (AAAI-08). 2008. P. 156–161.
77. Michalak, T., Sroka, J., Rahwan, T., Wooldridge, M., McBurney, P., Jennings, N.R. A distributed algorithm for anytime coalition structure generation. //In Autonomous Agents And MultiAgent Systems (AAMAS 2010). 2010. P.1007–1014.
78. Rahwan T., Michalak T. P., Jennings N. R. Minimum search to establish worst-case guarantees in coalition structure generation //Twenty-Second International Joint Conference on Artificial Intelligence. 2011. P. 338-343.
79. Tran-Thanh, L., Nguyen, T. D., Rahwan, T., Rogers, A., Jennings, N. R. An efficient vector-based representation for coalitional games //Twenty-Third International Joint Conference on Artificial Intelligence. 2013.
80. Michalak, T., Rahwan, T., Elkind, E., Wooldridge, M., Jennings, N. R. A hybrid exact algorithm for complete set partitioning //Artificial Intelligence. 2016. V. 230. P. 14-50.
81. Rahwan, T., Ramchurn, S.D., Jennings, N.R., Giovannucci, A. An anytime algorithm for optimal coalition structure generation // Journal of Artificial Intelligence Research. 2009. № 34. P. 521-567.
82. Ohta, N., Iwasaki, A., Yokoo, M., Maruono, K., Conitzer, V., Sandholm, T. A

compact repre- sentation scheme for coalitional games in open anonymous environments //AAAI. 2006. V. 6. P. 21.
83. Ohta, N., Conitzer, V., Ichimura, R., Sakurai, Y., Iwasaki, A., Yokoo, M. Coalition structure generation utilizing compact characteristic function representations //In International Conference on Principles and Practice of Constraint Programming. 2009. P. 623-638.
84. Voice, T., Ramchurn, S.D., Jennings, N.R. On coalition formation with sparse synergies // Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems. 2012. № 1. P. 223-230.
85. Chalkiadakis, G., Greco, G., Markakis, E. Characteristic function games with restricted agent interactions: core-stability and coalition structures // Artificial Intelligence. 2016. № 232. P. 76-113.
86. Di Mauro, N., Basile, T.M., Ferilli, S., Esposito, F. Coalition structure generation with GRASP // International Conference on Artificial Intelligence: Methodology, Systems, and Applications. 2010. P. 111-120.
87. Liu, Y., Zhang, G. F., Su, Z. P., Yue, F., Jiang, J. G. Using computational intelligence algorithms to solve the coalition structure generation problem in coalitional skill games //Journal of Computer Science and Technology. 2016. V. 31. №. 6. P. 1136-1150.
88. Khalouzadeh L., Nematbakhsh N., Zamanifar K. A decentralized coalition formation algorithm among homogeneous agents //Journal of Theoretical & Applied Information Technology. 2010. V. 22. N. 1. P. 36-42
89. Partsakoulakis I., Vouros G. Importance and Properties of Roles in MAS Organization: A review of methodologies and systems //Proceedings of the workshop on MAS Problem Spaces and Their Implications to Achieving Globally Coherent Behavior. 2002.
90. Kinny D., Georgeff M., Rao A. A methodology and modelling technique for systems of BDI agents //European workshop on modelling autonomous agents in a multi-agent world, LNAI 1038. 1996. P. 56-71.
91. Wooldridge M., Jennings N. R., Kinny D. The Gaia methodology for agent-oriented analysis and design //Autonomous Agents and multi-agent systems. 2000. V. 3. №. 3. P.

285-312.
92. DeLoach S.A., Wood M. Developing multiagent systems with agentTool //International Workshop on Agent Theories, Architectures, and Languages, LNAI 1986. 2001. P. 46-60.
93. Derakhshan F., Bench-Capon T., McBurney P. Dynamic assignment of roles, rights and responsibilities in normative multi-agent systems //Journal of Logic and Computation. 2013. V. 23. №. 2. P. 355-372.
94. Ward C.B., Henderson-Sellers B. Utilizing dynamic roles for agents //Journal of Object Technology. 2009. V. 8. № 5. P. 177 – 198.
95. Cabri G., Ferrari L., Leonardi L. Exploiting runtime bytecode manipulation to add roles to Java agents //Science of Computer Programming. 2005. Т. 54. №. 1. С. 73-98.
96. Ferrari L., Zhu H. Enabling dynamic roles for agents //2011 International Conference on Collaboration Technologies and Systems (CTS). IEEE, 2011. P. 500-507.
97. Осипов Г.С., Панов А.И. Отношения и операции в знаковой картине мира субъекта поведения //Искусственный интеллект и принятие решений. 2017. №. 4. С. 5.
98. Осипов Г.С. Целенаправленное поведение коалиции когнитивных агентов // Гибридные и синергетические интеллектуальные системы: сб. матер. IV Всерос. конф., Калининград, 2018. С. 81–85.
99. Panov A.I. Behavior planning of intelligent agent with sign world model //Biologically inspired cognitive architectures. 2017. V. 19. P. 21-31.
100. Киселев Г.А., Панов А.И. Знаковый подход к задаче распределения ролей в коалиции когнитивных агентов // Тр. СПИИРАН. 2018. № 57. С. 161–187.
101. Корбут А.А., Финкельштейн Ю.Ю. Дискретное программирование. - М.: Наука, 1969. - 368 с.
102. Lau H. C., Zhang L. Task allocation via multi-agent coalition formation: Taxonomy, algorithms and complexity //Proceedings. 15th IEEE International Conference on Tools with Artificial Intelligence. IEEE, 2003. P. 346-350.
103. Karabati S., Kouvelis P., Yu G. A min-max-sum resource allocation problem and its applications. Operations Research. 2001. V.49. N.6. P. 913-922.

104. Штильман Б.М. Формально-лингвистическая модель для решения задач дискретной оптимизации. I. Инструментарий формализации. Язык траекторий //Изв. АН СССР Техн. кибернет. 1985. № 3. С. 110-122.
105. Штильман Б.М. Формально-лингвистическая модель для решения задач дискретной оптимизации. II. Язык зон, переводы и проблема границ //Изв. АН СССР Техн. кибернет. 1985. № 4. С. 10-21.
106. Stilman B. A linguistic approach to geometric reasoning //Computers & Mathematics with Applications. 1993. V. 26. №. 7. P. 29-57.
107. Stilman B. Linguistic geometry: methodology and techniques //Cybernetics and System. 1995. V. 26. №. 5. P. 535-597.
108. Горященко А.С. Разработка модуля задания правил продукционной системы // в сб. трудов IV Всероссийской научной конференции молодых ученых с международным участием ИУСА-2016, Т.1, С. 105-109.
109. Frederiksen B. Applying expert system technology to code reuse with Pyke //Proceedings of the conference on the Python Conference (PyCon). 2008.
110. http://svn.python.org/projects/python/trunk/Objects/listsort.txt
111. Auger, N., Juge, V., Nicaud, C., Pivoteau, C. On the worst-case complexity of TimSort // 26th Annual European Symposium on Algorithms (ESA 2018). 2018. P. 4:1–4:13.
112. Hoffman A. J. On simple linear programming problems //Proceedings of Symposia in Pure Mathematics. 1963. V. 7. P. 317-327.
113. Alon, N., Cosares, S., Hochbaum, D. S., & Shamir, R. An algorithm for the detection and construction of Monge sequences //Linear Algebra and its Applications. 1989. V. 114. P. 669-680.
114. Shamir R. A fast algorithm for constructing Monge sequences in transportation problems with forbidden arcs //Discrete mathematics. 1993. V. 114. N. 1-3. P. 435-444.
115. Estes A. S., Ball M. O. Monge Properties, Optimal Greedy Policies, and Policy Improvement for the Dynamic Stochastic Transportation Problem //INFORMS Journal on Computing, Articles in Advance, 2020. P. 1–23.
116. Such, J.M., Alberola, J.M., Mulet, L., Espinosa, A., Garcia-Fornes, A., Botti, V.

Large-scale multiagent platform benchmarks // LADS. 2007. P. 192-204.
117. Яковлев, К.С., Баскин, Е.С. Графовые модели в задаче планирования траектории на плоскости // Искусственный интеллект и принятие решений. 2013. № 1. С. 5-12.
118. Hart, P.E., Nilsson, N.J., Raphael, B. A Formal Basis for the Heuristic Determination of Minimum Cost Paths // Systems Science and Cybernetics, IEEE Transactions 1968. V. 4. N 2. P. 100-107.
119. Sampurno G. I., Sugiharti E., Alamsyah A. Comparison of Dynamic Programming Algorithm and Greedy Algorithm on Integer Knapsack Problem in Freight Transportation //Scientific Journal of Informatics. 2018. V. 5. N. 1. P. 49.
120. Gu, Z., Zhu, Y., Wang, Y., Du, X., Guizani, M., Tian, Z. Applying artificial bee colony algorithm to the multidepot vehicle routing problem //Software: Practice and Experience. 2022. V. 52. N. 3. P. 756-771.
121. Wang K., Ye C., Ning A. Competitive decision algorithm for the split vehicle routing problem with simultaneous pickup and delivery and time windows //2010 International Conference on Future Information Technology and Management Engineering. IEEE, 2010. V. 2. P. 371-375.

Приложение 1. Свидетельство о регистрации программы для ЭВМ

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2021617131

Программный комплекс для решения распределительных задач с использованием коалиций интеллектуальных агентов

Правообладатель: ***Горященко Алексей Сергеевич (RU)***

Автор(ы): ***Горященко Алексей Сергеевич (RU)***

Заявка № **2021616341**
Дата поступления **26 апреля 2021 г.**
Дата государственной регистрации
в Реестре программ для ЭВМ ***11 мая 2021 г.***

Руководитель Федеральной службы
по интеллектуальной собственности

Г.П. Ивлиев

Приложение 2. Акт о внедрении в учебный процесс РУДН

УТВЕРЖДАЮ
Декан Факультета физико-
математических
и естественных наук Российского
университета дружбы народы

_____ Л.Г.Воскресенский

«___» 01.04. 2022 г.

АКТ

об использовании (внедрении) результатов диссертационной работы Горященко Алексея Сергеевича на тему «Оптимизация задач маршрутизации на основе взаимодействующих интеллектуальных транспортных агентов» в учебный процесс Российского университета дружбы народов (РУДН)

Настоящим актом подтверждаем, что в учебный процесс кафедры информационных технологий РУДН внедрены следующие результаты кандидатской диссертации Горященко А.С.:

1. Коалиционный метод решения задач маршрутизации в рамках мультиагентного подхода в условиях случайного изменения параметров агентов.
2. Алгоритмы формирования коалиций и решения задач маршрутизации, в которых агенты используют оценки успешности для повышения своих способностей.

Перечисленные результаты используются при чтении дисциплин «Интеллектуальные системы», «Моделирование вычислительных систем», при выполнении научно-исследовательских, курсовых и выпускных квалификационных работ.

Зав. кафедрой информационных технологий,
д.ф.-м.н., профессор Ю.Н.Орлов

к.ф.-м.н., доцент М.Б.Фомин

Приложение 3. Акт о внедрении результатов в ООО «РИ Технологии»

ИНТЕРНЕТ ТЕХНОЛОГИИ

Общество с ограниченной ответственностью
«РИ ТЕХНОЛОГИИ»
121205, Москва, инновационный Центр «Сколково»,
Большой бульвар, д. 42, стр.1.
Тел./факс: +7(499) 135-51-45 www.ritech.ru

Исх. № 56-22 от 21.03.2022г.

Утверждаю
Генеральный директор
ООО «РИТЕХ»
_____ Рыбкина Е.А.

АКТ
об использовании (внедрении) результатов

Результаты диссертационной работы Горященко Алексея Сергеевича на тему «Оптимизация задач маршрутизации на основе взаимодействующих интеллектуальных транспортных агентов» использованы в научно-аналитической деятельности Общества с ограниченной ответственностью «РИ Технологии» (ООО «РИТЕХ»).

Прошел успешную апробацию и принят в эксплуатацию разработанный Горященко А.С. метод решения задач маршрутизации на основе интеллектуальных агентов и их коалиций в условиях, когда характеристики агентов в процессе моделирования могут меняться. Это позволило существенно расширить класс решаемых задач маршрутизации с переменным составом элементов и их свойств.

Разработанный Горященко А.С. программный комплекс, предназначенный для моделирования решения распределительных задач большой размерности, позволяет проводить прототипирование и отладку алгоритмов поведения автономных интеллектуальных технических систем, в режиме времени близком к реальному.

Планируется в дальнейшем использовать результаты диссертационной работы Горященко А.С. в ООО «РИТЕХ» для решения практических задач логистики, распределения электроэнергии и моделирования чрезвычайных ситуаций.

Председатель комиссии,
директор по информационным технологиям,
к.ф.-м.н. Соченков И.В.

Члены комиссии:

В.н.с., к.т.н. Латышев А.В.

Н.с. Девяткин Д.А.