Міністерство освіти і науки України
Запорізький державний університет

ДО ЗАХИСТУ ДОПУЩЕНИЙ
Зав. каф. інформаційних технологій, доц.
_____ **С. Ю. Борю**
(підпис)
_____
(дата)

ДИПЛОМНА РОБОТА
МОДЕЛЮВАННЯ ВЗАЄМОДІЇ АВТОНОМНИХ РОБОТІВ ПРИ ЗБОРІ РАДІОАКТИВНИХ ВІДХОДІВ

Виконав
ст. групи 8229-1 А. О. Віротченко
(шифр) (підпис і дата) (ім'я, по батькові, прізвище)

Керівник доцент В. А. Єрмолаєв
(посада) (підпис і дата) (ім'я, по батькові, прізвище)

Нормо контролер
доцент І. А. Костюшко
(посада) (підпис) (ім'я, по батькові, прізвище)

Запоріжжя
2004

СОДЕРЖАНИЕ

РЕФЕРАТ

Дипломная работа: 57 страниц, 4 рисунка, 6 таблиц, 2 диаграммы, 1 приложение, 38 источников.

Цель работы – разработка модели кооперативного поведения группы автономных роботов, осуществляющих сбор радиоактивных отходов, программная реализация полученной модели и её анализ.

При выполнении работы в качестве платформы использована модель исследования пространства Стилза, программный инструментарий Delphi 6, а так же основные понятия мульти-агентских систем.

В результате работы построена модель кооперативного поведения группы автономных роботов, осуществляющих сбор радиоактивных отходов, построена модель окружающей среды, в которой проводится сбор радиоактивных отходов, выполнена программная реализация моделей и проведён анализ работы алгоритмов.

ВВЕДЕНИЕ

В данной работе мы рассмотрим возможность ликвидации последствий аварий на атомных электростанциях с помощью мобильных роботов. А именно: мы попытаемся создать модель поведения для автономных роботов, которые, взаимодействуя между собой, будут заниматься поиском и сбором радиоактивных отходов на некоторой загрязнённой территории.

Возникают вопросы: чем интересна эта проблема и актуальна ли она на сегодняшний день? Актуальность затрагиваемого вопроса очевидна. Каждому известно, что “мирный атом” широко используется в современной энергетике, а с проблемой ядерных катастроф человечество уже сталкивалось. Поэтому, несмотря на высокий уровень технологий и рост эффективности и надёжности систем безопасности на АЭС, всё ещё существует вероятность катастроф и возникновения аварийных ситуаций на них.

Сразу же отметим, что технические, физические и прочие характеристики ядерных реакторов, энергоблоков и АЭС нас не интересуют. Наша цель – создание алгоритма, следуя которому, система, состоящая из множества мобильных роботов, достигнет кооперативного поведения. Формулировка и более подробное описание терминов агент, кооперация, мобильность и др. будут даны в следующей главе.

Интерес к данному вопросу связан так же с тем, что наша проблема относится к широкому классу задач исследования и покрытия площади региона мобильными роботами. Такое применение робототехники используется, в основном, в двух случаях: если использование роботов уменьшает затраты времени и средств или в случае, когда работа и вмешательство человека являются невозможными (нежелательными) в некоторых условиях. Данный класс задач включает в себя исследование

поверхностей планет, дна морей, поиск полезных ископаемых, изучение вулканов, поиск и обезвреживание взрывчатых устройств, исследование и обезвреживание регионов, зараженных опасными веществами, и многие другие.

Так как многие из перечисленных выше задач уже давно попали в поле зрения учёных и исследователей, то не удивительно, что на сегодняшний день существует множество алгоритмов, решающих поставленные проблемы тем или иным образом. В данной работе будут приведены и описаны некоторые из них, будут указаны их достоинства и недостатки. В итоге, на их основании мы попытаемся синтезировать модель кооперативного поведения для рассматриваемого нами сценария.

Отметим так же, что основной и, возможно, главной трудностью на пути к решению данной проблемы является моделирование окружающей среды (реального мира). Проблема состоит в том, что при столкновении с некоторой ситуацией, реакция на которую не предусмотрена, робот должен как-то реагировать. Из этого следует, что мы должны максимально подробно сформулировать постановку задачи и наш алгоритм должен уметь приспосабливаться к окружающей среде.

Теперь пришло время определиться с методами, при помощи которых мы будем решать стоящую перед нами задачу. Наиболее подходящим среди существующих на сегодняшний день инструментов является применение мульти-агентских технологий.

Необходимо обратить внимание на то, что мульти-агентские системы являются новым направлением, но в нём уже развита строго формализованная система определений. Базовым понятием, которое будет использоваться в нашей работе, является понятие агент. Так же мы определим свойства, которыми агент должен обладать.

Агент – есть компьютерная система, способная к автономному действию в некоторой окружающей среде [1].

Интеллектуальный агент – компьютерная система, способная к гибким автономным действиям в некоторой окружающей среде [1].

Под гибкими действиями мы понимаем:

Автономность: агенты могут решать проблемы и выполнять задания без прямого вмешательства людей или других агентов, а также они способны демонстрировать контроль над своим внутренним состоянием.

Реактивность: Способность агента поддерживать взаимодействие с окружающей средой и реагировать на происходящие в ней изменения.

Про-активность: способность агентов действовать не просто в ответ на изменения в окружающей среде, а также проявлять инициативу. Демонстрировать поведение, управляемое поставленной целью (целями).

Социальный характер поведения: агент способен сотрудничать с другими агентами или людьми с целью решения собственных проблем.

В дополнение к этим необходимым свойствам некоторые авторы [1] предлагают несколько других, потенциально желаемых, характеристик. Они включают:

Мобильность – способность агента изменять своё физическое расположение для повышения эффективности решения проблемы.

Адаптация – способность агента изменять своё поведение со временем в ответ на свойства окружающей среды. Увеличение познаний о решаемой проблеме.

Правдивость – агент не станет сознательно передавать ложную информацию.

Моделируемые нами агенты будут действовать в реальном мире, который является динамическим.

Динамическая среда – это такая среда, над которой выполняются другие процессы и которая, таким образом, изменяется неподконтрольно данному агенту.

Выше нами было упомянуто, что моделируемые нами роботы (агенты) должны неким образом взаимодействовать между собой. Данное

взаимодействие необходимо для повышения эффективности в работе системы и носит название кооперации. Словарь Вебстера [2] определяет “кооперацию” как “объединение с другим, или другими, для взаимной, часто экономической выгоды”. В роботике основными определениями кооперации являются следующие:

1. “Объединение совместных поведений, направленных к некоторой цели, к которой проявлен обычный интерес или обещана награда” [3];
2. “Форма взаимодействия, обычно основанная на связи” [4];
3. “[объединение] вместе для выполнения каких-либо действий, которое улучшает результат, повышая качество выполнения задачи или экономя время” [5].

Таким образом, в данных нами определениях можно выделить три рациональных зерна: задание, механизм кооперации и систему выполнения.

Мы определим кооперативное поведение следующим образом: Даётся некоторая задача, определяемая проектировщиком, мульти-агентская система показывает кооперативное поведение, если некоторый механизм (т.е. механизм кооперации) увеличивает общую пользу системы.

Интуитивно примитивное кооперативное поведение влечёт за собой некоторый тип действий, приносящий выгоду. Механизм кооперации может быть создан разработчиком, как контроль или структура связи в аспектах спецификации задачи, в поведении динамического взаимодействия агентов и т.д.

Данных понятий на начальной стадии исследования должно быть достаточно. Более узкие и специализированные определения будут даваться в ходе изложения материала.

Обладая начальными сведениями об исследуемой нами проблеме и получив некоторое представление об инструментарии, пользуясь которым, мы будем решать поставленную задачу, можно перейти к подробной формулировке задания.

Итак, группа из нескольких (возможно одного) мобильных роботов будет заниматься поиском и сбором радиоактивных отходов на некоторой территории.

Сформулируем задачу следующим образом:

Во-первых, ландшафт местности неизвестен заранее, так как в результате взрыва он сильно изменился. Более того, он продолжает изменяться потому, что не исключена возможность дальнейших разрушений. Следовательно, создание агентами карты местности не имеет смысла.

Во-вторых, повышенный уровень радиации препятствует наличию какой-либо радиосвязи и централизованного управления. Следовательно, механизм взаимодействия должен опираться на другие методы (методы непрямой связи).

В-третьих, обнаружив и подобрав искомый предмет, являющийся источником радиации, агент должен доставить его в некоторую точку, в которой будет находиться ёмкость, предназначенная для помещения в неё радиоактивных отходов. Не исключено, что источники радиоактивного загрязнения могут находиться в скоплениях.

Итак, мы сформулировали постановку задачи. Проблема, которую мы рассматриваем, состоит в построении формальной модели автономного программного агента и сообщества программных агентов, которые будут управлять роботами настолько эффективно, насколько это возможно. Для этого нам понадобится оптимальная модель кооперативного поведения, которую можно использовать в рассматриваемой нами окружающей среде. С этой целью мы рассмотрим организацию кооперации с помощью различных методов.

1 МОДЕЛИ И МЕХАНИЗМЫ КООПЕРАЦИИ

Необходимо понимать, что выбор механизма кооперации играет ключевую роль в решении нашей проблемы, потому что именно от эффективности кооперации будет зависеть эффективность решения задачи. Существует несколько подходов к организации кооперативного поведения. Мы должны изучить и понять их для создания нашего собственного алгоритма. Иными словами, мы будем опираться на опыт и знания других разработчиков при решении нашей проблемы.

Поиск механизма кооперации есть поиск модели кооперативного поведения. Задача такого поиска может формулироваться следующим образом: Дана группа роботов, окружающая среда и задание, как организовать кооперативное поведение?

Для решения данной задачи следует проработать следующие аспекты [6].

Во-первых, реализация кооперативного поведения должна опираться на некоторую модель социальной инфраструктуры. Она охватывает такие концепции, как разнородность/однородность роботов, структура общения и др.

Во-вторых, необходимы механизмы для множественного сосуществования роботов в общей окружающей среде, для управления объектами в окружающей среде и для возможности общения друг с другом.

Третий параметр исследования – происхождение кооперации, обращён к тому, каким образом достигается кооперативное поведение.

Так как адаптация и гибкость являются основными чертами при решении проблемы группой роботов, мы рассмотрим обучение, как четвёртый ключ к достижению кооперативного поведения.

Рассмотрим данные аспекты более детально.

1.1 Социальная инфраструктура

Социальная инфраструктура кооперативной системы роботов обеспечивает инфраструктуру, которая определяет коллективное поведение и определяет способности и ограничения системы. Мы сейчас кратко обсудим некоторые ключевые особенности социальной инфраструктуры мобильных роботов: централизация/децентрализация, дифференциация, связь и способность симуляции других агентов.

1.1.1 Централизация/децентрализация

Наиболее фундаментальным решением, искомым при определении групповой архитектуры, является централизация или децентрализация системы, и, если система децентрализована, то иерархична она или распределена. Централизованная архитектура характеризуется единственным агентом, осуществляющим контроль. Децентрализованная архитектура не имеет такого агента. Существует два типа децентрализованной архитектуры: распределённая архитектура, в которой все агенты равны в отношении контроля, и иерархические архитектуры, которые централизованны локально.

В настоящее время доминирующей парадигмой является децентрализованный подход. Поведение децентрализованных систем часто описывается с помощью термина “самоорганизация”. Он широко используется в децентрализованных архитектурах, например, [4, 7, 8, 9]. Децентрализованные архитектуры имеют несколько преимуществ над централизованными: большая устойчивость к ошибкам, естественное использование параллелизма, надёжность и масштабируемость.

1.1.2 Дифференциация

Мы назовём группу роботов гомогенными, если их индивидуальные способности идентичны и гетерогенными в противоположном случае. В основном, гетерогенность представляется комплексными научными заданиями, для которых распределение является более сложным. В литературе в настоящее время доминируют работы, которые представляют гомогенные группы роботов. Однако некоторые известные архитектуры могут манипулировать гетерогенными, например, ACTRESS и ALLIANCE [10]. В гомогенных группах задание распределения может быть определено индивидуальной способностью, но в гетерогенных системах агенты могут нуждаться в дифференциации на различные роли, которые могут быть известны во время моделирования и повышают динамику во время работы программы.

1.1.3 Структура взаимодействия

Структура взаимодействия группы агентов определяется возможными режимами взаимодействия между агентами. Мы характеризуем три типа взаимодействия, которые являются наиболее распространёнными.

Взаимодействие через окружающую среду

Простейший и наиболее ограниченный тип взаимодействий возникает, когда окружающая среда сама является средой общения (эффект общей памяти) и нет явной связи между агентами. Системы, которые основаны на данной форме взаимодействия, включают [7, 8, 11, 12, 13, 14, 15].

Взаимодействие через ощущение

Передача на “длину рук” – отношение в теории организации [16], оно относится к локальным взаимодействиям, которые возникают между агентами как результат ощущения ими друг друга (явная связь). Коллективные поведения, которые могут использовать этот тип

взаимодействия, основаны на сгруппированости (“стадности”) и формировании образца (сохранение схожести с другими агентами в группе). Этот тип взаимодействия требует, чтобы агенты могли видеть различие между другими агентами и объектами в окружающей среде.

Взаимодействие через связь

Третья форма взаимодействия включает явную двустороннюю связь с другими агентами путём радиопередачи специальных сообщений. Так как архитектуры, использующие данный вид связи, подобны сетям, то возникает много стандартных проблем из области сетей, среди которых проблемы топологии сети и протоколов коммуникации.

1.1.4 Симуляция поведения других агентов

Симуляция намерений, желаний, действий, способностей и состояний других агентов может способствовать более эффективной кооперации между роботами. Если каждый агент наделён способностью симуляции других агентов, то наличие коммуникации становится менее необходимым. Заметим, что симуляция поведения других агентов влечёт за собой неявную связь через окружающую среду или восприятие. Симуляция требует, чтобы проектировщик имел некоторое представление об окружающих агентах и, чтобы это представление могло быть использовано для получения выводов о действиях других агентов.

1.2 Конфликт ресурса

Когда единственный неделимый ресурс запрашивается множеством агентов, то возникает конфликт ресурса. Эта проблема была изучена с различных точек зрения, особенно серьёзно была изучена проблема

алгоритмов взаимного исключения и мультидоступа в компьютерных сетях. Для множества роботов конфликт ресурса возникает, когда им необходимо разделить пространство, управлять объектами или они используют общий носитель связи. Поэтому мы сосредоточимся на проблеме разделения пространства (“контроль движения” - более точная формулировка), которая, прежде всего, решается путём мульти-агентского планирования проблем уклонений от столкновений и попаданий в тупики. В мульти-агентской системе каждый робот, очевидно, может планировать свой путь, который могут просчитать и другие роботы. Агенты также могут учитывать глобальную окружающую среду, конфигурации с параметрами простраенства-времени, явные модели других агентов и т.п. [17, 18]. Однако, исследования, проведённые на реальных мульти-агентских системах, как правило, показывают, что предварительное планирование пути невозможно. Поэтому, для упрощения задачи, роботов часто ограничивают предписанными путями или движением по правилам (подобно правилам дорожного движения в человеческом мире). Коммуникация при этом используется для избежания столкновений и попаданий в тупики [19, 20].

Гросман [21] классифицировал случаи проблемы контроля движения на три типа: (I) ограничение дорог, (II) возможность существования множества дорог, среди которых могут осуществлять свой выбор автономные роботы, (III) существование множества дорог с централизованным контролем движения роботов.

Когда индивидуальные агенты создают уникальный путь из одной точки в другую, то конфликт невозможен. В случае глобальных знаний и централизованного контроля легко предотвратить конфликт. Поэтому интересен случай (II), где роботам дана возможность самостоятельно выбирать дороги.

1.3 Происхождение кооперации

В большинстве работ кооперация явно моделируется при проектировании системы. Интересно рассмотрение проблемы, изучающей достижение кооперации, возникающей среди агентов без явного вмешательства человека. МакФарланд [22] провёл черту между двумя различными типами групп поведений, которые можно наблюдать в природе: социальное поведение и кооперативное поведение. Социальное поведение наблюдается у многих видов насекомых (например, колонии муравьёв или пчёл) как результат генетического определения индивидуального поведения. В таком обществе индивидуальные агенты имеют не очень большие способности, но, по-видимому, интеллектуальное поведение определяется их взаимодействием. Такая “кооперация” в поведении необходима для выживания индивидуумов в колониях.

С другой стороны МакФарланд [22] определил кооперативное поведение как социальное поведение, наблюдаемое у высших животных (позвоночных животных). Кооперация – результат взаимодействия между эгоистичными агентами. В отличие от социального поведения, кооперативное – не является врождённым поведением, но участники желают и намерены кооперироваться, с целью максимизировать индивидуальную выгоду.

В работах [23 - 26] и некоторых других предлагаются механизмы т.н.. возникающей кооперации (“emerging cooperation”) у эгоистичных рациональных агентов на базе экономических моделей или теории игр.

1.4 Обучение

Отыскание корректных значений для контроля параметров, которые необходимы для желаемого кооперативного поведения, может быть сложным, потребляющим много времени процессом для разработчика. Поэтому желательно, чтобы агенты в мульти-агентской системе имели

возможность *обучения* для контроля значений параметров и оптимизации решения задачи, а также для приспособления к изменениям в окружающей среде [27]. Закрепляющее обучение [28, 29] часто используется при кооперации роботов. Кроме того, методы, вдохновлённые биологической эволюцией, также используются при кооперации роботов.

1.5 Выводы

В данной главе мы рассмотрели и проанализировали некоторые основные способы кооперации и взаимодействия между агентами. Несложно видеть, что приведённые механизмы интересны и весьма разнообразны. Для иллюстрации изложенных методов приведём несколько примеров в главе 2. После ознакомления с примерами нам будет необходимо сделать некоторые выводы и определиться с методологией решения поставленной перед нами задачи в главе 3.

2 МЕТОДЫ И АЛГОРИТМЫ КООПЕРАЦИИ

В данной главе мы предоставим и проанализируем некоторые алгоритмы, демонстрирующие кооперативное поведение между агентами.

2.1 Метод Стилза исследования пространства

Люк Стилз [12] разработал архитектуру программного агента, осуществляющего исследование удалённой планеты. В постановке Стилза задача формулируется следующим образом.

Цель исследования – сбор образцов некоторой драгоценной породы на удалённой планете. Расположение образцов породы не известно заранее, но они обычно сгруппированы в некоторых местах. Доступно несколько автономных средств, которые могут передвигаться по планете, собирая образцы, и позже возвращаются на базовый корабль, что бы состыковаться с ним и вернуться на Землю. Нет детальной карты планеты, также известно, что ландшафт местности препятствует наличию прямой связи между роботами.

Решение проблемы использует два механизма, предложенных Стилзом. Первый – это градиентное поле. Агенты могут знать, в каком направлении расположен базовый корабль. Базовый корабль генерирует радиосигнал. Очевидно, что этот сигнал будет ослабевать с увеличением расстояния от источника. Для определения направления, в котором расположен базовый корабль, агенту необходимо передвигаться “вверх по градиенту” силы сигнала. Сигнал не должен нести какую либо информацию, он просто должен существовать.

Второй механизм позволяет агентам связываться друг с другом. Характеристики местности препятствуют прямой связи (такой, как

отправление сообщений), Стилзом был предложен метод непрямой связи. Идея заключается в том, что агенты будут переносить “радиоактивные частицы”, которые могут быть брошены, а затем обнаружены и подняты проходящими роботами. Таким образом, если агент бросит несколько таких частиц на пересечённой местности, позже другой агент, оказавшийся в этом месте, сможет обнаружить их. Такой простой механизм делает возможным взаимодействие без наличия прямой связи.

Индивидуальное поведение агента динамически составляется из элементарных правил, которые мы сейчас укажем. Сначала мы рассмотрим, как может быть запрограммирован агент для индивидуального сбора образцов. Затем мы рассмотрим, как агенты могут быть запрограммированы для генерации кооперативных решений.

Таблица 2.1 – Правила поведения агентов, работающих индивидуально

№	Правило	Комментарий
П1	Если обнаружено препятствие, то изменить направление движения (случайным образом).	Первым правилом индивидуального агента является правило уклонения от препятствий.
П2	Если переносится образец и база, то бросить образец.	Второе правило гласит, что любой образец, переносимый агентом, должен быть доставлен и оставлен на базовом корабле.
П3	Если переносится образец и не база, то двигаться вверх по градиенту.	Правило 3 объясняет как агенты, переносящие образцы, будут возвращаться на базовый корабль (руководствуясь правилом градиентного поля).

П4	Если обнаружен образец, то поднять его.	Данное правило показывает, как агенты будут подбирать образцы, когда найдут их
П5	Если ни одно из правил (1-4) не находится в состоянии выполнения, то двигаться случайно.	Финальное правило заключается в том, что агент не может сделать ничего лучшего, чем двигаться случайно.

Очевидно, что правило 5 принуждает агента к постоянному движению. Указанные правила можно представить в виде иерархии:

$$1<2<3<4<5.$$

Данная иерархия элементарных поведений гарантирует, что агент будет всегда поворачивать при обнаружении препятствия, если агент находится на материнском корабле с частицей на борту, то он всегда будет оставлять её, и, наконец, когда агенту больше нечего делать, он будет двигаться случайно. Несложно видеть, как эти простые поведения решают проблему: агенты будут искать образцы путём случайного движения и, найдя, будут доставлять их на материнский корабль.

Если образцы распределены на местности случайно, то, снарядив большое количество роботов этими очень простыми правилами, мы получим хороший результат. Но в рассматриваемой задаче образцы расположены в скоплениях. В этом случае было бы желательно взаимодействие между агентами. Например, при обнаружении одним агентом скопления породы, он должен помочь другим членам команды узнать место расположения источника. С учётом отсутствия прямой связи Стилз предлагает метод, частично основанный на поведении муравьёв в колонии. Идея состоит в создании агентом пути из радиоактивных частиц в случае нахождения породы. Путь будет создан, когда агент будет возвращаться с образцом породы на базовый корабль. Из этого вытекает, что если какой-либо агент в какой-то точке пересечёт путь, то ему просто будет необходимо двигаться вниз по градиенту

для того, что бы обнаружить источник. Данное изменение увеличивает эффективность рассматриваемой инженерной схемы.

Следует отметить ещё несколько деталей:

1. Агент должен поднимать частицы, попадающиеся на его пути, во время следования к источнику ресурса.
2. Путь прокладывается только при возвращении на базовый корабль. Если агент пришёл к источнику породы не по проложенному пути, то, доставляя образец к материнскому кораблю, он создаст ещё один путь, тем самым усилив его.
3. Если источник исчерпан, то после того, как несколько агентов проследуют по оставленному следу, путь будет ликвидирован.

Модифицированные правила выглядят следующим образом:

Таблица 2.2 – Модифицированные правила поведения агентов

№	Правило	Комментарий
П1	Если обнаружено препятствие, то изменить направление (случайным образом).	Преодоление препятствий (правило 1) остаётся неизменным.
П6	Если переносится образец и база, то бросить образец.	Правило 6 тоже не изменилось.
П7	Если переносится образец и не база, то бросить две частицы и двигаться вверх по градиенту.	Правило 7 предписывает агенту бросать частицы при возвращении на базу с образцом для создания или усиления пути.
П4	Если обнаружен образец, то поднять его.	Правило подбора образца (правило 4) остаётся неизменным.
П8	Если обнаружена частица, то поднять одну частицу и двигаться вниз по градиенту.	Данное правило описывает, как агент будет работать с частицами.

П5	Если ни одно из правил (1, 6, 7, 4, 8) не находится в состоянии выполнения, то двигаться случайно.	Правило случайного движения (правило 5) так же остаётся неизменным.

Модернизированные правила организовываются в следующую иерархию:

1<6<7<4<8<5

Алгоритм:

A) Получить сенсорный ввод из окружающей среды.

B) Если на пути обнаружено препятствие, то изменить направление движения.

C) Если переносится образец то

Если база, то бросить образец, иначе

Если не база, то бросить две частицы и передвинуться вверх по градиенту.

Вернуться к шагу A.

D) Если обнаружен образец то поднять его

Вернуться к шагу A.

E) Если обнаружена частица то поднять её и передвинуться вниз по градиенту

Вернуться к шагу A.

F) Двигаться в случайном направлении

G) Вернуться к шагу A.

КОНЕЦ.

Предложенная Стилзом архитектура – проста и эффективна. Она демонстрирует взаимодействие через окружающую среду (эффект общей памяти) используя непрямую связь. К тому же данная стратегия устойчива к исчезновению или выходу из строя членов команды. Стратегия Стилза не

использует модель окружающего мира, т.е. алгоритм будет работать в динамической окружающей среде.

В описанном алгоритме довольно примитивные агенты способны демонстрировать эффективное кооперативное поведение.

В принципе, агентская архитектура, предложенная Стилзом, может быть использована для решения нашей проблемы. Но хотелось бы рассмотреть ещё несколько интересных примеров децентрализованных архитектур, использующих непрямую связь.

2.2 Распределённое покрытие региона роботами-муравьями, использующими испаряющиеся следы

Данная абстрактная модель является децентрализованной мульти-агентской адаптируемой системой, использующей общую память. Окружающий мир представляется в виде графа, а задача сводится к его покрытию. Будут описаны три метода покрытия графа распределённым способом. Подробное описание этих методов и их полное математическое обоснование находятся в [30].

Описываемый алгоритм разработан для осуществления очистки пола в здании. Карта здания – неизвестна. Связь осуществляется с помощью испаряющихся следов, которые создают агенты. Если быть более точными, то след создаётся с помощью некоторого химического вещества, которое со временем испаряется. Следовательно, по интенсивности следа в некоторой точке можно определить время последнего её посещения. Предполагается, что пол выложен из плиток. Модель окружающего мира представлена в виде графа, вершинами которого являются плитки (возможно целые комнаты здания), а рёбрами – грани, соединяющие вершины (плитки или комнаты). Отметим, что скорость выполнения алгоритма зависит от количества вершин, т.е. чем больше вершин, тем дольше время покрытия графа и наоборот. Так

же следует отметить, что попытка применения данного алгоритма для ликвидации последствий ядерных и химических загрязнений, описана в [31].

Аналогом данной стратегии в природе является поведение муравьёв, которые оставляют за собой след из вещества, называемого феромоном. След так же имеет свойство испаряться и, используя это, муравьи прекрасно кооперируются.

Ещё может оказаться полезной информация о том, что существует несколько методов создания следа (маркировки пола). Помимо использования запаха [32] не менее эффективно можно использовать нагревание пола [33 - 35] или разметку пола различными цветами, яркость которых символизирует время посещения вершины.

2.2.1 Алгоритм ДВИЖЕНИЕ-МУРАВЬЯ-1 (покрытие без наличия индивидуальной памяти)

Мы рассмотрим граф *G=(V,E)* как модель для чистки пола. Каждая вершина в данном графе обозначает плитку пола, а каждое ребро – граница между двумя плитками. Так же мы назначим для каждого ребра *(u,v)* две “метки запаха”: *s(u,v),* которые являются временем последнего пересечения ребра в направлении от u к v и подобно *s(v,u)* для обратного направления. Все s-метки изначально сброшены в 0. Мы предположим, что при пересечении ребра в направлении от u к v маркировка свежести следа уничтожит старые и установит новые значения *s(u,v),* при этом значение *s(v,u)* остаётся неизменным.

В рассматриваемом алгоритме агент посещает вершину $u \in V(G)$, проверяя метки на всех рёбрах, выходящих из u, в направлениях исходящих из *u*. Потом он идёт к ребру, которое имеет наименьшее значение следа, это ребро не посещалось дольше всего. В направлении пересечения ребра агент оставляет постоянное количество химического вещества, создавая след. Количество вещества на ребре *(u,v),* обозначаемое *s(u,v),* медленно

уменьшается со временем, поэтому по “запаху” двух рёбер можно сказать, какое из них было посещено раньше. Для простоты мы будем обозначать следы временем их создания - *t*. Обозначим множество соседних вершин *u* как *N(u)*. Тогда формально алгоритм будет выглядеть следующим образом:

Алгоритм ДВИЖЕНИЕ-МУРАВЬЯ-1

A*) t:=t+1;*

B) Найти ребро (u,v) исходящее из u, такое что

$$s(u,v) = \min_{\omega \in N(u)} \{s(u,\omega)\}$$

(если найдено более чем одно такое ребро, то принять случайное решение)

(пока двигаемся из *u* в *v*, бросить феромон вдоль ребра *(u,v)*).

C) Идти к v и в процессе установить *s(u,v):=t;*

D) Вернуться к шагу A.

КОНЕЦ.

Заметим, что в одно и то же время одну плитку может занимать более чем один агент, но агенты покидают вершину в немного различные моменты времени. Порядок выходов из вершины может быть определён уникальным кодом, присвоенным каждому агенту, или при помощи других методов.

2.2.2 Алгоритм ДВИЖЕНИЕ-МУРАВЬЯ-2 (расширенный многоуровневый поиск)

Практическое тестирование алгоритма ДВИЖЕНИЕ-МУРАВЬЯ-1 показало, что существуют случаи, в которых агент попадает в “западню”, которая приводит к бесполезному многократному посещению некоторых участков графа. Этот алгоритм учитывает указанные недостатки и в основном работает более эффективно, но требует более сложной аппаратной поддержки. Этот алгоритм по большому счёту является обобщением

алгоритма глубокого первого поиска (ГПП) [36 – 38]. Основная идея ГПП – поиск соседних вершин, которые ещё не были посещены, если таковых не осталось, то агент возвращается вдоль ребра, по которому он попал в вершину, и продолжает поиск там и т.д. Если таких рёбер не существует, то вершина, в которой находится агент – первая, с которой он начал свой поиск, и, следовательно, задача выполнена.

Для описания данного алгоритма нам понадобятся два следующих определения. Обозначим через $t_{in}(u)$ время первого входа агента в вершину *u.*

$$t_{in}(u) = \min_{v \in N(u), s(v,u)>0} \{s(u,v)\} \tag{2.1}$$

и пусть $t_{out}(u)$ - время первого выхода из *u.*

$$t_{out}(u) = \min_{v \in N(u), s(u,v)>0} \{s(u,v)\} \tag{2.2}$$

Напомним, что в алгоритме глубокого первого поиска ребро никогда не посещается дважды в одном и том же направлении [38]. В начале работы алгоритма значения $t_{in}(u)$ и $t_{out}(u)$ считаются равными 0.

Могут быть сделаны следующие выводы:

1) Если $t_{in}(u)$ и $t_{out}(u)$ оба равны нулю, тогда *u* – “новая” вершина, т.е. она никогда не посещалась раньше.
2) Если $t_{in}(u) > t_{out}(u) > 0$ (т.е. вершина была покинута раньше, чем в неё вошли), тогда u – начальная вершина.

Алгоритм глубокого первого поиска

A) *t:=t+1;*

B) если $\exists v \in N(u)$ s(u,v)<1

Тогда устанавливаем *s(u,v):=t;*

Если $t_{in}(u)$<1 идти в *v;*

/*Я вернулся к источнику?*/

C) если $t_{in}(u) > t_{out}(u) > 0$, тогда остановиться (граф покрыт).

/*Вернуться. Все вершины старые*/

D) Найти ребро *(u,v)* такое, что $s(v,u) = t_{in}(u)$;

E) Установить $s(v,u) := t_{in}(u)$;

F) Идти в *v*.

G) Вернуться к шагу A.

КОНЕЦ.

Необходимо отметить, что при изменении графа или исчезновении маркировки на рёбрах граф никогда не будет покрыт. Алгоритм глубокого первого поиска не подходит для окружающей среды, в которой наблюдаются помехи или граф часто изменяется. Вторая проблема заключается в отсутствии кооперации в алгоритме глубокого первого поиска. Как было упомянуто выше, агент в финале выполнения своего задания возвращается в начальную точку и остаётся там “навсегда” вместо того, что бы помогать товарищам по команде.

Для преодоления вышеупомянутых проблем предложен многоуровневый подход в ГПП. Он заключается в том, что при попадании агента в ситуацию, в которой поиск больше не может продолжаться (нет ребра, ведущего назад), начинается новый уровень поиска. Это достигается увеличением значения переменной уровня-поиска (*r*), которая индивидуальна для каждого робота. Эта переменная запоминает время, когда “началась новая эпоха” и любое ребро (вершина), которые были посещены до этого, считаются теперь непосещёнными для данного агента. Если $t_{in}(u)$ <уровня-поиска(*r*) для робота *k,* тогда вершина u считается “новой” для этого агента.

Теперь алгоритм будет иметь следующий вид:

/*многоуровневый ГПП*/

/*изначально все уровни-поиска устанавливаются в 1*/

/*все s(.,.)=0*/

Алгоритм ДВИЖЕНИЕ-МУРАВЬЯ-2

A) *t:=t+1;*

B) если $\exists v \in N(u) : s(u,v) <$ уровня-поиска(r),

то установить *s(u,v):=t;*

Если $t_{in}(u) <$ уровня-поиска(r) идти к v;

/*Истощил ли я текущий уровень поиска ?*/

C) Если $t_{in}(u) > t_{out}(u) \geq$ уровня-поиска(r) тогда

Установить уровень-поиска(*r*):=*t;*

/*Возвращение. Все соседи старые*/

D) Найти ребро *(u,v)* такое, что *s(u,v)=* $t_{in}(u)$*;*

E) установить *s(u,v):=t;*

F) Идти к *v.*

G) Вернуться к шагу A.

КОНЕЦ

Обратим внимание, что шаг B выполняется если u имеет непосещённых соседей на текущем уровне поиска. Шаг D – шаг возвращения. На шаге C уровень поиска увеличивается, если текущий уровень поиска не может быть продолжен. Это может произойти в одном из трёх случаев:

1) Робот возвращается обратно к точке, откуда был начат текущий уровень поиска. Он сейчас начнёт новый поиск, установив новое значение уровня-поиска.
2) Изменения, произошедшие в графе, обычно делают возвращение невозможным.
3) При некоторых сенсорных ошибках агент верит, что выполняется пункт 1.

Отметим, что во втором алгоритме каждый робот нуждается в запоминании своего уровня поиска и делает больше вычислений в каждой посещённой точке. В награду мы получаем более эффективное выполнение задачи.

Описанные алгоритмы показывают великолепные результаты решения поставленной перед ними задачи. Но они содержат в себе очень серьёзный недостаток. Дело в том, что алгоритмы опираются на идеальную окружающую среду. А именно:

1. Окружающий мир должен быть разделён на комнаты, плитки или покрыт какой-либо другой “сетью”, которая интерпретируется как граф.
2. Ощущение силы следа должно быть достаточно точным иначе алгоритм будет неправильно функционировать.

Выполнение данных условий в реальном мире весьма проблематично, поэтому данные алгоритмы способны функционировать лишь в некоторых случаях.

2.3 Сравнительный анализ рассмотренных алгоритмов

Мы описали три типа мульти-агентских децентрализованных кооперативных архитектур, которые тем или иным образом связаны с рассматриваемым нами сценарием. Сейчас нам необходимо сделать выводы, которые помогут решить поставленную перед нами задачу. Для большей наглядности представим интересующие нас свойства алгоритмов в виде таблицы:

Таблица 2.3 – Сравнительный анализ алгоритмов

	Децентрализация	Используемая связь	Кооперация	Модель окружающей среды
Стилз	+	Непрямая	+	Практична
Д-М-1	+	Непрямая	+	Не практична
ГПП	+	Непрямая	-	Не практична
Д-М-2	+	Непрямая	+	Не практична

Теперь вспомним, какими свойствами должен обладать моделируемый нами алгоритм.

1. Агенты осуществляют совместный поиск, следовательно, алгоритм должен использовать механизмы кооперации.
2. Использование прямой связи в предложенных нами условиях невозможно.
3. Из-за отсутствия прямой связи мы должны использовать децентрализованный подход.
4. Модель окружающей среды должна быть предельно проста, но она должна учитывать параметры, используемые моделью кооперативного поведения.

Сопоставив выдвинутые нами требования и свойства описанных алгоритмов (Таб. 3) мы видим, что только модель Стилза удовлетворяет всем условиям.

Модель Стилза действительно может быть применена для решения задачи очистки региона от источников радиоактивного загрязнения. Нужно всего лишь немного адаптировать её к условиям решаемой задачи. Этим мы и займёмся в следующей главе.

2.4 Выводы

С помощью предложенных в данной главе примеров мы проиллюстрировали различные подходы достижения кооперации между автономными агентами. Более того, модель Стилза, описанная выше, удовлетворяет условиям решаемой нами задачи и будет использована в дальнейшем в качестве основы создаваемой модели.

3 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ И ВЫЧИСЛИТЕЛЬНЫЙ ЭКСПЕРИМЕНТ

В данной главе на основе имеющихся данных мы построим модель окружающей среды, алгоритм поведения агентов и выполним их программную реализацию.

3. 1 Моделирование окружающей среды и кооперативного поведения

Ещё раз напомним, что в основе моделируемого нами алгоритма лежит модель Стилза, предназначенная для сбора полезных ископаемых группой автономных роботов, находящихся на удалённой планете. Модель Стилза достаточно эффективна и нам остаётся только внести некоторые коррективы, связанные с постановкой нашей задачи.

Вспомним, что модель Стилза использует два механизма. Это градиентное поле и механизм создания агентом следа при возвращении на базовый корабль с образцом породы на борту. Мы тоже используем эти механизмы.

В модели Стилза источником градиента является базовый корабль. Для решения нашей задачи целесообразно заменить базовый корабль на ёмкость (резервуар), в которую будут доставляться отходы, найденные агентами. Соответственно резервуар становится источником градиента.

Теперь перейдём к моделированию окружающей среды. Нам известно, что на местности агент может встретить непреодолимые препятствия, источники радиации (искомый феномен), ёмкость для хранения отходов и следы, созданные агентами.

Простейшим вариантом модели является представление местности в виде “сетки” (Рис. 3.1).

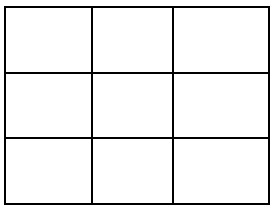

Рисунок 3.1 – Модель окружающей среды, представляемой в виде “сетки”.

Каждая ячейка данной “сетки” характеризуется двумя параметрами:

1. Сила градиента.
2. Характер ландшафта.

Под характером ландшафта мы понимаем одно из следующих состояний:

1. Непреодолимое препятствие.
2. Искомый феномен.
3. След, оставленный агентом.
4. Пустырь (участок местности, на котором ничего не расположено).
5. Резервуар для хранения отходов.

Необходимо отметить, что ячейки не могут одновременно содержать в себе несколько видов ландшафта (т.е. на одной ячейке одновременно не могут располагаться препятствие и искомый феномен и т.п.). Но несколько агентов могут находиться в одной ячейке в один и тот же момент времени.

Обратим внимание на то, что, руководствуясь правилом случайного движения, агент непрерывно перемещается. Предположим, что в единицу времени агент совершает перемещение на одну ячейку. Естественно, что агент может перейти только на одну из соседних ячеек.

Под соседней ячейкой подразумеваем ячейку, граничащую с той, в которой расположен агент. Как показано на рис. 3.2, каждая ячейка имеет восемь соседних ячеек.

1	2	3
4	A	5
6	7	8

Рисунок 3.2 – Модель окружающей среды. А – агент, 1-8 – восемь соседних ячеек.

В качестве алгоритма, управляющего поведением агента, возьмём алгоритм, предложенный Стилзом, но внесём в него некоторые коррективы. Вспомним, что алгоритм Стилза состоит из набора правил (Таб. 2.2), следуя которым группа агентов демонстрирует кооперативное поведение. Предположим, что агент способен определять, что находится в соседних ячейках. Тогда можно ввести дополнительные правила, предназначенные для выбора направления движения. Внесём их в таблицу поведений.

Таблица 3.1 – Дополнительные правила поведения агента

№	Правило	Комментарий
П9	Если в одной из соседних ячеек обнаружен образец, то перейти в эту ячейку.	Агент переходит на соседнюю ячейку с целью подбора образца.
П10	Если в одной из соседних ячеек обнаружена частица, то перейти в эту ячейку.	Агент переходит на соседнюю ячейку для того, что бы стать на след.

Дополнительно вводимые правила войдут в иерархию правил следующим образом:

$$1<6<7<4<8<9<10<5.$$

Блок-схема работы модифицированного алгоритма Стилза приведена на рисунке 3.3.

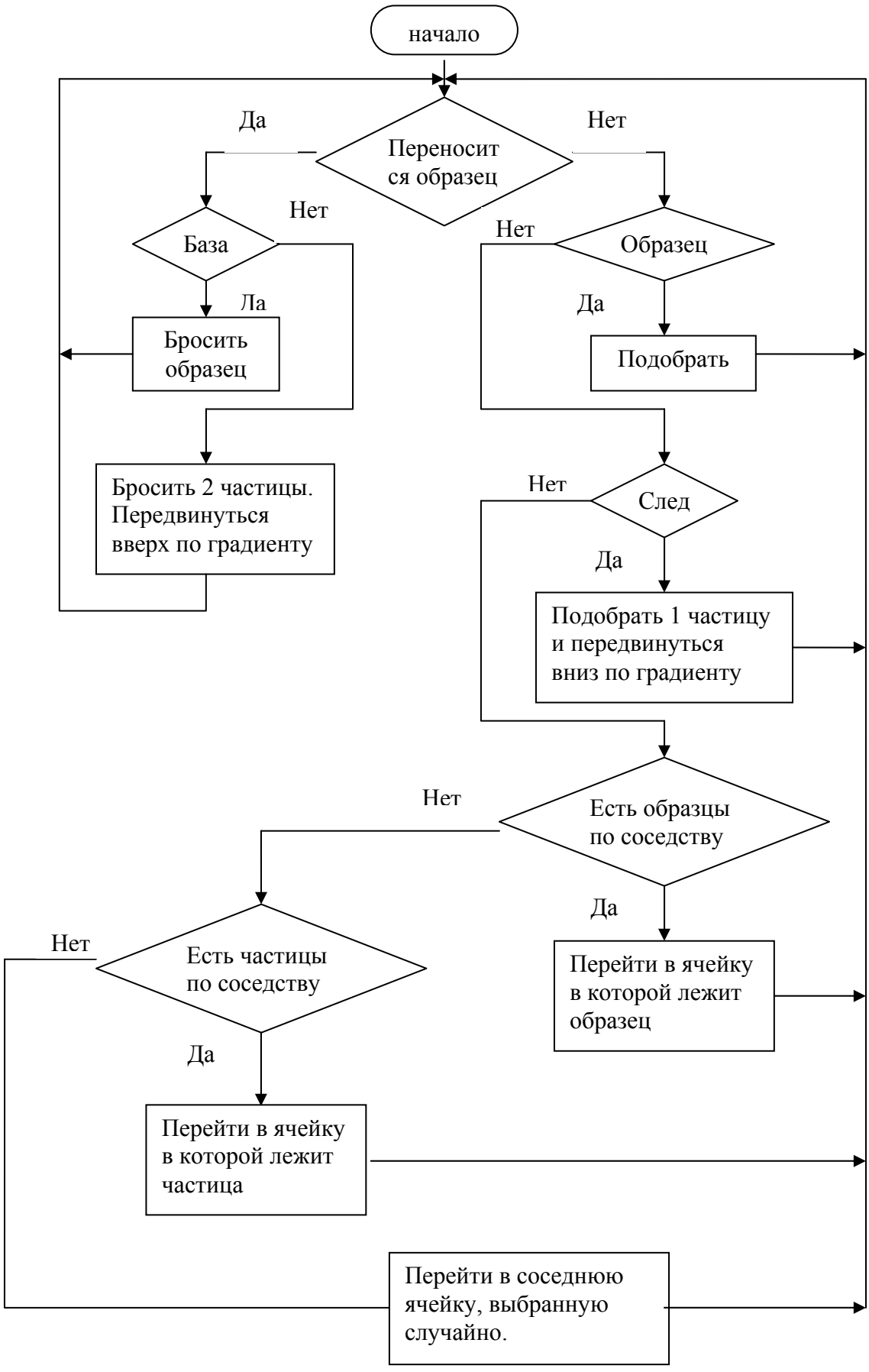

Рисунок 3.3 – Блок-схема работы модифицированного алгоритма Стилза.

Используя предложенную модель окружающей среды и модернизированный алгоритм Стилза кооперативного поведения агентов, мы можем построить тестовую программу, эмитирующую сбор радиоактивных отходов группой автономных агентов. Текст программы находится в Приложении А.

Графическое представление карты местности будет выглядеть следующим образом:

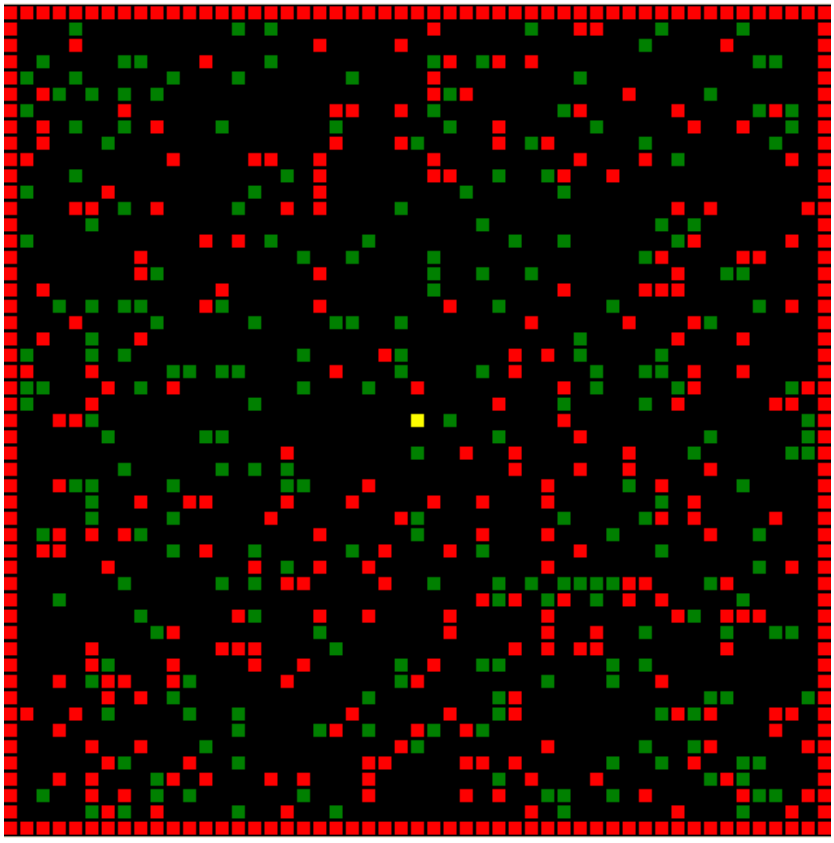

Рисунок 3.4 – Карта местности, на которой будет осуществляться поиск и сбор радиоактивных отходов.

На рисунке 3.4 виды местности представлены различными цветами. Сведём их интерпретацию в таблицу:

Таблица 3.2 – Интерпретация карты местности

Цвет	Значение
Красный	Непреодолимое препятствие
Зелёный	Источник радиации, который необходимо подобрать.
Жёлтый	Ёмкость для хранения радиоактивных отходов.
Чёрный	Участки, на которых ничего не расположено.

Агентов на карте мы будем изображать белым цветом, а след, оставленный ими – синим.

3.2 Анализ работы алгоритма

Для анализа результатов работы алгоритма создадим две карты местности. Размер карт будет равен 51×51. В центре карт (ячейка с координатами [26, 26]) расположим резервуар, в который агенты будут доставлять собранные образцы. Препятствия и образцы разместим на картах случайным образом. Различие будет состоять лишь в том, что на первой карте образцы будут равномерно распределены по всей площади, а на второй карте они будут находиться в скоплениях. Количество образцов на обеих картах установим равным 20%, количество препятствий будет таким же.

Тестирование и анализ будем проводить следующим образом. На каждой из карт осуществим сбор образцов 1, 2, 5, 10 и 15 агентами. Сбор будем осуществлять при помощи двух алгоритмов (модернизированный алгоритм Стилза и алгоритм случайного поиска без использования следа). Выполним сбор образцов каждым алгоритмом три раза, и занесём средние значения времени сходимости алгоритмов в таблицу. Под временем сходимости понимаем время подбора последнего образца на карте.

Таблица 3.3 – Результаты сбора образцов

Кол-во агентов	Кооперативный сбор (минуты)	Автономный сбор (минуты)	Кооперативный сбор (минуты)	Автономный сбор (минуты)
1	90	350	100	180
2	50	190	55	100
5	30	110	30	50
10	27	100	27	40
15	25	95	25	35
	Образцы в скоплениях		Образцы рассеяны	

Для облегчения понимания и анализа полученных данных представим их в виде диаграмм.

Диаграмма 3.1

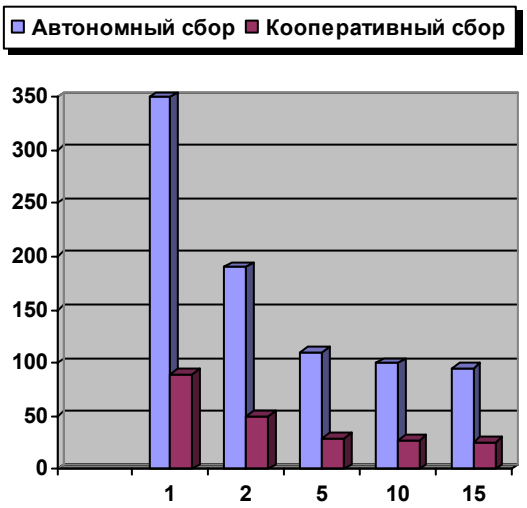

Диаграмма 3.2

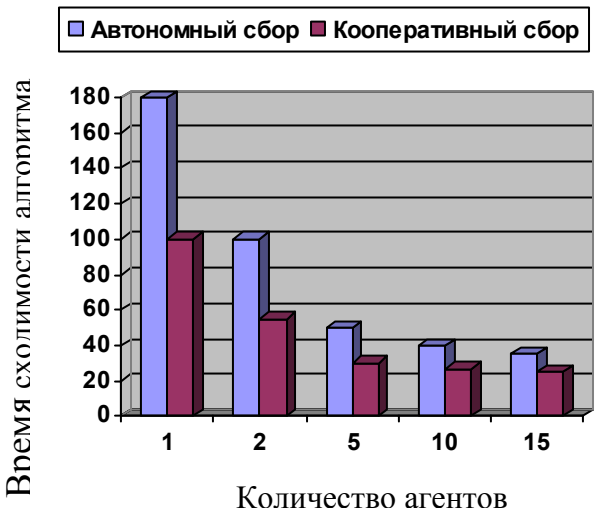

Диаграмма 3.1 иллюстрирует время сходимости алгоритмов в случае расположения феномена в виде скоплений, в случае, который демонстрирует диаграмма 3.2, образцы распределены равномерно.

Из результатов, полученных после тестирования, мы можем видеть, что алгоритм кооперативного поиска гораздо эффективнее, чем алгоритм случайного поиска. А именно: в случае равномерного распределения искомого феномена кооперативный сбор на 40% эффективнее случайного, если же феномен находится в скоплениях, то эффективность кооперативного поиска превосходит эффективность случайного поиска на 73%.

3.3 Выводы

На завершающем этапе нашей работы мы построили модель окружающей среды, в которой мы должны решать задачу сбора радиоактивных отходов. Так же мы адаптировали алгоритм Стилза к данной модели, внеся в него некоторые модификации. На основе полученных моделей и алгоритмов нами была создана программа, симулирующая сбор радиоактивных отходов на местности.

На основе данных, полученных при тестовых запусках программы в различных условиях, был проведён анализ эффективности алгоритма кооперации и сделаны интересующие нас выводы.

4 ОХРАНА ТРУДА

На предыдущих курсах нами были изучены такие предметы, как “Безопасность жизнедеятельности”, “Гражданская оборона”, “Охрана труда”. При выполнении дипломной работы мне удалось воспользоваться знаниями, полученными на этих дисциплинах.

В связи со спецификой получаемой мною специальности, основная часть работы выполнялась на персональном компьютере. Необходимо помнить, что персональный компьютер является потенциальным источником возникновения аварийных ситуаций, а несоблюдение основных предписаний при работе на компьютере влечёт за собой серьёзные заболевания.

Перечислим основные негативные факторы, которые возникают при работе компьютера:

1) Компьютер – это электроприбор, который питается электрическим током. Следовательно, должны соблюдаться правила обращения с электрическими приборами. Влажность внутри помещения должна находиться в пределах нормы. Не должна нарушаться целостность и изоляция электропроводки и шнуров. В случае неисправности элементов электрической сети или компьютера, работу нужно остановить, а прибор обесточить.
2) Т.к. компьютер является электрическим прибором, то, следовательно, он может быть источником возгорания. Поэтому возле компьютера не должно находиться легковоспламеняющихся предметов. В помещении должен находиться огнетушитель, предназначенный для гашения электроприборов (например, порошковый).
3) При работе монитора, возникает движение огромного количества α, β частиц в направлении пользователя. Эти частицы способны поражать организм человека на уровне ДНК и вызывать различные заболевания. Частицы представляют опасность при приближении человека к экрану

на расстояние менее 50 см. На расстоянии 50-70 см. от монитора α, β частицы опасности не представляют.

4) Так же работа компьютера может порождать статическое электричество. На компьютере накапливается заряд одного знака, а на пользователе – противоположного, в результате чего между пользователем и компьютером могут возникать электрические разряды. Для предотвращения данного явления электрический прибор должен быть заземлён.
5) Помимо α, β частиц компьютер излучает γ лучи (рентгеновское излучение). Излучение имеет небольшую силу, поэтому не представляет особой опасности. Оно направлено в противоположную от пользователя сторону. Поэтому желательно, что бы компьютер “упирался” в глухую стену. Если такой возможности нет, то работающие рядом пользователи должны находится на расстоянии не менее двух метров от соседних компьютеров.
6) Работающий компьютер ионизирует воздух. Плотность ионизации прямо пропорциональна времени работы компьютера. Ионизация воздуха приводит к появлению озона. Озон потребляется кровью в результате чего снижается активность крови к кислороду и возникает кислородное голодание. Проветривание помещения с интервалом 60-90 минут предотвращает явление повышенной ионизации.
7) Необходимо так же отметить, что работа на компьютере является причиной долгого пребывания в одной позе, эмоционального напряжения и утомления глаз. Что бы избежать этого необходимо через определённые интервалы времени прерывать работу на ПК и отдыхать или делать небольшую разминку.

Работая на персональном компьютере, я соблюдал перечисленные выше требования. Это было достигнуто следующим образом.

Были соблюдены правила использования электрических приборов. Заземление компьютера и монитора были осуществлены при помощи евро-разеток. Правила пожарной безопасности так же не были нарушены. Компьютер был установлен таким образом, что расстояние между мной и монитором было не меньше 70 см. В радиусе 2,5 метров компьютеров не находилось. Работа была организована в интервалы 60-90 минут. В перерывах выполнялось проветривание помещения.

Таким образом, при написании дипломной работы мной были выдержаны все основные предписания для работы на персональном компьютере.

ЗАКЛЮЧЕНИЕ

Итак, перед нами стояла задача моделирования поведения автономных роботов при совместном сборе радиоактивных отходов. Для решения стоящей перед нами задачи мы воспользовались мульти-агентским подходом.

Мы дали определение программного агента и перечислили свойства, которыми должен обладать интеллектуальный программный агент. Затем мы рассмотрели и проанализировали известные механизмы и методы достижения кооперации между агентами других авторов [30, 36, 37, 38]. Для иллюстрации перечисленных механизмов кооперации были приведены несколько алгоритмов использующих различные подходы. Среди рассмотренных алгоритмов в качестве основы создаваемой нами модели был выбран алгоритм Стилза [12]. Построив модель окружающей среды, мы адаптировали модель Стилза к условиям решаемой нами задачи. Используя имеющиеся данные, мы создали тестовую программу, моделирующую сбор радиоактивных отходов. На основе результатов работы тестовой программы был проведён сравнительный анализ работы алгоритма в различных условиях. Использованная нами модель проста в реализации. Выполненные в работе программная реализация и вычислительный експеримент демонстрируют ее высокую эффективность.

Хотелось бы отметить, что построенная нами модель и алгоритм являются простейшим решением поставленной задачи. Эту модель можно использовать в качестве основы при построении более сложных алгоритмов, которые будут демонстрировать лучшие результаты при выполнении работы.

Таким образом, можно отметить, что результаты выполненной дипломной работы могут быть использованы в качестве базиса для дальнейших исследований.

ПЕРЕЧЕНЬ ССЫЛОК

1 Michael Wooldridge: An introduction to multiagent systems, Departament of Computer Science, University of Liverpool, UK 2002.

2 Merriam-Webster. Webster's 7th Collegiate Dictionary. 1963.

3 D. Barnes and J. Gray. Behaviour synthesis for co-operating mobil control. In Int. Conf. on Control, 1135-1140, 1991.

4 M. Mataric. Interaction and Intelligent behavior. MIT AI Lab Technical Report, AI-TR-1945, August 1994.

5 S. Premvuti and S. Yuta. Consideration on the cooperation of multiple autonomous mobile robots. In IEEE/RSJ IROS, 59-63, 1990.

6 Cao, Y., Fukunaga, A. S., Kahng, A. B. & Meng, F. (1995), Cooperative Mobile Robotics: Antecedents and Directions, in `IEEE/TSJ International Conference on Intelligent Robots and Systems', Yokohama, Japan. http://citeseer.ist.psu.edu/article/cao95cooperative.html

7 R. Beckers, O. E. Holland, and J. L. Deneuborg. From local actions to global tasks : Stigmergy and collective robotics. In Proc. A-Life IV. MIT Press, 1994.

8 R. C. Arkin. Cooperation without communication: Multiagent schema-based robot navigation. Journal of Robotic Systems, 9(3):351-364, 1992.

9 L. Steels. A case study in the behavior-oriented design of autonomous agents. In Proc. Simulation of Adaptive Behavior, 1994.

10 L. Parker. Heterogeneous Multi-Robot Cooperation. PhD thesis, MIT EECS Dept., February 1994.

11 S. Goss and J. Deneubourg. Harvesting by a group of robots. In Proc. European Conf. on Artificial Life, 1992.

12 L. Steels. Cooperation between distributed agents through self-organization. In European Workshop on Modelling Autonomous Agents in a Multi-Agent World, 175-195, 1990.

13 B. Tung and L. Kleinrock. Distributed control methods. In Proceedings of the 2nd Int. Symp. On High Performance Distributed Computing, 206-215, 1993.

14 Y-C. Tung. Distributed Control Using Finite State Automata. PhD thesis, UCLA Computer Science Dept., 1994.

15 S. Sen, M. Sekaran, and J. Hale. Learning to coordinate whithout sharing information. In Proc. AAAI, 426-431, 1994.

16 C. Hewitt. Toward an open systems architecture. In Information Processing 89. Proceedings of the IFIP 11th World Computer Congress, 389-92, 1993.

17 T. Fukuda and K. Sekiyama. Communication reduction with risk estimate for multiple robotic system. In IEEE ICRA, 2864-2869, 1994.

18 M. Rude. Cooperation of mobile robots by event transforms into local space-time. In IEEE/RSJ IROS, 1501-1507, 1994.

19 P. Caloud, W. Choi, J.-C. Latombe, C. Le Pape, and M. Yin. Indoor automation with many mobile robots. In IEEE/RSJ IROS, 67-72, 1990.

20 H. Asama, M. K. Habib, I. Endo, K. Ozaki, A. Matsumoto, and Y. Ishida. Functional distribution among multiple mobile robots in an autonomous and decentralized robot system. In IEEE ICRA, 1921-6, 1991.

21 D. Grossman. Traffic control of multiple robot vehicles. IEEE Journal of Robotics and Automation, 4:491-497, 1988.

22 D. McFarland. Towards robot cooperation. In Proc. Simulation of Adaptive Behavior, 1994.

23 M. R. Genesereth, M. L. Ginsberg, and S. Rosenschein. Cooperation without communication. In Proc. AAAI, 51-57, 1986.

24 J. S. Rosenschein and M. R. Genesereth. Deals among rational agents. In Proc. Intl. Joint Conf. Artificial Intelligence, 91-99, 1985.

25 A. H. Bond and L. Gasser. Readings in Distributed Artificial Intelligence. Morgan Kaufmann Publishers, 1988.

26 J. S. Rosenschein and G. Zlotkin. Rules of Encounter: Designing conventions for automated negotiation among computers. MIT Press, 1994.

27 Ermolayev, V., Keberle, N., Kononenko, O., Plaksin, S., Terziyan, V.: Towards a framework for agent-enabled semantic web service composition. Int. J. of Web Services Research, 1(3), 2004, p. 63-87.

28 A. G. Barto, R. S. Sutton, and C. J. C. H. Watkins. Learning and sequential decision making. In M. Gabriel and J. Moore, editors, Learning and Computational Neuroscience: Foundations of Adaptive Networks, 539-603. MIT Press, 1983.

29 L. P. Kaelbling. Learning in Embedded systems. MIT Press, 1993.

30 Israel A. Wagner, Michael Lindenbaum, and Alfred M. Bruckstein: Distributed Covering by Ant-Robots Using Evaporating Traces, IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION, VOL. 15, NO.5, OCTOBER 1999.

31 S. Hedberg, "Robots cleaning up hazardous waste", AI Expert. New York: Springer-Verlag, May 1995, pp. 20-24, 1994.

32 R. A. Russel, "Laying and sensing odor markings as a strategy for assisting mobile robots navigation tasks," IEEE Robot. Automat. Mag., vol. 2, pp. 3-9, Sep. 95.

33 J. Borenstein, H. R. Everett, and L. Feng, Navigating Mobile Robots-Systems and Techniques. Wellesley, MA: A. K. Peters, 1996.

34 R. A. Russel, "Mobile robot guidance using a short-lived heat trail", Robotica, vol. 11, pp. 427-431, 1993.

35 R. A. Russel, "Heat trails as short-lived navigational markers for mobile robots", in Proc. 1997 IEEE Int. Conf. Robot. Automat., 1997., pp. 3534-3539.

36 R. Tarjan, "Depth-first search and linear graph algorithms", SIAM J. Comput., vol. 1, no. 2, pp. 146-160, 1972.

37 J. Hopcrof and R. Tarjan, "Efficient algorithms for graph manipulation", Commun. ACM, pp. 372-378, June 1973.

38 S. Even, Graph Algorithms. Rockville, MD: Comput. Sci. Press, 1979.

ПРИЛОЖЕНИЕ А

Программная реализация агентов была выполнена на языке программирования Delphi 6 и имеет следующий вид:

```
var x,stop_time,w,num,i,j,k: integer;
   flag: boolean; {true если феномен на борту}
   draw_par,coord_i, coord_j,old_par,old_i,old_j: array[1..100] of integer;
   flag_val: array [1..100] of boolean;
label beg, rep;
begin
    num:=StrToInt(Edit1.Text);
    stop_time:=100 div num;
    for i:=1 to 100 do
    begin    {начальные значения (все агенты начинают с центра не имея частиц на борту)}
        coord_i[i]:=26;
        coord_j[i]:=26;
        flag_val[i]:=false;
    end;
    w:=1;
    Randomize;
beg:
    map[26,26]:=0;
    i:=coord_i[w];
    j:=coord_j[w];
    flag:=flag_val[w];
    sleep(stop_time);
    old_i[w]:=i;
    old_j[w]:=j;
    map[i,j]:=4;
    for x:=1 to num do
    begin
    draw_par[x]:=map[coord_i[x],coord_j[x]];
    map[coord_i[x],coord_j[x]]:=4;
    end;
    draw;
    for x:=1 to num do
    map[coord_i[x],coord_j[x]]:=draw_par[x];
    if flag=false then
    begin
        if map[i+1,j-1]=1 then  {поиск источника}
        begin
            map[old_i[w],old_j[w]]:=old_par[w];
            i:=i+1;
            j:=j-1;
            flag:=true;
            coord_i[w]:=i;
```

```
    coord_j[w]:=j;
    if map[i,j]<>4 then old_par[w]:=3 else
    for x:=num downto 1 do
    if (coord_i[w]=coord_i[x]) and (coord_j[w]=coord_j[x]) and
    (old_par[x]<>4) then old_par[w]:=old_par[x];
    flag_val[w]:=flag;
    if w<num then w:=w+1 else w:=1;
    goto beg;
end;
if map[i+1,j]=1 then
begin
     map[old_i[w],old_j[w]]:=old_par[w];
     i:=i+1;
     flag:=true;
     coord_i[w]:=i;
     coord_j[w]:=j;
     if map[i,j]<>4 then old_par[w]:=3 else
     for x:=num downto 1 do
     if (coord_i[w]=coord_i[x]) and (coord_j[w]=coord_j[x]) and
     (old_par[x]<>4) then old_par[w]:=old_par[x];
     flag_val[w]:=flag;
     if w<num then w:=w+1 else w:=1;
     goto beg;
end;
if map[i+1,j+1]=1 then
begin
     map[old_i[w],old_j[w]]:=old_par[w];
     i:=i+1;
     j:=j+1;
     if map[i,j]<>4 then old_par[w]:=3 else
     for x:=num downto 1 do
     if (coord_i[w]=coord_i[x]) and (coord_j[w]=coord_j[x]) and
     (old_par[x]<>4) then old_par[w]:=old_par[x];
     flag:=true;
     coord_i[w]:=i;
     coord_j[w]:=j;
     flag_val[w]:=flag;
     if w<num then w:=w+1 else w:=1;
     goto beg;
end;
if map[i,j-1]=1 then
begin
     map[old_i[w],old_j[w]]:=old_par[w];
     j:=j-1;
     flag:=true;
     coord_i[w]:=i;
     coord_j[w]:=j;
     if map[i,j]<>4 then old_par[w]:=3 else
     for x:=num downto 1 do
     if (coord_i[w]=coord_i[x]) and (coord_j[w]=coord_j[x]) and
     (old_par[x]<>4) then old_par[w]:=old_par[x];
     flag_val[w]:=flag;
```

```
     if w<num then w:=w+1 else w:=1;
     goto beg;
end;
if map[i,j+1]=1 then
begin
     map[old_i[w],old_j[w]]:=old_par[w];
     j:=j+1;
     flag:=true;
     coord_i[w]:=i;
     coord_j[w]:=j;
     if map[i,j]<>4 then old_par[w]:=3 else
     for x:=num downto 1 do
     if (coord_i[w]=coord_i[x]) and (coord_j[w]=coord_j[x]) and
     (old_par[x]<>4) then old_par[w]:=old_par[x];
     flag_val[w]:=flag;
     if w<num then w:=w+1 else w:=1;
     goto beg;
end;
if map[i-1,j-1]=1 then
begin
     map[old_i[w],old_j[w]]:=old_par[w];
     i:=i-1;
     j:=j-1;
     flag:=true;
     coord_i[w]:=i;
     coord_j[w]:=j;
     if map[i,j]<>4 then old_par[w]:=3 else
     for x:=num downto 1 do
     if (coord_i[w]=coord_i[x]) and (coord_j[w]=coord_j[x]) and
     (old_par[x]<>4) then old_par[w]:=old_par[x];
     flag_val[w]:=flag;
     if w<num then w:=w+1 else w:=1;
     goto beg;
end;
if map[i-1,j]=1 then
begin
    map[old_i[w],old_j[w]]:=old_par[w];
    i:=i-1;
    flag:=true;
    coord_i[w]:=i;
    coord_j[w]:=j;
    if map[i,j]<>4 then old_par[w]:=3 else
    for x:=num downto 1 do
    if (coord_i[w]=coord_i[x]) and (coord_j[w]=coord_j[x]) and
    (old_par[x]<>4) then old_par[w]:=old_par[x];
    flag_val[w]:=flag;
    if w<num then w:=w+1 else w:=1;
    goto beg;
end;
if map[i-1,j+1]=1 then
begin
    map[old_i[w],old_j[w]]:=old_par[w];
```

```
    i:=i-1;
    j:=j+1;
    flag:=true;
    coord_i[w]:=i;
    coord_j[w]:=j;
    if map[i,j]<>4 then old_par[w]:=3 else
    for x:=num downto 1 do
    if (coord_i[w]=coord_i[x]) and (coord_j[w]=coord_j[x]) and
    (old_par[x]<>4) then old_par[w]:=old_par[x];
    flag_val[w]:=flag;
    if w<num then w:=w+1 else w:=1;
    goto beg;
end; {феномена по соседству не найдено}
{---Наступили на след---}
map[old_i[w],old_j[w]]:=old_par[w];
if map[old_i[w],old_j[w]]>100 then {агент уже наступил на след}
begin
    if old_par[w]=101 then old_par[w]:=3 else old_par[w]:=old_par[w]-1;
    if (map[i-1,j-1]>100) and (gradient[i-1,j-1]<gradient[i,j]) then
    begin
        i:=i-1;
        j:=j-1;
        coord_i[w]:=i;
        coord_j[w]:=j;
        if map[i,j]<>4 then old_par[w]:=map[i,j] else
        for x:=num downto 1 do
        if (coord_i[w]=coord_i[x]) and (coord_j[w]=coord_j[x]) and
        (old_par[x]<>4) then old_par[w]:=old_par[x];
        flag_val[w]:=flag;
        if w<num then w:=w+1 else w:=1;
        goto beg;
    end;
    if (map[i+1,j+1]>100) and (gradient[i+1,j+1]<gradient[i,j]) then
    begin
        i:=i+1;
        j:=j+1;
        coord_i[w]:=i;
        coord_j[w]:=j;
        if map[i,j]<>4 then old_par[w]:=map[i,j] else
        for x:=num downto 1 do
        if (coord_i[w]=coord_i[x]) and (coord_j[w]=coord_j[x]) and
        (old_par[x]<>4) then old_par[w]:=old_par[x];
        flag_val[w]:=flag;
        if w<num then w:=w+1 else w:=1;
        goto beg;
    end;
    if (map[i-1,j]>100) and (gradient[i-1,j]<gradient[i,j]) then
    begin
        i:=i-1;
        coord_i[w]:=i;
        coord_j[w]:=j;
        if map[i,j]<>4 then old_par[w]:=map[i,j] else
```

```
    for x:=num downto 1 do
    if (coord_i[w]=coord_i[x]) and (coord_j[w]=coord_j[x]) and
    (old_par[x]<>4) then old_par[w]:=old_par[x];
    flag_val[w]:=flag;
    goto beg;
end;
if (map[i+1,j]>100) and (gradient[i+1,j]<gradient[i,j]) then
begin
    i:=i+1;
    coord_i[w]:=i;
    coord_j[w]:=j;
    if map[i,j]<>4 then old_par[w]:=map[i,j] else
    for x:=num downto 1 do
    if (coord_i[w]=coord_i[x]) and (coord_j[w]=coord_j[x]) and
    (old_par[x]<>4) then old_par[w]:=old_par[x];
    flag_val[w]:=flag;
    if w<num then w:=w+1 else w:=1;
    goto beg;
end;
if (map[i-1,j+1]>100) and (gradient[i-1,j+1]<gradient[i,j]) then
begin
    i:=i-1;
    j:=j+1;
    coord_i[w]:=i;
    coord_j[w]:=j;
    if map[i,j]<>4 then old_par[w]:=map[i,j] else
    for x:=num downto 1 do
    if (coord_i[w]=coord_i[x]) and (coord_j[w]=coord_j[x]) and
    (old_par[x]<>4) then old_par[w]:=old_par[x];
    flag_val[w]:=flag;
    if w<num then w:=w+1 else w:=1;
    goto beg;
end;
if (map[i,j-1]>100) and (gradient[i,j-1]<gradient[i,j]) then
begin
    j:=j-1;
    coord_i[w]:=i;
    coord_j[w]:=j;
    if map[i,j]<>4 then old_par[w]:=map[i,j] else
    for x:=num downto 1 do
    if (coord_i[w]=coord_i[x]) and (coord_j[w]=coord_j[x]) and
    (old_par[x]<>4) then old_par[w]:=old_par[x];
    flag_val[w]:=flag;
    if w<num then w:=w+1 else w:=1;
    goto beg;
end;
if (map[i,j+1]>100) and (gradient[i,j+1]<gradient[i,j]) then
begin
    j:=j+1;
    coord_i[w]:=i;
    coord_j[w]:=j;
    if map[i,j]<>4 then old_par[w]:=map[i,j] else
```

```
            for x:=num downto 1 do
            if (coord_i[w]=coord_i[x]) and (coord_j[w]=coord_j[x]) and
            (old_par[x]<>4) then old_par[w]:=old_par[x];
            flag_val[w]:=flag;
            if w<num then w:=w+1 else w:=1;
            goto beg;
        end;
        if (map[i+1,j-1]>100) and (gradient[i+1,j-1]<gradient[i,j]) then
        begin
            i:=i+1;
            j:=j-1;
            coord_i[w]:=i;
            coord_j[w]:=j;
            if map[i,j]<>4 then old_par[w]:=map[i,j] else
            for x:=num downto 1 do
            if (coord_i[w]=coord_i[x]) and (coord_j[w]=coord_j[x]) and
            (old_par[x]<>4) then old_par[w]:=old_par[x];
            flag_val[w]:=flag;
            if w<num then w:=w+1 else w:=1;
            goto beg;
        end;
        goto rep;
    end else
    begin {ищем след}
    if map[i+1,j-1]>100 then
    begin
         i:=i+1;
         j:=j-1;
         coord_i[w]:=i;
         coord_j[w]:=j;
         if map[i,j]<>4 then old_par[w]:=map[i,j] else
         for x:=num downto 1 do
         if (coord_i[w]=coord_i[x]) and (coord_j[w]=coord_j[x]) and
         (old_par[x]<>4) then old_par[w]:=old_par[x];
         flag_val[w]:=flag;
         if w<num then w:=w+1 else w:=1;
         goto beg;
    end;
    if map[i+1,j]>100 then
    begin
         i:=i+1;
         coord_i[w]:=i;
         coord_j[w]:=j;
         if map[i,j]<>4 then old_par[w]:=map[i,j] else
         for x:=num downto 1 do
         if (coord_i[w]=coord_i[x]) and (coord_j[w]=coord_j[x]) and
         (old_par[x]<>4) then old_par[w]:=old_par[x];
         flag_val[w]:=flag;
         if w<num then w:=w+1 else w:=1;
         goto beg;
    end;
    if map[i+1,j+1]>100 then
```

```
begin
     i:=i+1;
     j:=j+1;
     coord_i[w]:=i;
     coord_j[w]:=j;
     if map[i,j]<>4 then old_par[w]:=map[i,j] else
     for x:=num downto 1 do
     if (coord_i[w]=coord_i[x]) and (coord_j[w]=coord_j[x]) and
     (old_par[x]<>4) then old_par[w]:=old_par[x];
     flag_val[w]:=flag;
     if w<num then w:=w+1 else w:=1;
     goto beg;
end;
if map[i,j-1]>100 then
begin
     j:=j-1;
     coord_i[w]:=i;
     coord_j[w]:=j;
     if map[i,j]<>4 then old_par[w]:=map[i,j] else
     for x:=num downto 1 do
     if (coord_i[w]=coord_i[x]) and (coord_j[w]=coord_j[x]) and
     (old_par[x]<>4) then old_par[w]:=old_par[x];
     flag_val[w]:=flag;
     if w<num then w:=w+1 else w:=1;
     goto beg;
end;
if map[i,j+1]>100 then
begin
     j:=j-1;
     coord_i[w]:=i;
     coord_j[w]:=j;
     if map[i,j]<>4 then old_par[w]:=map[i,j] else
     for x:=num downto 1 do
     if (coord_i[w]=coord_i[x]) and (coord_j[w]=coord_j[x]) and
     (old_par[x]<>4) then old_par[w]:=old_par[x];
     flag_val[w]:=flag;
     if w<num then w:=w+1 else w:=1;
     goto beg;
end;
if map[i-1,j-1]>100 then
begin
     i:=i-1;
     j:=j-1;
     coord_i[w]:=i;
     coord_j[w]:=j;
     if map[i,j]<>4 then old_par[w]:=map[i,j] else
     for x:=num downto 1 do
     if (coord_i[w]=coord_i[x]) and (coord_j[w]=coord_j[x]) and
     (old_par[x]<>4) then old_par[w]:=old_par[x];
     flag_val[w]:=flag;
     if w<num then w:=w+1 else w:=1;
     goto beg;
```

```
        end;
        if map[i-1,j]>100 then
        begin
            i:=i-1;
            coord_i[w]:=i;
            coord_j[w]:=j;
            if map[i,j]<>4 then old_par[w]:=map[i,j] else
            for x:=num downto 1 do
            if (coord_i[w]=coord_i[x]) and (coord_j[w]=coord_j[x]) and
            (old_par[x]<>4) then old_par[w]:=old_par[x];
            flag_val[w]:=flag;
            if w<num then w:=w+1 else w:=1;
            goto beg;
        end;
        if map[i-1,j+1]>100 then
        begin
            i:=i-1;
            j:=j+1;
            coord_i[w]:=i;
            coord_j[w]:=j;
            if map[i,j]<>4 then old_par[w]:=map[i,j] else
            for x:=num downto 1 do
            if (coord_i[w]=coord_i[x]) and (coord_j[w]=coord_j[x]) and
            (old_par[x]<>4) then old_par[w]:=old_par[x];
            flag_val[w]:=flag;
            if w<num then w:=w+1 else w:=1;
            goto beg;
        end;
        end;
rep:
        k:=Random(8); {случайное движение}
        if (k=0) and (map[i-1,j-1]<>2) and (map[i-1,j-1]<>4) then
        begin
            map[old_i[w],old_j[w]]:=old_par[w];
            i:=i-1;
            j:=j-1;
            coord_i[w]:=i;
            coord_j[w]:=j;
            if map[i,j]<>4 then old_par[w]:=map[i,j] else
            for x:=num downto 1 do
            if (coord_i[w]=coord_i[x]) and (coord_j[w]=coord_j[x]) and
            (old_par[x]<>4) then old_par[w]:=old_par[x];
            flag_val[w]:=flag;
            if w<num then w:=w+1 else w:=1;
            goto beg
        end else
        if (k=1) and (map[i-1,j]<>2) and (map[i-1,j]<>4)then
        begin
            map[old_i[w],old_j[w]]:=old_par[w];
            i:=i-1;
            coord_i[w]:=i;
            coord_j[w]:=j;
```

```
    if map[i,j]<>4 then old_par[w]:=map[i,j] else
    for x:=num downto 1 do
    if (coord_i[w]=coord_i[x]) and (coord_j[w]=coord_j[x]) and
    (old_par[x]<>4) then old_par[w]:=old_par[x];
    flag_val[w]:=flag;
    if w<num then w:=w+1 else w:=1;
    goto beg
end else
if (k=2) and (map[i-1,j+1]<>2) and (map[i-1,j+1]<>4)then
begin
    map[old_i[w],old_j[w]]:=old_par[w];
    i:=i-1;
    j:=j+1;
    coord_i[w]:=i;
    coord_j[w]:=j;
    if map[i,j]<>4 then old_par[w]:=map[i,j] else
    for x:=num downto 1 do
    if (coord_i[w]=coord_i[x]) and (coord_j[w]=coord_j[x]) and
    (old_par[x]<>4) then old_par[w]:=old_par[x];
    flag_val[w]:=flag;
    if w<num then w:=w+1 else w:=1;
    goto beg
end else
if (k=3) and (map[i,j-1]<>2) and (map[i,j-1]<>4)then
begin
    map[old_i[w],old_j[w]]:=old_par[w];
    j:=j-1;
    coord_i[w]:=i;
    coord_j[w]:=j;
    if map[i,j]<>4 then old_par[w]:=map[i,j] else
    for x:=num downto 1 do
    if (coord_i[w]=coord_i[x]) and (coord_j[w]=coord_j[x]) and
    (old_par[x]<>4) then old_par[w]:=old_par[x];
    flag_val[w]:=flag;
    if w<num then w:=w+1 else w:=1;
    goto beg
end else
if (k=4) and (map[i,j+1]<>2) and (map[i,j+1]<>4)then
begin
    map[old_i[w],old_j[w]]:=old_par[w];
    j:=j+1;
    coord_i[w]:=i;
    coord_j[w]:=j;
    if map[i,j]<>4 then old_par[w]:=map[i,j] else
    for x:=num downto 1 do
    if (coord_i[w]=coord_i[x]) and (coord_j[w]=coord_j[x]) and
    (old_par[x]<>4) then old_par[w]:=old_par[x];
    flag_val[w]:=flag;
    if w<num then w:=w+1 else w:=1;
    goto beg
end else
if (k=5) and (map[i+1,j-1]<>2) and (map[i+1,j-1]<>4)then
```

```
    begin
        map[old_i[w],old_j[w]]:=old_par[w];
        i:=i+1;
        j:=j-1;
        coord_i[w]:=i;
        coord_j[w]:=j;
        if map[i,j]<>4 then old_par[w]:=map[i,j] else
        for x:=num downto 1 do
        if (coord_i[w]=coord_i[x]) and (coord_j[w]=coord_j[x]) and
        (old_par[x]<>4) then old_par[w]:=old_par[x];
        flag_val[w]:=flag;
        if w<num then w:=w+1 else w:=1;
        goto beg
    end else
    if (k=6) and (map[i+1,j]<>2) and (map[i+1,j]<>4)then
    begin
        map[old_i[w],old_j[w]]:=old_par[w];
        i:=i+1;
        coord_i[w]:=i;
        coord_j[w]:=j;
        if map[i,j]<>4 then old_par[w]:=map[i,j] else
        for x:=num downto 1 do
        if (coord_i[w]=coord_i[x]) and (coord_j[w]=coord_j[x]) and
        (old_par[x]<>4) then old_par[w]:=old_par[x];
        flag_val[w]:=flag;
        if w<num then w:=w+1 else w:=1;
        goto beg
    end else
    if ((k=7) or (k=8)) and (map[i+1,j+1]<>2) and (map[i+1,j+1]<>4)then
    begin
        map[old_i[w],old_j[w]]:=old_par[w];
        i:=i+1;
        j:=j+1;
        coord_i[w]:=i;
        coord_j[w]:=j;
        if map[i,j]<>4 then old_par[w]:=map[i,j] else
        for x:=num downto 1 do
        if (coord_i[w]=coord_i[x]) and (coord_j[w]=coord_j[x]) and
        (old_par[x]<>4) then old_par[w]:=old_par[x];
        flag_val[w]:=flag;
        if w<num then w:=w+1 else w:=1;
        goto beg
    end else goto rep;
end else
begin {феномен на борту}
    if (old_par[w]>100) and (old_par[w]<105) then old_par[w]:=old_par[w]+1 else
    if old_par[w]<>1 then old_par[w]:=102;
    map[i,j]:=old_par[w];
    if gradient[i,j]=26 then
    begin
        map[26,26]:=0;
        flag:=false;
```

```
    old_par[w]:=0;
    flag_val[w]:=flag;
    if w<num then w:=w+1 else w:=1;
    goto beg;
end else
if (gradient[i-1,j-1]>gradient[i,j]) and (map[i-1,j-1]<>2) then
begin
    i:=i-1;
    j:=j-1;
    coord_i[w]:=i;
    coord_j[w]:=j;
    if map[i,j]<>4 then old_par[w]:=map[i,j] else
    for x:=num downto 1 do
    if (coord_i[w]=coord_i[x]) and (coord_j[w]=coord_j[x]) and
    (old_par[x]<>4) then old_par[w]:=old_par[x];
    flag_val[w]:=flag;
    if w<num then w:=w+1 else w:=1;
    goto beg;
end;
if (gradient[i-1,j]>gradient[i,j]) and (map[i-1,j]<>2) then
begin
    i:=i-1;
    coord_i[w]:=i;
    coord_j[w]:=j;
    if map[i,j]<>4 then old_par[w]:=map[i,j] else
    for x:=num downto 1 do
    if (coord_i[w]=coord_i[x]) and (coord_j[w]=coord_j[x]) and
    (old_par[x]<>4) then old_par[w]:=old_par[x];
    flag_val[w]:=flag;
    if w<num then w:=w+1 else w:=1;
    goto beg;
end;
if (gradient[i-1,j+1]>gradient[i,j]) and (map[i-1,j+1]<>2) then
begin
    i:=i-1;
    j:=j+1;
    coord_i[w]:=i;
    coord_j[w]:=j;
    if map[i,j]<>4 then old_par[w]:=map[i,j] else
    for x:=num downto 1 do
    if (coord_i[w]=coord_i[x]) and (coord_j[w]=coord_j[x]) and
    (old_par[x]<>4) then old_par[w]:=old_par[x];
    flag_val[w]:=flag;
    if w<num then w:=w+1 else w:=1;
    goto beg;
end;
if (gradient[i,j-1]>gradient[i,j]) and (map[i,j-1]<>2) then
begin
    j:=j-1;
    coord_i[w]:=i;
    coord_j[w]:=j;
    if map[i,j]<>4 then old_par[w]:=map[i,j] else
```

```
    for x:=num downto 1 do
    if (coord_i[w]=coord_i[x]) and (coord_j[w]=coord_j[x]) and
    (old_par[x]<>4) then old_par[w]:=old_par[x];
    flag_val[w]:=flag;
    if w<num then w:=w+1 else w:=1;
    goto beg;
end;
if (gradient[i,j+1]>gradient[i,j]) and (map[i,j+1]<>2) then
begin
    j:=j+1;
    coord_i[w]:=i;
    coord_j[w]:=j;
    if map[i,j]<>4 then old_par[w]:=map[i,j] else
    for x:=num downto 1 do
    if (coord_i[w]=coord_i[x]) and (coord_j[w]=coord_j[x]) and
    (old_par[x]<>4) then old_par[w]:=old_par[x];
    flag_val[w]:=flag;
    if w<num then w:=w+1 else w:=1;
    goto beg;
end;
if (gradient[i+1,j-1]>gradient[i,j]) and (map[i+1,j-1]<>2) then
begin
    i:=i+1;
    j:=j-1;
    coord_i[w]:=i;
    coord_j[w]:=j;
    if map[i,j]<>4 then old_par[w]:=map[i,j] else
    for x:=num downto 1 do
    if (coord_i[w]=coord_i[x]) and (coord_j[w]=coord_j[x]) and
    (old_par[x]<>4) then old_par[w]:=old_par[x];
    flag_val[w]:=flag;
    if w<num then w:=w+1 else w:=1;
    goto beg;
end;
if (gradient[i+1,j]>gradient[i,j]) and (map[i+1,j]<>2) then
begin
    i:=i+1;
    coord_i[w]:=i;
    coord_j[w]:=j;
    if map[i,j]<>4 then old_par[w]:=map[i,j] else
    for x:=num downto 1 do
    if (coord_i[w]=coord_i[x]) and (coord_j[w]=coord_j[x]) and
    (old_par[x]<>4) then old_par[w]:=old_par[x];
    flag_val[w]:=flag;
    if w<num then w:=w+1 else w:=1;
    goto beg;
end;
if (gradient[i+1,j+1]>gradient[i,j]) and (map[i+1,j+1]<>2) then
begin
    i:=i+1;
    j:=j+1;
    coord_i[w]:=i;
```

```
            coord_j[w]:=j;
            if map[i,j]<>4 then old_par[w]:=map[i,j] else
            for x:=num downto 1 do
            if (coord_i[w]=coord_i[x]) and (coord_j[w]=coord_j[x]) and
            (old_par[x]<>4) then old_par[w]:=old_par[x];
            flag_val[w]:=flag;
            if w<num then w:=w+1 else w:=1;
            goto beg;
          end;
          goto rep;
      end; {феномен на борту}
end;
```