Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
Институт космических и информационных технологий
Информатика

УТВЕРЖДАЮ
Заведующий кафедрой
_____А. С. Кузнецов
подпись инициалы, фамилия
« ____» _____ 2017 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

09.03.04 «Программная инженерия»

Вычислительный эксперимент по оценке эффективности алгоритма работы пастушьей собаки

Научный руководитель/ руководитель	_____ подпись, дата	доцент, канд. техн. наук должность, ученая степень	И.В. Евдокимов инициалы, фамилия
Выпускник	_____ подпись, дата		Е.А. Карпова инициалы, фамилия
Нормоконтролер	_____ подпись, дата		О.А. Антамошкин инициалы, фамилия

Красноярск 2017

Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
Институт космических и информационных технологий
Информатика

УТВЕРЖДАЮ
Заведующий кафедрой
_____А. С. Кузнецов
подпись инициалы, фамилия
« ____» _____ 2017 г.

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
в форме бакалаврской работы

Студенту Карповой Екатерине Александровне

Группа КИ13-18Б Направление (специальность) 09.03.04

Программная инженерия

Тема выпускной квалификационной работы: Вычислительный эксперимент по оценке эффективности алгоритма работы пастушьей собаки

Утверждена приказом по университету № 2930/с от 07 марта 2017

Руководитель ВКР: И. В. Евдокимов, доцент, канд. техн. наук

Исходные данные для ВКР: научные статьи, учебная литература, нормативно-правовые документы.

Перечень разделов ВКР: Теоретические основы постановки компьютерного эксперимента, Планирование эксперимента и анализ решений по управлению экспериментом, Компьютерный эксперимент

Перечень графического материала: таблицы, графики, формулы, иллюстрации

Руководитель ВКР _____ И. В. Евдокимов
подпись

Задание принял к исполнению _____ Е. А. Карпова
подпись

« ___ » _____ 2017

РЕФЕРАТ

Выпускная квалификационная работа по теме «Вычислительный эксперимент по оценке эффективности алгоритма работы пастушьей собаки» содержит 54 страницы текстового документа, 31 используемый источник, 9 иллюстраций, 6 таблиц, 19 формул, 3 приложения.

АЛГОРИТМ ПАСТУШЬЕЙ СОБАКИ, УПРАВЛЕНИЕ ГРУППОЙ, ВЫЧИСЛИТЕЛЬНЫЙ ЭКСПЕРИМЕНТ, ЭФФЕКТИВНОСТЬ АЛГОРИТМА.

Цель работы: оценивание эффективности алгоритма работы пастушьей собаки, поиск его применимости к живым и искусственным агентам в различных областях.

Задачи:

1. Изучить теоретические основы построения компьютерного эксперимента;
2. Рассмотреть алгоритм работы пастушьей собаки;
3. Провести анализ рынка программного обеспечения для выбора инструментов разработки;
4. Реализовать алгоритм работы пастушьей собаки и провести соответствующие вычисления;
5. Проанализировать результаты эксперимента;
6. Выявить области применения алгоритма.

В ходе работы был выполнен компьютерный эксперимент. Была установлена зависимость времени выполнения алгоритма от количества автономных объектов, а также типа пастуха. Оказалось, что более эффективно использование виртуального пастуха. Вероятность успеха выполнения алгоритма с преградами на пути – 0,88. Также были выявлены области применения алгоритма с возможными вариантами усовершенствования.

Алгоритм работы пастушьей собаки является эффективным средством для управления группой объектов.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В современном мире тяжело переоценить роль информационных технологий. Практически невозможно представить жизнь без вычислительной техники, так как с ее помощью человек автоматизирует многие сферы своей деятельности, в том числе производство, что позволяет значительно повысить производительность труда. В России по состоянию на 2015 год 84,8% организаций используют специальные программные средства, где 15,1% из них предназначены для управления автоматизированным производством или отдельными техническими средствами и технологическими процессами [1]. Средствами автоматизации выступают приборы и автоматические устройства, одними из которых являются роботы.

Роботов можно разделить на два класса: манипуляционные и мобильные [2, с. 8-9]. Манипуляционный робот – автоматическая машина, стационарная или передвижная, которая состоит из исполнительного устройства в виде манипулятора, имеющего несколько степеней подвижности, и устройства программного управления, служащего для выполнения в производственном процессе двигательных и управляющих функций [2, с. 6]. Мобильные робототехнические системы имеют движущиеся шасси с автоматически управляемыми приводами. Они могут быть колесными, шагающими, колесно-шагающими и т.д [2, с. 9].

Если раньше в промышленности внедрялись и использовались в основном манипуляционные роботы, то сейчас все больше возрастает интерес к мобильным роботам, а задача нахождения оптимального алгоритма управления данным классом роботов и по сей день остается актуальной. Большая часть алгоритмов направлена на управление одиночным мобильным роботом, в то время как существует ряд задач, где эффективнее или даже необходимо задействовать большое количество роботов. Одним из таких алгоритмов является алгоритм работы пастушьей собаки, разработанный

совместно учеными Швеции и Великобритании в 2014 году. Помимо того, что данный алгоритм позволяет роботизировать выпас скота, авторы утверждают, что он также применим для управления автономными, взаимодействующими между собой агентами в различных ситуациях [3].

Таким образом, целью выпускной квалификационной работы является оценивание эффективности алгоритма работы пастушьей собаки, поиск его применимости к живым и искусственным агентам в различных областях.

Для достижения поставленной цели необходимо выполнить следующие задачи:

1. Изучить теоретические основы построения компьютерного эксперимента;
2. Рассмотреть алгоритм работы пастушьей собаки;
3. Провести анализ рынка программного обеспечения для выбора инструментов разработки;
4. Реализовать алгоритм работы пастушьей собаки и провести соответствующие вычисления;
5. Проанализировать результаты эксперимента;
6. Выявить области применения алгоритма.

Объектом исследования данной работы является групповое управление мобильными объектами. Предметом исследования выпускной квалификационной работы является алгоритм работы пастушьей собаки.

1 Теоретические основы постановки компьютерного эксперимента

Во многих областях исследований проведение физического эксперимента может быть затруднительно или даже невозможно. Благодаря вычислительному эксперименту можно исследовать определенный круг вопросов, накапливать результаты, а затем гибко применять их для решения задач в различных сферах деятельности [4].

В зависимости от используемого программного обеспечения может варьироваться форма и даже содержание вычислительного эксперимента. Благодаря языкам программирования высокого уровня или специально разработанным пакетам прикладных программ процедуры анализа систем возможно полностью автоматизировать. Также может быть разработан интерфейс для интерактивного взаимодействия пользователя с ЭВМ [5, 6].

Прежде чем приступить к выполнению компьютерного эксперимента, должна быть построена модель. Под моделью понимается некий упрощенный объект, отражающий особенности реального объекта, процесса или явления. Моделирование – исследование объектов познания на их моделях; построение моделей реально существующих предметов, явлений или процессов [7]. Моделирование довольно тяжело заключить в жесткие рамки, поскольку оно является творческим процессом. Однако можно выделить основные этапы моделирования:

1. Постановка задачи;
2. Разработка модели;
3. Проведение вычислительного эксперимента;
4. Анализ полученных результатов.

На этапе постановки задачи определяется проблема, которую необходимо решить. Происходит изучение объекта для выявления его основных свойств и характеристик.

Для подготовки задачи к решению на компьютере ее представляют в формальном виде. Для моделирования формализация является неотъемлемой частью. Основной тезис формализации: возможность разделения объекта и его обозначения. При формальном описании задачи средствами математики результатом становится математическая модель, позволяющая получить решение исходной задачи с заданной степенью точности.

После построения математической модели ее необходимо преобразовать в компьютерную форму. Тип компьютерной модели для проведения вычислительного эксперимента зависит от решаемой задачи. Существует вариант, когда модель представляется как данные, обрабатываемые в последствие какой-либо программой. Другой вариант компьютерной модели – программа для проведения расчетов [4].

Когда компьютерная модель построена, можно непосредственно приступить к компьютерному эксперименту.

Сам компьютерный эксперимент в наше время имеет огромную популярность, как метод исследования в современном научном познании. В широком смысле под экспериментом можно понимать организацию и наблюдение каких-либо явлений, осуществляемых в условиях, близких к естественным, либо имитируют их. Как уже отмечалось ранее, вычислительный эксперимент обладает некоторыми отличительными способностями и преимуществами перед натуральным экспериментом. Например, при использовании вычислительной техники снижается стоимость постановки эксперимента, а также экономится время. Это обеспечивается за счет вариативности выполняемых расчетов и простоты модификации имитационных математических моделей. Для проведения компьютерного эксперимента необходимо составить план и технологию моделирования.

План моделирования – должен четко отражать последовательность работы с моделью. Обычно план выглядит, как последовательность пронумерованных пунктов, где описываются действия, необходимые исследователю для осуществления работы с компьютерной моделью. На этом

этапе не обязательно конкретизировать, как пользоваться программным инструментарием. Можно сказать, что подробный план отражает стратегию компьютерного эксперимента.

В таком плане в первую очередь необходимо разработать тест, а затем тестирование модели. Под тестированием понимается процесс проверки правильности модели. Сам тест – это набор исходных данных с заранее известным результатом. Для уверенности в правильности получаемых результатов моделирования следует предварительно провести компьютерный эксперимент на модели для составленного теста. При проведении такого эксперимента стоит помнить следующее:

1. Тест не отражает смыслового содержания компьютерной модели, а проверяет ее разработанный алгоритм функционирования;

2. Исходные данные в тесте могут совершенно не отражать реальную ситуацию. Это может быть любая совокупность простейших чисел или символов. Важно то, чтобы вы могли заранее знать ожидаемый результат при конкретном варианте исходных данных.

Если тестирование было выполнено успешно и подтвердилась правильность функционирования модели, то можно перейти к этапу технологии моделирования.

Технология моделирования – совокупность целенаправленных действий пользователя над компьютерной моделью. Каждый эксперимент должен сопровождаться осмыслением результатов, которые станут основой анализа результатов моделирования [8].

После выполнения компьютерного эксперимента проводится изучение поведения модели, а также оценка эффективности алгоритма. На основании полученных результатов либо делается вывод о возможности применения модели для практических задач, либо принимается решение о проведении дополнительной серии экспериментов и корректировки модели, и тогда весь цикл исследований необходимо повторить [4].

1.1 Оценка эффективности алгоритмов

Как уже отмечалось ранее, после построения модели, а также проведения компьютерного эксперимента возникает необходимость оценить ее эффективность. Для этих целей существует множество методик, выбор которых зависит от самого алгоритма, рода задач, решаемых с его помощью, критериев оценки качества и т.д.

К настоящему времени существует несколько классификаций таких методик. В одной из них методики оценки делятся на:

1) Субъективные;
2) Объективные:
    a) Системные;
    b) Прямые:
        i) Аналитические;
        ii) Эмпирические:
            (1) Контролируемые;
            (2) Автоматические [9].

Наибольшей популярностью пользуются субъективные (визуальные) методики оценки. Их суть заключается в том, что результаты, полученные благодаря выполнению алгоритма, визуализируются с помощью различных графиков, диаграмм и т.д., после чего экспертами проводится визуальный анализ данных [10]. Несмотря на свою простоту, субъективная методика оценки эффективности обладает недостатком, отраженном в ее названии, – субъективностью, так как оценка проводится человеком. Таким образом различные эксперты могут получить различные результаты.

Еще одним способом оценки эффективности алгоритма является применение объективной методики, которую можно подразделить на системные и эмпирические.

При использовании системной методики алгоритм оценивается на основе конечных результатов работы всей системы подобных моделей. Такой подход помогает определить наиболее подходящий результат для дальнейшей обработки, но не всегда помогает определить качество работы алгоритма [9].

Прямые методики непосредственно имеют дело с самим алгоритмом или результатом его работы. В зависимости от этого выделяются аналитические и эмпирические методики.

Аналитические методики рассматривают алгоритм независимо от его результатов. При этом изучаемыми свойствами алгоритма могут быть сложность, стратегия реализации, ресурсоемкость, возможность распараллеливания и т.д. Сложность алгоритма можно оценить по порядку величины. Алгоритм имеет сложность O(f(n)), если при увеличении размерности входных данных N, время выполнения алгоритма возрастает с той же скоростью, что и функция f(N). При этом используется только такая часть, которая возрастает быстрее всего. Таким образом стоит уделить особое внимание циклам и вызовам процедур, в том числе рекурсиям [11]. Подобные свойства не имеют прямого отношения к качеству работы алгоритма. Однако данная методика может оказаться очень полезной, если имеются какие-либо ограничения по времени выполнения алгоритма, аппаратной части, либо другие.

Если аналитические методики предназначены для работы с самим алгоритмом, то эмпирические оценивают непосредственно результат его работы на некотором тестовом наборе. Их можно подразделить на контролируемые и автоматические.

Контролируемые методики используют количественные меры различия результатов работы алгоритма с некоторыми эталонными значениями, зачастую созданными искусственно. Такие значения являются идеальными результатами эксперимента с точки зрения экспериментатора. Возможна ситуация, когда для каждого алгоритма исследуется несколько его свойств. В этом случае каждому тестовому значению может соответствовать сразу

несколько эталонных значений. Использование таких методик дает хорошую оценку, однако существует ряд алгоритмов, для которых создание эталонных значений становится довольно трудоемким процессом и вносит элемент субъективизма.

В автоматических методиках производится количественная оценка некоторых результатов моделирования, полученных при выполнении компьютерного эксперимента, на основе чего делается вывод о качестве алгоритма. В отличие от контролируемых методик уже не требуются какие-либо эталонные значения, что, возможно, является их основным достоинством. Это свойство позволяет осуществлять контроль и самообучение в системах реального времени [9].

Прежде чем сделать выбор в пользу той или иной методики оценивания эффективности, следует рассмотреть сам алгоритм.

1.2 Алгоритм работы пастушьей собаки

Модель предлагает решение проблемы выпаса. С помощью набора простых эвристик пастух может пасти автономных, взаимодействующих между собой агентов в направлении к пункту назначения. Эвристика пастуха основана на адаптивном переключении между сбором агентов, когда они слишком рассредоточены, и их вождением, как только они агрегируют. Эти правила действуют для уменьшения вероятности того, что группа распадается, и позволяют пастуху вести группу в направлении целевого местоположения. Движение пастуха из стороны в сторону также является следствием этих правил, в отличие от других моделей, где такое поведение было жестко закодировано в правилах движения управляющего объекта для повышения эффективности работы [12, 13, 14].

Прежде всего приведем в таблице 1 обозначения, описания и типичные значения параметров модели.

Таблица 1 – Параметры модели. Обозначения, описания и типичные значения

Обозначение	Описание	Типичные значения
L	Длина стороны квадрата генерации агентов	150 м
параметры агентов		
N	Общее число агентов	1 – 201
n	Количество ближайших соседей	1 – 200
r_s	Расстояние обнаружения пастуха	65 м
r_a	Расстояние для начала взаимодействия агентов	2 м
ρ_a	Относительная сила отталкивания от других агентов	2
c	Относительная сила притяжения к n ближайшим соседям	1,05
ρ_s	Относительная сила отталкивания от пастуха	1
h	Относительная сила инерции	0,5
e	Относительная сила углового шума	0,3
δ	Смещение агента за временной шаг	1 м/временной шаг
p	Вероятность движения за один временной шаг	0,05
параметры пастуха		
δ_s	Смещение пастуха за временной шаг	1,5 м/временной шаг
P_d	Позиция для вождения	$r_a\sqrt{N}$ за группой
P_c	Позиция для сбора	r_a м от ближайшего агента
e	Относительная сила углового шума	0,3

Первоначально N стайных агентов выпускаются в случайных позициях в верхней правой четверти квадратного поля L x L, и пастух выпускается за пределами этой области. Каждый агент стремится держаться подальше от пастуха, оставаясь как можно ближе к своим n ближайшим соседям. Такое поведение притягиваться к близлежащим соседей и отталкивается от потенциальных угроз характерно для овец и многих других пастушьих

животных [15]. На рисунке 1 показаны правила, которыми руководствуются агенты.

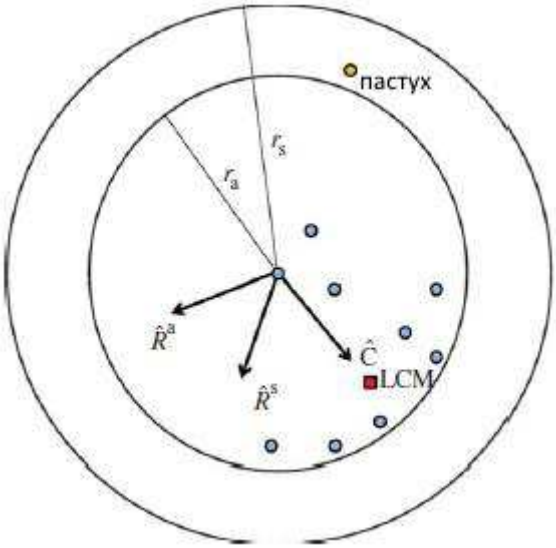


Рисунок 1 – Правила передвижения агентов

Если агент находится от пастуха на расстоянии дальше, чем r_s, то он остается неподвижным, за исключением случайных движений, происходящих время от времени. Вне зависимости от расстояния до пастуха, агенты отталкиваются от других агентов на очень коротких расстояниях, которые меньше, чем r_a, а единичный вектор $\hat{R}^a$ указывает направление этого локального отталкивания. Если агент находится в пределах расстояния r_s от пастуха, агент притягивается к местному центру масс (LCM) своих n ближайших соседей, в направлении единичного вектора $\widehat{C}$, и в то же время отталкивается непосредственно от пастух в направлении $\hat{R}^s$. Новое направление агента $\overline{H}'$ является линейной комбинацией этих трех векторов, взвешенных с помощью соответствующих параметров модели ρ_a, с, ρ_s плюс слабая инерция $d\widehat{H}$ и малый шум $e\epsilon$. $\overline{H}'$ является нормированным, и агент перемещается на дистанцию δ в этом направлении. ρ_a является относительной

силой отталкивания от других агентов, c – относительной силой притяжения к другим агентам и ρ_s – относительной силой отталкивания от пастуха.

Задача пастуха состоит в том, чтобы собрать и загнать всех агентов в нижний левый угол поля. Если пастух находится в пределах $3r_a$ от любого агента, он не движется в этом временном шаге. В противном случае, пастух совершает одно из двух действий, в зависимости от положения агентов (рисунок 2).

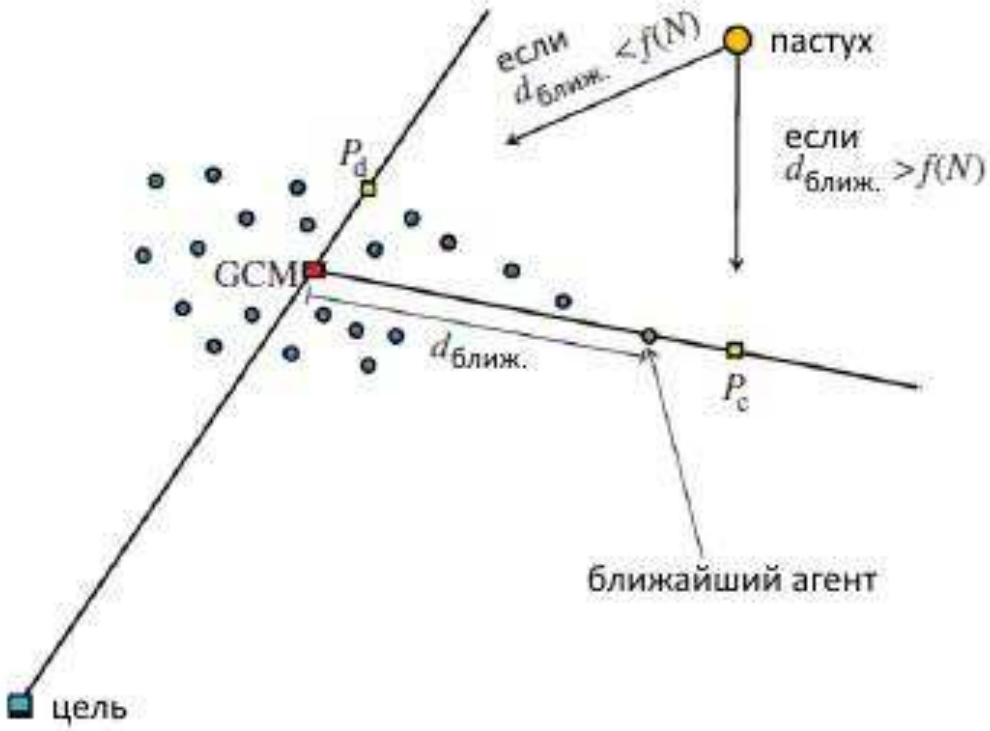


Рисунок 2 – Выбор движения пастуха

Если все агенты находятся в пределах расстояния $f(N)$ от их глобального центра масс (GCM), то пастух принимает позицию ведущего P_d непосредственно позади стада относительно цели. Пометим такое поведение как "вождение". Если по крайней мере один агент находится на расстоянии дальше, чем $f(N)$ от GCM, то вместо этого пастух принимает позицию сбора в направлении P_c непосредственно позади агента, находящегося на

наибольшем расстоянии от GCM. Мы называем это поведение "сбор". При сборе или вождении пастух перемещается на расстояние δ_s.

Первоначально N агентов расположены случайным образом в верхней правой четверти квадрата L x L, а пастух выпущен в нижней левой четверти. Квадрат никак не огорожен, поэтому агенты и пастух могут выйти за его пределы позднее. Обозначим позицию пастуха S и положение i-го агента – A_i. Если агент находится на расстоянии дальше, чем r_s от пастуха, то он пасется. То есть они, как правило, стационарные, но демонстрирует небольшие случайные движения. Если расстояние до пастуха меньше r_s, то каждый i-й агент будет отталкиваться непосредственно от него в направлении $\overline{R_i^s} = \bar{A}_i - \bar{S}$ и в то же время будет притягиваться к центру масс его n ближайших соседей $(L\bar{C}M)_i$ в направлении $\overline{C_i} = L\bar{C}M_i - \bar{A}_i$. Агенты также локально отталкиваются друг от друга, поэтому если два или более агентов находятся в пределах расстояния r_a друг от друга, то возникнет отталкивающая сила. Точнее, если i-й агент имеет k соседей на расстоянии r_a в положениях A_1, …, A_k, сила отталкивания i-го агента определяется по формуле

$$\overline{R_i^a} = \sum_{j=1}^{k} \frac{1}{|\overline{A_i} - \overline{A_j}|}\left(\bar{A}_i - \bar{A}_j\right). \quad (1)$$

Направление агента i в следующем временном шаге будет $\overline{H_i'}$, представляющее собой линейную комбинацию этих сил (нормированных) плюс инерция $\widehat{H_i}$ и ошибка $\hat{\epsilon}$ (рисунок 1). Его можно описать следующим образом:

$$\overline{H_i'} = h\widehat{H_i} + c\widehat{C_t} + \rho_a\widehat{R_i^a} + \rho_s\widehat{R_i^s} + e\widehat{\epsilon_i}, \quad (2)$$

где веса выбраны таким образом, что $\rho_a > c > \rho_s > h$. Причины такого неравенства в том, что отталкивание ρ_a типа агент-агент должно быть

доминирующим для поддержания групп любого размера. Также в реальном мире овцы, как правило, агрегируют, а не сразу расходиться в присутствии собаки [6]. Таким образом, мы предполагаем, что локальное притяжение между агентами сильнее, чем отталкивания от пастуха $c > \rho_s$. И, наконец, тенденция действовать в прежнем направлении h включается, чтобы предотвратить резкие повороты и сгладить траектории. Типичные значения для этих параметров при моделировании: ρ_a = 2, c = 1,05, ρ_s = 1 и h = 0,5. Агент i передвинется на расстояние δ в этом направлении $\overline{H_i'}$, и его новое положение задается по формуле

$$\overline{A_i'} = \overline{A_i} + \delta\widehat{H_i'}. \quad (3)$$

Задача пастуха заключается в том, чтобы собрать всех агентов в одно стадо и загнать их в левый нижний угол квадрата L x L. Когда центр массы стада (GCM) находится в пределах определенного расстояния от начала координат, задача пастуха завершена. Пока задача не решена, пастух выбирает один из двух возможных ходов, собирать или загнать, на каждом временном шаге. Это решение зависит от того, находятся ли все агенты в пределах расстояния f(N) от GCM (рисунок 2). Чтобы стадо было идеально круглым, f(N) должно быть $r_a\sqrt{N}$, поэтому, чтобы учесть асимметрию стада, мы возьмем $f(\mathrm{N}) = r_a N^{2/3}$. Если все агенты находятся за пределами f(N), то пастух движется по направлению к точке P_c, чтобы собрать самого удаленного от GCM агента в позиции A_f. Если стадо собрано, то есть все агенты находятся в пределах f(N), то пастух позиционирует себя на P_d для вождения стада. Пастух пытается двигаться по прямой линии к этим точкам, но если он попадает в зону $3r_a$ агента, его скорость δ_s устанавливается на уровне $0.3r_a$. Пастух испытывает тот же шум, что и агенты $e\hat{\epsilon}$; этот шум имеет решающее значение для разрешения тупиковых ситуаций. Пастух будет повторять все эти действия до

тех пор, пока GCM не будет находиться в пределах определенного расстояния от начала координат [3].

1.3 Постановка задачи исследования

Для проведения компьютерного эксперимента необходимо реализовать алгоритм работы пастушьей собаки. Также следует дополнить алгоритм возможностью управления группой объектов с помощью виртуального пастуха. Использование виртуального пастуха подразумевает, что управляющий объект будет моментально занимать необходимую позицию, не затрачивая время на преодоление дистанции между текущей и целевой позициями. Кроме того, следует добавить реакцию агентов на другие объекты, а также возможность прокладывать маршрут в обход препятствиям, возникающим на пути к цели, что должно реализовываться с помощью введения подцелей. Подцели должны задаваться так, чтобы группа могла обогнуть возникающий на пути объект. Как только подцель будет достигнута, следует выбрать новую подцель. Если на пути к основной цели больше не останется преград, то подцелью необходимо выбрать саму цель. Также в программе должна быть добавлена возможность выбора размера и координат области генерации агентов, координат цели и начальных координат пастуха.

После реализации программы следует выполнить серию экспериментов для получения результатов, позволяющих провести анализ данного алгоритма. Эксперименты будут направлены на выявление зависимости времени выполнения алгоритма от таких параметров, как количество автономных объектов, виртуальности пастуха и количества преград.

Оценку эффективность данного алгоритма, в виду его специфики, провести, использую субъективные методики, с помощью визуального анализа данных.

На основе оценки эффективности стоит произвести анализ применимости рассматриваемого алгоритма относительно живых и искусственных агентов.

2 Планирование эксперимента и анализ решений по управлению экспериментом

Планирование эксперимента напрямую связано с разработкой и исследованием математической модели объекта исследования.

Планирование эксперимента – это процедура выбора числа и условий проведения опытов, необходимых и достаточных для решения поставленной задачи с требуемой точностью [16, с. 7].

При этом, как учит нас теория, необходимо придерживаться следующих ограничений:

1. общее число опытов должно быть по возможности минимальным;
2. необходимо одновременно изменять все переменные, определяющие (влияющие) процесс. Причем это изменение должно происходить по определенным правилам–алгоритмам;
3. при описании исследований необходимо использовать математический аппарат, формализующий действия экспериментатора;
4. в процессе проведения и планирования эксперимента необходимо придерживаться четкой стратегии, позволяющей принимать обоснованные решения после каждой серии экспериментов [16, с. 7–8].

Прежде чем приступать к эксперименту, необходимо однозначно и непротиворечиво сформулировать основную цель эксперимента, определиться с параметром оптимизации. Параметр оптимизации должен быть единственным, хотя он и может принимать различные значения. Под параметром оптимизации понимают характеристику цели, заданную

количественно. Параметр оптимизации является откликом на воздействие факторов, которые определяют поведение исследуемой системы. Каждый реальный объект может характеризоваться несколькими или одним параметром оптимизации. Параметр оптимизации необходимо выбирать с учетом комплекса требований. Он должен:

1. быть количественным, т.е. иметь числовую оценку;
2. обладать однозначностью в статистическом смысле. Заданному набору значений факторов должно соответствовать одно значение параметра оптимизации, при этом обратное утверждение неверно: одному и тому же значению параметра могут соответствовать разные наборы значений факторов;
3. быть универсальным и всесторонне отражать характеристики объекта, процесса, явления. Универсальными обычно являются экономические и технико-экономические параметры (себестоимость, надежность и др.);
4. быть эффективным как с точки зрения достижения цели, так и в статистическом смысле. Если, например, за параметр оптимизации принять себестоимость восстановления детали, то он не будет характеризовать надежность ее работы. Поэтому в качестве параметра оптимизации целесообразно выбирать себестоимость при допустимой износостойкости или износостойкость при допустимой себестоимости. Статистически эффективным параметром оптимизации является тот, который имеет наименьшие ошибки измерений;
5. иметь ясный физический смысл. Это требование не только определяет цель исследования, но и облегчает интерпретацию полученных результатов эксперимента [17, с. 15].

Ранее уже было отмечено, что параметром оптимизации является время выполнения алгоритма.

Необходимо определиться с факторами, влияющими на ход эксперимента и с тем, какие значения принимают эти факторы. Влияющих факторов, вообще говоря, может быть сколько угодно, при этом каждый из них

может принимать бесконечное число значений. Однако не следует забывать, что в зависимости от числа факторов и их уровней катастрофически растет и число экспериментов [16, с. 25, 17]. В постановке задачи исследования уже были определены необходимые факторы:

1. Количество агентов;
2. Виртуальность пастуха;
3. Количество преград.

Математическая задача планирования эксперимента состоит в том, чтобы найти уравнение поверхности отклика:

$$y = \varphi(u_1, \dots, u_m) + h, \tag{4}$$

где y – выход объекта, т.е. параметр оптимизации,

u_i – фактор, оказывающий влияние на выход,

h – центрированная помеха [18, с. 70].

Таким образом, математическое планирование связано с изучением формы поверхности отклика; следовательно, оптимальному значению выхода соответствуют максимальные или минимальные точки этой поверхности. Для большинства реальных процессов вид поверхности отклика заранее неизвестен, поэтому при экспериментальном поиске оптимальных условий функцию y представляют в виде степенного ряда [18, 19].

$$y = a_0 + \sum_{i=1}^{m} a_i u_i + \sum_{i \le} \sum_{j} a_{ij} u_i u_j + \cdots + \sum_{i \le} \dots \sum_{j} a_{i \dots j} u_i \dots u_j. \tag{5}$$

Точность подобной аппроксимации определяется порядком степенного ряда и диапазоном изменения переменных. Поверхность отклика изучается обычно в сравнительно узком интервале варьирования факторов, поэтому без большой погрешности можно отбросить члены высших порядков.

Коэффициенты степенного ряда (коэффициенты регрессии) можно оценить с помощью выборочных коэффициентов регрессии.

Для каждого фактора выбирают условный нулевой (исходный) уровень u_i^0, диапазон и шаг варьирования переменных Δu_i. Диапазон изменения факторов равен разности между верхним и нижним пределами данного фактора. В процессе установления варьирования решается вопрос о масштабе фактора, для выбора которого необходимо обратиться к теоретическим источникам, либо иметь некоторые априорные представления [19, с. 14–16].

Для параметра «количество агентов» $u_1^0 = 55{,}5$, диапазон – 91, $\Delta u_1 = 45{,}5$, для виртуальности пастуха выбираем следующие параметры $u_2^0 = 0{,}5$, диапазон – 1, $\Delta u_2 = 0{,}5$ и для количества преград $u_3^0 = 2$, диапазон – 4, $\Delta u_3 = 2$.

Так как параметр оптимизации влияют всего три фактора, достаточно удобно будет провести полный факторный эксперимент для построения линейной модели. Полным факторным экспериментом называется эксперимент, в котором реализуются все возможные сочетания уровней факторов [18, с. 78]. При построении линейной модели используем полный факторный эксперимент типа 2^3. В таблице 2 для него приведена матрица планирования.

Таблица 2 – Матрица планирования для полного факторного эксперимента 2^3

Номер эксперимента	x_1	x_2	x_3	x_1x_2	x_1x_3	x_2x_3	$x_1x_2x_3$	y_i
1	+	+	+	+	+	+	+	y_1
2	-	+	+	-	-	+	-	y_2
3	+	-	+	-	+	-	-	y_3
4	-	-	+	+	-	-	+	y_4
5	+	+	-	+	-	-	-	y_5
6	-	+	-	-	+	-	+	y_6

Окончание таблицы 2

Номер эксперимента	x_1	x_2	x_3	x_1x_2	x_1x_3	x_2x_3	$x_1x_2x_3$	y_i
7	+	-	-	-	-	+	+	y_7
8	-	-	-	+	+	+	-	y_8

Само уравнение модели выглядит следующим образом:

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_{12} x_1 x_2 + \beta_{13} x_1 x_3 + \beta_{23} x_2 x_3 + \\ + \beta_{123} x_1 x_2 x_3, \quad (6)$$

где x_i – безразмерный входной параметр,

β_i – новые коэффициенты модели.

Переход от размерных входных переменных к безразмерным, а также расчет коэффициентов β_i из a_i и обратно показаны ниже.

$$x_i = \frac{u_i - u_i^0}{\Delta u_i}. \quad (7)$$

$$\beta_0 = a_0,\ \beta_i = a_i \Delta u_i. \quad (8)$$

$$a_0 = \beta_0,\ a_i = \frac{\beta_i}{\Delta u_i}. \quad (9)$$

После выполнения нашего эксперимента коэффициенты β_i будут рассчитываться так:

$$\beta_0 = \frac{y_1+y_2+y_3+y_4+y_5+y_6+y_7+y_8}{8},\ \beta_1 = \frac{y_1-y_2+y_3-y_4+y_5-y_6+y_7-y_8}{8},$$
$$\beta_2 = \frac{y_1+y_2-y_3-y_4+y_5+y_6-y_7-y_8}{8},\ \beta_3 = \frac{y_1+y_2+y_3+y_4-y_5-y_6-y_7-y_8}{8},$$
$$\beta_{12} = \frac{y_1-y_2-y_3+y_4+y_5-y_6-y_7+y_8}{8},\ \beta_{13} = \frac{y_1-y_2+y_3-y_4-y_5+y_6-y_7+y_8}{8},$$
$$\beta_{23} = \frac{y_1+y_2-y_3-y_4-y_5-y_6+y_7+y_8}{8},\ \beta_{123} = \frac{y_1-y_2-y_3+y_4-y_5+y_6+y_7-y_8}{8}. \quad (10)$$

Выполнив планирование эксперимента можно непосредственно приступить к разработке программы, позволяющей получить необходимые данные.

2.1 Анализ рынка программного обеспечения

Прежде чем приступить к разработке приложения, реализующего исследуемый алгоритм, проанализируем рынок программного обеспечения с целью выбора наиболее подходящих инструментальных средств.

Для проведения научных вычислений существует множество различных прикладных программ. Самым популярным пакетом является MATLAB, так как он имеет большой функционал для решения задач технических вычислений, а также свой встроенный одноименный язык [20].

Ближайшим бесплатным аналогом MATLAB является Scilab – бесплатное программное обеспечение с открытым исходным кодом, предоставляющее мощную среду для инженерных и научных вычислений [21]. Данный продукт также имеет множество математических функций и инструментов и встроенный язык, имеющий сходство с MATLAB.

Еще одна популярная программа MapleSim – это средство моделирования и симуляции на системном уровне, которое применяет современные методы, чтобы значительно сократить время разработки модели, обеспечить более глубокое понимание поведения системы и обеспечить быстрое и точное моделирование [22].

Проведем сравнительный анализ вышеперечисленных программных продуктов.

Таблица 3 – Сравнение программных продуктов

Параметр	MATLAB	Scilab	MapleSim
Математика и вычисления	+	+	+
Визуализация данных	+	+	+
Работа с внешними интерфейсами	+	+	+
Разработка алгоритмов	+	-	-
Встроенный язык программирования	+	+	+
Лицензия	проприетарное	свободная	проприетарное

Как видно из таблицы 3 все пакеты обладают большим наборов инструментов для моделирования и анализа полученных данных. Однако продукты MATLAB и MapleSim являются проприетарным. Также стоит отметить, что Scilab и MapleSim не нацелены на разработку алгоритмов. Кроме того, исследуемая модель обладает определенной спецификой, так как является алгоритмом управления групповыми объектами. Возможности данных средств не позволяют реализовать его в полной мере, поэтому следует разработать отдельную программу, выполняющую данный алгоритм. Плюс такого подхода заключается в том, что появляется возможность задействовать вычислительные ресурсы в том объеме, который требуются для выполнения поставленной задачи. Также при правильном проектировании достаточно просто модернизировать и внедрять алгоритм в другие рабочие программы.

После того, как был сделан выбор в пользу разработки программы для выполнения алгоритма, следует определиться с языком разработки. Существующие на сегодняшний день языки программирования можно выделить в следующие группы:

1. Универсальные языки высокого уровня;
2. Специализированные языки разработчика ПО;
3. Специализированные языки пользователя;
4. Языки низкого уровня.

Для реализации алгоритма следует выбрать один из универсальных языков высокого уровня, позволяющий производить научные расчеты с необходимой временными затратами и точностью. Такими языками являются C++, C#, Java и т.д.

Одним из самых популярных языков программирования является C++. Его используют для написания операционных систем, различный драйверов устройств, прикладных программ, развлекательных и других приложений. Свой синтаксис данный язык унаследовал от C.

C++ обладает следующими достоинствами:

1. Код, написанный на языке C, можно скомпилировать компилятором C++. Иногда все же требуется провести некоторые изменения, но в большинстве случаев это сделать довольно просто;
2. Высокая вычислительная производительность. Программист имеет возможность работы с памятью на низком уровне, а также контролировать все аспекты структуры и порядка исполнения программы;
3. Наличие различных стилей и технологий программирования. При написании кода можно придерживаться объектно-ориентированного стиля программирования, структурного программирования, обобщенного программирования и т.д.;
4. Автоматический вызов деструкторов в порядке, обратном вызову конструкторов. Это упрощает и повышает надежность управления памятью и другими ресурсами;
5. Перегрузка операторов. Таким образом программист может кратко и емко записывать выражения над пользовательскими типами в естественной алгебраической форме;
6. Шаблоны C++ позволяют создавать обобщенные контейнеры и алгоритмы для различных типов данных;
7. Наличие большого количества учебной литературы.

Однако C++ имеет свои недостатки:

1. Плохая поддержка модульности. Использование #include существенно замедляет компиляцию при использовании большого количества модулей. Также необходимо писать объявление функций в заголовочных файлах и их определение в файлах с исходным кодом;

2. Использование макросов #define. Они одновременно являются мощным и опасным средством;

3. Не интуитивность некоторых преобразований типов. Например, операция над беззнаковым и знаковым числами выдаёт беззнаковый результат;

4. Сложность и избыточность, из-за которых C++ трудно изучать [23].

Еще одним универсальным языком программирования, чьи достоинства и недостатки также будут рассмотрены, является C#. Данный язык обладает C-подобным синтаксисом, наиболее близким к C++ и Java.

Его достоинствами являются:

1. Простой и надежный. По сравнению с языками C и C++ были упрощены работы с пользовательскими типами, контейнерами и другими элементами языка. Также были исключены некоторые проблематичные модули, такие как множественное наследование;

2. Сборка мусора позволяет программистам не волноваться об утечках памяти;

3. Благодаря каркасу Framework.Net программы, написанные на C#, достаточно просто запускать на различных операционных системах.

К недостаткам данного языка можно отнести:

1. Медленную работу динамической компиляции с проверкой во время выполнения относительно предшественников C и C++;

2. Отрицательное влияние сборщика мусора на производительность системы, так как все потоки программы должны периодически приостанавливаться [24].

Последним рассмотрим язык программирования Java. Данный язык является сильно типизированным объектно-ориентированным. Приложения

Java транслируются в байт-код, благодаря чему могут работать на любой компьютерной архитектуре, где установлена виртуальная Java-машина.

Достоинства языка Java:

1. Независимость байт-кода от операционной системы и оборудования, что уже было отмечено выше;
2. Строгая типизация вносит больше ограничений на действия с переменными и величинами различных типов, устраняя возможные потери данных;
3. Наличие готовых шаблонов проектирования повышает производительность работы программистов;
4. Огромное количество профессиональных бесплатных средств разработки.

Недостатками языка Java являются:

1. Снижение производительности программ и алгоритмов из-за концепции виртуальной машины. Однако для современных версий языка данный недостаток практически перестал быть актуальным;
2. Малопригодность для разработки оконных приложений. В первую очередь язык Java направлен на создание клиент-серверных приложений [25].

После рассмотрения нескольких универсальных языков высокого уровня выбор был сделан в пользу C++. Для проверки алгоритма на эффективность необходима высокая производительность программы, что может быть достигнуто средствами работы с памятью данного языка. Кроме того, выбор языка определялся опытом и знаниями конкретного разработчика.

После выбора языка программирования необходимо осуществить выбор среды разработки.

Разработка программы будет осуществляться на операционной системе Windows 8.1. Для этой платформы популярными фреймворками являются Microsoft Visual Studio и Qt.

Выбор был сделан в пользу Qt, так как написанные в нем программы запускаются в большинстве современных операционных систем путём

простой компиляции программы для каждой ОС без изменения исходного кода. Также решающим фактором стала возможность использовать Meta Object Compiler (MOC) — предварительную системы обработки исходного кода. MOC увеличивает мощь библиотек в несколько раз, используя слоты и сигналы. Кроме того, это позволяет сделать код более лаконичным. Утилита MOC ищет в заголовочных файлах на C++ описания классов, содержащие макрос Q_OBJECT, и создаёт дополнительный исходный файл на C++, содержащий метаобъектный код [26]. Также Qt распространяется под свободной лицензией, что является немаловажным преимуществом [27].

Сама программа, выполняющая алгоритм пастушьей собаки, будет распространяться также под свободной лицензией.

2.1.1 Применений свободных лицензий в России: теория, практика, перспективы развития

Сегодня свободные лицензии имеют широкое распространение на практике во многих странах мира. Они предоставляют возможность творческим людям самовыражаться, используя размещенные в свободном доступе ресурсы, а также самим публиковать свои работы на определенных условиях. И в России в гражданском кодексе уже существуют статьи, регулирующие некоторые вопросы открытых лицензий.

Согласно [28] «Открытая лицензия является договором присоединения». При этом условия данного типа лицензии должны быть доступны неопределенному кругу лиц и размещены так, чтобы лицензиат ознакомился с ними перед началом использования соответствующего ресурса.

Кроме того, «Лицензиар может предоставить лицензиату право на использование принадлежащего ему произведения для создания нового результата интеллектуальной деятельности». То есть, если иное не предусмотрено открытой лицензией, считается, что лицензиар сделал

предложение заключить договор об использовании принадлежащего ему произведения любым лицам, желающим использовать новый результат интеллектуальной деятельности, созданный лицензиатом на основе этого произведения, в пределах и на условиях, которые предусмотрены открытой лицензией. Акцепт такого предложения считается также акцептом предложения лицензиара заключить лицензионный договор в отношении этого произведения.

В законе также говорится, что открытая лицензия является «безвозмездной, если ею не предусмотрено иное». Для программ срок действия открытой лицензии равен всему сроку действия исключительного права, если его не определить. Также, если не указана территория, на которой допускается использовать программу для ЭВМ, то ее использование возможно на территории всего мира.

Свободные лицензии, в отличие от открытой лицензии, применяются к произведениям, которые являются объектами авторского права [29]. Существует множество свободных лицензий. Их классифицируют по масштабу ограничений, по назначению или даже по авторам. Наиболее значимые лицензии:

1. GNU General Public License Version 3 (GNU GPL). На данный момент является самой известной свободной лицензией. При использовании данной лицензии программы возможно распространять как бесплатно, так и за деньги. Однако всегда необходимо соблюдать одно условие – пользователю передается весь комплекс прав: право на свободное распространение программы, исходный код программы, право на модификацию программы и использование ее элементов в собственных разработках и т. д.

2. GNU Lesser General Public License Version 3 (GNU LGPL). Такая лицензия в отличие от GNU GPL предназначена для библиотек и позволяет использовать их в проприетарном ПО.

3. Mozilla Public License Version 2.0 (MPL). Особенностью данной лицензии является то, что при использовании неизмененной библиотеки под

MPL, как части большего проекта, можно лишь указать ссылки на исходники этой библиотеки. Также, в отличие от GPL и LGPL, лицензия распространяется на файлы, а не проекты.

4. Apache 2.0. Данная лицензия не ставит условием неизменность лицензии распространения программного обеспечения, и не настаивает даже на сохранении его бесплатного и открытого статуса. Единственным условием, накладываемым лицензией Apache, является информирование получателя о факте использования исходного кода.

Конечно, был представлен далеко не весь перечень свободных лицензий, и используются также другие виды. Выбор того или иного вида лицензии осуществляется в зависимости от потребностей авторов и пользователей.

Но все-таки в российском законодательстве существуют некоторые проблемы. До сих пор нет четкого механизма реализации открытых лицензий в РФ, что может приводить к ущемлению интересов авторов. Также существует проблема установления первичного обладателя прав на результат творческой интеллектуальной деятельности – лицензиара [30].

Все же российским законодателем был сделан достаточно значимый шаг навстречу потребностям информационного общества [31]. Вполне возможно, что в скором будущем будут решены имеющиеся проблемы.

2.1.2 Коммуникационные каналы ИТ-разработчиков открытого кода

Условия распространения программ с открытым кодом зависят от конкретного вида лицензии. Использование свободных лицензий обычно не накладывает никаких ограничений на каналы распространения. Таким образом, возможно использование любых носителей. Но самым распространенным коммуникационным каналом для ИТ-разработчиков открытого кода является интернет.

Крупнейшим веб-сервисом для хостинга ИТ-проектов является GitHub. На нем можно абсолютно бесплатно размещать проекты с открытым исходным кодом. Помимо размещения кода участники могут общаться, комментировать правки друг друга, а также следить за новостями знакомых. У каждого проекта есть личная страница, где можно подробно описать его. Также на сайте можно просмотреть файлы проектов с подсветкой синтаксиса для большинства языков программирования. С использование данного сайта любой желающий может ответвиться от проекта, внести какие-либо изменения, доработки и поделиться ими с окружающими.

Таким образом, для разработчиков открытого программного обеспечения существует множество каналов распространения своих работ.

2.2 Основной способ экспериментальной оценки повторяемости и воспроизводимости результатов эксперимента

Помимо основных факторов, влияние которых исследуется при проведении эксперимента, на объект действуют и случайные факторы. Для учета случайных погрешностей находится ошибка воспроизводимости опытов, оценка которой находится в зависимости от имеющихся у экспериментатора ресурсов.

Если для каждой экспериментальной точки плана проводилось одинаковое количество опытов, то среднее значение замеряемого параметра рассчитывается следующим образом:

$$\overline{y_j} = \frac{1}{m}\sum_{i=1}^{m} y_{ji}, \tag{11}$$

где y_{ji} – значение параметра в j-ой точке плана в i-ом опыте,

m – число опытов в j-ой точке плана.

Для расчета дисперсии воспроизводимости используют следующую формулу:

$$\sigma^2_{\text{восп.}} = \frac{1}{n\cdot(m-1)}\sum_{j=1}^{n}\sum_{i=1}^{m}(y_{ji} - \overline{y_j})^2, \quad (12)$$

где n – количество экспериментов.

Для облегчения расчетов формулу (12) можно записать так:

$$\sigma^2_{\text{восп.}} = \frac{1}{n}\sum_{j=1}^{n}\sigma_j^2. \quad (13)$$

Дисперсия воспроизводимости используется для проверки адекватности полученной модели.

2.3 Методы проверки адекватности модели

Построенная модель в обязательном порядке должна быть подвергнута проверке на адекватность. Один из способов проверки осуществляется с помощью критерия Фишера: если $F_{\text{расч.}} < F_{\text{табл.}}$, то уравнение регрессии является адекватным, иначе – неадекватным. Расчетное значение критерия $F_{\text{расч.}}$ находится по следующей формуле:

$$F_{\text{расч.}} = \frac{\sigma^2_{\text{ост.}}}{\sigma^2_{\text{восп.}}}, \quad (14)$$

где $\sigma^2_{\text{ост.}}$ – остаточная дисперсия (дисперсия адекватности).

Дисперсия адекватности вычисляется по формуле:

$$\sigma^2_{\text{ост.}} = \frac{m}{n-r}\sum_{j=1}^{n}(\widetilde{y_j} - \overline{y_j})^2, \quad (15)$$

где r – число значимых коэффициентов в полученном уравнении регрессии;

$\widetilde{y_j}$ – значение изучаемого параметра, вычисленное по уравнению регрессии со значимыми коэффициентами для j-ого эксперимента.

Табличное значение $F_{\text{табл.}}$ берется из таблицы критических точек распределения Фишера. Но прежде необходимо задать уровень значимости α и степени свободы $k_1 = n - r$ и $k_2 = n \cdot (m - 1)$.

Существует также другой способ проверки адекватности уравнения регрессии. Для этого используется коэффициент β_0, который оценивает в совокупности коэффициент b_0 и все коэффициенты b_{ii}, стоящие перед x_i^2 полиномиального уравнения сигнальной части объекта. Кроме того для коэффициента b_0 можно получить оценку β_0^0 усреднением результатов опытов в центре плана. Разность $\beta_0 - \beta_0^0$ представляет из себя оценку для суммы коэффициентов $\sum b_{ii}$. Если такая сумма нулю, то полученная модель хорошо описывает объект, иначе гипотеза об адекватности отвергается. Для проверки гипотезы используется статистика Стьюдента:

$$t = \frac{(\beta_0 - \beta_0^0)}{\widehat{\sigma_{\beta_0}}}, \tag{16}$$

где $\widehat{\sigma_{\beta_0}}$ – оценка дисперсии для коэффициента β_0, вычисляемая по формуле ниже:

$$\widehat{\sigma_{\beta_0}} = \frac{\widehat{\sigma_y}}{\sqrt{n}}. \tag{17}$$

Гипотеза адекватности принимается, если $|t| < t_{v,\alpha}$. v – число степеней свободы оценки $\widehat{\sigma_y}$, а α – уровень значимости.

После рассмотрения двух методов по оценке адекватности модели был сделан выбор в пользу первого метода с использованием статистики Фишера.

3 Компьютерный эксперимент

Перед проведением вычислений была написана программы, выполняющая алгоритм работы пастушьей собаки. Руководство программиста с описанием исходного кода программы представлено в приложении А, а руководство пользователя – в приложении Б.

При планировании эксперимента было установлено, что для исследования модели необходимо провести восемь типов опытов. Для каждого типа проводилось по 50 экспериментов. В общей сложности было проведено 400 опытов.

В результате проведения эксперимента были получены наборы значений времени выполнения алгоритма в секундах, с которыми можно ознакомиться в таблице 4.

Таблица 4 – Результаты компьютерного эксперимента в секундах

Опыт	Результат							
	1	2	3	4	5	6	7	8
y_1	23,11	0,3	20,24	0,3	23,39	0,27	19,68	0,25
y_2	28,86	0,39	20,45	0,31	24,03	0,31	19,75	0,28
y_3	35,31	0,41	20,81	0,34	24,83	0,39	19,97	0,28
y_4	39,12	0,42	20,86	0,38	25,05	0,39	19,99	0,3
y_5	42,36	0,42	20,9	0,38	25,61	0,39	20,06	0,31
y_6	43,01	0,45	20,97	0,38	25,69	0,41	20,07	0,31
y_7	51,62	0,45	21,28	0,39	25,98	0,42	20,07	0,33
y_8	62,7	0,47	21,38	0,39	26,05	0,42	20,12	0,34
y_9	24,68	0,48	21,68	0,39	26,41	0,42	20,18	0,36
y_{10}	27,38	0,49	22,06	0,41	27,55	0,44	20,23	0,38
y_{11}	34,04	0,52	22,19	0,41	27,66	0,44	20,3	0,38
y_{12}	34,94	0,52	22,25	0,41	28,05	0,44	20,41	0,38
y_{13}	35,33	0,53	23,47	0,41	29,87	0,45	20,45	0,39
y_{14}	36,57	0,53	20,55	0,42	31,07	0,47	20,52	0,39

Продолжение таблицы 4

Опыт	Результат							
	1	2	3	4	5	6	7	8
y_{15}	38,51	0,56	20,72	0,44	32,35	0,47	20,55	0,39
y_{16}	41,76	0,59	20,72	0,44	32,45	0,49	20,58	0,39
y_{17}	42,82	0,69	20,75	0,44	32,73	0,5	20,59	0,39
y_{18}	46,7	0,7	21	0,44	32,73	0,5	20,62	0,41
y_{19}	51,5	0,72	21,05	0,45	32,86	0,5	20,63	0,41
y_{20}	54,3	0,74	21,07	0,45	34,01	0,53	20,7	0,41
y_{21}	67,49	0,74	21,55	0,45	34,92	0,53	20,71	0,41
y_{22}	62,96	0,74	21,74	0,45	35,74	0,54	20,73	0,41
y_{23}	23,68	0,74	22,01	0,45	35,83	0,55	20,84	0,42
y_{24}	25,51	0,75	22,05	0,47	36,22	0,58	20,86	0,42
y_{25}	38,85	0,84	23,3	0,47	36,89	0,59	20,9	0,42
y_{26}	42,57	0,91	23,39	0,47	37,37	0,61	20,91	0,42
y_{27}	43,41	0,97	23,55	0,48	38,02	0,63	20,95	0,42
y_{28}	45,27	1,02	26,34	0,48	40,11	0,71	20,96	0,44
y_{29}	52,93	1,06	32,61	0,48	40,86	0,8	21,04	0,44
y_{30}	63,3	1,42	20,12	0,48	43,09	0,94	21,04	0,44
y_{31}	23,59	1,74	20,95	0,49	43,33	0,94	21,15	0,44
y_{32}	26,76	1,94	21,51	0,5	43,36	1,08	21,21	0,45
y_{33}	28,03	2	22,17	0,52	44,04	1,39	21,37	0,45
y_{34}	32,59	2,14	22,57	0,52	45,58	2	21,76	0,45
y_{35}	33,6	2,5	22,68	0,53	46,57	2,36	21,79	0,47
y_{36}	36,39	2,99	22,92	0,53	46,9	2,59	21,95	0,47
y_{37}	39,43	4,25	25,48	0,53	47,54	3,06	22,12	0,47
y_{38}	40,48	5,7	25,59	0,55	48,49	4,03	22,12	0,48
y_{39}	43,6	5,77	20,91	0,56	52,57	4,81	22,26	0,48
y_{40}	46,02	6,64	21,25	0,58	52,88	4,95	22,38	0,48
y_{41}	47,15	6,72	21,27	0,58	52,96	6,17	22,6	0,48
y_{42}	47,72	7,33	21,35	0,59	53,86	6,89	22,75	0,48
y_{43}	48,32	8,3	22,16	0,59	55,86	6,93	23,04	0,5
y_{44}	51,93	9,39	22,33	0,59	55,97	7,82	23,16	0,5

Окончание таблицы 4

Опыт	Результат							
	1	2	3	4	5	6	7	8
y_{45}	52,75	10,48	22,49	0,61	56,57	10,09	23,39	0,52
y_{46}	62,77	12,01	24,52	0,64	59,74	10,31	23,82	0,52
y_{47}	65,24	12,27	24,65	0,66	68,49	10,75	23,87	0,52
y_{48}	67,77	14,68	20,36	0,66	72,95	12,07	24,86	0,53
y_{49}	39,73	16,41	20,63	5,5	75,17	13,46	26,83	0,54
y_{50}	95,48	17,02	20,84	45,29	98,54	32,55	30,64	0,63

Эксперимент считался удачным, если все автономные агенты были доведены до пункта назначения. Пример успешной работы алгоритма проиллюстрирован на рисунке 3.

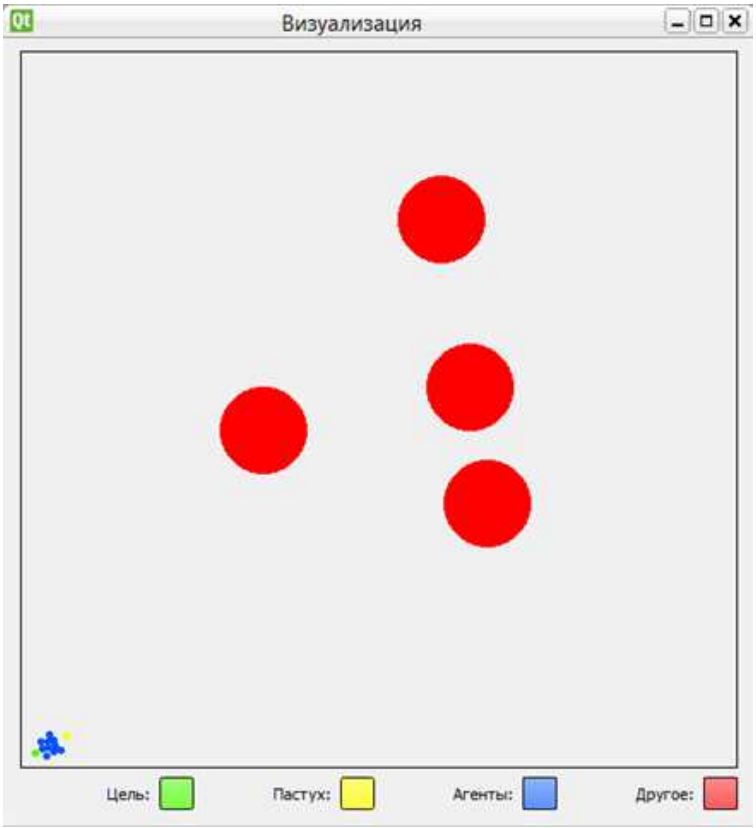


Рисунок 3 – Удачное выполнение алгоритма

Однако бывали ситуации, когда агенты разделялись. Если между ними оказывался другой объект, то возникала вероятность зависания алгоритма, так как пастух стремился вернуть «потерявшегося» агента и занимал позицию за ним. Такой вариант показан на рисунке 4.

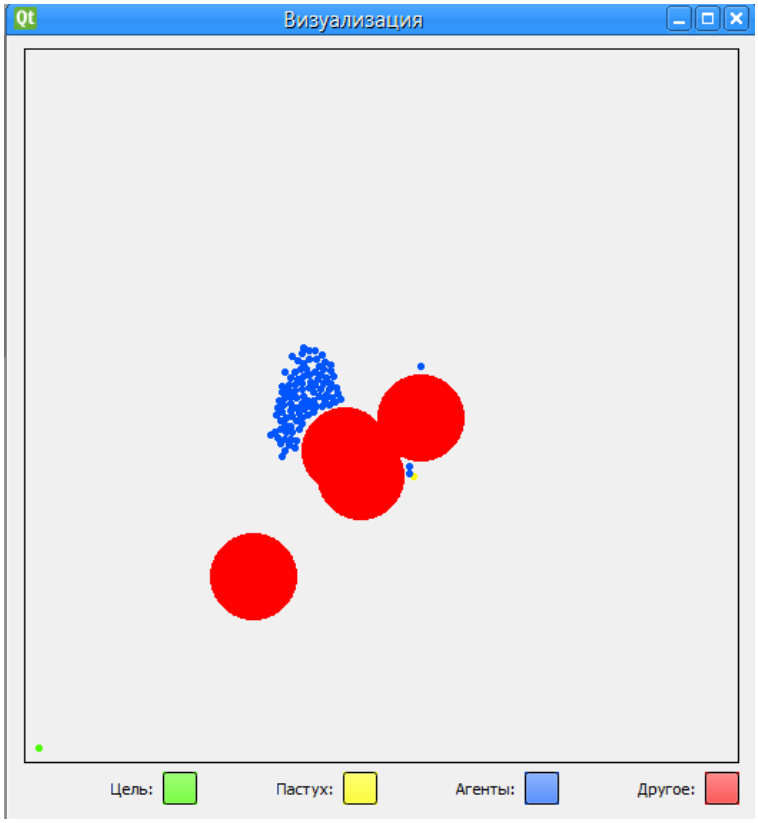


Рисунок 4 – Зависание алгоритма

Помимо нахождения зависимости времени он параметров модели, также должна быть найдена вероятность успешной работы алгоритма. Для этого была выполнена дополнительная серия экспериментов для различного количества объектов. С результатами можно ознакомиться в таблице 5.

Таблица 5 – Количество неудачных экспериментов в зависимости от количества агентов

Количество агентов	Количество ошибок	
	реальный пастух	виртуальный пастух
10	2	1
11	3	1
12	2	1
13	4	3
14	5	2
15	2	4
16	2	3
17	5	4
18	5	4
19	4	4
20	2	4
21	2	4
22	3	2
23	3	2
24	5	4
25	4	3
26	4	4
27	4	2
28	5	3
29	5	4
30	3	3
31	5	4
32	2	2
33	4	4
34	4	3
35	4	1
36	2	2
37	2	4
38	5	3
39	2	4
40	2	2
41	4	1
42	5	3
43	4	4
44	3	2
45	3	2
46	3	1
47	4	3
48	5	3
49	3	3
50	5	1
51	4	3
52	3	4

Продолжение таблицы 5

Количество агентов	Количество ошибок	
	реальный пастух	виртуальный пастух
53	4	1
54	3	1
55	3	4
56	2	2
57	5	4
58	3	3
59	3	3
60	5	3
61	4	3
62	2	4
63	3	4
64	4	4
65	2	4
66	3	3
67	2	3
68	5	4
69	3	1
70	5	2
71	3	4
72	2	1
73	2	1
74	3	4
75	2	2
76	5	2
77	2	1
78	3	1
79	4	3
80	5	3
81	3	1
82	4	1
83	4	3
84	2	1
85	5	1
86	4	4
87	3	4
88	2	2
89	4	1
90	3	1
91	5	1
92	2	2
93	4	1
94	3	3
95	4	4
96	2	1

Окончание таблицы 5

Количество агентов	Количество ошибок	
	реальный пастух	виртуальный пастух
97	2	1
98	5	4
99	3	2
100	5	4
101	2	3

3.1 Проведение вычислений и анализ результатов

После проведения серий экспериментов необходимо выполнить вычисления и проанализировать полученные результатов. В качестве результата эксперимента были взяты усредненные значения для каждого типа. Напомним, что x_1 – количество автономных объектов, x_2 – виртуальность пастуха и x_3 – количество преград.

В таблице 6 показана матрица планирования для обработки результатов эксперимента, где в последней колонке указаны усредненные значения времени выполнения алгоритма в зависимости от типа эксперимента.

Таблица 6 – Матрица планирования для обработки результатов

Номер эксперимента	x_1	x_2	x_3	x_1x_2	x_1x_3	x_2x_3	$x_1x_2x_3$	$\overline{y_i}$
1	+	+	+	+	+	+	+	43,8
2	-	+	+	-	-	+	-	3,4
3	+	-	+	-	+	-	-	22,15
4	-	-	+	+	-	-	+	1,47
5	+	+	-	+	-	-	-	41,38
6	-	+	-	-	+	-	+	3,19
7	+	-	-	-	-	+	+	21,55
8	-	-	-	+	+	+	-	0,42

Следующим шагом были вычислены коэффициенты уравнения регрессии по формулам 10:

$$\beta_0 = \frac{43{,}8+3{,}4+22{,}15+1{,}47+41{,}38+3{,}19+21{,}55+0{,}42}{8} = \frac{137{,}36}{8} = 17{,}17,$$
$$\beta_1 = \frac{43{,}8-3{,}4+22{,}15-1{,}47+41{,}38-3{,}19+21{,}55-0{,}42}{8} = \frac{120{,}4}{8} = 15{,}05,$$
$$\beta_2 = \frac{43{,}8+3{,}4-22{,}15-1{,}47+41{,}38+3{,}19-21{,}55-0{,}42}{8} = \frac{46{,}18}{8} \approx 5{,}77,$$
$$\beta_3 = \frac{43{,}8+3{,}4+22{,}15+1{,}47-41{,}38-3{,}19-21{,}55-0{,}42}{8} = \frac{4{,}28}{8} \approx 0{,}54,$$
$$\beta_{12} = \frac{43{,}8-3{,}4-22{,}15+1{,}47+41{,}38-3{,}19-21{,}55+0{,}42}{8} = \frac{36{,}78}{8} \approx 4{,}6,$$
$$\beta_{13} = \frac{43{,}8-3{,}4+22{,}15-1{,}47-41{,}38+3{,}19-21{,}55+0{,}42}{8} = \frac{1{,}76}{8} = 0{,}22,$$
$$\beta_{23} = \frac{43{,}8+3{,}4-22{,}15-1{,}47-41{,}3-3{,}19+21{,}55+0{,}42}{8} = \frac{0{,}98}{8} \approx 0{,}12,$$
$$\beta_{123} = \frac{43{,}8-3{,}4-22{,}15+1{,}47-41{,}38+3{,}19+21{,}55-0{,}42}{8} = \frac{2{,}66}{8} \approx 0{,}33.$$

После нахождения коэффициентов их проверяют на значимость. Для этого были определены выборочные дисперсии:

$$\sigma_1^2 = 204{,}32,$$
$$\sigma_2^2 = 21{,}04,$$
$$\sigma_3^2 = 4{,}37,$$
$$\sigma_4^2 = 40{,}49,$$
$$\sigma_5^2 = 237{,}35,$$
$$\sigma_6^2 = 30{,}58,$$
$$\sigma_7^2 = 3{,}75,$$
$$\sigma_8^2 = 0{,}01.$$

По формуле (13) была рассчитана дисперсия воспроизводимости:

$$\sigma^2_{\text{восп.}} = \frac{1}{8}\sum_{i=1}^{8}\sigma_i^2 = \frac{1}{8}(204{,}32 + 21{,}04 + 4{,}37 + 40{,}49 + 237{,}35 + 30{,}58 + 3{,}75 + 0{,}01) = \frac{1}{8}\cdot 541{,}91 \approx 67{,}74.$$

Таким образом среднее квадратическое отклонение коэффициентов составило:

$$\sigma_{\text{коэф.}} = \sqrt{\frac{\sigma^2_{\text{восп.}}}{8\cdot 50}} = \sqrt{\frac{67{,}74}{400}} \approx 0{,}41.$$

Из таблицы распределения Стьюдента по числу степеней свободы $8 \cdot 49 = 392$ при уровне значимости $\alpha = 0{,}05$, был найден $t_{\text{кр.}} = 1{,}96$. Следовательно,

$$t_{\text{кр.}} \cdot \sigma_{\text{коэф.}} = 1{,}96 \cdot 0{,}41 \approx 0{,}8.$$

При сравнении значения $t_{\text{кр.}} \cdot \sigma_{\text{коэф.}}$ с полученными коэффициентами было замечено, что β_3, β_{13}, β_{23} и β_{123} меньше по абсолютной величине, чем 0,8. Это означает, что данные коэффициенты не являются значимыми. Таким образом уравнение регрессии с безразмерными переменными принимает вид:

$$\hat{y} = 17{,}17 + 15{,}05 \cdot x_1 + 5{,}77 \cdot x_2 + 4{,}6 \cdot x_1 x_2. \quad (18)$$

Далее необходимо проверить полученное уравнение на адекватность по критерию Фишера.

По найденному уравнению были получены следующие результаты:

$$\widetilde{y_1} = 17{,}17 + 15{,}05 + 5{,}77 + 4{,}6 = 42{,}69,$$
$$\widetilde{y_2} = 17{,}17 - 15{,}05 + 5{,}77 - 4{,}6 = 3{,}19,$$
$$\widetilde{y_3} = 17{,}17 + 15{,}05 - 5{,}77 - 4{,}6 = 21{,}95,$$

$$\widetilde{y_4} = 17{,}17 - 15{,}05 - 5{,}77 + 4{,}6 = 0{,}85,$$
$$\widetilde{y_5} = 17{,}17 + 15{,}05 + 5{,}77 + 4{,}6 = 42{,}69,$$
$$\widetilde{y_6} = 17{,}17 - 15{,}05 + 5{,}77 - 4{,}6 = 3{,}19,$$
$$\widetilde{y_7} = 17{,}17 + 15{,}05 - 5{,}77 - 4{,}6 = 21{,}95,$$
$$\widetilde{y_8} = 17{,}17 - 15{,}05 - 5{,}77 + 4{,}6 = 0{,}85.$$

Таким образом была найдена остаточная дисперсия:

$$\sigma^2_{\text{ост.}} = \frac{50}{8-4}((42{,}69 - 43{,}8)^2 + (3{,}19 - 3{,}4)^2 + (21{,}95 - 22{,}15)^2 + (0{,}85 - 1{,}47)^2 + (42{,}69 - 41{,}38)^2 + (3{,}19 - 3{,}19)^2 + (21{,}95 - 21{,}55)^2 + (0{,}85 - 0{,}42)^2) = \frac{50}{4}(1{,}23 + 0{,}04 + 0{,}04 + 0{,}38 + 1{,}72 + 0 + 0{,}16 + 0{,}18) \approx 46{,}88.$$

Расчетное значение критерия Фишера определили по формуле:

$$F_{\text{расч.}} = \frac{\sigma^2_{\text{ост.}}}{\sigma^2_{\text{восп.}}} = \frac{46{,}88}{67{,}74} \approx 0{,}69.$$

Табличное значение критерия Фишера было взято из таблицы критических точек распределения Фишера при уровне значимости $\alpha = 0{,}05$ по соответствующим степеням свободы $k_1 = 8 - 4 = 4$ и $k_2 = 8 \cdot 49 = 392$:

$$F_{\text{табл.}} = 2{,}4.$$

Так как $F_{\text{расч.}} = 0{,}69 < F_{\text{табл.}} = 2{,}4$, то полученное уравнение регрессии адекватно.

Последним этапом было получение уравнения регрессии в натуральных переменных, использую формулу (18):

$$\hat{y} = 17{,}17 + 15{,}05 \cdot \frac{u_1 - 55}{45} + 5{,}77 \cdot \frac{u_2 - 0{,}5}{0{,}5} + 4{,}6 \cdot \frac{u_1 - 55}{45} \cdot \frac{u_2 - 0{,}5}{0{,}5}.$$

Выполнив преобразование, мы получили следующее уравнение, описывающее зависимость времени работы алгоритма пастушьей собаки от количества автономных агентов и типа пастуха:

$$\hat{y} = 0{,}23 \cdot u_1 + 0{,}32 \cdot u_2 + 0{,}2 \cdot u_1 u_2 - 1{,}9. \quad (19)$$

Анализируя полученные результаты, мы можем отметить, что наибольшее влияние на время выполнения алгоритма оказывает количество автономных объектов. Вторым фактором по значимости является тип управляющего объекта. На рисунке 5 с помощью формулы (19) построен график зависимости времени выполнения алгоритма от количества автономных объектов, где видно, что при использовании виртуального пастуха время выполнения алгоритма сокращается примерно в 2 раза.

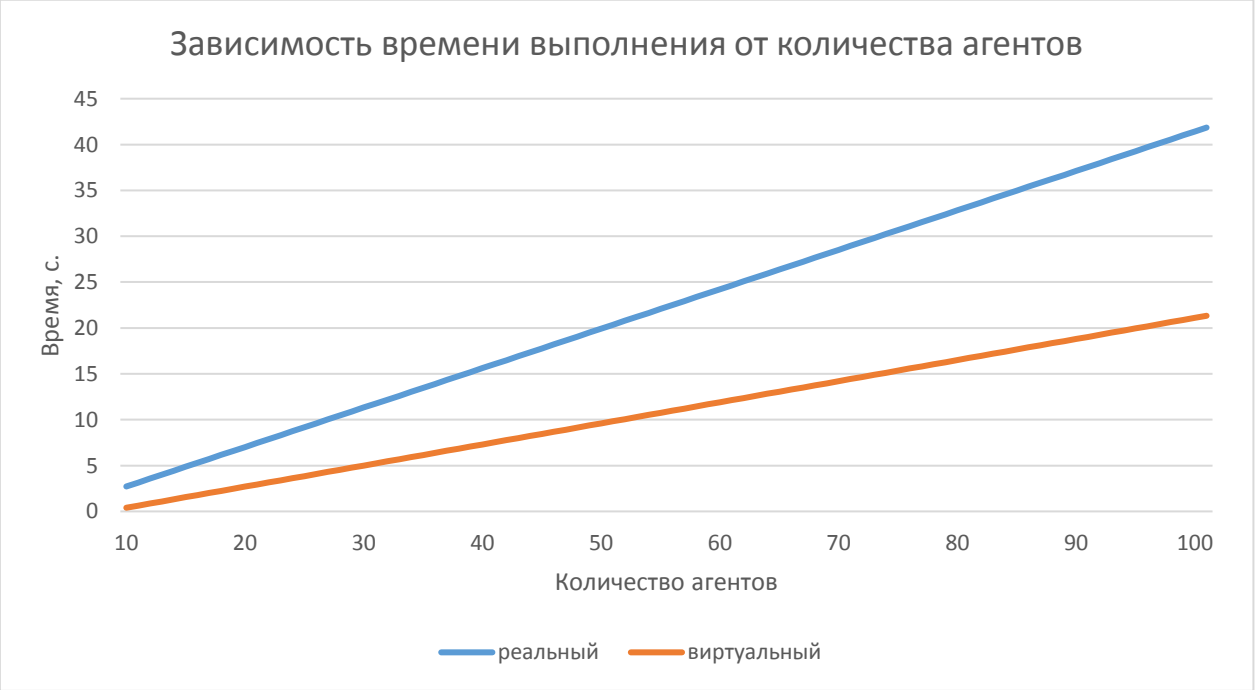


Рисунок 5 – Зависимость времени работы алгоритма от количества управляемых агентов

На рисунке 6 представлена диаграмма, где показано среднее время выполнения алгоритма. Для реального пастуха оно составляет 22,29 секунды, а для виртуального – 10,87 секунд.

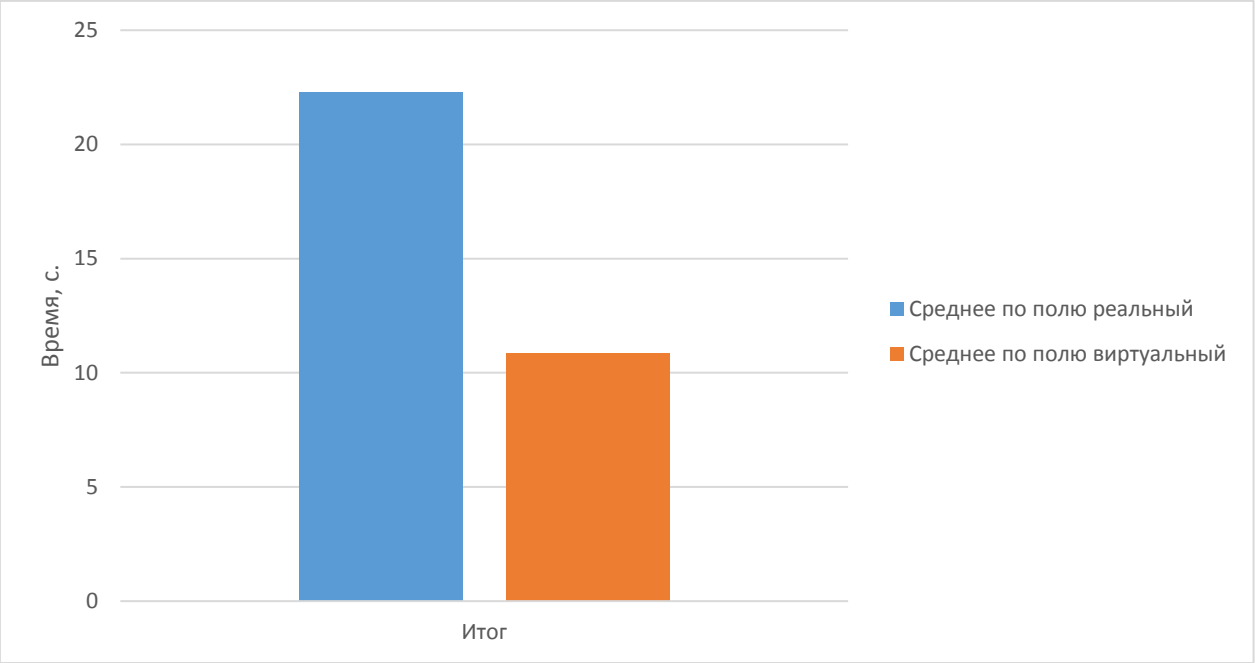


Рисунок 6 – Среднее время работы алгоритма

Минимальное время работы алгоритма в зависимости от типа пастуха показано на рисунке 7. Для реального пастуха оно составляет 2,72 секунды, а для виртуального – 0,4 секунды.

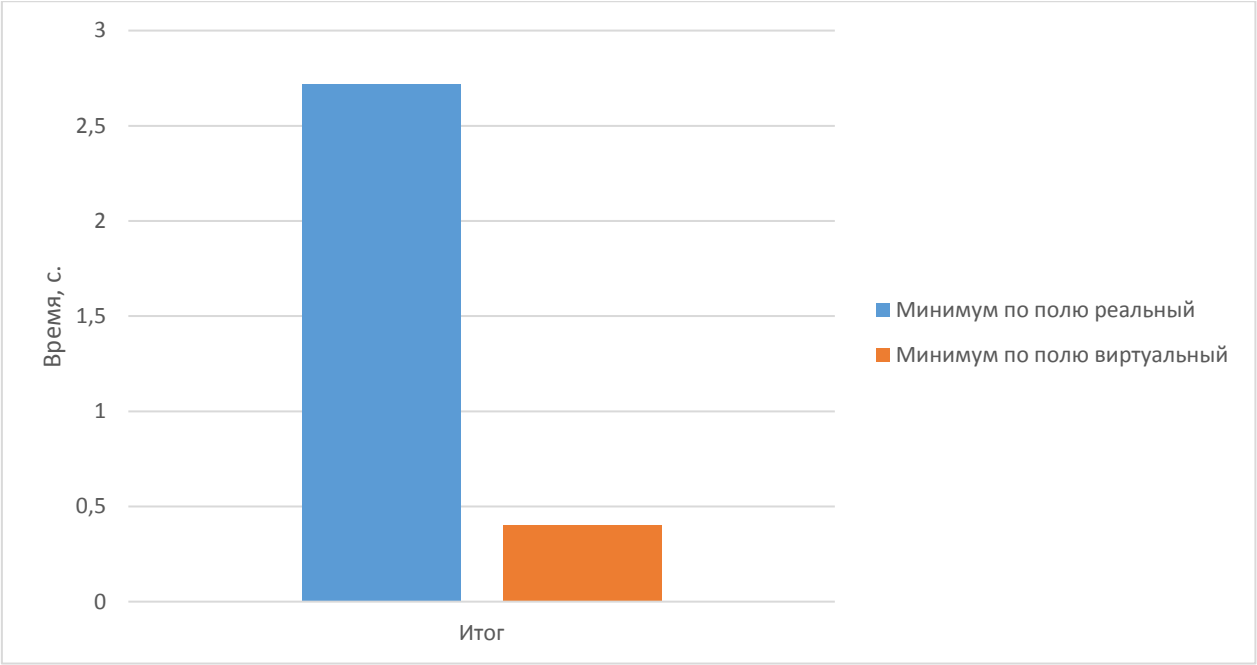


Рисунок 7 – Минимальное время работы алгоритма в зависимости от типа пастуха

Максимальное время работы алгоритма можно увидеть на рисунке 8. Для реального пастуха оно составляет 41,85 секунды, а для виртуального – 21,33 секунды.

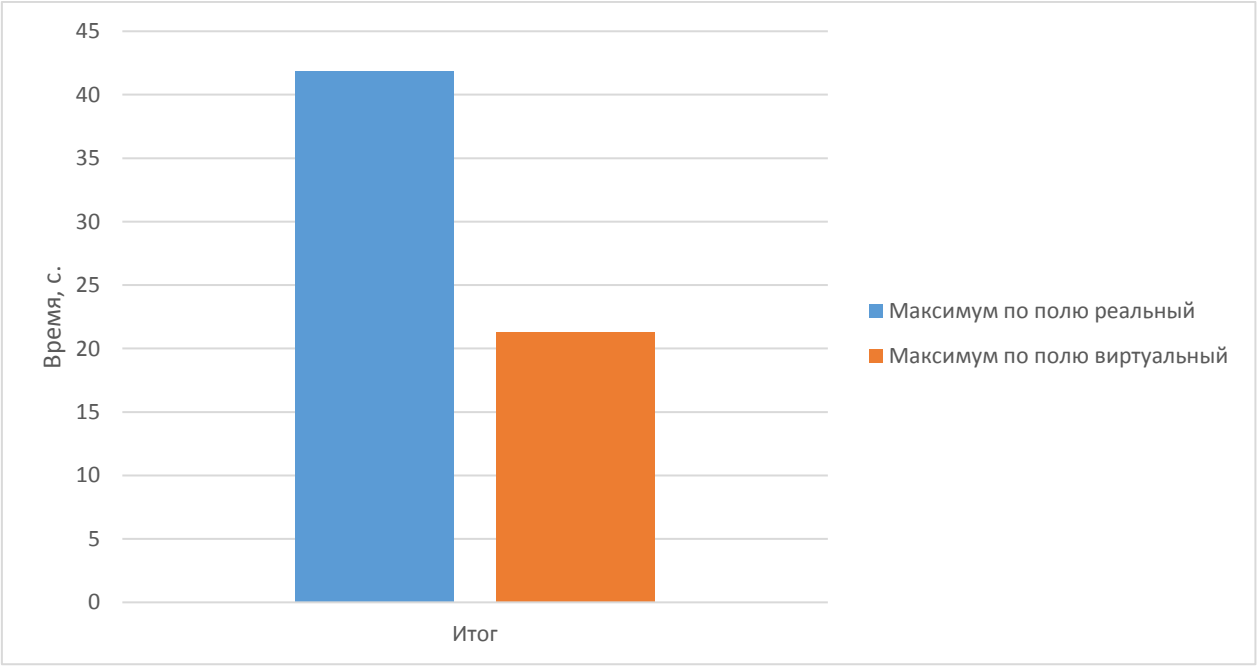


Рисунок 8 – Максимальное время работы алгоритма в зависимости от типа пастуха

В ходе построения уравнения зависимости времени выполнения алгоритма от различных факторов было замечено, что наличие или отсутствие преград на пути к цели не оказывает существенного влияния. Таким образом алгоритм пастушьей собаки эффективен при преодолении дистанции с иными объектами на маршруте.

Как отмечалось ранее, иногда возникает зависание алгоритма, когда на рабочем поле присутствуют иные объекты. Во время проведения эксперимента было замечено, что на каждые 50 опытов приходиться около четырех таких ошибок в независимости от количества агентов. С результатами можно ознакомиться на рисунке 9, где показан график вероятности успешной работы алгоритма.

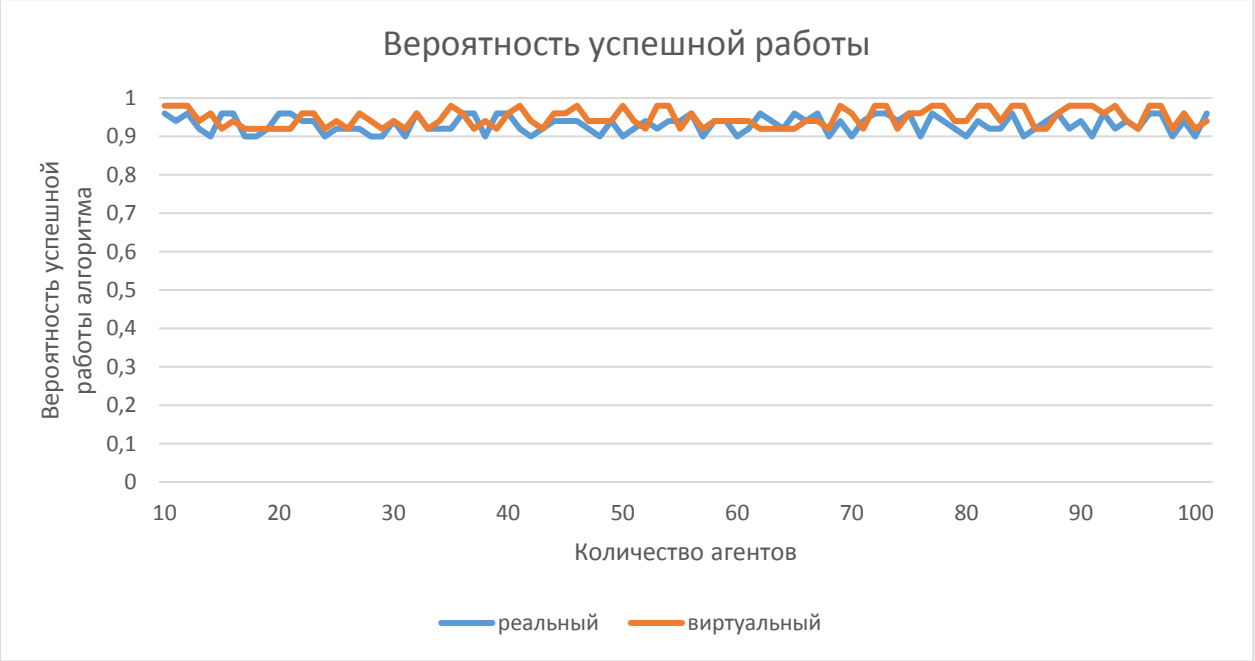


Рисунок 9 – Вероятность успешной работы алгоритма в зависимости от количества агентов

Все же количество ошибок не превышает 12% от общего количества проведенных экспериментов, поэтому можно говорить о достаточной надежности алгоритма.

Если говорить об общей эффективности алгоритма, то стоит его сравнить с аналогами. На самом деле алгоритмов, позволяющих вести группу к одной конечной цели не так много. В основном они направлены на выбор траектории движения отдельного агента, предотвращая столкновения с другими объектами. В качестве примера можно привести алгоритм Steering Behavior, позволяющий проложить путь в обход препятствиям. Однако использование данного алгоритма относительно АПС будет наименее эффективно, так как в этом случае каждый агент будет «сам за себя», что увеличит нагрузку на алгоритм. Также будет велика вероятность нарушения целостности группы, что в некоторых случаях может быть недопустимым. Более близким к АПС является метод роевого интеллекта для управления группой агентов. Отличительной особенностью данного метода является разделение на «ведущих» и «ведомых» роботов. Агенты, получившие информацию о направлении движения, становятся «ведущими», остальные – «ведомыми». Таким образом, когда «ведущие» начинают двигаться к цели, то ведомые начинают следовать за ними. Для этого используются силы отталкивания и притяжения к агентам подобно тому, как это было в АПС. Однако данный алгоритм становится невозможно применять по отношению к живым объектам. К тому же возрастает вычислительная нагрузка на самих агентов, так как в любой момент «ведущие» и «ведомые» могут поменяться. Среди рассмотренных алгоритмов АПС является самым универсальным решением, так как позволяет решать самые разнообразные задачи по управлению автономными объектами.

На основе полученных результатов можно сказать, что алгоритм работы пастушьей собаки является эффективным и универсальным для управления группой автономных объектов, как с использование реального управляющего, так и с виртуальным пастухом, в независимости от наличия преград на пути к цели.

3.2 Алгоритм работы пастушьей собаки применительно к живым и искусственным агентам

Ранее говорилось, что исследуемый алгоритм разрабатывался на основе работы пастушьей собаки. Прямым его назначением может служить роботизация выпаса овец или других животных, имеющих склонность собираться в группы, стаи, стада и т.п. К живым объектам возможно применить только алгоритм с реальным пастухом. В таком случае можно использовать одного или несколько роботов, на которых и будут реагировать объекты. Также рассматриваемый алгоритм возможно попробовать применить для управления толпой людей в чрезвычайных ситуациях. Для этого также можно использовать мобильных роботов или иные средства, не позволяющих людям покинуть безопасную зону.

Если говорить об искусственных объектах, то для них можно использовать алгоритм как с реальным пастухом, так и виртуальным. Такими объектами могут являться роботы, которых необходимо довести с базы до места их работы и наоборот после завершения каких-либо работ собрать их в одну группу и вернуть на базу. Для выполнения такого рода задач эффективнее использовать виртуального пастуха, при необходимости посылая сигнал об его положении агентам.

При управлении группой роботов для каждого из них может быть задан свой алгоритм передвижения в зависимости от типа робота и вида местности. В таком случае алгоритм работы пастушьей собаки будет дополнять основной алгоритм передвижения конкретного робота. Виртуальный пастух будет получать координаты всех участвующих агентов и рассчитывать глобальный центр масс группы. Если какой-либо участник будет находиться дальше допустимого расстояния от центра масс, то пастух будет посылать ему сигнал с координатами центра масс для движения. Для самого передвижения в нужную точку будет выполняться конкретный алгоритм данного агента. Когда

группа будет собрана, пастух может передавать сигнал всей группе для движения к пункту назначения.

Таким образом рассматриваемый алгоритм можно использовать для одновременного управления разными типами искусственных объектов.

Поскольку работу пастуха, как реального, так и виртуального, разделяется на сбор и вождение группы, то возможно использование только одного из режимов работы, если того требует задача.

При постоянном движении нескольких объектов может потребоваться, чтобы отдельные объекты не выбивались из группы. Для этого можно применять работу сбора алгоритма пастушьей собаки при отдалении объекта. В качестве примера можно привести сплав древесины по течению реки. Управлять направлением движения бревен в таком варианте не требуется, однако с помощью алгоритма сбора группы, можно выполнять поддержку во избежание их расплывания. Отслеживать положение бревен можно с помощью специальных датчиков, а собирать уплывающие единицы можно с использованием манипуляционного робота в качестве пастуха, плывущего впереди или сзади группы.

Из рассмотренных выше примеров становиться ясно, что алгоритм работы пастушьей собаки помимо своего основного назначения можно полностью или частично применять в самых разнообразных областях.

3.3 Поиск применимости АПС в новых областях на основе таблицы разрешения технических противоречий

Для поиска применимости алгоритма пастушьей собаки в новых областях можно воспользоваться таблицей разрешения технических противоречий, предложенной Г. С. Альтшуллером. Сама таблица, а также приемы устранения технических противоречий приведены в приложении В.

Если речь идет об искусственных объектах, то алгоритм можно применять для управления группой исследовательских роботов. Таких агентов возможно отправлять для изучения труднодоступной местности. Ручное управление группой роботов в таких ситуациях становиться довольно трудозатратным. Для этого нужна высокая степень автоматизации работы всей системы. В таком случае может ухудшиться точность измерений, так как группа может периодически покидать необходимую область, а также снизится производительность системы, так как придется выполнять большое количество вычислений для принятия решений передвижения в ту или иную сторону. Первую проблему согласно таблице разрешения технических противоречий можно решить, используя принцип «посредника». Можно ввести дополнительный легкоуправляемый объект, вокруг которого и будут передвигаться остальные агента. Решение второго технического противоречия возможно путем разбиения большой группы на несколько небольших подгрупп со своим виртуальным пастухом. Таким образом можно задать для каждой подгруппы изучение своей отдельной подобласти, увеличивая эффективность проводимых работ и снижая нагрузку на виртуальных пастухов.

При появлении нескольких групп возникает идея использования алгоритма пастушьей собаки относительно групп объектов, а не одиночных агентов. В этом случае в каждой группе имеется свой виртуальный пастух, выполняющий только роль поддержания целостности группы, а также имеется главный пастух, для которого управляемым объектом будет являться одна группа. Этот пастух будет собирать группы вместе и вести их к заданной точке. При этом возможно сохранять некоторую дистанцию между группами подобно тому, как сохраняли бы между собой дистанцию одиночные объекты. Таким образом можно было бы повысить эффективность работы АПС в отдельных группах за счет уменьшения количества агентов в них.

ЗАКЛЮЧЕНИЕ

Целью данной выпускной квалификационной работы было оценивание эффективности алгоритма пастушьей собаки, а также поиск его применимости в различных областях. Для достижения поставленной цели был выполнен компьютерный эксперимент. Однако прежде были изучены теоретические основы построения компьютерного эксперимента, алгоритм пастушьей собаки, выбраны средства разработки, позволяющие эффективно реализовать алгоритм, в также было выполнено планирование компьютерного эксперимента. Сам алгоритм был дополнен возможностью использования виртуального пастуха, добавлением объектов-преград, реакцией агентов на такие объекты, а также способностью прокладывать маршрут в обход препятствующих объектов путем добавления промежуточных целей.

В общей сложности было выполнено 400 опытов. Благодаря проведенному анализу полученных результатов была установлена зависимость времени выполнения алгоритма от количества автономных объектов, а также типа пастуха. Также было замечено, что использование виртуального пастуха является более эффективным средством для управления группой искусственных объектов, так как в случаи использования виртуального пастуха исключаются случайные взаимодействия пастуха и агентов. Поскольку виртуальный пастух, в отличии от реального, не является материальным объектом, ему не нужно затрачивать время на передвижение до заданной позиции.

Если присутствуют объекты, препятствующие движению по направлению к заданной цели, вероятность успешного выполнения алгоритма составляет около 0,88, что является хорошим результатом.

Также в результате выполнения работы было выявлено, что алгоритм пастушьей собаки полностью или частично возможно применять в самых

разнообразных сферах от выпаса скота и контроля толпы людей до управления различными типами роботов и поддержки целостности группы объектов.

В итоге можно сделать вывод, что алгоритм пастушьей собаки является эффективным и универсальным средством для управления группой живых или искусственных объектов.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Удельный вес организаций, использовавших специальные программные средства, в общем числе обследованных организаций [Электронный ресурс] : по данным формы федерального статистического наблюдения № 3-информ "Сведения об использовании информационных и коммуникационных технологий и производстве вычислительной техники, программного обеспечения и оказании услуг в этих сферах // Федеральная служба государственной статистики. – Режим доступа: http://www.gks.ru.

2. Попов, Е. П. Основы робототехники: введение в специальность: учеб. для вузов по специальности «Робототехн. системы и комплексы» / Е. П. Попов, Г. В. Письменный. – Москва : Высшая школа, 1990. – 224 с.

3. Daniel Strombom, Richard P. Mann, Alan M. Wilson, Stephen Hailes, A. Jennifer Morton, David J. T. Sumpter, Andrew J. King. Solving the shepherding problem: heuristics for herding autonomous, interacting agents // J. R. Soc. Interface. – Nov. 2014 – Vol. 11, no. 100.

4. Корешкова, И. А. История математического моделирования и технологии вычислительного эксперимента: научная статья / И. А. Корешкова // Научный журнал «Научные исследования в образовании». – 2009. – №4. – С. 1–12.

5. Казакова, А. Н. Развитие теории, методологии и практики компьютерного эксперимента в социально-экономических исследованиях и задачах управления: научная статья / А. Н. Казакова // Международный научный журнал «Символ науки». – 2015. – №6. – С. 119–120.

6. Петров, П. В. Технология вычислительного эксперимента: научная статья / П. В. Петров, Р. А. Сунарчин, В. А. Целищев // Вестник Уфимского государственного авиационного технического университета. – 2008. – Т. 10, №1. – С. 30–35.

7. Большой энциклопедический словарь [Электронный ресурс] : бесплатная онлайн энциклопедия с полнотекстовым поиском и поддержкой морфологии русских слов. – Режим доступа: https://www.vedu.ru.

8. Сафонов, В. И. Компьютерное моделирование: учебное пособие / В. И. Сафонов; Мордов. гос. пед. ин-т. – Саранск, 2009. – 92 с.

9. Кольцов, П. П. О количественной оценке эффективности алгоритмов анализа изображений: научная статья / П. П. Кольцов, А. С. Осипов, А. С. Куцаев, А. А. Кравченко, Н. В. Котович, А. В. Захаров // Научный журнал «Компьютерная оптика». – 2015. – Т. 39, №4. – С. 542–556.

10. Виноградов, Г. П. Оценка эффективности метода кластеризации, использующего субъективные оценки: научная статья / Г. П. Виноградов, А. А. Мальков // Международный научный журнал «Программные продукты и системы». – 2009. – №2. – С. 137–141.

11. Оценка сложности алгоритмов [Электронный ресурс] : статья // Коллективный блог «Хабрахабр». – Режим доступа: https://habrahabr.ru.

12. Lien J, Bayazit OB, Sowell RT, Rodriguez S, Amato AM. Shepherding behaviors // In Proc. IEEE Int. Conf. on Robotics and Automation. – 2004. – pp. 4159–4164.

13. Bennett B, Trafankowski M. 2012 A Comparative investigation of herding algorithms. In Proc. Symp. on Understanding and Modelling Collective Phenomena // UMoCoP, Birmingham, UK. – 2–6 July 2012. – pp. 33–38.

14. Lien JM, Pratt E. 2009 Interactive planning for shepherd motion // In Proc. AAAI Spring Symp., Palo, Alto, CA. – 23–25 March 2009. – pp. 95–102. –Menlo Park, CA: AAAI Press.

15. Hamilton WD. 1971 Geometry for the selfish herd. J. Theor. Biol. 21, 295–311.

16. Любченко Е. А., Чуднова О. А. Планирование и организация эксперимента: учебное пособие. Часть 1. – Владивосток: Изд-во ТГЭУ, 2010. – 156 с.

17. Макаричев Ю. А. Методы планирование эксперимента и обработки данных: учеб. пособие / Макаричев Ю. А., Иванников Ю. Н. – Самара: Самар. гос. техн. ун-т, 2016. – 131 с.: ил.

18. Рубан, А. И. Методы анализа данных: учеб. пособие. 2-е изд., исправл. и доп. / А. И. Рубан. Красноярск: ИПЦ КГТУ. – 2004. – 319 с.

19. Кузнецова Е. В. Математическое планирование эксперимента: Учебно-методическое пособие для студентов очного и заочного обучения специальностей «Технология обработки металлов давлением», «Динамика и прочность машин», «Компьютерная механика», «Компьютерная биомеханика». – Пермь: Перм. гос. техн. ун-т, 2011. – 35 с.

20. MathWorks [Электронный ресурс] : официальный сайт программного продукта «MATLAB». – Режим доступа: http://www.mathworks.com.

21. Scilab [Электронный ресурс] : официальный сайт программного продукта «Scilab». – Режим доступа: http://www.scilab.org.

22. MapleSoft [Электронный ресурс] : официальный сайт программного продукта «MapleSim». – Режим доступа: http://www.maplesoft.com.

23. C++ [Электронный ресурс] : статья // Свободная энциклопедия «ВикипедиЯ». – Режим доступа: https://ru.wikipedia.org.

24. C Sharp [Электронный ресурс] : статья // Свободная энциклопедия «ВикипедиЯ». – Режим доступа: https://ru.wikipedia.org.

25. Java [Электронный ресурс] : статья // Свободная энциклопедия «ВикипедиЯ». – Режим доступа: https://ru.wikipedia.org.

26. Qt [Электронный ресурс] : статья // Свободная энциклопедия «ВикипедиЯ». – Режим доступа: https://ru.wikipedia.org.

27. Qt [Электронный ресурс] : официальный сайт программного продукта «Qt». – Режим доступа: https://info.qt.io.

28. ГК РФ Статья 1286.1. Открытая лицензия на использование произведения науки, литературы или искусства [Электронный ресурс] : Гражданский кодекс Российской Федерации часть 4 от 18.12.2006 N 230-ФЗ (ред. от 03.07.2016, с изм. от 13.12.2016) (с изм. и доп., вступ. в силу с

01.01.2017) // Справочная правовая система «КонсультантПлюс». – Режим доступа: http://www.consultant.ru.

29. Малышева, М. Ф. Об актуальности введения в российское законодательство свободных лицензий: научная статья / М. Ф. Малышева, И. А. Стрельникова // Научный журнал «Вестник университета». – 2013. – №19. – С. 169–173.

30. Солопова, Н. С. Некоторые проблемы открытых лицензий в авторском праве Российской Федерации: научная статья / Н. С. Солопова // Научный журнал «Правопорядок: история, теория, практика». – 2016. – Т. 8, №1. – С. 44–47.

31. Матвеев, А. Г. Создание правовых основ так называемых свободных лицензий в гражданском кодексе Российской Федерации // Вестник Пермского университета. Сер.: Юридические науки. 2014. Вып. 3(25). С. 125–135.

ПРИЛОЖЕНИЕ А

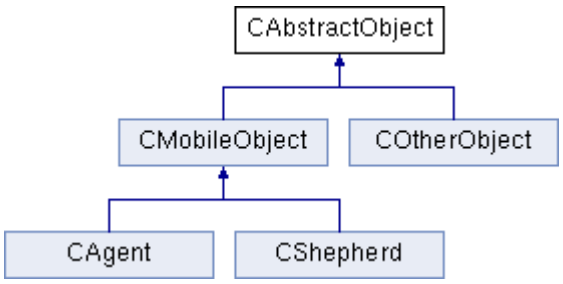
Техническая документация

Класс CAbstractObject

Родительский класс для всех объектов алгоритма

`#include <abstract_object.h>`

Граф наследования:CAbstractObject:



Открытые члены

void **set_position** (QPoint **position**)

Устанавливает позицию объекта

QPoint **get_position** ()

Возвращает текущую позицию объекта

void **set_vector_position** (QVector2D **position**)

Устанавливает позицию объекта

QVector2D **get_vector_position** ()

Возвращает текущую позицию объекта

void **set_size** (int **size**)

Устанавливает размер объекта

int **get_size** ()

Возвращает размер объекта

Защищенные данные

QVector2D **position**

Позиция объекта

int **size**

Радиус объекта

Подробное описание

Родительский класс для всех объектов алгоритма

Автор:

Карпова Е.А.

Версия:

1.0

Дата:

Май 2017 года

Данный класс содержит в себе свойства и методы, характерные для всех объектов алгоритма

Методы

QPoint CAbstractObject::get_position ()

Возвращает текущую позицию объекта

Возвращает:

Позицию объекта в виде точки

```
15 {
16     return QPoint(position.x(), position.y());
17 }
```

int CAbstractObject::get_size ()

Возвращает размер объекта

Возвращает:

Радиус объекта

```
38 {
39     return size;
40 }
```

QVector2D CAbstractObject::get_vector_position ()

Возвращает текущую позицию объекта

Возвращает:

Позицию объекта в виде вектора

```
25 {
26     return position;
27 }
```

void CAbstractObject::set_position (QPoint *position*)

Устанавливает позицию объекта

Аргументы:

position	Позиция в виде точки, которую необходимо установить объекту

Переводит переданную позицию в вектор и устанавливет объекту

```
9 {
10     this->position.setX(position.x());
11     this->position.setY(position.y());
12 }
```

void CAbstractObject::set_size (int *size*)

Устанавливает размер объекта

Аргументы:

size	Радиус объекта

```
30 {
31     if (size > 0)
32     {
33         this->size = size;
34     }
```

```
35 }
```

void CAbstractObject::set_vector_position (QVector2D *position*)

Устанавливает позицию объекта

Аргументы:

position	Позиция, которую необходимо установить объекту

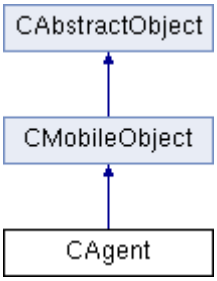
```
20 {
21     this->position = position;
22 }
```

Объявления и описания членов классов находятся в файлах:

C:/Projects/Shepherding_model/model/abstract_object.h
C:/Projects/Shepherding_model/model/abstract_object.cpp

Класс CAgent

Класс для автономных агентов
`#include <agent.h>`
Граф наследования:CAgent:



Открытые члены

void **set_direction** (QVector2D direction)
Устанавливает направление движения агента

QVector2D **get_direction** ()
Возвращает направление движения агента

Дополнительные унаследованные члены

Подробное описание

Класс для автономных агентов

Автор:
Карпова Е.А.

Версия:
1.0

Дата:
Май 2017 года

Является наследником класса **CMobileObject** с дополнительными свойствами и методами, характерными автономным агентам

Методы

QVector2D CAgent::get_direction ()

Возвращает направление движения агента

Возвращает:
Направление движения в виде вектора

```
14 {
15     return direction;
16 }
```

void CAgent::set_direction (QVector2D *direction*)

Устанавливает направление движения агента

Аргументы:

direction	Направление движения

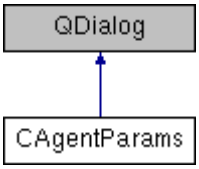
```
9 {
10     this->direction = direction;
11 }
```

Объявления и описания членов классов находятся в файлах:

C:/Projects/Shepherding_model/model/agent.h
C:/Projects/Shepherding_model/model/agent.cpp

Класс CAgentParams

Класс окна редактирования параметров агентов
`#include <agent_params.h>`
Граф наследования:CAgentParams:



Открытые слоты

void **take_agent_params** ()
Передает параметры из формы в алгоритм

Открытые члены

CAgentParams (QWidget *parent=0)

Подробное описание

Класс окна редактирования параметров агентов

Автор:
Карпова Е.А.

Версия:
1.0

Дата:
Май 2017 года

В окне данного класса задаются параметры алгоритма, касающиеся агентов

Объявления и описания членов классов находятся в файлах:
C:/Projects/Shepherding_model/agent_params.h
C:/Projects/Shepherding_model/agent_params.cpp

Класс CAlgorithm

Класс для объектов-преград
`#include <algorithm.h>`

Открытые члены

int **get_goal_radius** ()
Возвращает радиус цели

void **execution** ()
Запускает одну итерацию алгоритма

void **set_virtual_shepherd** (bool flag)
Устанавливает тип пастуха

void **set_gen_center** (QPoint gen_center)
Устанавливает центр генерации агентов

void **set_gen_size** (QPoint size)
Устанавливает размер поля генерации агентов

void **set_agent_quantity** (int quantity)
Устанавливает количество агентов

void **set_goal** (QPoint goal)
Устанавливает координаты цели

void **set_agent_nearest_quantity** (int quantity)
Устанавливает количество ближайших соседей для агента

void **set_agents_agent_reaction** (int radius)
Устанавливает радиус реакции агентов на других агентов

void **set_agents_shepherd_reaction** (int radius)
Устанавливает радиус реакции агентов на пастуха

void **set_agents_other_reaction** (int radius)
Устанавливает радиус реакции агентов на объекты-преграды

void **set_agents_agent_repulsion** (double strength)
Устанавливает силу отталкивания агентов друг от друга

void **set_agents_agent_attraction** (double strength)
Устанавливает силу притяжения агентов друг к другу

void **set_agents_shepherd_repulsion** (double strength)
Устанавливает силу отталкивания агентов от пастуха

void **set_agents_other_repulsion** (double strength)
Устанавливает силу отталкивания агентов от объектов-преград

void **set_agents_inertia** (double inertia)
Устанавливает коэффициент инерции для агентов

void **set_agents_noise** (double noise)
Учтанавливает коэффициент шума для направления агентов

void **set_agents_movement_probability** (double probability)
Устанавливает вероятность случайных передвижений агентов

void **set_agent_speed** (double speed)
Устанавливает скорость передвижения агентов

void **set_shepherd_speed** (double speed)
Устанавливает скорость предвижения пастуха

void **set_shepherd_noise** (double noise)
Устанавливает коэффициент шума для направления пастуха

void **add_object** (QString name, QPoint position, int radius)
Добавляет новый объект

void **change_object** (int index, QString name, QPoint position, int radius)

Меняет параметры имеющегося объекта

void **delete_object** (int index)

Удаляет имеющийся на поле объекта

void **clear_objects** ()

Удаляет все имеющиеся на поле объекты-преграды

bool **gen_agents** ()

Генерирует агентов в пределах заданного ранее поля генерации

bool **gen_shepherd** (QPoint position)

Генерация пастуха

bool **test_goal** ()

Проверяет положение группы относительно цели

QList< QPoint > **get_agents_positions** ()

Возвращает список позиций агентов

QList< **COtherObject** * > **get_objects** ()

Возвращает объекты-преграды

COtherObject * **get_object** (int index)

Возвращает указатель на определенный объект

QPoint **get_shepherd_position** ()

Возвращает позицию пастуха

QPoint **get_goal** ()

Возвращает координаты цели

QPoint **get_field_size** ()

Возвращает размеры поля

QVector2D **get_goal_position** ()

Возвращает координаты цели в виде вектора

Статические открытые данные

static **CAlgorithm algorithm**

алгоритм

Подробное описание

Класс для объектов-преград

Автор:

Карпова Е.А.

Версия:

1.0

Дата:

Май 2017 года

Данный класс отвечает за работу алгоритма пастушьей собаки

Методы

void CAlgorithm::add_object (QString *name*, QPoint *position*, int *radius*)

Добавляет новый объект

Аргументы:

name	Имя объекта

position	Позиция объекта
radius	Радиус объекта

```
249 {
250     COtherObject* object = new COtherObject;
251
252     object->set_name(name);
253     object->set position(position);
254     object->set size(radius);
255
256     objects.push_back(object);
257 }
```

void CAlgorithm::change_object (int *index*, QString *name*, QPoint *position*, int *radius*)

Меняет параметры имеющегося объекта

Аргументы:

index	Номер объекта
name	Новое имя объекта
position	Новая позиция объекта
radius	Новый радиус объекта

```
260 {
261     if (index >= 0 && index < objects.length())
262     {
263         objects.at(index)->set name(name);
264         objects.at(index)->set_position(position);
265         objects.at(index)->set_size(radius);
266     }
267 }
```

void CAlgorithm::delete_object (int *index*)

Удаляет имеющийся на поле объекта

Аргументы:

index	Номер объекта

```
270 {
271     if (index >= 0 && index < objects.length())
272     {
273         delete objects.takeAt(index);
274     }
275 }
```

bool CAlgorithm::gen_agents ()

Генерирует агентов в пределах заданного ранее поля генерации

Возвращает:

Статус при завершении генерации

Если генерация была выполнена успешно, то возвращается True, иначе - False

```
278 {
279     if (!test gen field())
280     {
281         return false;
282     }
283
284     clear agents();
285
286     for (int i = 0; i < agent quantity; i++)
287     {
288         CAgent* agent = new CAgent;
289         agent->set position(gen position());
290         agents.push back(agent);
291     }
292
```

```
293     if (!agents.isEmpty())
294     {
295         nearest agent = agents.first();
296 
297         test_object_position();
298 
299         return true;
300     }
301 
302     return false;
303 }
```

bool CAlgorithm::gen_shepherd (QPoint *position*)

Генерация пастуха

Аргументы:

position	Начальная позиция пастуха

Возвращает:

Статус при завершении генерации

Если генерация была выполнена успешно, то возвращается True, иначе - False

```
306 {
307     shepherd.set position(position);
308 
309     return true;
310 }
```

QList< QPoint > CAlgorithm::get_agents_positions ()

Возвращает список позиций агентов

Возвращает:

Список позиций агентов

```
335 {
336     QList<QPoint> positions;
337 
338     QList<CAgent*>::Iterator i;
339 
340     for (i = agents.begin(); i != agents.end(); i++)
341     {
342         positions.push_back((*i)->get_position());
343     }
344 
345     return positions;
346 }
```

QPoint CAlgorithm::get_field_size ()

Возвращает размеры поля

Возвращает:

Точка, содержащая размеры поля

```
374 {
375     return field size;
376 }
```

QPoint CAlgorithm::get_goal ()

Возвращает координаты цели

Возвращает:

Координаты цели

```
369 {
370     return QPoint(subgoal.x(), subgoal.y());
371 }
```

QVector2D CAlgorithm::get_goal_position ()

Возвращает координаты цели в виде вектора

Возвращает:

Вектор цели

```
379 {
380     return goal;
381 }
```

int CAlgorithm::get_goal_radius ()

Возвращает радиус цели

Возвращает:

Радиус цели

```
52 {
53     return goal_radius;
54 }
```

COtherObject * CAlgorithm::get_object (int *index*)

Возвращает указатель на определенный объект

Аргументы:

index	Номер объекта

Возвращает:

Указатель на объект

```
354 {
355     if (index >= 0 && index < objects.length())
356     {
357         return objects.at(index);
358     }
359 
360     return new COtherObject;
361 }
```

QList< COtherObject * > CAlgorithm::get_objects ()

Возвращает объекты-преграды

Возвращает:

Список указателей на все имеющиеся объекты-преграды

```
349 {
350     return objects;
351 }
```

QPoint CAlgorithm::get_shepherd_position ()

Возвращает позицию пастуха

Возвращает:

Координаты пастуха

```
364 {
365     return shepherd.get_position();
```

```
366 }
```

void CAlgorithm::set_agent_nearest_quantity (int *quantity*)

Устанавливает количество ближайших соседей для агента

Аргументы:

quantity	Количество соседей

```
177 {
178     agent nearest quantity = quantity;
179 }
```

void CAlgorithm::set_agent_quantity (int *quantity*)

Устанавливает количество агентов

Аргументы:

quantity	Количество агентов

```
145 {
146     agent_quantity = quantity;
147 
148     agent_nearest_quantity = --quantity;
149     goal_radius = agents_agent_reaction * sqrt(agent_quantity);
150 }
```

void CAlgorithm::set_agent_speed (double *speed*)

Устанавливает скорость передвижения агентов

Аргументы:

speed	Скорость

```
234 {
235     agent speed = speed;
236 }
```

void CAlgorithm::set_agents_agent_attraction (double *strength*)

Устанавливает силу притяжения агентов друг к другу

Аргументы:

strength	Сила притяжения

```
204 {
205     agents_agent_attraction = strength;
206 }
```

void CAlgorithm::set_agents_agent_reaction (int *radius*)

Устанавливает радиус реакции агентов на других агентов

Аргументы:

radius	Радиус реакции на агентов

```
182 {
183     agents agent reaction = radius + agent size;
184 
185     goal radius = agents agent reaction * sqrt(agent quantity);
186 }
```

void CAlgorithm::set_agents_agent_repulsion (double *strength*)

Устанавливает силу отталкивания агентов друг от друга

Аргументы:

strength	Сила отталкивания

```
199 {
200     agents agent repulsion = strength;
201 }
```

void CAlgorithm::set_agents_inertia (double *inertia*)

Устанавливает коэффициент инерции для агентов

Аргументы:

inertia	Инерция

Устанавливаемый коэффициент влияет на направление агента по инерции

```
219 {
220     agents inertia = inertia;
221 }
```

void CAlgorithm::set_agents_movement_probability (double *probability*)

Устанавливает вероятность случайных передвижений агентов

Аргументы:

probability	Вероятность

```
229 {
230     agents movement probability = probability;
231 }
```

void CAlgorithm::set_agents_noise (double *noise*)

Учтанавливает коэффициент шума для направления агентов

Аргументы:

noise	Шум

```
224 {
225     agents_noise = noise;
226 }
```

void CAlgorithm::set_agents_other_reaction (int *radius*)

Устанавливает радиус реакции агентов на объекты-преграды

Аргументы:

radius	Радиус реакции на преграды

```
194 {
195     agents other reaction = radius;
196 }
```

void CAlgorithm::set_agents_other_repulsion (double *strength*)

Устанавливает силу отталкивания агентов от объектов-преград

Аргументы:

strength	Сила отталкивания

```
214 {
215     agents other repulsion = strength;
216 }
```

void CAlgorithm::set_agents_shepherd_reaction (int *radius*)

Устанавливает радиус реакции агентов на пастуха

Аргументы:

radius	Радиус реакции на пастуха

```
189 {
190     agents shepherd reaction = radius + shepherd.get size();
191 }
```

void CAlgorithm::set_agents_shepherd_repulsion (double *strength*)

Устанавливает силу отталкивания агентов от пастуха

Аргументы:

strength	Сила отталкивания

```
209 {
210     agents_shepherd_repulsion = strength;
211 }
```

void CAlgorithm::set_gen_center (QPoint *gen_center*)

Устанавливает центр генерации агентов

Аргументы:

gen_center	Точка-центр генерации агентов

```
135 {
136     this->gen center = gen center;
137 }
```

void CAlgorithm::set_gen_size (QPoint *size*)

Устанавливает размер поля генерации агентов

Аргументы:

size	Точка содержит в себе длину и ширину поля генерации агентов

```
140 {
141     this->gen_size = size;
142 }
```

void CAlgorithm::set_goal (QPoint *goal*)

Устанавливает координаты цели

Аргументы:

goal	Точка, содержащая в себе координаты цели

```
169 {
170     this->goal.setX(goal.x());
171     this->goal.setY(goal.y());
172
173     subgoal = this->goal;
174 }
```

void CAlgorithm::set_shepherd_noise (double *noise*)

Устанавливает коэффициент шума для направления пастуха

Аргументы:

noise	Шум

```
244 {
245     shepherd noise = noise;
246 }
```

void CAlgorithm::set_shepherd_speed (double *speed*)

Устанавливает скорость предвижения пастуха

Аргументы:

speed	Скорость

```
239 {
240     shepherd speed = speed;
241 }
```

void CAlgorithm::set_virtual_shepherd (bool *flag*)

Устанавливает тип пастуха

Аргументы:

flag	Тип пастуха

Если flag равен True, то в алгоритме используется виртуальный пастух, иначе - реальный

```
130 {
131     virtual shepherd = flag;
132 }
```

bool CAlgorithm::test_goal ()

Проверяет положение группы относительно цели

Возвращает:

Результат проверки

Если группа находится в пределах радиуса цели, то возвращается True, иначе - False

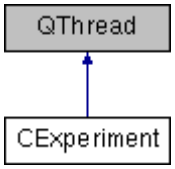
```
313 {
314     QVector2D global_mass_center = get_mass_center(agents);
315 
316     QVector2D distance = global mass center - subgoal;
317 
318     if (distance.length() <= goal_radius)
319     {
320         if (goal == subgoal)
321         {
322             return true;
323         }
324         else
325         {
326             subgoal = goal;
327             test object position();
328         }
329     }
330 
331     return false;
332 }
```

Объявления и описания членов классов находятся в файлах:

C:/Projects/Shepherding_model/model/algorithm.h
C:/Projects/Shepherding_model/model/algorithm.cpp

Класс CExperiment

Класс для проведения экспериментальных вычислений
`#include <experiment.h>`
Граф наследования:CExperiment:



Сигналы

void **finished** ()
Сигнал о завершении эксперимента

Открытые члены

CExperiment ()
void **run** ()
Выполняет эксперимент

Подробное описание

Класс для проведения экспериментальных вычислений

Автор:
Карпова Е.А.

Версия:
1.0

Дата:
Май 2017 года

Конструктор(ы)

CExperiment::CExperiment ()

Устанавливаем начальные значения свойств класса

Разрешаем принудительную остановку потока

```
16 {
17     QTime midnight(0,0,0);
18     qsrand(midnight.secsTo(QTime::currentTime()));
19 
21     agent_quantity = 101;
22     quantity = 0;
23     fail_experiment = 0;
24     middle_time = 100000;
25 
27     setTerminationEnabled(true);
28 }
```

Методы

void CExperiment::run ()

Выполняет эксперимент

Указываем файл для резервного копирования результатов эксперимента
Генерируем объекты-преграды
Запускаем таймер
Начинаем выполнение алгоритма
Если превышено время выполнения алгоритма, то считаем опыт неудачным
Если алгоритм успешно выполнился, то сохраняем время его выполнения

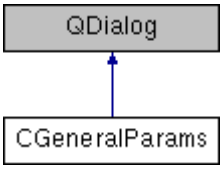
```
31 {
32     QTime timer;
33 
35     QFile fileOut("result.txt");
36 
38     random_objects();
39 
41     timer.start();
42 
44     while(!CAlgorithm::algorithm.test_goal())
45     {
46         CAlgorithm::algorithm.execution();
47 
49         if (timer.elapsed() > middle time)
50         {
51             QMessageBox::information(0, "Information", "Fail!");
52             save_data();
53 
54             fail experiment++;
55 
56             emit finished();
57 
58             return;
59         }
60     }
61 
63     int ms = timer.elapsed();
64 
65     time.push_back(ms);
66     qSort(time.begin(), time.end());
67     quantity++;
68     save data();
69 
70     if(fileOut.open(QIODevice::WriteOnly | QIODevice::Text))
71     {
72         QTextStream writeStream(&fileOut);
73 
74         writeStream << QString("%1 %2").arg(quantity).arg(fail_experiment);
75 
76         fileOut.close();
77 
78     }
79 
80 
81     if (quantity < QUANTITY)
82     {
83         emit finished();
84     }
85     else
86     {
87         QMessageBox::information(0, "Information", "Operation Complete");
88         quantity = 0;
89         fails.push back(fail experiment);
90 
91         time.clear();
92     }
93 }
```

Объявления и описания членов классов находятся в файлах:

C:/Projects/Shepherding_model/experiment.h
C:/Projects/Shepherding_model/experiment.cpp

Класс CGeneralParams

Класс окна редактирования основных параметров алгоритма
`#include <general_params.h>`
Граф наследования:CGeneralParams:



Открытые слоты

void **take_general_params** ()
Передает основные параметры в алгоритм

Сигналы

void **change_lenght** (int lenght)
Сигнал об изменении длины рабочего поля
void **change_width** (int width)
Сигнал об изменении ширины рабочего поля

Открытые члены

CGeneralParams (QWidget *parent=0)

Подробное описание

Класс окна редактирования основных параметров алгоритма

Автор:
Карпова Е.А.
Версия:
1.0
Дата:
Май 2017 года
В окне данного класса задаются основные параметры алгоритма

Методы

void CGeneralParams::change_lenght (int *lenght*) `[signal]`

Сигнал об изменении длины рабочего поля

Аргументы:

lenght	Новая длина поля

void CGeneralParams::change_width (int *width*) `[signal]`

Сигнал об изменении ширины рабочего поля

Аргументы:

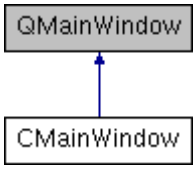
width	Новая ширина рабочего поля

Объявления и описания членов классов находятся в файлах:

C:/Projects/Shepherding_model/general_params.h
C:/Projects/Shepherding_model/general_params.cpp

Класс CMainWindow

Класс главного окна программы
`#include <main_window.h>`
Граф наследования:CMainWindow:



Сигналы

void **take_params** ()
Сигнал, призывающий к передаче параметров в алгоритм

void **paint** ()
Сигнал, призывающий нарисовать кадр работы алгоритма

Открытые члены

CMainWindow (QWidget *parent=0)

Подробное описание

Класс главного окна программы

Автор:
Карпова Е.А.

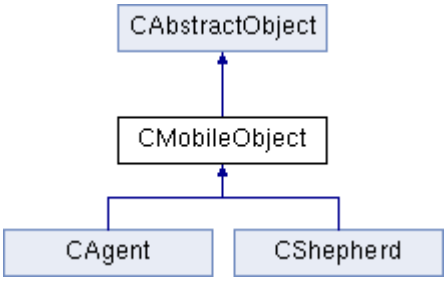
Версия:
1.0

Дата:
Май 2017 года

Объявления и описания членов классов находятся в файлах:
C:/Projects/Shepherding_model/main_window.h
C:/Projects/Shepherding_model/main_window.cpp

Класс CMobileObject

Класс для мобильных объектов алгоритма
`#include <mobile_object.h>`
Граф наследования:CMobileObject:



Открытые члены

void **set_speed** (double **speed**)
Устанавливает скорость объекта

double **get_speed** ()
Возвращает скорость объекта

Защищенные данные

double **speed**
Скорость

Подробное описание

Класс для мобильных объектов алгоритма

Автор:
Карпова Е.А.

Версия:
1.0

Дата:
Май 2017 года

Данный класс является наследником **CAbstractObject** и содержит дополнительные свойства и методы для мобильных объектов

Методы

double CMobileObject::get_speed ()

Возвращает скорость объекта

Возвращает:
Скорость

```
19 {
20     return speed;
```

```
21 }
```

void CMobileObject::set_speed (double *speed*)

Устанавливает скорость объекта

Аргументы:

speed	Скорость

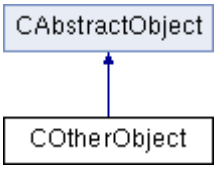
```
11 {
12     if (speed > 0)
13     {
14         this->speed = speed;
15     }
16 }
```

Объявления и описания членов классов находятся в файлах:

C:/Projects/Shepherding_model/model/mobile_object.h
C:/Projects/Shepherding_model/model/mobile_object.cpp

Класс COtherObject

Класс для объектов-преград
`#include <other_object.h>`
Граф наследования:COtherObject:



Открытые члены

void **set_name** (QString name)
Устанавливает имя объекта

QString **get_name** ()
Возвращает имя объекта

Дополнительные унаследованные члены

Подробное описание

Класс для объектов-преград

Автор:
Карпова Е.А.

Версия:
1.0

Дата:
Май 2017 года

Данный класс является наследником класса **CAbstractObject** и содержит дополнительные методы и свойства, помогающие идентифицировать объекты

Методы

QString COtherObject::get_name ()

Возвращает имя объекта

Возвращает:
Имя объекта в виде строки

```
14 {
15     return name;
16 }
```

void COtherObject::set_name (QString *name*)

Устанавливает имя объекта

Аргументы:

name	Имя объекта в виде строки

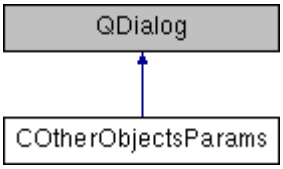
```
9 {
10     this->name = name;
11 }
```

Объявления и описания членов классов находятся в файлах:

C:/Projects/Shepherding_model/model/other_object.h
C:/Projects/Shepherding_model/model/other_object.cpp

Класс COtherObjectsParams

Класс окна редактирования объектов-преград
`#include <other_objects_params.h>`
Граф наследования:COtherObjectsParams:



Сигналы

void **change** ()
Сигнал об изменениях

Открытые члены

COtherObjectsParams (QWidget *parent=0)

Подробное описание

Класс окна редактирования объектов-преград

Автор:
Карпова Е.А.

Версия:
1.0

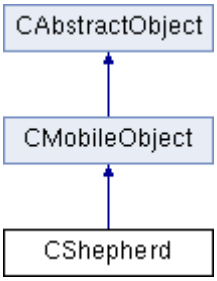
Дата:
Май 2017 года

В окне данного класса задаются основные параметры объектов-преград

Объявления и описания членов классов находятся в файлах:
C:/Projects/Shepherding_model/other_objects_params.h
C:/Projects/Shepherding_model/other_objects_params.cpp

Класс CShepherd

Класс для управляющего объекта
`#include <shepherd.h>`
Граф наследования:CShepherd:



Открытые члены

void **set_blind_angle** (int blind_angle)
Устанавливает слепую зону для пастуха

int **get_blind_angle** ()
Возвращает угол слепой зоны пастуха

Дополнительные унаследованные члены

Подробное описание

Класс для управляющего объекта

Автор:
Карпова Е.А.

Версия:
1.0

Дата:
Май 2017 года

Является наследником класса **CMobileObject** с дополнительными свойствами и методами, характерными для управляющего объекта

Методы

int CShepherd::get_blind_angle ()

Возвращает угол слепой зоны пастуха

Возвращает:
Угол слепой зоны сзади пастуха

Предупреждения:
Данный метод не используется в программе

```
21 {
22     return blind_angle;
```

```
23 }
```

void CShepherd::set_blind_angle (int *blind_angle*)

Устанавливает слепую зону для пастуха

Аргументы:

blind_angle	Угол слепой зоны сзади пастуха

Предупреждения:

Данный метод не используется в программе

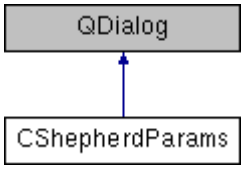
```
13 {
14     if (blind_angle >= 0 && blind_angle <= 360)
15     {
16         this->blind_angle = blind_angle;
17     }
18 }
```

Объявления и описания членов классов находятся в файлах:

C:/Projects/Shepherding_model/model/shepherd.h
C:/Projects/Shepherding_model/model/shepherd.cpp

Класс CShepherdParams

Класс окна редактирования параметров пастуха
`#include <shepherd_params.h>`
Граф наследования:CShepherdParams:



Открытые слоты

void **take_shepherd_params** ()
Передает параметры пастуха в алгоритм

void **set_shepherd_x_max** (int x)
Устанавливает максимальное значение положения пастуха по оси x.

void **set_shepherd_y_max** (int y)
Устанавливает максимальное значение полржения пастуха по оси y.

Открытые члены

CShepherdParams (QWidget *parent=0)

Подробное описание

Класс окна редактирования параметров пастуха

Автор:
Карпова Е.А.

Версия:
1.0

Дата:
Май 2017 года

В окне данного класса задаются основные параметры пастуха

Методы

void CShepherdParams::set_shepherd_x_max (int *x*) [slot]

Устанавливает максимальное значение положения пастуха по оси x.

Аргументы:

x	Максимальное значение x

```
27 {
28     ui->x shepherd->setMaximum(x);
29 }
```

void CShepherdParams::set_shepherd_y_max (int *y*) [slot]

Устанавливает максимальное значение полржения пастуха по оси y.

Аргументы:

y	Максимальное значение y

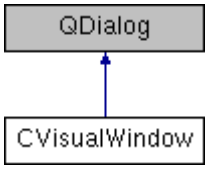
```
32 {
33     ui->y_shepherd->setMaximum(y);
34 }
```

Объявления и описания членов классов находятся в файлах:

C:/Projects/Shepherding_model/shepherd_params.h
C:/Projects/Shepherding_model/shepherd_params.cpp

Класс CVisualWindow

Класс окна визуализации работы алгоритма
`#include <visual_window.h>`
Граф наследования:CVisualWindow:



Открытые слоты

void **set_field_length** (int length)
Устанавливает длину поля для рисования

void **set_field_width** (int width)
Устанавливает ширину поля для рисования

Сигналы

void **paint** (int time)

Открытые члены

CVisualWindow (QWidget *parent=0)

Защищенные члены

void **paintEvent** (QPaintEvent *)
Рисование

Подробное описание

Класс окна визуализации работы алгоритма

Автор:
Карпова Е.А.

Версия:
1.0

Дата:
Май 2017 года

Методы

void CVisualWindow::set_field_length (int *length*) `[slot]`

Устанавливает длину поля для рисования

Аргументы:

length	Длина

```
37 {
```

```
38     ui->field->setFixedWidth(length);
39 }
```

void CVisualWindow::set_field_width (int *width*) [slot]

Устанавливает ширину поля для рисования

Аргументы:

width	Ширина

```
42 {
43     ui->field->setFixedHeight(width);
44 }
```

Объявления и описания членов классов находятся в файлах:

C:/Projects/Shepherding_model/visual_window.h
C:/Projects/Shepherding_model/visual_window

РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

Программа для изучения поведения алгоритма пастушьей собаки

ПРИЛОЖЕНИЕ Б

Красноярск 2017

1 Назначение программы

Программа предназначена для изучения поведения алгоритма пастушьей собаки в зависимости от входных данных.

2 Условия выполнения программы

Для работы с программой необходимо наличие компьютера с клавиатурой и манипулятором типа «мышь» либо ноутбук.

Минимальные системные требования:

1. Видеопамять: 8 МБ;
2. Процессор: Pentium 233 МГц;
3. Оперативная память (ОЗУ): 64 МБ;
4. Свободного места на диске: 500 КБ;
5. Операционная система: Windows 7.

3 Выполнение программы

3.1 Установка и запуск программы

Установка программы не требуется. Для работы необходимо загрузить программу на компьютер и запустить исполняемый файл Shepherding_model.exe. Также возможно создать ярлык и использовать его для запуска программы.

3.2 Главное окно программы

Главное окно программы представлено на рисунке Б.1. В заголовке отображено название алгоритма. В верхней части программы находится меню выбора команд и панель инструментов.

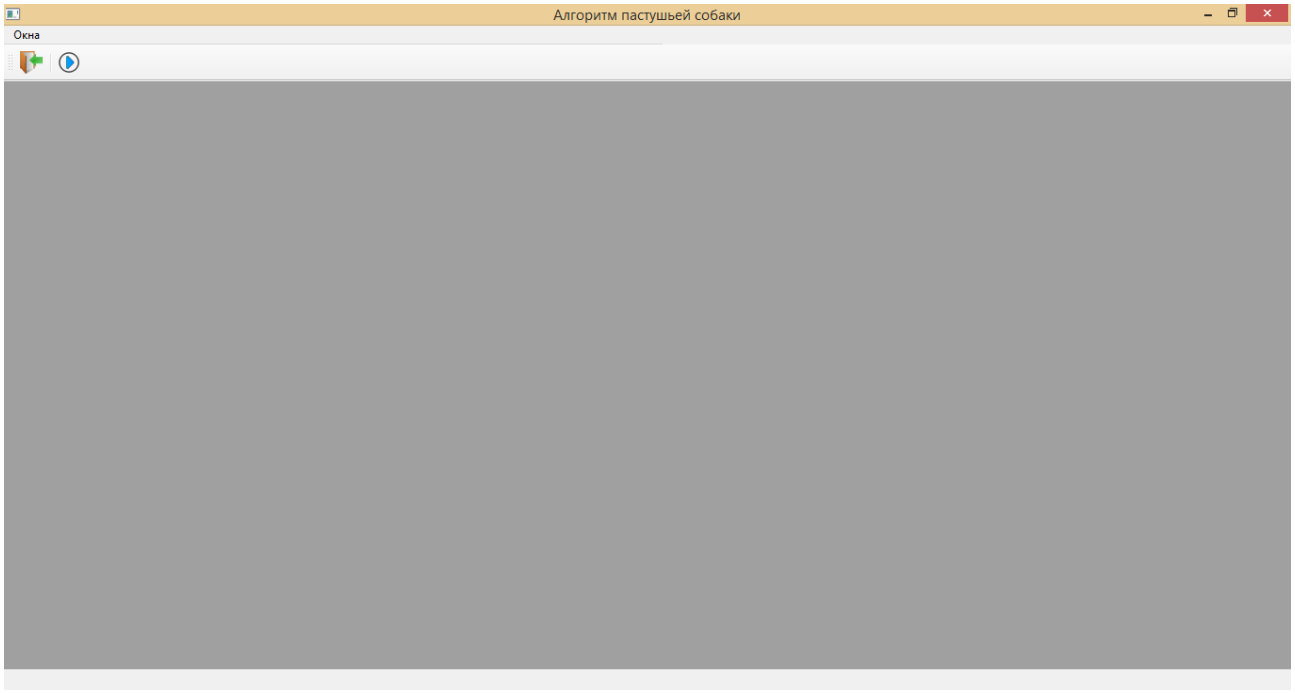


Рисунок Б.1 – Главное окно программы

При выборе «Окна» выпадает список окон, которые возможно использовать в программе. Данный список показан на рисунке Б.2. Существует пять различных окон, четыре из которых отвечают за настройку параметров модели. При выборе определенного пункта меню открывается соответствующее окно в рабочей области программы.

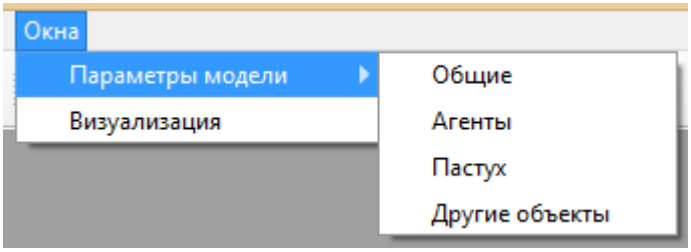


Рисунок Б.2 – Меню выбора окна

На панели инструментов, изображенной на рисунке Б.3, находятся две кнопки: «Выход из приложения» и «Начать эксперимент».

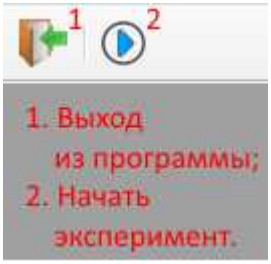


Рисунок Б.3 – Панель инструментов

3.3 Окно редактирования основных параметров алгоритма

Данное окно изображено на рисунке Б.4. В окне «Основные параметры» можно изменять размер поля, редактируя его длину и ширину. Также имеется возможность настройки области генерации агентов. Для этого можно менять координаты центра генерации агентов и размеры области генерации. Кроме того, данное окно позволяет выбрать координаты цели.

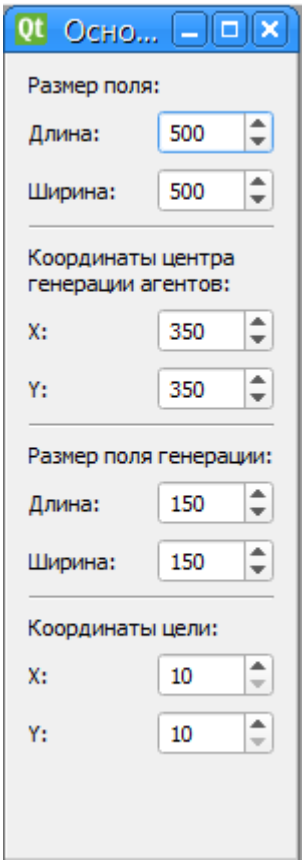


Рисунок Б.4 – Окно редактирования основных параметров алгоритма

3.4. Окно редактирования параметров агентов

Данное окно показано на рисунке Б.5. В нем задаются параметры для автономных управляемых объектов. Возможно задать:

1. Количество агентов, участвующих в алгоритме;
2. Количество ближайших соседей для агента;
3. Скорость передвижения агента;
4. Радиус реакции на пастуха;
5. Радиус реакции на других агентов;
6. Радиус реакции на иные объекты;

7. Силу отталкивания от соседей;

8. Силу притяжения к ближайшим соседям;

9. Силу отталкивания от пастуха;

10. Силу отталкивания от других объектов;

11. Коэффициент для инерции агента. Указывает, насколько необходимо учитывать предыдущее направление агента;

12. Коэффициент шума;

13. Вероятность случайных передвижений.

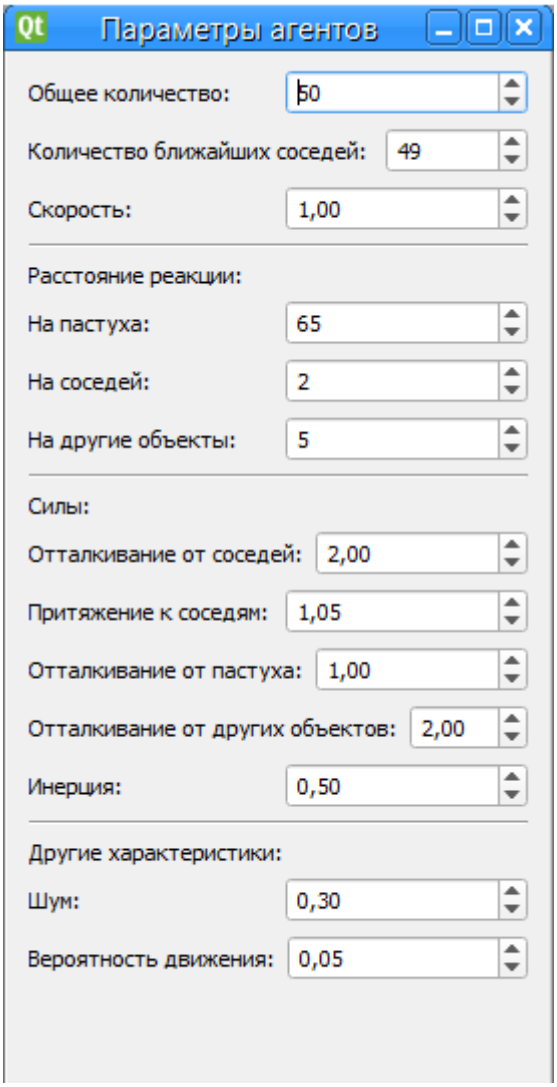


Рисунок Б.5 – Окно редактирования параметров агентов

3.5 Окно редактирования параметров пастуха

Окно «Параметры пастуха» изображено на рисунке Б.6. Здесь можно выбрать тип пастуха, его начальные координаты, скорость передвижения, а также коэффициент шума.

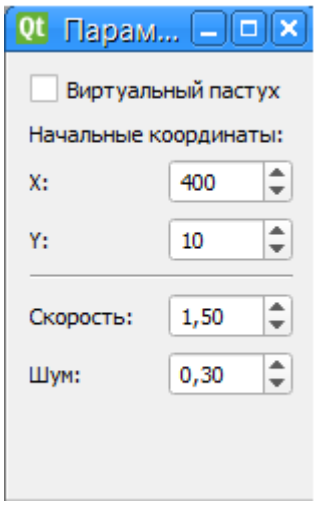


Рисунок Б.6 – Окно редактирования параметров пастуха

3.6 Окно редактирования объектов

Окно, позволяющее редактировать объекты, проиллюстрировано на рисунке Б.7. В этом окне возможно добавлять, редактировать, а также удалять объекты-преграды. Такие объекты в качестве параметров имеют имя, координаты и радиус. Список имеющихся объектов находиться в правой части окна.

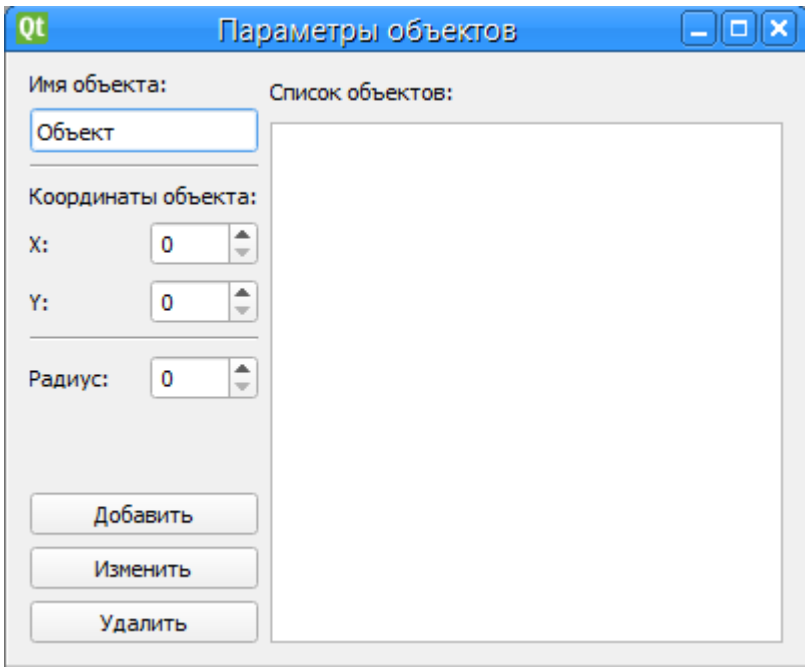


Рисунок Б.7 – Окно редактирования объектов

Для добавления нового объекта необходимо заполнить его параметры и нажать кнопку «Добавить». После этого действия добавленный объект отобразится в списке объектов.

Для того, чтобы изменить параметры существующего объекта, следует выбрать его из списка объектов, поменять нужные характеристики и нажать кнопку «Изменить».

Чтобы удалить объект, нужно выбрать его из списка объектов и нажать на кнопку «Удалить».

3.7 Окно визуализации алгоритма

Окно, отображающее работу алгоритма, показано на рисунке Б.8. В данном окне показывается текущее положение всех объектов алгоритма, а также цель. В

нижней части окна имеются кнопки, позволяющие выбрать удобный цвет для каждого типа участников.

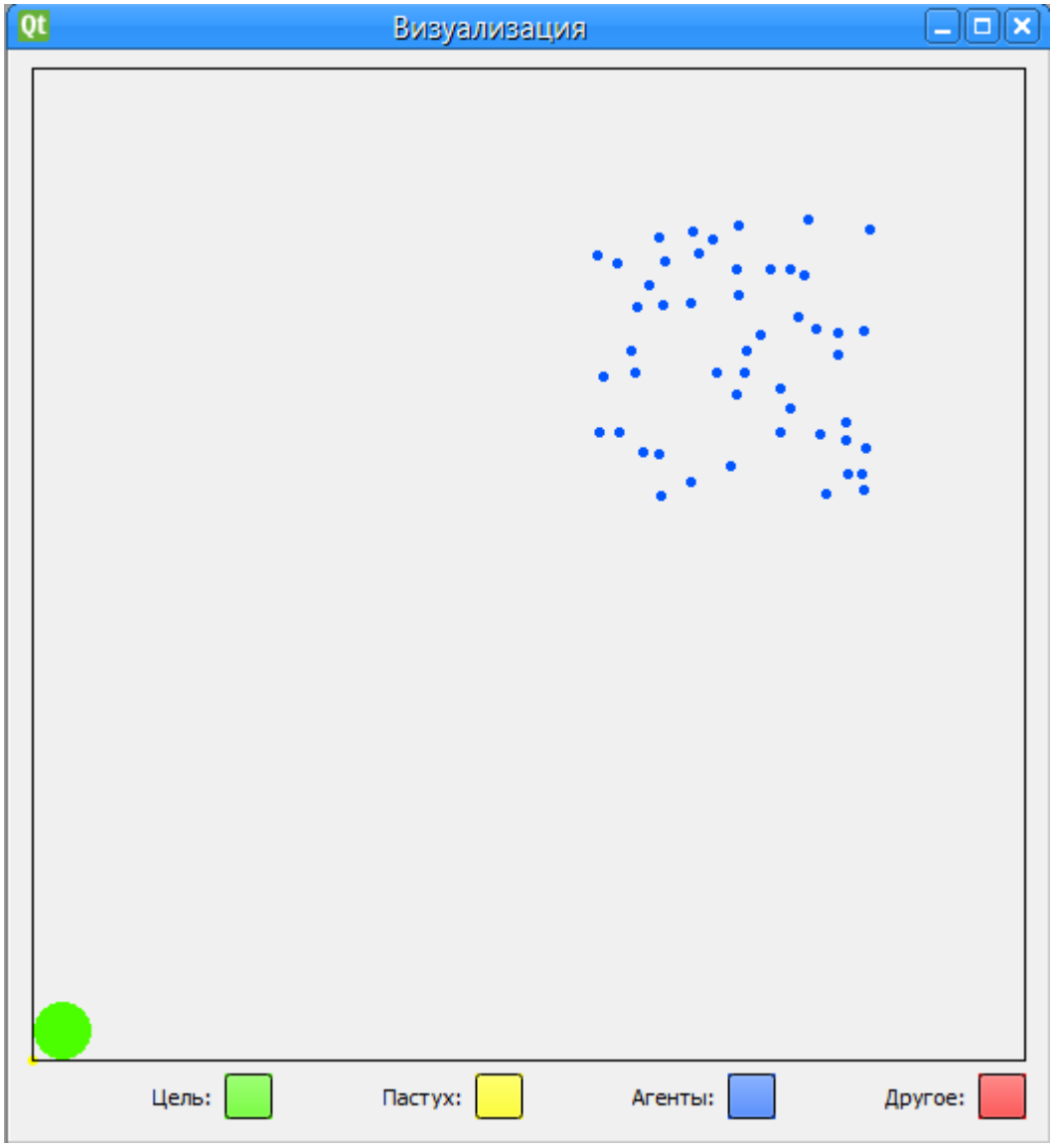


Рисунок Б.8 – Окно визуализации алгоритма

3.8 Выход из программы

Для выхода из программы следует нажать соответствующую кнопку на панели инструментов, либо на красный крестик в правом верхнем углу приложения, как показано на рисунке Б.9.

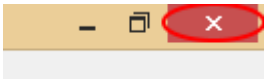


Рисунок Б.9 – Кнопка, закрывающая приложение

4 Сообщения оператору

Если алгоритм был успешно выполнен, то на экран выведется сообщение, изображенное на рисунке Б.10.

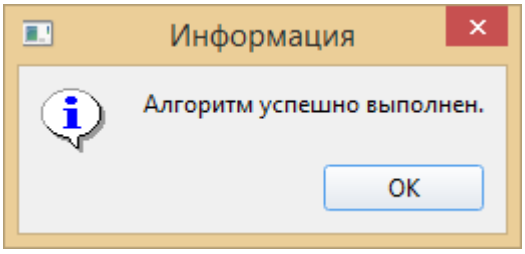


Рисунок Б.10 – Сообщение об успешном завершении алгоритма

Если по каким-либо причинам, алгоритм не может завершиться, то выведется соответствующее сообщение.

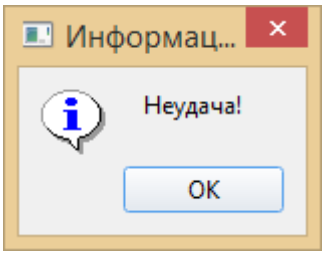


Рисунок Б.11 – Сообщение о неудачном выполнении алгоритма

ПРИЛОЖЕНИЕ В

Приемы устранения технических противоречий

	Что ухудшается при изменении → / Что нужно изменить по условиям задачи ↓	1 Вес подвижного объекта	2 Вес неподвижного объекта	3 Длина подвижного объекта	4 Длина неподвижного объекта	5 Площадь подвижного объекта	6 Площадь неподвижного объекта	7 Объем подвижного объекта	8 Объем неподвижного объекта	9 Скорость	10 Сила	11 Напряжение, давление	12 Форма	13 Устойчивость состава объекта	14 Прочность	15 Время действия подвижного объекта	16 Время действия неподвижного объекта	17 Температура
1	Вес подвижного объекта			15,8 29,34		29,17 38,34		29,2 40,28		2,8 15,38	8,10 18,37	10,36 37,40	10,14 35,40	11,35 19,39	28,27 18,40	5,34 31,35		6,29 4,38
2	Вес неподвижного объекта				10,1 29,35		35,30 13,2		5,35 14,2		8,10 19,35	13,29 10,18	13,10 29,14	26,39 1,40	28,2 10,27		2,27 19,6	28,19 32,22
3	Длина подвижного объекта	8,15 29,34				15,7 4		7,17 4,35		13,4 8	17,10 4	1,8 35	1,8 10,29	1,8 15,34	8,35 29,34	19		10,15 19
4	Длина неподвижного объекта		35,28 40,29				17,7 10,40		35,8 2,14		28,1	1,14 35	13,14 15,7	39,37 35	15,14 28,26		1,40 35	3,35 38,18
5	Площадь подвижного объекта	2,17 29,4		14,15 18,4				7,14 17,4		29,30 4,34	19,30 35,2	10,15 36,28	5,34 29,4	11,2 13,39	3,15 40,14	6,3		2,15 16
6	Площадь неподвижного объекта		30,2 14,18		26,7 9,39						1,18 35,36	10,15 36,37		2,38	40		2,10 19,30	35,39 38
7	Объем подвижного объекта	2,26 29,40		1,7 35,4		1,7 4,17				29,4 38,34	15,35 36,37	6,35 36,37	1,15 29,4	28,10 1,39	9,14 15,7	6,35 4		34,39 10,18
8	Объем неподвижного объекта		35,10 19,14	19,1	35,8 2,14						2,18 37	24,4	7,2 35	34,28 35,40	9,14 17,15		35,34 38	35,6 4
9	Скорость	2,28 13,38		13,1		29,30 34		7,29 34			13,28 15,19	6,18 38,40	35,15 18,34	28,33 1,18	8,3 26,14	3,19 35,5		28,30 36,2
10	Сила	8,1 37,18	18,13 1,28	17,19 9,36	28,1	19,10 15	1,18 36,37	15,9 12,37	2,36 18,37	13,28 15,12		18,21 11	10,35 40,34	35,10 21	35,10 14,27	19,2		35,10 21
11	Напряжение, давление	10,36 37,40	13,29 10,18	35,10 36	35,1 14,16	10,15 36,28	10,15 36,37	6,35 10	35,2	6,35 36	36,35 21		35,4 15,10	35,33 2,40	9,18 3,40	19,3 27		35,39 19,2
12	Форма	8,10 29,40	15,10 26,3	29,34 5,4	13,14 10,7	5,34 4,10		14,4 15,22	7,2 35	35,15 34,18	35,10 37,40	34,15 10,14		33,1 18,4	30,14 10,40	14,26 9,25		22,14 19,32
13	Устойчивость состава объекта	21,35 2,39	26,39 1,40	13,15 1,28	37	2,11 13	39	28,10 19,39	34,28 35,40	33,15 28,18	10,35 21,16	2,35 40	22,1 18,4		17,9 15	13,27 10,35	39,3 35,23	35,1 32
14	Прочность	1,8 40,15	40,26 27,1	1,15 8,35	15,14 28,26	3,34 40,29	9,40 28	10,15 14,7	9,14 17,15	8,13 26,14	10,18 3,14	10,3 18,40	10,30 35,40	13,17 35		27,3 26		30,10 40
15	Время действия подвижного объекта	19,5 34,31		2,19 9		3,17 19		10,2 19,30		3,35 5	19,2 16	19,3 27	14,26 28,25	13,3 35	27,3 10			19,35 39
16	Время действия неподвижного объекта		6,27 19,16		1,40 35				35,34 38					39,3 35,23				19,18 36,40
17	Температура	36,22 6,38	22,35 32	15,19 9	15,19 9	3,35 39,18	35,4	34,39 40,18	35,6 4	2,28 36,30	35,10 3,21	35,39 19,2	14,22 19,32	1,35 32	10,30 22,40	19,13 39	19,18 36,40	
18	Освещенность	19,1 32	2,35 32	19,32 16		19,32 26		2,13 10		10,13 19	26,19 6		32,3	32,3 27	35,2	2,19 6		32,35 19
19	Затраты энергии подвижным объектом	12,18 28,31		12,3		15,19 25		35,13 18		8,15 35	16,26 21,1	23,14 25	12,2 29	19,13 17,24	5,19 9,35	28,35 6,18		19,24 3,14
20	Затраты энергии неподвижным		19,9 6,27								36,4			27,4 29,18	35			

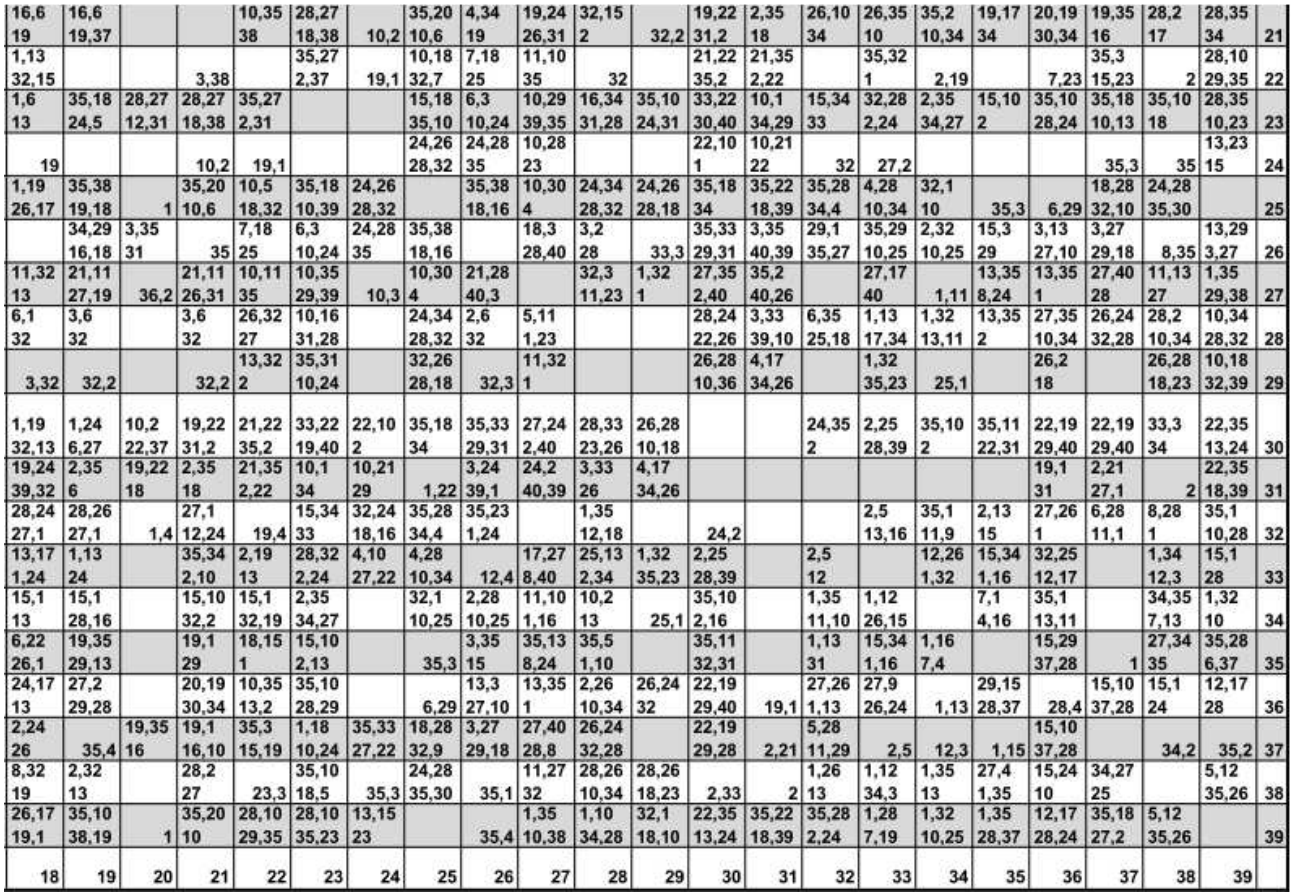
Рисунок В.1 – Таблица выбора приемов устранения технических противоречий, лист 1

21	Мощность	8,36 38,31	19,26 17,27	1,10 35,37		19,4	17,32 13,38	35,6 38	30,6 25	15,35 2	26,2 36,35	22,10 35	29,14 2,40	35,32 15,31	26,10 28	19,35 10,38	16	2,14 17,25
22	Потери энергии	15,6 19,28	19,6 18,9	7,2 6,13	6,28 7	15,26 17,30	17,7 30,18	7,18 23	7	16,35 38	36,4			14,2 39,6	26			19,38 7
23	Потери вещества	35,6 23,40	35,6 22,32	14,29 10,39	10,28 24	35,2 10,31	10,18 39,31	1,29 30,36	3,39 18,31	10,13 28,38	14,15 18,40	3,36 37,10	29,35 3,5	2,14 30,40	35,28 31,40	28,27 3,18	27,16 18,38	21,36 39,31
24	Потери информации	10,24 35	10,35 5	1,26	26	30,3	30,2		2,22	26,3						10	10	
25	Потери времени	10,20 37,35	10,20 26,5	15,2 29	30,24 14,5	26,4 5,16	10,35 17,4	2,5 34,10	35,16 32,18		10,37 36,5	37,36 4	4,10 34,17	35,3 22,5	29,3 28,18	20,10 28,18	28,20 10,16	35,29 21,18
26	Количество вещества	35,6 18,31	27,26 18,35	29,14 35,18		15,15 29	2,18 40,4	15,20 29		35,29 34,28	35,14 3	10,36 14,3	35,1	15,2 17,40	14,35 34,10	3,35 10,40	3,35 31	3,17 39
27	Надежность	3,8 10,40	3,10 8,28	15,9 14,4	15,29 28,11	17,10 14,16	32,35 40,4	3,10 14,24	2,35 24	21,35 11,28	8,28 10,3	10,24 35,19	35,1 16,11		11,3	2,35 3,25	34,27 6,40	3,35 10
28	Точность измерения	32,35 26,28	28,35 25,26	28,26 5,16	32,28 32,3	26,28 32,2	26,28 32,3	32,13 6		28,13 32,24	32,2	6,28 32	6,28 32	32,35 13	28,6 32	28,6 32	10,26 24	6,19 28,24
29	Точность изготовления	28,32 13,18	28,35 27,9	10,28 29,37	2,32 10	28,33 29,32	2,29 18,36	32,28 2	25,10 35	10,28 32	28,19 34,36	3,35	32,30 40	30,2	3,27	3,27 40		19,3
30	Вредные факторы, действующие на объект	22,21 27,39	2,22 13,24	17,1 39,4	1,18	22,1 33,28	27,2 39,35	22,23 37,35	34,39 19,27	21,22 35,28	13,35 39,18	22,2 3,35	22,1 3,35	35,24 30,18	18,35 37,1	22,15 33,28	17,1 40,33	22,33 35,2
31	Вредные факторы самого объекта	19,22 15,39	35,22 1,39	17,15 16,22		17,2 18,39	22,1 40	17,2 40	30,18 35,4	35,28 3,23	35,28 1,40	2,33 27,18	35,1	35,40 27,39	15,35 22,2	15,22 33,31	21,39 16,22	22,35 2,24
32	Удобство изготовления	28,29 15,16	1,27 36,13	1,29 13,17	15,17 27	13,1 26,12	16,4	13,29 1,40	35	35,13 8,1	35,1	35,19 1,37	1,28 13,27	11,13 1	1,3 10,32	27,1 4	35,2	27,26 18
33	Удобство эксплуатации	25,2 13,15	6,13 1,25	1,17 13,12		1,17 13,16	18,16 15,39	1,16 35,15	4,18 39,31	18,13 34	28,13 35	2,32 12	15,34 29,28	32,35 30	32,40 3,28	29,3 8,25	1,16 25	26,27 13
34	Удобство ремонта	2,27 35,11	2,27 35,11	1,28 10,25	3,18 31	15,13 32	16,3	25,2 35,11	1	34,9	1,11	13	1,13 2,4	2,35	1,11 2,9	11,29 28,27	1	4,1
35	Адаптация, универсальность	1,6 15,8	19,15 29,16	35,1 29,2	1,35 16	35,30 29,7	15,2	15,35 29		35,10 14	15,17 20	35,2	15,37 1,8	35,30 14	35,3 32,6	13,1 35	2,16	27,2 3,35
36	Сложность устройства	26,30 34,36	2,26 35,39	1,19 26,24	26	14,1 13,16	6,36	34,26 6	1,16	34,10 28	26,2	19,1 35	29,13 28,15	2,22 17,19	2,13 28	10,4 28,15		2,17 13
37	Сложность контроля и измерения	27,26 28,13	6,13 28,1	16,17 26,24	26	2,13 18,17	2,39 30,16	29,1 4,16	2,18 26,31	3,4 16,35	36,28 40,19	35,36 37,32	27,13 1,39	11,22 39,30	27,3 15,28	19,29 25,39	25,34 6,35	3,27 35,16
38	Степень Автоматизации	28,26 18,35	28,26 35,10	14,13 28,17	23	17,14 13		35,13 16		28,1	2,35	13,4	15,32 1,13	18,1	25,1	6,9		26,2 19
39	Производительность	35,26 24,37	28,27 15,3	18,4 28,38	30,7 14,26	10,26 34,31	10,35 17,7	2,6 34,10	35,37 10,2		28,15 10,36	10,37 14	14,10 34,40	35,3 22,39	29,28 10,18	35,10 2,18	20,10 16,38	35,21 28,10
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17

Рисунок В.1, лист 2

18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	
Освещенность	Затраты энергии подвижным объектом	Затраты энергии неподвижным объектом	Мощность	Потери энергии	Потери вещества	Потери информации	Потери времени	Количество вещества	Надежность	Точность измерения	Точность изготовления	Вредные факторы, действующие на объект	Вредные факторы самого объекта	Удобство изготовления	Удобство эксплуатации	Удобство ремонта	Адаптация, универсальность	Сложность устройства	Сложность контроля и измерения	Степень Автоматизации	Производительность	
19,1	35,12		12,36	6,2	5,35	10,24	10,35	3,26	3,11	28,27	28,35	22,21	22,35	27,28	35,3	2,27	29,5	26,30	28,29	26,35	35,3	
32	34,31		18,,3	34,19	3,31	35	20,28	18,31	1,27	35,26	26,18	18,27	31,39	1,36	2,24	28,11	15,8	36,34	26,32	18,19	24,37	1
35		18,19	15,19	18,19	5,8	10,15	10,20	19,6	10,28	18,26	10,1	2,19	35,22	28,1	6,13	2,27	19,15	1,10	25,28	2,26	1,28	
19,32		28,1	18,22	28,15	13,30	35	35,26	18,26	803	28	35,17	22,37	1,39	9	1,32	28,11	29	26,39	17,15	35	15,35	2
	8,35			7,2	4,29		15,2		10,14	28,32	10,28	1,15		1,29	15,29	1,28	14,15	1,19	35,1	17,24	14,4	
32	24		1,35	35,39	23,10	1,24	29	29,4	29,40	4	29,37	17,24	17,2	17	39,4	10	1,16	26,24	26,24	26,16	28,29	3
					10,28		30,29		15,29	32,28	2,32			15,17							30,14	
3,25			12,8	6,28	24,35	24,3	14		28	3	10	1,18		27	2,25	3	1,35	1,26	26		7,26	4
15,32			19,10	15,17	10,35			29,30		26,28		22,33	17,2	13,1	15,17	15,13		14,1	2,36	14,30	10,26	
19,13	19,3		32,18	30,26	2,39	30,3	26,4	6,13	29,9	32,3	2,32	28,1	18,39	26,24	13,16	10,1	15,3	13	26,18	28,23	34,2	5
				17,7	10,14		10,35	2,18	32,35	26,28	2,29	27,2	22,1					1,18	2,35		10,15	
			17,3	30	18,39	30,2	4,18	40,4	40,4	32,3	18,36	39,35	40	40,2	16,4	16	15,2	36	30,18	23	17,7	6
10,13			35,6	7,15	36,39		2,6	29,30	14,1	25,26	25,28	22,21	17,2	29,1	15,13				29,26	35,34	10,6	
2	35		13,18	13,16	34,10	2,22	34,10	7	40,11	28	2,16	27,35	40,1	40	30,12	10	15,3	26,1	4	16,24	2,34	7
					10,39		35,16		2,35		35,10	34,39	30,18						2,17		35,37	
			30,6		35,34		32,18	35,3	16		25	19,27	35,4	35		1		1,31	26		10,2	8
10,13	8,15		19,35	14,20	10,13			10,19	11,35	28,32	10,28	1,28	2,24	35,13	32,28	34,2	15,10	10,28	3,34			
19	35,38		38,2	19,35	28,38	13,3		29,38	27,28	1,24	32,25	35,23	35,21	8,1	13,12	28,27	26	4,34	27,16	10,2		9
	19,17	1,16	19,35		8,35		10,37	14,29	3,35	35,10	28,29	1,35	13,3	15,37	1,28	15,1	15,17	26,35	36,37		3,28	
	10	36,37	18,37	14,2	40,5		36	18,36	13,21	23,24	37,36	40,18	36,24	18,1	3,25	11	18,20	10,18	10,19	2,35	35,37	10
	14,24		10,35	2,36	10,36		37,36	10,14	10,13	6,28		22,2	2,33	1,35				19,1	2,36		10,14	
	10,37		14	25	3,37		4	36	19,35	25	3,35	37	27,18	16	11	2	35	35	37	35,2	35,37	11
13,15	2,6				35,29		14,10		10,40	28,32	32,30	22,1		1,32	32,15	2,13	1,15	16,29	15,13	15,1	17,26	
32	34,14		4,6 2	14	3,5		34,17	36,2	16	1	40	2,35	35,1	17,28	26	1	29	1,28	39	32	34,10	12
32,3		27,4	32,35	14,2	2,14			15,32				35,24	35,40		32,35	2,35	35,30	2,35	35,22	1,8	23,35	
27,15	13,2	2918	27,31	39,6	30,40		35,3	35		13	18	18,30	27,39	35,2	30	10,16	34,2	22,26	39,23	35	40,3	13
	19,35		10,26		35,28		29,3	29,10		3,27		18,35	15,35	11,3	32,40	27,11	15,3	2,13	27,3		29,35	
35,2	10	35	35,28	35	31,40		28,10	27	11,3	16	3,27	37,1	22,2	10,32	28,2	3	32	28	15,40	15	10,14	14
2,19	28,6		19,10		28,27		20,10	3,35	11,2		3,27	22,15	21,39	27,1		29,10	1,35	10,4	19,29		35,17	
4,35	35,18		35,38		3,18	10	28,18	10,40	13	3	16,40	33,28	16,22	4	12,3	27	13	29,15	39,35	6,1	14,19	15
					27,16		28,20	3,35	34,27	10,26		17,1							25,34		20,10	
			16		18,38	10	10,16	31	6,40	24		40,33	22	35,1	1	1	2		6,35	1	16,38	16
32,30	19,15		2,14	21,17	21,36		35,28	3,17	19,35	32,19		22,33	22,35			4,10	2,18	2,17	3,27	26,2	15,28	
21,16	3,17		17,25	35,38	29,31		21,18	30,39	3,10	24	24	35,2	2,24	26,3	26,3	16	27	16	35,31	19,16	35	17
	32,1	32,1		19,16			19,1			11,15			35,19	19,35	28,26	15,17	15,1	6,32		2,26	2,25	
	19	1,15	32	1,6	13,1	1,6	26,17	1,19		32	3,32	15,2	32,39	28,26	19	13,16	1,19	13	32,2	10	16	18
2,15			6,19	12,22	35,24		35,38	34,23	19,21	3,1		1,35	2,35	28,26		1,15	15,17	2,29			12,28	
19			37,18	15,24	18,5		19,18	16,18	11,27	32		6,27	6	30	19,4	17,28	13,16	27,28	35,4	32,2	35	19
19,2					28,27			3,35	10,36			10,2	19,22						19,35			
35,32					18,31			31	23			22,37	18	1,4					16,25		1,6	20

Рисунок В.1, лист 3



16,6	16,6			10,35	28,27		35,20	4,34	19,24	32,15		19,22	2,35	26,10	26,35	35,2	19,17	20,19	19,35	28,2	28,35	
19	19,37			38	18,38	10,2	10,6	19	26,31	2	32,2	31,2	18	34	10	10,34	34	30,34	16	17	34	21
1,13					35,27		10,18	7,18	11,10			21,22	21,35		35,32				35,3		28,10	
32,15			3,38		2,37	19,1	32,7	25	35	32		35,2	2,22		1	2,19		7,23	15,23	2	29,35	22
1,6	35,18	28,27	28,27	35,27			15,18	6,3	10,29	16,34	35,10	33,22	10,1	15,34	32,28	2,35	15,10	35,10	35,18	35,10	28,35	
13	24,5	12,31	18,38	2,31			35,10	10,24	39,35	31,28	24,31	30,40	34,29	33	2,24	34,27	2	28,24	10,13	18	10,23	23
							24,26	24,28	10,28			22,10	10,21								13,23	
19			10,2	19,1			28,32	35	23			1	22	32	27,2				35,3	35	15	24
1,19	35,38		35,20	10,5	35,18	24,26		35,38	10,30	24,34	24,26	35,18	35,22	35,28	4,28	32,1			18,28	24,28		
26,17	19,18	1	10,6	18,32	10,39	28,32		18,16	4	28,32	28,18	34	18,39	34,4	10,34	10	35,3	6,29	32,10	35,30		25
	34,29	3,35		7,18	6,3	24,28	35,38		18,3	3,2		35,33	3,35	29,1	35,29	2,32	15,3	3,13	3,27		13,29	
	16,18	31	35	25	10,24	35	18,16		28,40	28	33,3	29,31	40,39	35,27	10,25	10,25	29	27,10	29,18	8,35	3,27	26
11,32	21,11		21,11	10,11	10,35		10,30	21,28		32,3	1,32	27,35	35,2		27,17		13,35	13,35	27,40	11,13	1,35	
13	27,19	36,2	26,31	35	29,39	10,3	4	40,3		11,23	1	2,40	40,26		40	1,11	8,24	1	28	27	29,38	27
6,1	3,6		3,6	26,32	10,16		24,34	2,6	5,11			28,24	3,33	6,35	1,13	1,32	13,35	27,35	26,24	28,2	10,34	
32	32		32	27	31,28		28,32	32	1,23			22,26	39,10	25,18	17,34	13,11	2	10,34	32,28	10,34	28,32	28
				13,32	35,31		32,26		11,32			26,28	4,17		1,32			26,2		26,28	10,18	
3,32	32,2		32,2	2	10,24		28,18	32,3	1			10,36	34,26		35,23	25,1		18		18,23	32,39	29
1,19	1,24	10,2	19,22	21,22	33,22	22,10	35,18	35,33	27,24	28,33	26,28			24,35	2,25	35,10	35,11	22,19	22,19	33,3	22,35	
32,13	6,27	22,37	31,2	35,2	19,40	2	34	29,31	2,40	23,26	10,18			2	28,39	2	22,31	29,40	29,40	34	13,24	30
19,24	2,35	19,22	2,35	21,35	10,1	10,21		3,24	24,2	3,33	4,17							19,1	2,21		22,35	
39,32	6	18	18	2,22	34	29	1,22	39,1	40,39	26	34,26							31	27,1	2	18,39	31
28,24	28,26		27,1		15,34	32,24	35,28	35,23		1,35					2,5	35,1	2,13	27,26	6,28	8,28	35,1	
27,1	27,1	1,4	12,24	19,4	33	18,16	34,4	1,24		12,18		24,2			13,16	11,9	15	1	11,1	1	10,28	32
13,17	1,13		35,34	2,19	28,32	4,10	4,28		17,27	25,13	1,32	2,25		2,5		12,26	15,34	32,25		1,34	15,1	
1,24	24		2,10	13	2,24	27,22	10,34	12,4	8,40	2,34	35,23	28,39		12		1,32	1,16	12,17		12,3	28	33
15,1	15,1		15,10	15,1	2,35		32,1	2,28	11,10	10,2		35,10		1,35	1,12		7,1	35,1		34,35	1,32	
13	28,16		32,2	32,19	34,27		10,25	10,25	1,16	13	25,1	2,16		11,10	26,15		4,16	13,11		7,13	10	34
6,22	19,35		19,1	18,15	15,10			3,35	35,13	35,5		35,11		1,13	15,34	1,16		15,29		27,34	35,28	
26,1	29,13		29	1	2,13		35,3	15	8,24	1,10		32,31		31	1,16	7,4		37,28	1	35	6,37	35
24,17	27,2		20,19	10,35	35,10			13,3	13,35	2,26	26,24	22,19		27,26	27,9		29,15		15,10	15,1	12,17	
13	29,28		30,34	13,2	28,29		6,29	27,10	1	10,34	32	29,40	19,1	1,13	26,24	1,13	28,37	28,4	37,28	24	28	36
2,24		19,35	19,1	35,3	1,18	35,33	18,28	3,27	27,40	26,24		22,19		5,28				15,10				
26	35,4	16	16,10	15,19	10,24	27,22	32,9	29,18	28,8	32,28		29,28	2,21	11,29	2,5	12,3	1,15	37,28		34,2	35,2	37
8,32	2,32		28,2		35,10		24,28		11,27	28,26	28,26			1,26	1,12	1,35	27,4	15,24	34,27		5,12	
19	13		27	23,3	18,5	35,3	35,30	35,1	32	10,34	18,23	2,33	2	13	34,3	13	1,35	10	25		35,26	38
26,17	35,10		35,20	28,10	28,10	13,15			1,35	1,10	32,1	22,35	35,22	35,28	1,28	1,32	1,35	12,17	35,18	5,12		
19,1	38,19	1	10	29,35	35,23	23		35,4	10,38	34,28	18,10	13,24	18,39	2,24	7,19	10,25	28,37	28,24	27,2	35,26		39
18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	

Рисунок В.1, лист 4

Приемы устранения технических противоречий:

1) Принцип дробления:
   a) Разделить объект на независимые части;
   b) Выполнить объект разборным;
   c) Увеличить степень дробления объекта.
2) Принцип вынесения. Отделить от объекта "мешающую" часть ("мешающее" свойство) или, наоборот, выделить единственно нужную часть или нужное свойство.
3) Принцип местного качества:

a) Перейти от однородной структуры объекта или внешней среды (внешнего воздействия) к неоднородной;
b) Разные части объекта должны выполнять различные функции;
c) Каждая часть объекта должна находиться в условиях, наиболее благоприятных для ее работы.

4) Принцип асимметрии:
a) Перейти от симметричной формы объекта к асимметричной;
b) Если объект уже асимметричен, увеличить степень асимметрии.

5) Принцип объединения:
a) Соединить однородные или предназначенные для смежных операций объекты;
b) Объединить во времени однородные или смежные операции.

6) Принцип универсальности. Объект выполняет несколько разных функций, благодаря чему отпадает необходимость в других объектах.

7) Принцип "матрешки":
a) Один объект размещен внутри другого, который, в свою очередь, находится внутри третьего и т. д.;
b) Один объект проходит сквозь полость в другом объекте.

8) Принцип антивеса:
a) Компенсировать вес объекта соединением с другим объектом, обладающим подъемной силой;
b) Компенсировать вес объекта взаимодействием со средой (преимущественно за счет аэро- и гидродинамических сил).

9) Принцип предварительного антидействия. Если по условиям задачи необходимо совершить какое-то действие, надо заранее совершить антидействне.

10) Принцип предварительного действия:

a) Заранее выполнить требуемое действие (полностью или хотя бы частично);
b) Заранее расставить объекты так, чтобы они могли вступить в действие без затрат времени на доставку и с наиболее удобного места.

11) Принцип "заранее подложенной подушки". Компенсировать относительно невысокую надежность объекта заранее подготовленными аварийными средствами.
12) Принцип эквипотенциальности. Изменить условия работы так, чтобы не приходилось поднимать или опускать объект.
13) Принцип "наоборот":
a) Вместо действия, диктуемого условиями задачи, осуществить обратное действие;
b) Сделать движущуюся часть объекта или внешней среды неподвижной, а неподвижную - движущейся. в. Перевернуть объект "вверх ногами", вывернуть его.
14) Принцип сфероидальносги:
a) Перейти от прямолинейных частей к криволинейным, от плоских поверхностей к сферическим, от частей, выполненных в виде куба или параллелепипеда, к шаровым конструкциям;
b) Использовать ролики, шарики, спирали;
c) Перейти от прямолинейного движения к вращательному, использовать центробежную силу.
15) Принцип динамичности:
a) Характеристики объекта (или внешней среды) должны меняться так, чтобы быть оптимальными на каждом этапе работы;
b) Разделить объект на части, способные перемещаться относительно друг друга;

c) Если объект, в целом неподвижен, сделать его подвижным, перемещающимся.

16) Принцип частичного или избыточного действия. Если трудно получить 100% требуемого эффекта, надо получить "чуть меньше" или "чуть больше" - задача при этом может существенно упроститься.

17) Принцип перехода в другое измерение:

a) Трудности, связанные с движением (или размещением) объекта по линии, устраняются, если объект приобретает возможность перемещаться в двух измерениях (т. е. на плоскости). Соответственно задачи, связанные с движением (или размещением) объектов в одной плоскости, устраняются при переходе к пространству трех измерений;

b) Использовать многоэтажную компоновку объектов вместо одноэтажной;

c) Наклонить объект или положить его "набок";

d) Использовать обратную сторону данной площади;

e) Использовать оптические потоки, падающие на соседнюю площадь или на обратную сторону имеющейся площади.

18) Использование механических колебаний:

a) Привести объект в колебательное движение;

b) Если такое движение уже совершается, увеличить его частоту (вплоть до ультразвуковой);

c) Использовать резонансную частоту;

d) Применить вместо механических вибраторов пьезовибраторы;

e) Использовать ультразвуковые колебания в сочетании с электромагнитными полями.

19) Принцип периодического действия:

a) Перейти от непрерывного действия к периодическому (импульсному);

b) Если действие уже осуществляется периодически, изменить периодичность;
c) Использовать паузы между импульсами для другого действия.

20) Принцип непрерывности полезного действия:
a) Вести работу непрерывно (все части объекта должны все время работать с полной нагрузкой);
b) Устранить холостые и промежуточные ходы.

21) Принцип проскока. Вести процесс или отдельные его этапы (например, вредные или опасные) на большой скорости.

22) Принцип "обратить вред в пользу":
a) Использовать вредные факторы (в частности, вредное воздействие среды) для получения положительного эффекта;
b) Устранить вредный фактор за счет сложения с другими вредными факторами;
c) Усилить вредный фактор до такой степени, чтобы он перестал быть вредным.

23) Принцип обратной связи:
a) Ввести обратную связь;
b) Если обратная связь есть, изменить ее.

24) Принцип "посредника":
a) Использовать промежуточный объект, переносящий или передающий действие;
b) На время присоединить к объекту другой (легкоудаляемый) объект.

25) Принцип самообслуживания:
a) Объект должен сам себя обслуживать, выполняя вспомогательные и ремонтные операции;
b) Использовать отходы (энергии, вещества).

26) Принцип копирования:
    a) Вместо недоступного, сложного, дорогостоящего, неудобного или хрупкого объекта использовать его упрощенные и дешевые копии;
    b) Заменить объект или систему объектов их оптическими копиями (изображениями). Использовать при этом изменение масштаба (увеличить или уменьшить копии);
    c) Если используются видимые оптические копии, перейти к копиям инфракрасным или ультрафиолетовым.
27) Дешевая недолговечность взамен дорогой долговечности. Заменить дорогой объект набором дешевых объектов, поступившись при этом некоторыми качествами (например, долговечностью).
28) Замена механической схемы:
    a) Заменить механическую схему оптической, акустической или "запаховой";
    b) Использовать электрические, магнитные и электромагнитные поля для взаимодействия с объектом;
    c) Перейти от неподвижных полей к движущимся, от фиксированных к меняющимся во времени, от неструктурных к имеющим определенную структуру;
    d) Использовать поля в сочетании с ферромагнитными частицами.
29) Использование пневмо- и гидроконструкций. Вместо твердых частей объекта использовать газообразные и жидкие: надувные и гидронаполяемые, воздушную подушку, гидростатические и гидрореактивные.
30) Использование гибких оболочек и тонких пленок:
    a) Вместо обычных конструкций использовать гибкие оболочки и тонкие пленки;

b) Изолировать объект от внешней среды с помощью гибких оболочек и тонких пленок.

31) Применение пористых материалов:

a) Выполнить объект пористым или использовать дополнительные пористые элементы (вставки, покрытия и т. д.);

b) Если объект уже выполнен пористым, предварительно заполнить поры каким-то веществом.

32) Принцип изменения окраски:

a) Изменить окраску объекта или внешней среды;

b) Изменить степень прозрачности объекта или внешней среды;

c) Для наблюдения за плохо видимыми объектами или процессами использовать красящие добавки;

d) Если такие добавки уже применяются, использовать люминофоры.

33) Принцип однородности. Объекты, взаимодействующие с данным объектом, должны быть сделаны из того же материала (или близкого ему по свойствам).

34) Принцип отброса и регенерация частей:

a) Выполнившая свое назначение или ставшая ненужной часть объекта должна быть отброшена (растворена, испарена и т. п.) или видоизменена непосредственно в ходе работы;

b) Расходуемые части объекта должны быть восстановлены непосредственно в ходе работы.

35) Изменение агрегатного состояния объекта. Сюда входят не только простые переходы, например от твердого состояния к жидкому, но и переходы к "псевдосостояниям" ("псевдожидкость") и промежуточным состояниям, например использование эластичных твердых тел.

36) Применение фазовых переходов. Использовать явления, возникающие при фазовых переходах, например изменение объема, выделение или поглощение тепла и т. д.

37) Применение теплового расширения:

a) Использовать тепловое расширение (или сжатие) материалов;
b) Использовать несколько материалов с разными коэффициентами теплового расширения.

38) Применение сильных окислителей:

a) Заменить обычный воздух обогащенным;
b) Заменить обогащенный воздух кислородом;
c) Воздействовать на воздух или кислород, ионизирующими излучениями;
d) Использовать озонированный кислород;
e) Заменить озонированный (или ионизированный) кислород озоном.

39) Применение инертной среды:

a) Заменить обычную среду инертной;
b) Вести процесс в вакууме. Этот прием можно считать антиподом предыдущего.

40) Применение композиционных материалов перейти от однородных материалов к композиционным.

Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
Институт космических и информационных технологий
Информатика

УТВЕРЖДАЮ
Заведующий кафедрой
_____А. С. Кузнецов
подпись инициалы, фамилия
«08» 06 2017 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

09.03.04 «Программная инженерия»

Вычислительный эксперимент по оценке эффективности алгоритма работы пастушьей собаки

Научный руководитель/
руководитель _____ 09.06.2017 доцент, канд. техн. наук И.В. Евдокимов
подпись, дата должность, ученая степень инициалы, фамилия

Выпускник _____ 31.05.2017 г. Е.А. Карпова
подпись, дата инициалы, фамилия

Нормоконтролер _____ 07.06.17 О.А. Антамошкин
подпись, дата инициалы, фамилия

Красноярск 2017

Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
Институт космических и информационных технологий
Информатика

УТВЕРЖДАЮ
Заведующий кафедрой
_____А. С. Кузнецов
подпись инициалы, фамилия
«_18_» _05_____ 2017 г.

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
в форме бакалаврской работы

Студенту Карповой Екатерине Александровне

Группа КИ13-18Б Направление (специальность) 09.03.04

Программная инженерия

Тема выпускной квалификационной работы: Вычислительный эксперимент по оценке эффективности алгоритма работы пастушьей собаки

Утверждена приказом по университету № 2930/с от 07 марта 2017

Руководитель ВКР: И. В. Евдокимов, доцент, канд. техн. наук

Исходные данные для ВКР: научные статьи, учебная литература, нормативно-правовые документы.

Перечень разделов ВКР: Теоретические основы постановки компьютерного эксперимента, Планирование эксперимента и анализ решений по управлению экспериментом, Компьютерный эксперимент

Перечень графического материала: таблицы, графики, формулы, иллюстрации

Руководитель ВКР _____ И. В. Евдокимов
подпись

Задание принял к исполнению _____ Е. А. Карпова
подпись

« 18 » мая 2017

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

КАФЕДРА «ИНФОРМАТИКА»

ОТЗЫВ
руководителя о качестве работы студента в процессе выполнения выпускной квалификационной работы

Студента Карповой Екатерины Александровны

Направление подготовки 09.03.04 Программная инженерия

Тема ВКР: «Вычислительный эксперимент по оценке эффективности алгоритма работы пастушьей собаки».

Характерной особенностью данной работы, её отличительной чертой является смелая идея автора, Карповой Екатерины Александровны, развить идеи биоинспирированного алгоритма, предложенного междисциплинарной исследовательской командой под руководством Эндрю Кинга и Даниэля Стрёмбома.

Материалы пояснительной записки ВКР имеют связное повествование, взаимоувязаны и грамотно составлены, выводы в разделах ВКР, рекомендации и предложения автора аналитически обоснованы. Графические материалы пояснительной записки удовлетворяют требованиям к ВКР. В период работы над проектом Карпова Екатерина Александровна проявила высокую самостоятельность в выборе проектных решений, высокий уровень теоретической подготовки, умение ориентироваться в научно-технической, специальной литературе и мировых информационных ресурсах. Екатерина Александровна показала глубокие знания в области разработки программного обеспечения, проведения вычислительного эксперимента и оценки эффективности алгоритмов.

Кроме того, Екатерина Александровна в 2016 г. заочно участвовала в ИТ-конференции в г.Уфе и имеет публикацию в сборнике «Новая наука: от идеи к результату», программные наработки Екатерины Александровны по данной ВКР рекомендованы к участию в «Конкурсе работ о современных тенденциях авторского права», проводимого в рамках проекта «Ноосфера. Запуск». С учётом сказанного, считаю, что Екатерине Александровне можно рекомендовать продолжить развитие своего научного и творческого потенциала по магистерской программе «Программная инженерия».

Итак, Карпову Екатерину Александровну можно характеризовать как амбициозного специалиста с аналитическим складом ума, способного самостоятельно вести программные проекты. На основе вышеизложенного можно сделать вывод о том, что автор данной выпускной квалификационной работы, Карпова Екатерина Александровна, заслуживает, присуждения степени бакалавра по направлению подготовки 09.03.04 Программная инженерия, а ВКР Карповой Екатерины Александровны заслуживает, я полагаю, отличной оценки.

Место работы и должность руководителя ВКР
кафедра «Информатика» , доцент научно-учебной лаборатории программного обеспечения кафедры «Информатика» института космических и информационных технологий ФГАОУ ВО «СФУ», канд.техн.наук, доцент
Фамилия, имя, отчество Евдокимов Иван Валерьевич
Подпись _____ «13» июня 2017 г.