ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САРАТОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ ГАГАРИНА Ю.А.»

На правах рукописи

Королёв Михаил Сергеевич

МОДЕЛИ И АЛГОРИТМЫ ПОДДЕРЖКИ ПРИНЯТИЯ РЕШЕНИЯ ПО ОПТИМИЗАЦИИ ВЫБОРА РАЗМЕЩЕНИЯ СЕНСОРОВ СИСТЕМЫ НАБЛЮДЕНИЯ НА ТЕХНИЧЕСКИХ ОБЪЕКТАХ

Специальность: 05.13.01 - Системный анализ, управление и обработка информации (в технической отрасли)

Диссертация
на соискание ученой степени
кандидата технических наук

Научный руководитель –
кандидат физ.-мат. наук,
профессор Печенкин В.В.

Саратов – 2019

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Информационные технологии являются основой развития науки и техники в XXI веке. Это подтверждается не только их фактической востребованностью при решении широкого спектра задач, как повседневных, так и сложных производственных, управленческих и исследовательских, но и заинтересованностью государства в данной сфере [1]. В целях реализации Стратегии развития информационного общества в Российской Федерации на 2017 – 2030 года, утвержденной Указом Президента Российской Федерации от 9 мая 2017 г. № 203 "О Стратегии развития информационного общества в Российской Федерации на 2017 - 2030 годы", по инициативе Президента Российской Федерации, предложена и утверждена программа «Цифровая экономика Российской Федерации», отмечающая технологии идентификации и комплексные системы обеспечения безопасности в качестве одного из наиболее перспективных направлений для оборонно-промышленного комплекса, программных технологий поддержки принятия решений в реальном времени.

Таким образом, можно сделать вывод о перспективности и актуальности разработки комплексных систем обеспечения безопасности для решения задач в разных областях. Потребность в разработке новых программных продуктов проектирования систем безопасности также обусловлена возрастающим спросом на комплексные системы проектирования с возможностью выбора размещения сенсоров наблюдения, зависящие от разных факторов: индустриальное развитие городов; опережающий рост развития систем безопасности, в особенности камер видеонаблюдения; появление новых услуг и сервисов, предоставляющих охранные услуги, в частности, установку и обслуживание систем видеонаблюдения на промышленных предприятиях; противодействие терроризму на критически важных объектах (АЭС, ГЭС, ГРЭС и т.п.). Поэтому оптимизация размещения средств наблюдения является актуальной проблемой и исследования в этой области востребованы.

Использование комплексных систем видеонаблюдения в России началось сравнительно недавно и влияет на их актуальность и совершенствование. В России происходит активное развитие как методов комплексного анализа систем

видеонаблюдения, так и практик их использования на технических объектах. Например, запатентован способ определения оптимальной конфигурации системы видеомониторинга леса для обнаружения и определения местоположения лесных пожаров (Пархачев В.В.) [2]. Разработан аппаратно-программный комплекс «Безопасный город» - сложный комплекс видеонаблюдения, анализа и оперативного управления силами правопорядка, позволяющий оперативно реагировать на различные ситуации городской жизни, например, автоматически выявлять скопление или неадекватное поведение людей (Орлов С.С.) [3].

Однако, перечисленные выше комплексные системы видеонаблюдения являются узконаправленными и требуются дополнительные исследования для применения полученных результатов в других областях деятельности.

Системному анализу задачи оптимизации размещения сенсоров систем видеонаблюдения на технических объектах посвящены работы Давидюка Н.В. [4], Ahn J.W.[5], Chang T.W.[5], Lee Sung. H.[5], Seo Y. W.[5], Hengel V.[6], Dong M.[7], Wang N.[7], Schlüter M.[8], Civera J.[9], Attar E.[10], Fiore L.[11], Zhao J.[12], Hanel M.[13], Holt R.[14], Liu J.[15], Gonzalez-Barbosa J.J.[16], Hörster E.[17], Zhang M.[18], Luo W.[18], Wang X.[19], Egea A.[20], Antelo L. T.[20], Alonso A. A.[20], Banga J. R.[20], Parpinelli R[21]. Последний предложил модифицированный алгоритм муравьиной колонии на основе классификационных правил, этот алгоритм выбран в качестве базового в данной работе.

В исследованиях, представленных выше авторов показано, что в общем случае решаемая задача оптимизации размещения камер является NP–полной, поэтому необходимы методы, работающие в приемлемое время и основанные на эвристиках. Одной из важных задач является описание в используемой формальной модели определения видимости узлов сцены с учётом параметров источников освещения. Такие характерные ситуации требуют введения в модель функции видимости физических свойств объектов сцены и их взаимодействия с источниками освещения, что влияет на видимость. Во-первых, применение такой модели позволит повысить эффективность определения видимости узлов на сцене за счет интегрального показателя, зависящего от положения сенсора в трехмерной сцене, фокусного

расстояния, теней от источника света, шумов и т.п. Во-вторых, описанные в литературе алгоритмы поиска оптимального расположения камер наблюдения строятся, в основном, на генетических и эвристических алгоритмах поиска, применении алгоритмов предварительного ранжирования объектов сцены.

Таким образом, разработка методов и алгоритмов поддержки принятия решения по выбору конфигурации размещения сенсоров видеонаблюдения на технических инфраструктурных объектах и промышленных предприятиях является актуальной задачей. Приемлемые по времени работы алгоритмы решения задачи основаны на введении модели формально определённого графа видимости узлов сцены, определении функции видимости узлов и сцены целиком и эвристическом подходе к поиску с использованием предварительного ранжирования.

В исследовании сформулированы различные задачи оптимизации размещения камер наблюдения, которые будут отличаться как выбором алгоритма, сложностью их решения и критериями оптимизации. Каждая из таких задач предполагает нахождение некоторого множества положений камер, которое является оптимальным с точки зрения предлагаемого для каждого случая критерия видимости. К этим задачам относятся следующие: определение положение камер, при котором достигается максимальная совокупная видимость одного объекта; определение положения камер, при котором достигается максимальная совокупная видимость множества объектов на выделенной области; определение «слепых зон» для текущей конфигурации камер; минимизация «слепых зон»; максимизация покрытия выделенной области; минимизация количества камер наблюдения при сохранении минимального порога видимости сцены.

В рамках исследования каждая из перечисленных оптимизационных задач размещения камер наблюдения на технических объектах, промышленных предприятиях решается как однокритериальная задача с целью определения наилучшего положения камер. Применение комплексных систем обеспечения безопасности с возможностью оптимизации выбора размещения сенсоров наблюдения для технических объектов, промышленных предприятий и критически важных объектов (АЭС,

ГРЭС, ГЭС и т.п.) органично включается в процедуры принятия решений и управления в реальном времени.

Цель работы – оптимизация выбора размещения сенсоров системы наблюдения на технических объектах с целью определения положения камер, при котором достигается максимальная совокупная видимость выделенной области.

Задачи работы. Для достижения поставленной цели необходимо решить следующие задачи:

1. Формализация задачи дискретной оптимизации размещения средств видеонаблюдения в трехмерной сцене.
2. Описание математической модели видимости сложной трехмерной сцены, соответствующей параметрам сцены и конфигурации сенсоров наблюдения.
3. Разработка и реализация функции видимости объекта и сцены целиком для определенной конфигурации и параметров камер наблюдения с учетом физических свойств объектов.
4. Разработка комплекса алгоритмов оптимизации размещения камер наблюдения в трехмерной сцене с использованием различных критериев оптимизации.
5. Разработка трёхфазного алгоритма решения задачи максимизации видимости сцены с заданными ограничениями на основе эвристического алгоритма муравьиной колонии и предварительного ранжирования графа видимости.
6. Апробация разработанных моделей и алгоритмов применительно к нескольким техническим объектам различной сложности.

Объект исследования – математическое, информационное и программное обеспечение для оптимизации выбора размещения средств наблюдения на техническом объекте.

Предмет исследования – методы и алгоритмы оптимизации выбора размещения камер наблюдения с целью максимизации совокупной видимости выделенной области технического объекта в трёхмерной сцене.

Методы исследования. В диссертационной работе применяются методы системного анализа, теории графов, теории дискретной оптимизации, теории

алгоритмов, специальные подходы к разработке алгоритмов ранжирования объектов предметной области, вычислительных экспериментов и статистического анализа.

Научная новизна

1. Предложена, обоснована и исследована математическая модель видимости сложной трехмерной сцены, **соответствующая** её реальным параметрам и конфигурации сенсоров. Такой подход, **по сравнению** с существующими. принципиально **расширяет** возможности определения видимости узлов выделенной области на технических объектах.

2. Разработана и реализована функция видимости объекта и сцены целиком для определённой конфигурации и параметров камер наблюдения, которая **в отличии** от существующих решений представляет не только геометрические характеристики объектов на сцене, но также учитывает их физические свойства.

3. Разработан комплекс алгоритмов оптимизации размещения сенсоров в трехмерной сцене с несколькими критериями оптимизации (минимизации «слепых зон», максимизации покрытия выделенной области сцены и некоторые другие). Такой подход, **по сравнению** с существующими, **позволяет** решить задачу дискретной оптимизации выбора размещения сенсоров системы видеонаблюдения на технических объектах в условиях, когда традиционные методы поиска принципиально не могут быть реализованы в реальном времени и требуют существенных вычислительных затрат.

4. **Предложен трёхфазный алгоритм** решения задачи максимизации видимости выделенных зон или наблюдаемых объектов на технических объектах с заданными ограничениями на основе эвристического алгоритма муравьиной колонии, предварительного ранжирования вершин графа видимости и вычислении предложенной функции видимости объекта и сцены целиком.

Практическая значимость. Разработанные алгоритмы и методы, а также их реализация в программном комплексе, могут быть использованы при решении задач дискретной оптимизации выбора размещения камер наблюдения на

технических инфраструктурных объектах, промышленных предприятиях с целью максимизации совокупной видимости множества объектов на выделенной области.

Разработанный программный комплекс «Конфигуратор промышленных помещений и объектов инфраструктуры», предназначен для использования организациями, оказывающими услуги проектирования и установки систем видеонаблюдения, с целью поддержки принятия решения по выбору размещения сенсоров наблюдения. Полученные с применением эвристических алгоритмов поиска и предварительного ранжирования результаты способствуют повышению эффективности выбора оптимальной конфигурации сенсоров видеонаблюдения на наблюдаемом объекте. Таким образом, использование разработанного программного комплекса может способствовать поддержанию необходимого уровня безопасности на технических объектах, промышленных предприятиях и критически важных объектах.

Реализация и внедрение результатов работы. Предложенные в работе модели и алгоритмы реализованы в виде программного комплекса «Оптимизации размещения камер наблюдения с максимизацией видимости выделенных зон или наблюдаемых объектов», получено свидетельство о регистрации программы для ЭВМ.

Результаты работы использованы при выполнении НИР в рамках проектной части государственного задания Минобрнауки РФ в сфере научной деятельности – задание № 9.2108.2017/ПЧ.

Результаты работы используются в учебном процессе Института прикладных информационных технологий и коммуникаций ФГБОУ ВО «Саратовский государственный технический университет имени Гагарина Ю.А.».

Разработанные модели и алгоритмы процесса поддержки принятия решения по выбору размещения камер видеонаблюдения, а также их реализация в программном комплексе «Конфигуратор промышленных помещений и объектов инфраструктуры» используются компанией ООО ЧОО «Волга Плюс» при проектировании, установке и управлении системами видеонаблюдения на промышленных предприятиях.

Достоверность и обоснованность результатов работы определяется корректной постановкой задач, применением методов системного анализа, теории графов, теории оптимизации, эвристических алгоритмов поиска, методов математической статистики. Результаты исследования подтверждены вычислительными экспериментами и результатами внедрения.

Положения, выносимые на защиту:

1. Разработанный подход определения узлов сетки, наложенной на выделенную область технических объектов на основе формальной модели ориентированного графа видимости сложной трехмерной сцены, соответствующего параметрам сцены и конфигурации сенсоров.
2. Разработанная и обоснованная функция видимости объекта для определённой конфигурации камер и параметров 3D-сцены, учитывающая физические свойства технических объектов (источники освещения, тени от источников света, параметры сенсоров, шумы и т.п.).
3. Разработанный комплекс алгоритмов для оптимизации выбора размещения сенсоров в трехмерной сцене с предложенными и обоснованными критериями оптимизации, способный решать данную задачу в реальном времени.
4. Разработанный и обоснованный трёхфазный алгоритм решения задачи максимизации видимости сцены с заданными ограничениями на основе эвристического алгоритма муравьиной колонии, предварительного ранжирования вершин графа видимости и вычисления предложенной функции видимости объекта и сцены целиком.
5. Разработанный программный комплекс «Конфигуратор промышленных помещений и объектов инфраструктуры», предназначенный для использования организациями, оказывающими услуги проектирования и установки систем видеонаблюдения, оптимизации выбора размещения сенсоров наблюдения.

Апробация результатов исследования. Результаты диссертационного исследования докладывались и обсуждались на конференциях: 26-я Всероссийская научно-практическая конференция по графическим информационным технологиям и системам КОГРАФ-2016 в рамках 9-го международного форума

информационных технологий «ITForum 2020/ИТ-Джем» (Нижний Новгород, НГТУ, 2016); Международная научно-практическая конференция «Проблемы управления в социально-экономических и технических системах» (Саратов, СГТУ, 2016); V научно-исследовательская конференция аспирантов и молодых ученых «BRINGING SCIENCE TO LIFE» (Саратов, СГТУ, 2016); Международная научная конференция «Информационно-коммуникационные технологии в науке, производстве и образовании (ICIT-2016) (Саратов, СГТУ, 2016); XXX - Международная научная конференция «Математические Методы в Технике и Технологиях» (ММТТ- 30) (Саратов, СГТУ, 2017); XIII Международная научно-практическая конференция «Проблемы управления в социально-экономических и технических системах» (Саратов, СГТУ, 2017); Международная научная конференция «Информационно-коммуникационные технологии в науке, производстве и образовании (ICIT-2017) (Саратов, СГТУ, 2017); XIV Международная научно-практическая конференция «Проблемы управления в социально-экономических и технических системах» (Саратов, СГТУ, 2018); XXXI - Международная научная конференция «Математические Методы в Технике и Технологиях» (ММТТ-31) (Саратов, СГУ, 2018); Международная научная конференция «Информационно-коммуникационные технологии в науке, производстве и образовании (ICIT-2019) (Саратов, СГТУ, 2019); XV Международная научно-практическая конференция «Проблемы управления в социально-экономических и технических системах» (Саратов, СГТУ, 2019); научно-практических семинарах ИнПИТ (Саратов, СГТУ, 2015-2019 гг.).

Соответствие темы диссертации требованиям паспорта специальностей научных работников.

Диссертационная работа соответствует п. 4, 9, 12 паспорта специальности 05.13.01 «Системный анализ, управление и обработка информации».

Публикации. Результаты работы опубликованы в 12 изданиях, 2 из которых являются изданиями, рекомендованными ВАК Минобрнауки РФ, 3 индексируются в базе WoS и SCOPUS, 1 – свидетельство о регистрации программы для ЭВМ.

Структура работы. Диссертационная работа состоит из введения, 4 глав, заключения, списка использованной литературы и приложений. Работа содержит 141 страницу, 30 рисунков, 2 таблицы, библиографический список из 104 наименований и 4 приложения.

ГЛАВА 1. МОДЕЛИ И АЛГОРИТМЫ РЕШЕНИЯ ЗАДАЧ ДИСКРЕТНОЙ ОПТИМИЗАЦИИ РАЗМЕЩЕНИЯ СЕНСОРОВ НАБЛЮДЕНИЯ В ТРЕХМЕРНОЙ СЦЕНЕ

1.1 Оптимизационный подход к проблемам принятия решений по оптимизации выбора размещения сенсоров наблюдения

Мировой рынок камер видеонаблюдения стремительно растет. Согласно данным исследования IMS 2018 года, показанным на рис. [1.1.1], рынок камер наблюдения, как ожидается, вырастет в 1,5 раза или более в ближайшие пять лет. Это связано с тем, что камеры наблюдения используются не только для предотвращения и раскрытия преступлений или управления трафиком. Сейчас они необходимы для производства сборочных конвейеров или наблюдения за стихийными бедствиями [22, 23], также комплексные системы видеонаблюдения значат не маловажную роль и в оборонно-промышленной сфере для обеспечения безопасности технических объектов, промышленных предприятий и критически важных объектов (АЭС, ГРЭС, ГЭС и т.п.). С развитием технологий обработки изображений больших данных стало возможным не только наблюдать за изображениями с сенсоров наблюдения, но и извлекать из них необходимые данные [24].

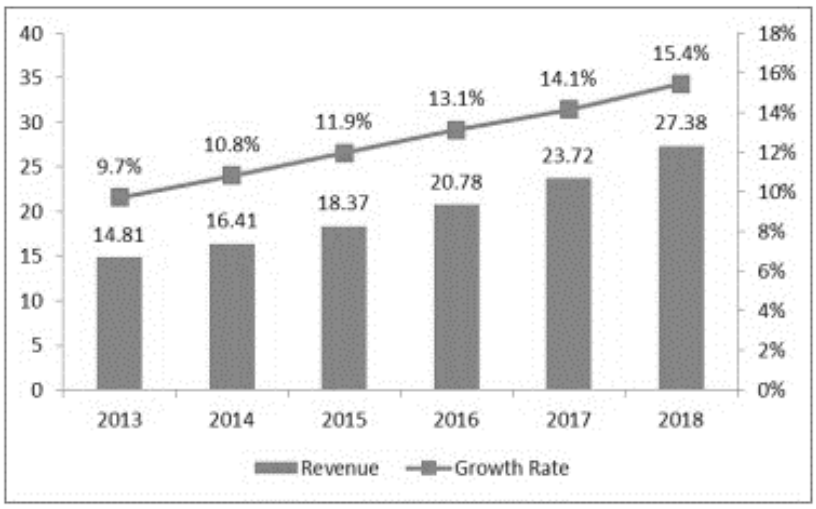


Рисунок 1.1.1 – Прогнозирование размера рынка сенсоров наблюдения

Таким образом, можно сделать вывод о перспективности и актуальности разработки комплексных систем обеспечения безопасности для решения задач, как

повседневных, так и сложных производственных, управленческих и исследовательских.

Потребность в разработки новых программных продуктов проектирования систем безопасности, также обусловлена возрастающим спросом на комплексные системы проектирования с возможностью выбора оптимального размещения сенсоров наблюдения. Если размещение камер неэффективно, даже со многими установленными камерами, эффект может быть неудовлетворительным. Для эффективного размещения камер наблюдения несколько исследований [25-30] исследовали проблему оптимального размещения камер. Проблема оптимального размещения камеры, иногда называемая проблемой развертывания сети камер, определяется как адекватное размещение камер для максимального охвата при определенных условиях [6, 26]. Эта задача оптимального размещения камеры состоит в нахождении минимального количества камер, удовлетворяющего определенному охвату, или нахождении максимального охвата с заданным количеством камер [25].

Интерес к проблеме оптимального размещения сенсоров наблюдения зависит и от разнообразия визуальных датчиков на рынке видеонаблюдения. С каждым годом появляется десятки компаний, которые предлагают свои разработки в данной области, в том числе сенсоры видеонаблюдения.

В существующих исследованиях наиболее рекомендуемое решение проблемы оптимального размещения камеры основано на двоичном целочисленном программировании (BIP) [5]. Однако из-за NP-полной(трудной) характеристики проблемы оптимального размещения камеры трудно найти решение для сложной реальной проблемы с использованием BIP. BIP предлагает глобальное оптимальное решение, однако исследования, основанные на BIP, корректно работают только на задачах с ограниченными, простыми условиями задачи [25].

Известно, что задача оптимального расположения камер наблюдения относится к задачам теории оптимизации и является NP-полной (трудной). Такая оценка достигается за счет экспоненциального роста количества комбинаций расположения сенсоров, что позволяет использовать комбинаторные методы теории графов. В такой постановке решением данной задачи является алгоритм полного перебора,

но для выполнения вычислений на графах большой размерности требуется большое количество операций, экспоненциальный рост времени выполнения просчета сцены [27].

Поскольку проблема оптимального размещения камеры является NP-полной [28], большинство существующих исследований сосредоточились на поиске эффективных и эффектных алгоритмов аппроксимации, а не на поиске оптимального решения.

Предложенные в современных исследованиях алгоритмы аппроксимации решает задачу непосредственно при высоком разрешении требуемого уровня, с одной стороны, такой подход на выходе дает оптимальное решение, с другой стороны, при использовании входных данных высокого разрешения в 10 раз увеличивается время просчета сцены и не позволяет использовать данные алгоритмы в реальном времени.

В качестве решения задач дискретной оптимизации размещения сенсоров наблюдения предлагаются эвристические или генетические алгоритм поиска, что в значительной мере позволяет уменьшить сложность алгоритма выполнения просчета сцены, данные алгоритмы является основой диссертационного исследования.

1.1.1 Используемые понятия и определения теории оптимизации

В основе современных методов решения задач дискретной оптимизации размещения сенсоров наблюдения лежат принципы оптимизации, позволяющие сконструировать систему и управлять ею наилучшим образом. Математической основой решения указанных задач является теория оптимизации.

Оптимизация — задача нахождения экстремума (минимума или максимума) целевой функции в некоторой области конечномерного векторного пространства, ограниченной набором линейных и/или нелинейных равенств и/или неравенств [29-31].

Понятие точки минимума, либо максимума требует уточнения. Точка $\vec{x} \in X$ называется:

Точкой локального минимума функции f на множестве X, или локальным решением задачи (1.1.1), если существует такое число $\varepsilon > 0$, что $f(\vec{x}^*) \leq f(\vec{x})$ при всех $\vec{x}$ таких, что $\left\| \vec{x} - \vec{x}^* \right\| \leq \varepsilon$;

Точкой глобального минимума функции f на множестве X, или глобальным решением задачи (1.1), если $f(\vec{x}^*) \leq f(\vec{x})$ при всех $\vec{x} \in X$.

Глобальное решение является одновременно и локальным, обратное – неверно.

При этом $f(\vec{x})$ будем называть целевой функцией, X – допустимым множеством, любой элемент $\vec{x} \in X$– допустимой точкой задачи (1.1.1)

Для того чтобы использовать математические результаты и численные методы теории оптимизации для решения конкретных задач, необходимо установить границы подлежащей оптимизации системы, определить количественный критерий, на основе которого можно произвести анализ вариантов с целью выявления "наилучшего", осуществить выбор внутрисистемных переменных, которые используются для определения характеристик и идентификации вариантов, и, наконец, построить модель, отражающую взаимосвязь между переменными. Данная последовательность действий составляет содержание процесса постановки задачи оптимизации [22-24].

Общая постановка задачи оптимизации.

Под оптимизационной задачей понимается задача, в которой необходимо найти решение, в некотором смысле наилучшее или, как говорят оптимальное. Оптимизационных задач достаточно много, и они могут иметь весьма разнообразный характер. Однако постановка, но не решение, всех оптимизационных задач имеет сходные черты [32, 33].

Во-первых, при постановке оптимизационной задачи указывается исходное множество вариантов (решений) и выбирается оптимальное решение. Это исходное множество решений называется пространством решений, назовем его X.

Во-вторых, некоторые решения априорно отвергаются в качестве возможных оптимальных решений. Т.е. на пространстве решений задаются ограничения, которым должны удовлетворять оптимальные решения. Эти ограничения выделяют в пространстве решений X некоторое подмножество S тех решений, которые удовлетворяют заданным ограничениям D. Это множество называется множеством допустимых решений.

В-третьих, указывается принцип сравнения двух любых допустимых решений с тем, чтобы можно было выяснить, какое из них лучше (оптимальной) в интересующем нас аспекте. Как правило, этот способ сравнения задается с помощью критерия оптимальности. Критерий оптимальности представляет собой отображение (т.е. функцию), определенное на множестве решений и принимающее в качестве значений вещественные неотрицательные чисел.

1.1.2 Методы решения однокритериальной и многокритериальной задач дискретной оптимизации

Постановка однокритериальной задачи оптимизации в зависимости от смысла выбранного критерия "наилучшего" системы, либо способу управления всегда соответствует "минимальное" или "максимальное" значение показателя, характеризующего качество функционирования системы, либо варианта управления [34].

Однокритериальная задача оптимизации формулируется следующим образом. Заданы множество $X = \{x_1, x_2, \dots, x_n\}$, характеризующее, например, некоторые внутренние параметры системы и функция $f(\vec{x})$, определенная на этом множестве и характеризующая качество функционирования системы. Требуется найти такие значения $\vec{x}^* = \{x_1^*, x_2^*, \dots, x_n^*\}$внутренних параметров, при которых функция $f(\vec{x})$ принимает максимальное или минимальное значение [30, 35].

Задача оптимизации *либо*
$$\left.\begin{array}{l} \text{а) } f(\vec{x}) \Rightarrow min, \vec{x} \in X \\ \text{б) } f(\vec{x}) \Rightarrow max, \vec{x} \in X \end{array}\right\} \qquad (1.1.1)$$

Рассматриваются конечномерные задачи оптимизации, т.е. задачи, допустимое множество решений которых лежит в евклидовом пространстве R^n.

Решения задачи (1.1.1), т.е. точки $\vec{x}^*$ минимума или максимума функции f на множестве x, называют ещё точками экстремума, а задачи (1.1.1) – экстремальными задачами.

Задача $+f(\vec{x}) \Rightarrow \underset{\vec{x}\in x}{max}$ эквивалентна задаче $-f(\vec{x}) \Rightarrow \underset{\vec{x}\in x}{min}$.

Это обстоятельство дает возможность переносить результаты, полученные для задачи минимизации, на задачи максимизации, и наоборот. К однокритериальным задачам оптимизации относятся: производственные и транспортные задачи.

В случае если имеется несколько целевых функций, которые необходимо максимизировать или минимизировать, то появляется

Задача 1.1.2 вида $f_i(\vec{x}) \Rightarrow max(min)$

$x\ D$

где $i>1$, D – допустимое множество, R_n - множество действительных чисел, а $f_i(\vec{x})$ – гладкие функции на D, называется **задачей многокритериальной оптимизации [30].**

Главная возникающая сложность при решении многокритериальной задачи в неоднозначности оптимального решения: в точке, где один из критериев достигает своего максимума(минимума), другой может быть далек не только от экстремума, но и от какой-либо приемлемой величины. При решении многокритериальной задачи оптимизации [30, 34-35] используются различные методы:

Метод обобщенного критерия. Метод перехода от нескольких критериев $f_1, f_2, \ldots f_m$ к одному, задаваемому новой функцией $z = \sum_{j=1}^{m} a_j f_j$, называется свертkой или методом обобщенного критерия. Числа αj называются весовыми коэффициентами. Чем больше α_j, тем больший «вклад» вносит j-й критерий в обобщенный критерий z. Иногда требуют, чтобы $z = \sum_{j=1}^{m} a_j = 1$.

Метод приоритетов. Метод приоритетов решения многокритериальных задач применяется в том случае, когда критерии f_i упорядочены по их относительной важности.

На первом шаге решения задачи отбирают множество исходов, которые имеют максимальную оценку по важнейшему критерию. Если исход единственный, то он и является оптимальным. Если же исходов несколько, то среди них выбирают те, которые имеют максимальную оценку по второму по важности критерию. Если опять исходов несколько, то процесс повторяют для следующих критерий.

Метод идеальной точки. Метод идеальной точки является «геометрическим» методом для многокритериальных задач.

В рамках диссертационного исследования при решении оптимизационной задачи размещения камер наблюдения на технических объектах, промышленных предприятиях перечисляется множество критерий, однако, решается как однокритериальная задача.

1.2 Подходы к оптимизации размещения сенсоров системы наблюдения

В исследовании рассматривается задача оптимизации размещения средств наблюдения различного типа в трехмерной сцене. Под трехмерной сценой понимается виртуальная модель реальной площадки с размещенными на ней объектами, за которыми необходимо установить наблюдение. Сама сцена имеет препятствия для средств наблюдения либо в виде геометрических трехмерных объектов, либо в виде поверхностей. При выборе позиций для размещения средств наблюдения (будем далее называть их камерами или сенсорами) на сцене образуются так называемые «слепые зоны», которые должны быть минимизированы. Решение задачи в этой постановке основывается на критерии оптимальности с точки зрения количественных характеристик слепых зон на сцене. В данной работе не рассматривается задача подбора оборудования для наблюдения по его характеристикам, однако эта проблема является актуальной и ее решение требует отдельного внимания [4].

Кратко опишем основные существующие в настоящее время подходы. Современные исследования [16,17 ,19 ,26] выдвинули гипотезу о непрерывном пространстве, которое упрощается как двумерная (2D) сетка точек. Здесь точки сетки являются дискретными точками на осях на минимальное расстояние, которое учитывает пространственную частоту дискретизации ($f_x = f_y : \Delta = 1/f_x = 1/f_y$) после

упрощения реального пространства в 2D [16]. При моделировании фиксированной местности, используя описанный выше метод, решение качество камеры оптимальным размещением проблем с более высоким разрешением, как правило, лучше, чем с меньшим разрешением, потому что соотношение реальной местности, что отражается в области моделирования с высоким разрешением (большие f_a, небольшие Δ) с использованием большего числа точек сетки выше, чем с низким разрешением (малые f_a,, большие Δ), используя меньшее количество точек сетки. Таким образом, Hörster и Lienhart [17] утверждали, что необходимо учитывать большое количество точек сетки.

Одной из частых проблем при решении подобных задач, а именно оптимизации размещения сенсоров, является необходимость регулярного выполнения перекалибровки устройств с целью определения их параметров. С подобными вопросами помогают справиться многоэтапные алгоритмы самокалибровки [10], позволяющие вычислять матрицу камеры, а также ее основные характеристики на основе реконструкции поверхности. Применение фильтрующих алгоритма в совокупности с анализом результатов предыдущих калибровок позволяет «экстраполировать» характеристики камеры, если необходимо добиться быстрого расчета в определенный момент времени [9].

Динамический характер объектов наблюдения требует отдельного подхода [11]. Позиции камер могут вычисляться как на плоскости, так и в трехмерном пространстве на основе данных о перемещениях объектов на текущий момент времени. При этом направление движения заранее обычно считается известным. Развитие такого решения позволяет определять калибровочные характеристики для фиксированных камер на основе данных о перемещениях в зонах высокой активности [36].

В случаях, когда расположение объектов-целей на сцене заранее неизвестно и их необходимо обнаружить путем обследования всей сцены, средства наблюдения должны быть размещены так, чтобы максимально покрыть всю область наблюдения, минимизировав количество «слепых зон» - областей, которые не наблюдаются ни одним сенсором. Сцена при этом может иметь сложную геометрическую

структуру, что должно учитываться при вычислениях. Одним из эффективных способов выполнения данной задачи является решение оптимизационной задачи эффективного размещения средств наблюдения, которое основывается на приближенных методах поиска оптимальных решений [12].

При отсутствии информации об объектах наблюдения необходимо получать её на основе реальных данных, которые получены в процессе сканирования сцены сенсорами. Первой задачей в таком случае является определение расстояния между целью и средством наблюдения. В этом случае используемый алгоритм определения оптимального положения камер должен учитывать комплексное расположение средств наблюдения в сцене, с целью минимизации ошибки определения дистанций [13]. Когда система, управляющая средствами наблюдения, не имеет предварительной информации о размещении целей и препятствий в сцене, трехмерная реконструкция сцены является необходимым решением для последующей оптимизации положения камер [15, 38].

Когда наблюдение происходит за сценами, имеющими большую площадь, требуется большое количество камер (несколько десятков или сотен). Сама размерность задачи не позволяет руководствоваться традиционными способами решения. В таких случаях удобно использовать трехмерное моделирование сцены, на которой предстоит решать задачу, и вычисления производятся на этой модели. В работе [36] предложено именно такое решение, которое для трехмерной модели здания формирует оптимальное положение набора из 400 камер наблюдения в целях максимизации площади покрытия зоны наблюдения.

Возможны ситуации, когда необходимо произвести эту же работу на открытой местности, имеющей не менее сложные геометрические параметры. В работе [39] рассматривается способ определения оптимальной конфигурации системы видеомониторинга леса для обнаружения и определения местоположения лесных пожаров. Такие системы применяются на практике сравнительно недавно, их актуальность возрастает, поскольку проблема является одной из наиболее серьезных и нерешенных на сегодняшний день. В этом же русле решается задача в работе [40],

дающей описание алгоритма нахождения позиций для размещения средств наблюдения на холмистом ландшафте, что позволяет минимизировать их количество.

Одним из подвидов задач оптимизации размещения подобных сенсоров, является задача оптимального их размещения на границе контролируемой области. Благодаря использованию комбинаторных методов, данный подход позволяет использовать вероятностные модели локализации проникновения на охраняемую территорию [41].

При рассмотрении слепых зон, как правило, рассматриваются соответствующие задачи определения слепых зон транспортных средств, вероятности нахождения объектов в данных зонах в реальном времени на основе изображения со средств видео-фиксации [42, 43].

В первой главе исследования предлагается описание алгоритма вычисления «слепых зон» для трехмерной сцены и заданной конфигурации расположения камер наблюдения, описывается структура программного пакета, в котором реализуется этот алгоритм и результатов проведенного вычислительного эксперимента. Некоторые предварительные идеи и определения, используемые нами, представлены в работах [43,44].

1.3 Постановка задачи диссертационного исследования

В исследовании перечислены различные задачи оптимизации размещения камер наблюдения, которые будут отличаться как выбором алгоритма, так и сложностью их решения. Каждая из таких задач предполагает нахождение некоторого множества положений камер, которое является оптимальным с точки зрения предлагаемого для каждого случая критерия видимости.

- определение положение камер, при котором достигается максимальная совокупная видимость одного объекта;
- определение положения камер, при котором достигается максимальная совокупная видимость множества объектов на выделенной области;
- определение «слепых зон» для текущей конфигурации камер;
- минимизация «слепых зон»;

- максимизация покрытия выделенной области;
- минимизация количества камер наблюдения.

В рамках анализа подходов к задаче дискретной оптимизации размещения сенсоров наблюдения отмечено, что при решении данной задачи, по одному из предложенных критериев оптимизации, а именно минимизация «слепых зон», существующие решения при выполнении на графах большой размерности требуют большой вычислительной мощности следовательно, не могут быть выполнены в реальном времени. Также существующие подходы направленны на оптимизацию размещения камер на всей области охраняемого объекта, без выделения конкретной(приоритетной) зоны, за которой необходимо установить наблюдение.

В рамках исследования оптимизационная задача размещения камер наблюдения на технических объектах, промышленных предприятиях решается как однокритериальная задача, в первой главе, с целью определения положения камер, при котором достигается минимизация «слепых зон» на выделенной области в реальный момент времени.

Применение комплексных систем обеспечения безопасности с возможностью оптимизации выбора размещения сенсоров наблюдения на технических объектах, промышленных предприятиях и критически важных объеках (АЭС, ГРЭС, ГЭС и т.п.), позволит выполнять поддержку принятия решений в реальном времени, а также оптимизировать выбор размещения сенсоров наблюдения.

Неформальная постановка задачи. В общем виде неформальная постановка задачи может быть описана следующим образом. На сцене, имеющую сложную геометрию, располагаются объекты-цели, которые необходимо контролировать с помощью средств наблюдения. Камеры могут располагаться в ограниченном наборе позиций внутри сцены. Препятствия на сцене, ее рельеф могут создавать «слепые зоны» для средств наблюдения, причем данные зоны могут быть «слепыми» для сенсоров различного типа: аудиальных, визуальных, датчиков движения. Главной задачей является вычисление позиций сенсоров, позволяющих минимизировать размеры таких зон за счет выбора позиций для средств наблюдения.

Эта задача является двойственной по отношению к задаче максимизации «обозреваемости» сцены.

Считаем, что имеется произвольное количество объектов, которые необходимо наблюдать с помощью определенного числа камер. В первой части исследования мы не рассматриваем вопросы идентификации объектов наблюдения, определения траектории их движения, определения их местоположения. Считаем, что их совокупность задается набором возможных позиций на сцене, в частности координатами на плоской сцене. Последние определяются исходя из трехмерной модели, позволяющей определить, может ли объект наблюдаться в определенной позиции камерой, находящейся в одной из фиксированных точек (см. рис. 1.3.1).

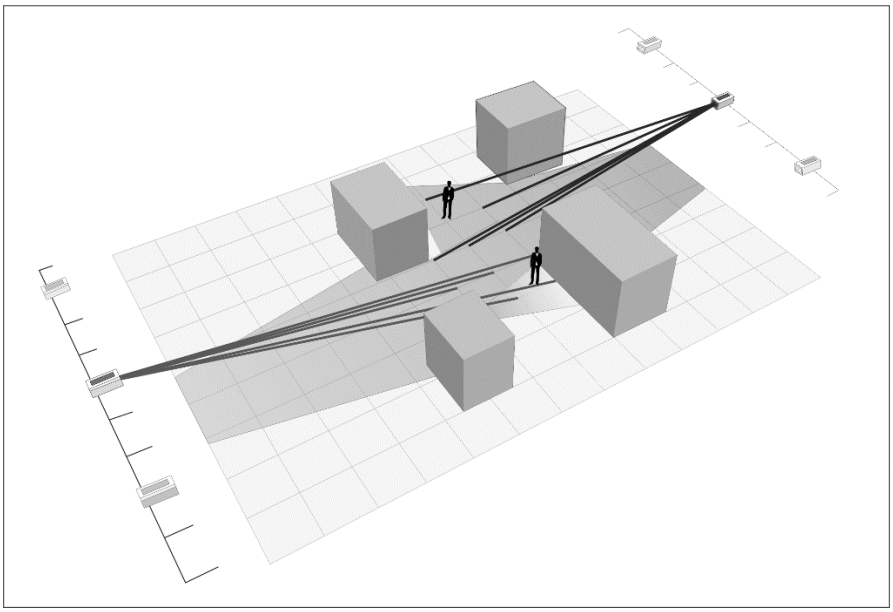


Рисунок 1.3.1 - Наблюдаемые объекты на трехмерной сцене

Предполагаем, не умаляя общности, что на обозреваемую трехмерную сцену наложена сетка (мы рассматриваем плоскую, но это не умаляет общности подхода), имеющая определенный шаг и размеры, узлы которой рассматривается как места возможного размещения объектов-целей, на сцене имеются препятствия, под которыми узлы сетки не рассматриваются. Такой прием часто используется в задачах оптимизации расположения камер [см., например, 20]. Камеры могут находиться не в любом положении, а только в конечном множестве дискретных позиций, определяемых шагом сетки, предустановленными параметрами и ограничениями. Такая постановка позволяет рассматривать широкий класс прикладных задач

оптимизации расположения камер как в закрытых помещениях, так и на достаточно обширных площадках (например, промышленных).

Далее задача поиска расположения камер сводится к оптимизационной задаче на специальным образом построенном графе, наложенным на рельеф сцены.

Формальная постановка задачи. В процессе формализации не учитываются параметры объектов и тех предметов на сцене, которые могут формировать слепые зоны для камер наблюдения. Задача рассматривается в варианте, когда определение наблюдаемости сводится к плоской карте, на которой наблюдаемость узла определяется возможностью проведения прямой линии от сенсора к позиции узла, не проходящей через элементы–препятствия на сцене. При этом данная линия может обозначать как линию видимости, так, например, и «слышимость» данного узла аудиальным устройством. Позиция сенсора размещена в трехмерной сетке, а позиции, за которыми необходимо вести наблюдение – в плоской сетке.

Фактически задача оптимального расположения сенсоров решается в соответствии с определением степени наблюдаемости участков плоской карты, в которых могут располагаться объекты (см. рис. 1.3.2). Далее мы предлагаем определение взвешенного графа, соответствующего описанию сцены и возможным положениям объектов и сенсоров.

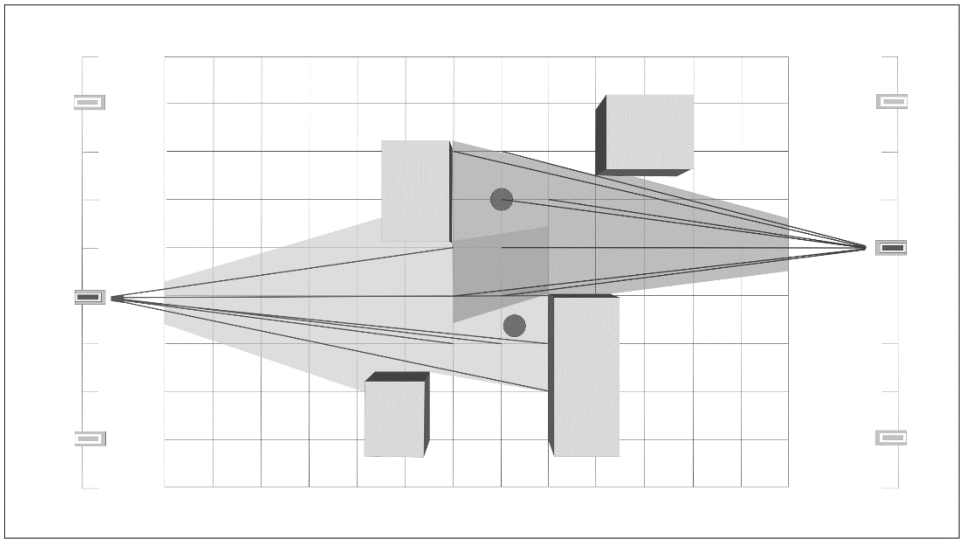


Рисунок 1.3.2 - Переход к «плоской» сцене

Определим ориентированный взвешенный граф наблюдаемости позиций объектов следующим образом (в этом определении предполагается наличие k сенсоров).

Графом наблюдаемости позиций объектов (далее просто граф наблюдаемости) является система

$$G= (V \cup A, E, f), \text{ где} \qquad (1.3.1)$$

V – множество вершин, соответствующее узлам сетки, возможным позициям объектов, в которых они могут располагаться, $A = \cup\{A_i\}_{i=1, k}$ – объединение набора множеств вершин, соответствующих узлам трехмерной сетки, в которых могут располагаться сенсоры. Каждое множество A_i – является множеством вершин графа, которые соответствуют узлам сетки, в которых может располагаться i–й сенсор (каждый сенсор имеет набор параметров, их уточнение является отдельным вопросом).

E – множество ребер графа, которое определено следующим образом. Вершина $v \in V$ соединяется с вершиной $u \in A_i$ в том и только в том случае, если узел сетки на сцене, соответствующий вершине v, обозревается соответствующим сенсором в позиции u.

Пусть $|V|=n$, $|A|=k$. То есть на сцене имеется n позиций, в которых могут располагаться объекты, и k сенсоров, которые могут находиться в вершинах, соответствующих их возможным позициям в множествах A_i.

Вершина $v \in V$ наблюдается i–ым сенсором из позиции u, если $(v, u) \in E$ и $u \in A_i$. Принадлежность вершины u множеству A_i указывает на конкретный сенсор, позволяющий наблюдать за объектом в позиции v. Если несколько объектов находятся в одном узле нашей сетки, они будут представлены одной вершиной графа наблюдаемости G.

Для каждого ребра графа определяется его вес следующим образом:

$f : E \to R$, функция меры наблюдаемости позиции.

Функция $f(e)$ принимает значение в множестве вещественных чисел. Конкретный вид функции $f(e)$ может быть определен, исходя из геометрических атрибутов

сцены, экспертных оценок наблюдаемости или оценок, основанных на процедурах калибровки сенсора, параметрах освещенности сцены. В данной работе используется вес ребра равный 1, если вершина видна из соответствующей позиции сенсора. В более общем виде вычисление значения функции наблюдаемости может быть основано на интегральном показателе, зависящем от положения сенсора в трехмерной сцене, фокусного расстояния камеры, вектора направления ее обзора, ширины и высоты области обзора, расстояния между камерой и наблюдаемой позицией, характеристиках освещения.

1.3.1 Определение конфигурации сенсоров и конфигурации объектов

Множество вершин $\bar{a} = \{a_1, a_2, \dots a_k\}$, где $\underset{1 \le i \le k}{\forall} a_i \in A_i$, назовем конфигурацией расположения сенсоров. Множество всех возможных конфигураций расположения сенсоров обозначаем как $\overline{A}$.

Назовем любое подмножество *V* конфигурацией расположения объектов на сцене, то есть $\bar{v} = \{v_{1,} v_2, \dots, v_l\}$, где *l* – количество объектов в конфигурации. Множество всех таких конфигураций является множеством всех подмножеств *V* и обычно обозначается как *P(V)*.

Для любого объекта, находящегося в позиции, соответствующей вершине *v*, определяем наблюдаемость объекта при конфигурации сенсоров $\bar{a}$ как функцию $F(v, \bar{a}) = \sum_{1 \le i \le k} f(v, a_i)$. То есть, функция наблюдаемости объекта при определенной конфигурации камер является отображением следующего вида:

$$F\colon V \times \bar{A} \rightarrow R \qquad (1.3.2)$$

Таким образом, имеется *(k+1)* – дольный граф, в котором могут быть соединены только вершины, соответствующие позициям наблюдаемых объектов с вершинами, соответствующими позициям сенсоров. Если позиция не наблюдается сенсором из соответствующего положения, эти две вершины не соединены ребром. Если же вершина наблюдается, существует ребро, вес которого определяет функцией наблюдаемости этой позиции на реальной сцене.

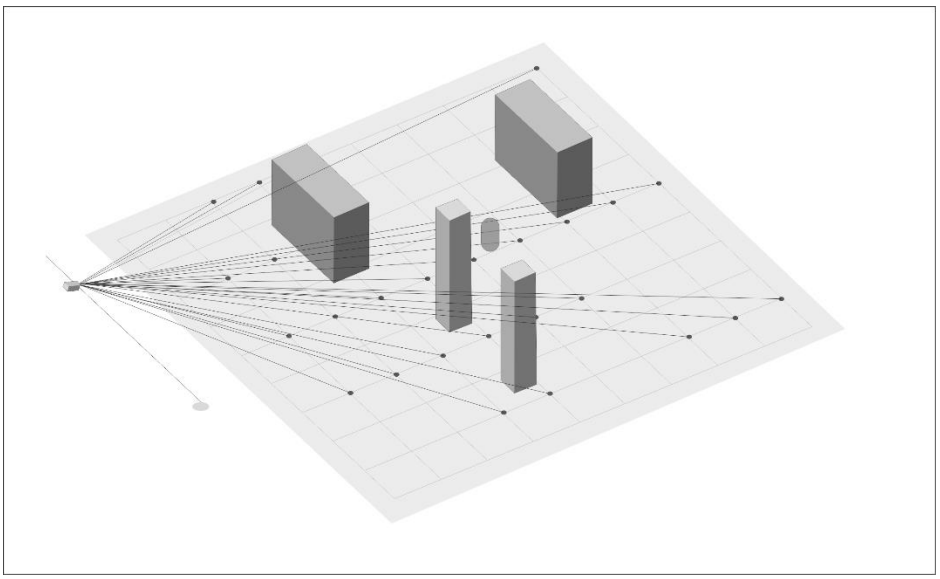


Рисунок 1.3.3 - Пример графа наблюдаемости для камеры в определенном положении

Целью исследования является определение и минимизация слепых зон на сцене. Минимизация достигается размещением сенсоров в некотором множестве возможных их положений, которое минимизирует количество позиций недоступных для наблюдения. Дадим формальное определение задачи в терминах графа видимости, определенного выше.

1.4 Алгоритм минимизации «слепых зон» в трехмерной сцене

Формализация задачи минимизации «слепых зон». Слепой зоной для данной конфигурации сенсоров наблюдения $\bar{a} \in \overline{A}$ называется множество вершин $B(\bar{a}) \in V$ графа наблюдаемости G, определенного следующим образом

$$B(\bar{a}) = \{\ v \in V | F(v, \bar{a}) = \ 0\} \quad (1.4.1)$$

Задача минимизации слепых заключается в поиске конфигурации камер наблюдения, для которых количество узлов сетки вне зоны наблюдения будет минимальным:

$$|B(\bar{a})| \to \min_{\bar{a} \in \bar{A}} \qquad (1.4.2)$$

Псевдокод алгоритма. По определению графа наблюдаемости вершина $v \in V$ сцены принадлежит слепой зоне для данной конфигурации $\bar{a}$ только в том случае, когда $F(v, \bar{a}) = 0$. То есть для всех $a_i \in \bar{a}$ должно выполняться условие:

$$[|\{v|(v, a_i) \in E\}| = 0] \qquad (1.4.3)$$

По этой причине при вычислении размера слепой зоны для заданной конфигурации используется алгоритм, который не выполняет проверку видимости всех позиций сцены для всех сенсоров. Значительное сокращение времени работы алгоритма происходит за счет того, что он осуществляет проверку всех позиций сцены только для одного сенсора (выбирается исходя из их экспертного ранжирования или случайным образом). Для остальных полученная при первом прогоне слепая зона «сужается», из нее удаляются те позиции *v*, для которых функция меры наблюдаемости *f* не равна нулю, как это должно быть в соответствии с (1.4.4).

В общем случае при полной проверке видимости всех позиций для каждого сенсора сложность алгоритма поиска слепой зоны для одной конфигурации имеет порядок $n \times k$, где k – количество сенсоров наблюдения, $n = |V|$. То есть зависимость имеет линейный вид от k. Результаты и особенности вычислительного эксперимента описаны далее.

Входные данные: Граф $G = (V \cup \{A_i\}, E, f)$, конфигурация сенсоров $\bar{a}$.

Выходные данные: *к*оличество вершин N в слепой зоне для конфигурации

Описанная ниже функция возвращает число вершин графа наблюдаемости *G*, которые входят в слепую зону при конфигурации $\bar{a}$. Для хранения вершин графа, которые входят в слепую зону, используется структура данных «список».

```
Function GetBlindArea(G, ā): Integer
 N := |V|
 //Текущие вершины в слепой зоне
  BlindNodeList.Create
 //Добавляем все вершины сцены в слепую зону
 for всех вершин v ∈ V do BlindNodeList.Add(v)
 for всех сенсоров a конфигурации ā  do begin
   If Not BlindNodeList.IsEmpty then
     for всех вершин v ∈ BlindNodeList do begin
//удаляем вершину v из слепой зоны, если она видима
       if (v,a) ∈ E then BlindNodeList.DeleteNode(v)
```

```
    end
  end
  N := BlindNodeList.Size
  Return(N)
end
```

1.4.1 Алгоритм последовательного сокращения слепых зон для одной конфигурации сенсоров наблюдения

Для моделирования трехмерной сцены и решения задачи оптимизации размещения сенсоров разработан программный комплекс, который позволяет строить и обновлять в реальном времени граф видимости, а также решать формализованную задачу (предусмотрены варианты получения оптимального решения или приближенного решения, полученного эвристическими алгоритмами).

- Инструментарий позволяет строить виртуальную сцену большой сложности и содержит следующие типы объектов:

- Цель. Представляет собой объект, расположенный на сцене, установление наблюдения за которым и является целью данного комплекса.

- Препятствия. Имитируют реальные препятствия на сцене, мешают установлению визуального контакта камеры с объектом-целью. Могут иметь геометрическую форму различной сложности.

- Средства наблюдения. Могут иметь различные алгоритмы работы, основываясь на различных функция наблюдения.

Программный комплекс содержит в себе три группы классов:

Классы описания – содержат описание сущностей и объектов сцены (Рис. 1.4).

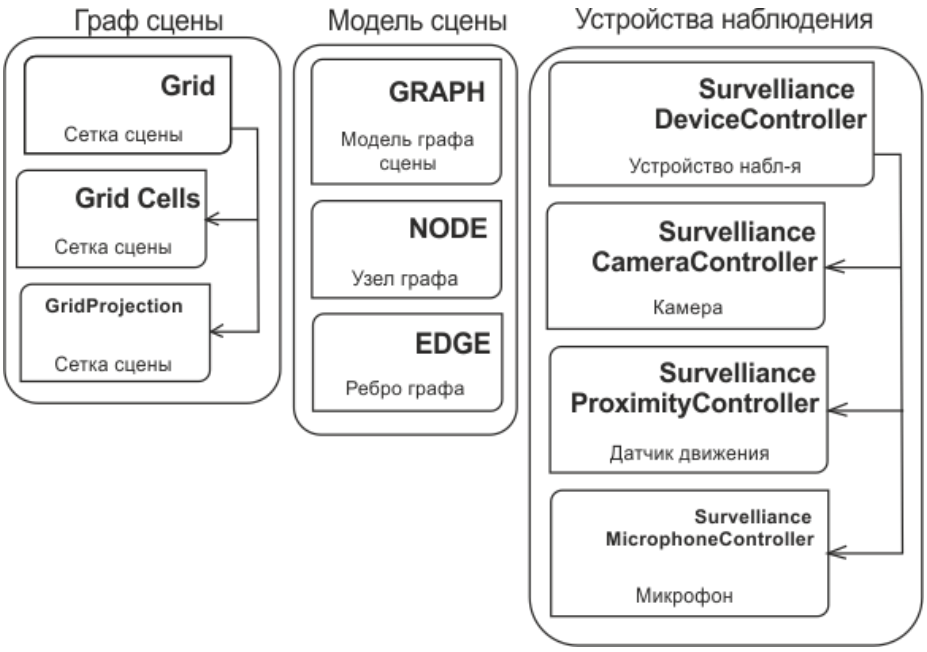


Рисунок 1.4.1 - Структура основных классов комплекса

Программный комплекс позволяет оптимизировать расположение сенсоров в трехмерной сцене с целью минимизации «слепых зон» (Рис. 1.4.2). При вычислении областей наблюдения сенсоров учитываются установленные параметры средств наблюдения (в случае камер – их угол обзора, дальность видимости, фокусное расстояние), что позволяет добиться возможности тонкой настройки и высокой точности расчетов.

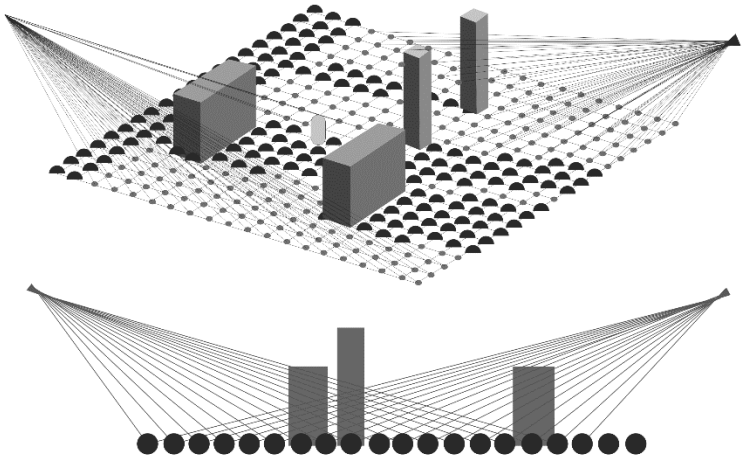


Рис 1.4.2 - Применение двух сенсоров наблюдения для определения «слепых зон» сцены

При выполнении расчетов вычисляются дополнительные параметры как самой сцены, так и самого процесса обработки данных: процент покрытия площади, процент перекрытия областей воздействия сенсоров, время расчета текущей конфигурации.

1.4.2 Анализ результатов вычислительного эксперимента

Для оценки скорости обработки сцены с целью поиска конфигурации сенсоров с минимальной по размеру слепой зоной был проведен вычислительный эксперимент, который выполнялся на ноутбуке с процессором Intel Core i5-6300HQ, с 8GB DDR3 SDRAM. Моделирование сцены и расчет осуществлялся с помощью программного обеспечения Unity 3D 5.4. 1. В качестве сцены, которая моделировалась в эксперименте, рассматривалась модель плоской сцены с различным количеством случайно размещенных объектов–препятствий и сенсоров. Видимость объекта определяется с помощью стандартной функции RayCast, реализованной в используемом программном обеспечении.

Дадим общую формулу вычисления времени работы алгоритма поиска минимальной слепой зоны для заданной сцены и количества камер наблюдения. Введем следующие обозначения:

T_1 – время сканирования для одного сенсора сцены, на которой находится один объект–препятствие;

T_2 – время, на которое увеличивается время сканирования одной конфигурации при добавлении одного объекта–препятствия;

m – количество объектов препятствий, которые формируют граф видимости;

k – количество используемых сенсоров.

На рис. 1.8 приведены графики характера временной сложности алгоритма вычисления слепой зоны для заданной конфигурации сенсоров с помощью полной проверки всех позиций для каждого сенсора (линейная функция) и с помощью алгоритма **GetBlindArea**. На графике показано, как растет вычислительная сложность в том и в другом случае при увеличении числа препятствий m, в частности при $m=10$. Вычисления производились для случайным образом сгенерированных

сцен, которые имеют физические размеры порядка 10 тыс. м.^{2} с шагом сетки равным 1 метру.

Вычислительный эксперимент показал, что поиск слепой зоны для заданной конфигурации со случайным расположением препятствий по времени зависит от количества сенсоров (k) с видом близким к логарифмической зависимости $T_1+log(1+T_1\times(k-1)\times\alpha)$, где $T_1 = 14$ *мс* для нашей сцены. Величина параметра α определяется в ходе вычислительного эксперимента, его оптимальное значение, наиболее точно соответствующее полученным результатам, было определено эмпирическим путем и равно 2. Каждое новое препятствие увеличивает время обсчета всей сцены для одной камеры на время $T_2 = 0{,}01$ *мс*. Для решения же полной задачи необходимо перебрать все возможные конфигурации сенсоров, что потребует числа операций порядка

$$(T_1 + \log(1 + T_1 \times (k-1) \times \alpha) + T_2 \times (m-1) \times \kappa) \times \prod_{i=1}^{k} |A_i| \qquad (1.4.1)$$

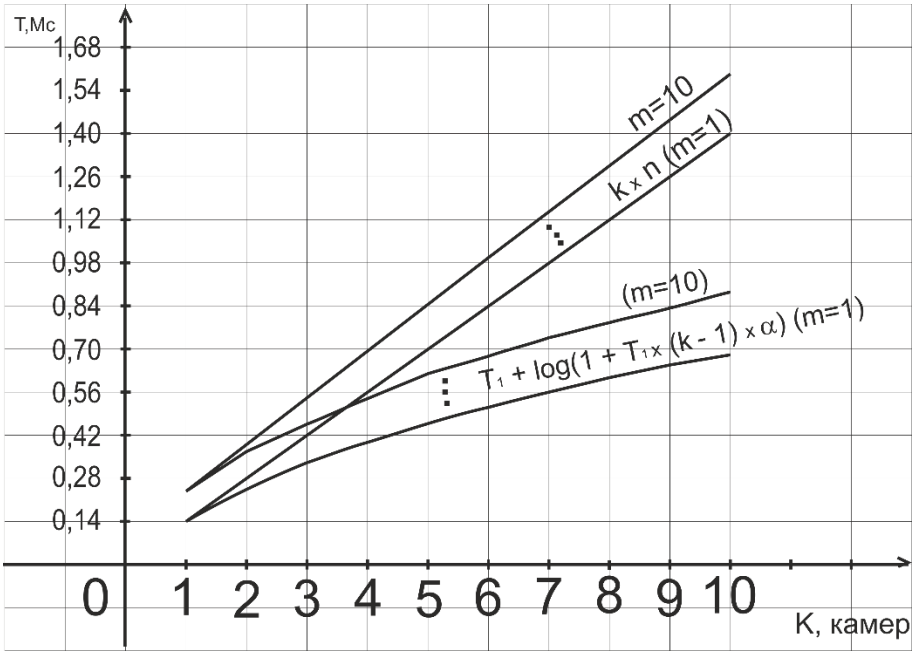


Рисунок 1.4.3 - Время выполнения проверок RayCast

Существуют конфигурации сенсоров и объектов–препятствий, которые могут в худшем случае приводить к количеству операций, растущему линейным образом, но при генерации случайных позиций и геометрических размеров препятствий на

сцене при усреднении результатов вычислений справедлива логарифмическая зависимость, как это показано на рис. 1.4.3.

Известно, что задача оптимизации расположения камер в общем случае является *NP–полной* (см. об этом в [27]). Такая оценка достигается за счет экспоненциального роста количества комбинаций расположения сенсоров (последний множитель в формуле 6). Предложенный алгоритм позволяет сократить время обсчета одной конфигурации и тем самым уменьшить время решения всей задачи поиска оптимального расположения сенсоров. Для сокращения сложности задачи необходимо использовать экспертные оценки возможных положений сенсоров, что позволяет за приемлемое время получить оптимальное решение.

1.5 Выводы по главе 1

В первой главе исследования проведен анализ моделей и алгоритмов решения задач дискретной оптимизации размещения сенсоров наблюдения в трехмерной сцене, на основе которого сделаны следующие выводы.

Задача оптимального расположения камер наблюдения относится к задачам теории оптимизации, которая является NP-полной (труднорешаемой). В такой постановке решением данной задачи является алгоритм полного перебора, но для выполнения вычислений на графах большой размерности количество операций растёт экспоненциально и выполнение алгоритма становится невозможным даже для относительно небольших объектов.

Большинство существующих исследований решения проблемы оптимального размещения камер сосредотачиваются на поиске эффективных алгоритмов аппроксимации, а не на поиске оптимального решения. Предложенные подходы, использующие алгоритмы аппроксимации к решению задачи оптимизации размещения сенсоров наблюдения на выделенной области по одному из предложенных критериев оптимизации, а именно, – минимизация «слепых зон», при выполнении на графах большой размерности не могут быть выполнены в реальном времени. Существуют подходы, направленные на оптимизацию размещения камер на всей области охраняемого объекта с большой размерностью входных параметров, без

выделения конкретной (приоритетной) зоны, за которой необходимо установить наблюдение. Такие подходы позволяют получить решение близкое к оптимальному, но при использовании реальных входных данных на несколько порядков увеличивается время просчета сцены, что не позволяет использовать данные алгоритмы за приемлемое время.

В рамках предлагаемого исследования оптимизационная задача размещения камер наблюдения на технических объектах решается как однокритериальная задача с различными критериями оптимизации. Эта задача предполагает нахождение некоторого множества позиций камер, которое является оптимальным с точки зрения предлагаемого для каждого случая критерия видимости. В данной главе был выбран критерий минимизации «слепых зон». В ходе решения этой задачи представлена формализация задачи определения оптимального расположения камер с помощью определённого в работе графа видимости. Для этого графа сформулирована задача нахождения «слепых зон» на выделенной области. Предложен алгоритм их поиска, который по результатам вычислительного эксперимента существенно сокращает трудоемкость процедуры для одной конфигурации сенсоров и конфигурации объектов–препятствий на трёхмерной сцене, что позволяет решить поставленную задачу. В главе представлены результаты вычислительного эксперимента и определены временные параметры работы алгоритма расчета одной конфигурации сенсоров, используемого для поиска оптимального их расположения.

ГЛАВА 2. ИСПОЛЬЗОВНИЕ ЭВРИСТИЧЕСКИХ АЛГОРИТМОВ ПОИСКА ПРИ РЕШЕНИИ ЗАДАЧ ДИСКРЕТНОЙ ОПТИМИЗАЦИИ РАЗМЕЩЕНИЯ СЕНСОРОВ НАБЛЮДЕНИЯ

2.1 Анализ эвристических алгоритмов оптимизации размещения сенсоров наблюдения

Задача дискретной оптимизации расположения камер наблюдения в трехмерной сцене является NP-полной (труднорешаемой). Анализируя существующие подходы, для точного решения данной задачи в современных исследованиях предлагается использовать алгоритмы полного перебора, аппроксимации и эвристические алгоритмы, решая задачу непосредственно при высоком разрешении, с одной стороны, такой подход на выходе дает оптимальное решение, с другой стороны, данные алгоритмы затруднительно использовать в реальном времени.

Решение задачи оптимизации расположения сенсоров наблюдения в трехмерной сцене с помощью полного обсчёта сцены (узлов графа) приводит к образованию экспоненциальной по сложности задачи, где необходимо разместить k сенсоров в m позициях (местах размещения), что приводит к порядку числа операций m^k и невозможности получения решения задачи оптимизации размещения сенсоров в сложной трехмерной сцене в приемлемое время.

Аппроксимационные алгоритмы используются при решении NP-полных задач, к которым относится задача оптимизации размещения средств наблюдения в сложной 3D-сцене. Существующие решения данной задачи, направленны на поиск эффективных алгоритмов поиска приближённого решения, а не на поиск оптимального, что позволяет использовать данные алгоритмы в реальном времени, обрабатывать в приемлемое время достаточно сложные сцены. Однако, алгоритмы аппроксимации, в отличии от генетических и эвристических алгоритмов поиска, обеспечивают решение близкое к оптимальному за приемлемое время, дают неплохое качество решения задачи оптимизации размещения сенсоров, в заранее определенные границы времени. Алгоритмы аппроксимации хорошо себя

зарекомендовали при решении задач, для которых используются точные алгоритмы, выполняющиеся за полиномиальное время, но выполняющиеся долго.

В качестве получения достаточного результата в реальном времени, при решении задачи оптимизации расположения камер наблюдения, предлагается использовать эвристические или генетические алгоритм поиска, что в значительной мере позволяет уменьшить сложность алгоритма выполнения вычислений на графах большой размерности.

В работе [6] рассматривается метод определения лучшего расположения большого количества камер в сложных помещениях. В качестве входных параметров берется 3D модель здания, которая затем используется в работе генетического алгоритма, для поиска положений, в которых достигается наибольшее покрытие и, если необходимо, пересечение областей видимости камер. Камеры описываются 6 пространственными параметрами (3 параметра обозначают позицию в трехмерном пространстве, 3 оставшихся – поворот). Также, камеры имеют внутренние параметры, такие как увеличение, угол обзор, фокусное расстояние и т.п.

Для оценки текущего положения камеры используется функция определения качества используемого размещения. Данная функция учитывает следующие параметры:

1. Общий объект покрываемой зоны;
2. Пересечение обозреваемой зоны с обозреваемыми зонами других камер;
3. Размер обозреваемой цели в кадре должен быть приближен к определенному значению.

Построенное решение позволяет строить карту обозреваемой площади, обнаруживать слепые зоны и зоны пересечения областей видимости камер. Однако, данный алгоритм не позволяет выделять конкретные зоны, за которыми необходимо установить наблюдение.

В исследовании [16] изучается проблема оптимального расположения набора камер как узконаправленных, так и с широким углом обзора (вплоть до кругового обзора). В предлагаемом решении, рабочая область представлена в виде сетки

наложенной на карту. Предлагается компьютерный алгоритм, основанный на эвристическом алгоритме поиска для определения мест размещения моделей камер.

Целью алгоритма является найти оптимальные позиции, поворот и минимальное количество камер для покрытия имеющейся площади. При этом предполагается, что камеры являются фиксированными.

Однако, данный алгоритм рассчитан на использовании лишь в двухмерное пространство, т.е. в плоскости XZ, при этом высота исследуемой области не учитывается. Его плюсом является возможность совместного использования камер различных характеристик, что может быть использовано в нашей работе.

В работе [45] приводится формализация задачи определения видимости объекта, которая является неотъемлемой частью в задачах оптимизации расположения средств наблюдения. Данная формализация может быть применена для оптимизационных задач в сценах любой сложности, учитывать различные характеристики сцены (освещенность, препятствия и т.п.), средств наблюдения и целей наблюдения. Приводится результаты работы алгоритма, а именно результаты симуляции, отображающих качество размещения камер и их эффективности.

Данный алгоритм строится на генетическом алгоритме поиска, позволяет производить расчет на основе имеющихся двумерных планов сцены, учитывая при этом объекты-препятствия различной формы.

Однако, средства наблюдения, применяющиеся при расчетах, представляют собой простые точки в пространстве, не ограниченные углом обзора, и не имеющие другие внутренние параметры.

В исследовании [20, 46] обсуждается проблема проектирования интегрированных технологических процессов и систем управления. Формулируется задача смешанного целочисленного нелинейного программирования с дифференциально-алгебраическими ограничениями. Этот класс задач часто является невыпуклым, поэтому методы локальной оптимизации обычно не позволяют найти глобальное решение. В качестве решения предлагается глобальный алгоритм оптимизации, основанный на расширении метаэвристической оптимизации колоний муравьев, чтобы решить этот сложный класс задач эффективным и надежным способом.

Объясняются идеи методологии и на основе различных полномасштабных тематических исследований оценивается эффективность подхода.

Метод Oracle [47] новый универсальный метод штрафной функции. Данный метод предназначен, для применения в стохастической метаэвристике, в генетических алгоритмах поиска, оптимизации алгоритма Роя частицы или оптимизация алгоритмы муравьиной колонии. Новизна этого метода заключается в том, что это продвинутый подход, который требует настройки только одного параметра – Oracle. Метод Oracle добавляет штрафные функции в целевую функцию, ограниченная задача преобразуется в неограниченную.

Алгоритм MOCoDE - новая модифицированная композитная дифференциальная эволюция на основе функции штрафа Oracle для задач ограниченной оптимизации [7]. Данный алгоритм предназначен для решения более сложных задач с дискретными, целочисленными или двоичными переменными. В MOCoDE вводится метод обработки дискретных переменных для решения сложных задач с переменными mix. Этот метод оценивается по одиннадцати ограниченным оптимизационным эталонным функциям и семи хорошо изученным инженерным задачам в реальной жизни. Экспериментальные результаты показывают, что MOCoDE достигает конкурентной производительности по отношению к некоторым другим современным подходам в эволюционных алгоритмах с ограниченной оптимизацией, но немного уступает в максимизации покрытия выделенной области.

В работе [48] рассматривается задача оптимального размещения камер наблюдения в распределенной системе видеонаблюдения с целью минимизации количества камер. Эффективность распределенных систем видеонаблюдения зависит от множества факторов, таких как используемые алгоритмы распознавания, технические характеристики камер и средств обработки данных. Предложенный алгоритм оптимизации топологии на основе метода роя частиц, позволяет сократить время проектирования системы видеонаблюдения, оценивая качество распознанного объекта на видеокадре.

Предложенный подход к оптимизации процесса распознавания объектов сложной конфигурации в реальном времени [49, 50]. Позволяет рассматривать

объект, распределенный в пространстве и времени, как множество мелких объектов, которые нужно распознать, чтобы получить результат. Наблюдение за такими объектами часто осуществляется с помощью распределенных систем видеонаблюдения с большим количеством камер. Большой объем данных приводит к проблеме обеспечения режима реального времени из-за нехватки вычислительных ресурсов. Предложено решение организации процесса распознавания распределенных объектов для обеспечения высокого уровня достоверности в условиях дефицита времени на принятие решений и вычислительных ресурсов. Подход основан на разделении процесса распознавания на подзадачи в виде дерева и последующем присвоении им приоритетов. Подзадачи могут решаться с запаздыванием, а приоритет меняется со временем, учитывая его устаревание и вероятность распознавания образов. Это обеспечит обнаружение распределенных объектов высокого уровня достоверности путем отбрасывания подзадач, вероятность успешного решения которых невелика.

В работах [51, 52] описана простая процедура оптимизации для задач на основе ограничений, которая работает с упрощенной функцией затрат или даже без нее. Упрощение постановки задачи делает этот метод особенно привлекательным. Новый метод поддается параллельным вычислениям и хорошо подходит для удовлетворения ограничений, ограниченной оптимизации и задач центрирования проектирования. Еще одним активом является его свойство самостоятельного управления, что делает новый метод простым в использовании.

На основе выше представленного анализа эвристических алгоритмов оптимизации размещения сенсоров наблюдения в качестве решения данной задачи предлагается использовать эвристический алгоритм поиска муравьиной колонии.

2.2. Особенности использования эвристического алгоритма муравьиной колонии при решении задачи оптимизации размещения сенсоров наблюдения

Муравьи относятся к насекомым образующих социальные сообщества(коллективы). При решении сложных динамических задач принято использовать

коллективную систему для выполнения совместной работы, которая без внешнего управления, координации или контроля неспособна выполняться каждым элементом системы в отдельности в разнообразных средах. В таких ситуация применяется Swarm intelligence (роевой интеллект), в качестве продуманного способа кооперирования, то есть способу выживания.

Основным фактом подтверждения оптимальности поведения муравьиных колоний является, расположение сети гнёзд их муравейника близка к минимальному остовному дереву графа [53, 54].

Самоорганизация составляет основу поведения муравьиной колонии, на основе низкоуровневого взаимодействия, обеспечивает достижения общих целей колонии. Колония не имеет центрального управления, обмен локальной информацией происходит только между отдельными особями (прямой обмен – визуальные и химические контакты) и непрямой обмен, лежат в основе муравьиных алгоритмах. Таким образом, в общем случае рассматриваются муравьи неспособные ощущать расстояние до пищи - «слепые муравьи» [55].

Stigmergy - непрямой обмен, представляет собой взаимодействие распространенное во времени, когда один муравей исследует некоторую область окружающей среды, а другие особи могут позже смогут воспользоваться данной информацией, если окажутся в ней. Биологи установили, что pheromone (феромон) – специальное химическое вещество, секретных специальных желёз, откладываемых при перемещении муравьев способствует отложенному взаимодействию. Концентрация феромона на пути определяет предпочтительность маршрутов [56].

Динамический характер адаптивного поведения, реализующийся за счёт испарения феромона, подтверждается проведением некоторой аналогии между «глобальной» памятью муравейника и распределением в окружающую среду феромона, которой воспринимается особями в течение нескольких дней.

Marco Dorigo впервые предложил использовать описанный природный механизм, вдохновленный кормовым поведением естественных колоний муравьев для решения задач оптимизации [57, 58]. Многоагентные системы используются муравьями во время добывания пищи, агенты данных систем работают по простым

правилам, определить кратчайший путь от муравейника к источникам пищи, помечая маршрут движения феромонами на земле. Данные алгоритмы часто используются при решении сложных NP-полных, комбинаторных задач – таких, как задача дискретной оптимизации размещения сенсоров видеонаблюдения на выделенной области и задача коммивояжера, первая из решенных с использованием данного типа алгоритмов.

2.2.1 Классический муравьиный алгоритм

Классический муравьиный алгоритм моделирует многоагентную систему. муравьев(Агентов). Муравьи в многоаспектной системе схожи с настоящими муравьями, которые довольно просто устроены: небольшое количество памяти, предназначенной для выполнения своих обязанностей и выполнение несложных вычислений на каждом шаге работы.

В памяти каждого муравья хранится список пройдённых маршрутов, называющийся «Tabu list» (список запретов) или памятью муравья [59]. Выбор узла для следующего шага зависит от информации хранящийся в памяти о уже пройденных узлах и не рассматривающийся в качестве возможного перехода. На каждом последующем шаге список запретов пополняется новыми узлами, а перед новой итерацией алгоритма – опустошается, когда муравей вновь проходит путь, т.е. при прохождении муравьем нового маршрута, память о старых маршрут очищается.

Кроме списка запретов, при выборе узла для перехода муравей руководствуется «привлекательностью» ребер, зависящая, во-первых, от веса ребра (расстояния между узлами), а во-вторых, от оставшихся на ребре следов феромонов других муравьев, которые ранее прошли по данному ребру. Естественно, что в отличие от весов ребер являющиеся константными, следы феромонов обновляются на каждом обновлении алгоритма: со временем на оставленных маршрутах следы испаряются, а часто используемых, напротив, усиливают их [60].

Пусть муравей находится в узле i, а узел j – это один из узлов, доступных для перехода: Обозначим вес ребра, соединяющего узлы i и j, как w_{ij}, а интенсивность феромона на нем – как t_{ij}. Тогда вероятность перехода муравья из i в j будет равна:

$$p_{ij} = \frac{t_{ij}^{a} + \frac{1}{w_{ij}^{B}}}{\sum l \in s_i (t_{il}^{a} + \frac{1}{w_{ij}^{B}})} \qquad (2.2.1)$$

где α и β – это регулируемые параметры, определяющие важность составляющих (веса ребра и уровня феромонов) при выборе пути. Очевидно, что при алгоритме превращается в классический жадный алгоритм, а при нём быстро сойдется к некоторому субоптимальному решению. Выбор правильного соотношения параметров является предметом исследований, и в общем случае производится на основании опыта.

После, как муравей успешно проходит маршрут, он оставляет на всех пройденных ребрах след [57], обратно пропорциональный длине пройденного пути:

$$\Delta_{ij} = \begin{cases} \frac{k}{L}, (ij) \in P \\ 0, (ij) \notin P \end{cases} \qquad (2.2.2)$$

где L – длина пути, а k – регулируемый параметр. Кроме этого, следы феромона испаряются, то есть интенсивность феромона на всех ребрах уменьшается на каждой итерации алгоритма. Таким образом, в конце каждой итерации необходимо обновить значения интенсивностей.

2.2.2 Модификации классического алгоритма муравьиной колонии

В первые алгоритм муравьиной колонии применили для решения задачи коммивояжёра, результаты первых экспериментов с применением муравьиного алгоритма для данной задачи были многообещающими, однако далеко не лучшими по сравнению с уже существовавшими методами. Простота классического муравьиного алгоритма (названного «муравьиной системой») оставляла возможности для доработок – и именно алгоритмические усовершенствования стали предметом дальнейших исследований Marco Dorigo [57,58, 60] и других специалистов в области комбинаторной оптимизации. В основном, эти усовершенствования связаны с большим использованием истории поиска и более тщательным исследованием областей вокруг уже найденных удачных решений. Ниже рассмотрены наиболее примечательные из модификаций.

Elitist Ant System. Одним из таких усовершенствований является введение в алгоритм так называемых «элитных муравьев». Опыт показывает, что, проходя ребра, входящие в короткие пути, муравьи с большей вероятностью будут находить еще более короткие пути. Таким образом, эффективной стратегией является искусственное увеличение уровня феромонов на самых удачных маршрутах. Для этого на каждой итерации алгоритма каждый из элитных муравьев проходит путь, являющийся самым коротким из найденных на данный момент [50].

Эксперименты показывают, что, до определенного уровня, увеличение числа элитных муравьев является достаточно эффективным, позволяя значительно сократить число итераций алгоритма. Однако, если число элитных муравьев слишком велико, то алгоритм достаточно быстро находит субоптимальные решение и застревает в нем [61]. Как и другие изменяемые параметры, оптимальное число элитных муравьев следует определять опытным путем.

Ant-Q. Luca M. Gambardella и Marco Dorigo опубликовали в 1995 году работу, в которой они представили муравьиный алгоритм, получивший свое название по аналогии с методом машинного обучения Q-learning [62]. В основе алгоритма лежит идея о том, что муравьиную систему можно интерпретировать как систему обучения с подкреплением. Ant-Q усиливает эту аналогию, заимствуя многие идеи.

Алгоритм хранит Q-таблицу, сопоставляющую каждому из ребер величину, определяющую «полезность» перехода по этому ребру. Эта таблица изменяется в процессе работы алгоритма – то есть обучения системы. Значение полезности перехода по ребру вычисляется исходя из значений полезностей перехода по следующим ребрам в результате предварительного определения возможных следующих состояний. После каждой итерации полезности обновляются исходя из длин путей, в состав которых были включены соответствующие ребра.

Ant Colony System. В 1997 году те же исследователи опубликовали работу, посвященную еще одному разработанному ими муравьиному алгоритму [63]. Для повышения эффективности по сравнению с классическим алгоритмом, ими были введены три основных изменения.

Во-первых, уровень феромонов на ребрах обновляется не только в конце очередной итерации, но и при каждом переходе муравьев из узла в узел. Во-вторых, в конце итерации уровень феромонов повышается только на кратчайшем из найденных путей. В-третьих, алгоритм использует измененное правило перехода: либо, с определенной долей вероятности, муравей безусловно выбирает лучшее – в соответствие с длиной и уровнем феромонов – ребро, либо производит выбор так же, как и в классическом алгоритме.

Max-min Ant System. В том же году Томас Штютцле (Tomas Stützle) и Хольгер Хоос (Holger Hoos) предложили муравьиный алгоритм, в котором повышение концентрации феромонов происходит только на лучших путях из пройденных муравьями [64]. Такое большое внимание к локальным оптимумам компенсируется вводом ограничений на максимальную и минимальную концентрацию феромонов на ребрах, которые крайне эффективно защищают алгоритм от преждевременной сходимости к субоптимальным решениям.

На этапе инициализации, концентрация феромонов на всех ребрах устанавливается равной максимальной. После каждой итерации алгоритма только один муравей оставляет за собой след – либо наиболее успешный на данной итерации, либо, аналогично алгоритму с элитизмом, элитный. Этим достигается, с одной стороны, более тщательное исследование области поиска, с другой – его ускорение.

ASrank. Bernd Bullnheimer, Richard F. Hartl и Christine Strauß разработали модификацию классического муравьиного алгоритма, в котором в конце каждой итерации муравьи ранжируются в соответствие с длинами пройденных ими путей [65]. Количество феромонов, оставляемого муравьем на ребрах, таким образом, назначается пропорционально его позиции. Для более тщательного исследования окрестностей уже найденных удачных решений, алгоритм использует элитных муравьев.

AntMiner. Впервые применение алгоритма муравьиной колонии для классификации было представлено R. Parpinelli в 2002 году [21]. Метод извлечения классификационных правил [66] на основе муравьиного алгоритма. Цель метода — получить простые правила вида если условие, то следствие.

Предполагается, что все атрибуты категориальны (имеют категорию). Т.е. термы представлены в виде Атрибут = Значение, например, пол = мужской. AntMiner последовательно формирует упорядоченный список правил. Вычисление начинается с пустого множества правил и после формирования первого, все тестовые единицы данных, покрытые этим правилом, удаляются из тестового набора.

В исследовании предлагается использовать данный алгоритм для решения задачи оптимизации размещения средств наблюдения. Рассмотрим алгоритм на тестовом наборе вершин графа. Алгоритм использует при работе направленный граф, где каждому атрибуту сопоставляется столько вершин, сколько возможных значений он принимает в тестовом наборе. Соответственно, предполагается, что перед началом работы алгоритма из тестового набора выделены множества атрибутов и их возможных значений, а также множество возможных классов.

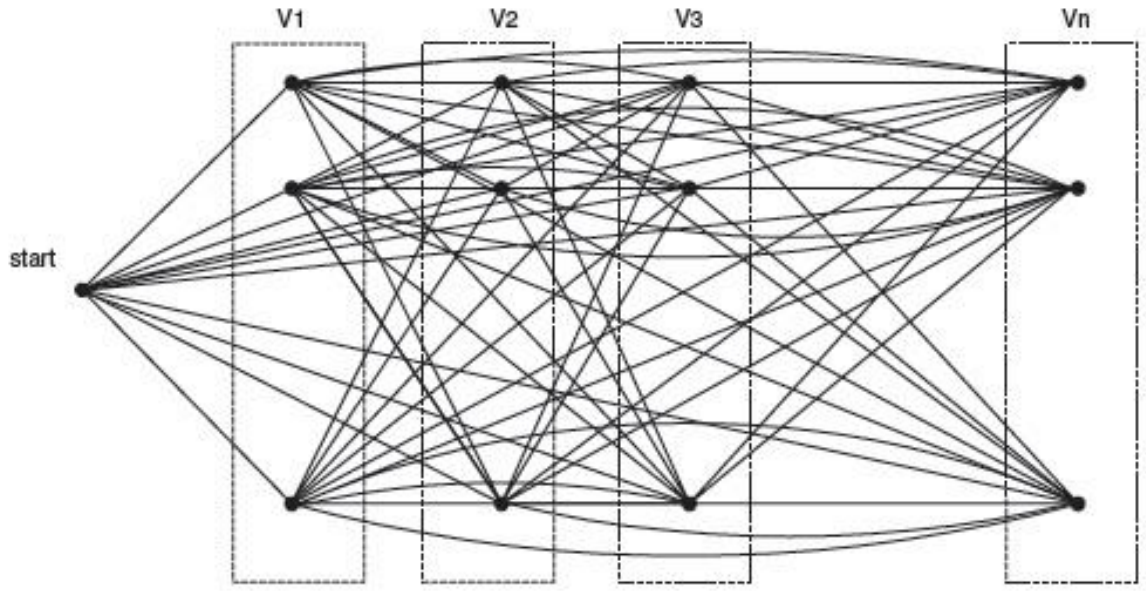


Рисунок 2.2.1 – Тестовый набор вершин

Псевдокод алгоритма AntMiner.

testSet = Тестовый набор вершин графа

ПОКА *(|testSet| > макс_число_непокрытых_случаев)*

i = 0;

ВЫПОЛНЯТЬ

i = i + 1;

i-й муравей последовательно строит классификационное правило;

Обновление феромонов на пути, представленным правилом i-го муравья;

***ПОКА** (i < Число_муравьёв)*

Выбрать лучшее из всех правил;

testSet = случаи, не покрытые лучшим правилом;

КОНЕЦ ЦИКЛА

Алгоритма Ant Miner при решении задачи оптимизации размещения средств наблюдения в трехмерной сцене. Далее последовательно рассмотрим шаги модифицированного алгоритма муравьиной колонии AntMiner.

Шаг 1. Инициализация параметров узлов графа.

Все ребра графа инициализируются начальным значением феромона по следующей формуле:

$$\tau_{ij}(t=0) = \frac{1}{\sum_{i=1}^{a} b_i} \qquad (2.2.3)$$

, где a — общее число атрибутов, b_i— количество возможных значений i-го атрибута.

Шаг 2. Начальное расположение муравьев в узлы сцены.

Каждое правило в алгоритме AntMiner содержит антецедент (набор термов Атрибут = Значение) и представляемый им класс. В оригинальном алгоритме исходные данные содержат только категориальные атрибуты. Предположим, что $терм_{ij} \approx A_i = V_{ij}$, где A_i это i-й атрибут, а V_{ij} это j-ое значение A_i.

Вероятность, что этот терм будет добавлен в текущее частичное правило, составляемое муравьём определяется следующей формулой:

$$P_{ij}(t) = \frac{\tau_{ij}(t)*\eta_{ij}}{\sum_{i}^{a}\sum_{j}^{b_i}\tau_{ij}(t)*\eta_{ij}, \forall i \in I} \qquad (2.2.4)$$

, где η_{ij} это значение эвристики (будет определена далее), τ_{ij} значение феромона на этом ребре графа, а I множество атрибутов, еще не использованных текущим муравьём.

Шаг 3. Эвристика. Определение видимых узлов графа

В традиционном алгоритме муравьиной колонии для решения задачи оптимизации размещения средств наблюдения веса ребер используются совместно со значением феромона для принятия решения о выборе следующего узла сцены (вершины графа). В алгоритме AntMiner эвристикой является количество информации — термин, использующийся в теории информации. Качеством здесь измеряется с помощью энтропии, для предпочтения одних правил над другими:

$$\eta_{ij} = \frac{log_2(k) - inf\ o T_{ij}}{\sum_i^a \sum_j^{b_i} log_2(k) - inf\ o T_{ij}} \qquad (2.2.5)$$

$$inf\ o\ T_{ij} = -\sum_{w=1}^{k} \left[\frac{freq T_{ij}^w}{|T_{ij}|}\right] * log_2 \left[\frac{freq T_{ij}^w}{|T_{ij}|}\right] \qquad (2.2.6)$$

где k — число классов, T_{ij} — подмножество, содержащее все единицы данных, где атрибут A_i имеет значение V_{ij}, $|T_{ij}|$ число элементов в T_{ij}, $freq(T_{wij})$ — число единиц данных, имеющих класс w, a — общее число атрибутов, b_i — число возможных значений i-го атрибута. Чем выше значение $Info\ T_{ij}$, тем меньше шансов что муравей выберет данный терм.

Шаг 4. Обновление феромона.

После того как каждый муравей завершит конструирование правила, выполняется обновление феромонов по следующей формуле: $\tau_{ij}(t+1) = \tau_{ij}(t) + \tau_{ij}(t).Q$ $\forall терм_{ij} \in текущее\ правило$, где Q — качество правила, которое вычисляется по следующей формуле:

$$Q = \left(\frac{TruePos}{TruePos + FalseNeg}\right) x \left(\frac{TrueNeg}{FalsePos + TrueNeg}\right) \qquad (2.2.7)$$

Здесь $TruePos$ — количество единиц данных, которые покрываются правилом и класс которых совпадает с классом, представляемым правилом, $FaslePos$ — количество единиц данных, которые покрываются правилом, но класс которых отличается от класса, представляемым правилом, $TrueNeg$ — количество единиц данных, которые не покрываются правилом и класс которых не совпадает с классом, представляемым правилом, $FalseNeg$ — количество единиц данных, которые не покрываются правилом, но класс которых совпадает с классом, представляемым правилом.

Также требуется выполнять пересчёт феромонов, имитирующий его испарение. Это можно сделать простой нормализацией значений с учётом обновленного значения феромонов.

Шаг 5. Улучшенное обновление феромона.

Для более гибкой настройки выполнения алгоритма распознания вершин графа (узлов сцены) используется следующая формула обновления вероятностей ребер, принадлежащих текущему правилу:

$$\tau_{ij}(t) = (1-p) * \tau_{ij}(t-1) + (1-\frac{1}{1+Q}) * \tau_{ij}(t-1) \qquad (2.2.8)$$

где p – коэффициент испарения (обычно устанавливается значение ~0,1), Q качество правила, описанное выше. Ребра, не покрытые текущим правилом, просто нормализуются. Этот способ обеспечивает снижение значения феромона при низком Q и повышение при высоком (в отличие от оригинального способа).

Шаг 5. Запуск элитных муравьев.

Запуская и постепенно увеличивая число элитных муравьев, позволяет добиться максимально эффективного решения поиска оптимального маршрута от узла камеры к узлу сцены.

Оригинальный алгоритм выбора терм исполняется по следующему псевдокоду:

ДЛЯ ВСЕХ $(i,j) \in J_i$

ЕСЛИ $q \le \sum P_{ij}$ ***ТО*** *Выбрать терм*$_{ij}$

Где q случайная(ый) величина(узел) [0..1] с равномерным распределением. Т.е. используется плотность вероятностей термов. Для введения элитных муравьёв алгоритм выбора терма изменяется следующим образом:

ЕСЛИ $q1 \le \phi$ ***ТО***

ДЛЯ ВСЕХ $(i,j) \in J_i$

ЕСЛИ $q2 \le \sum Pij$ ***ТО*** *Выбрать терм*$_{ij}$

ИНАЧЕ

Выбрать терм с $P = max\ P_{ij}$

Здесь q1 и q2 случайные величины [0..1] с равномерным распределением.

Модификация эвристического алгоритм муравьиной колонии для классификационных правил AntMiner , разработанный R. Parpinelli, выбран в качестве базового в нашем диссертационном исследовании по оптимизации размещения сенсоров наблюдения.

2.3 Повышение адекватности функции видимости узлов сетки с целью учета ее параметров

По сравнению с задачами мониторинга периметра или отдельных объектов автоматизация размещения и контроль динамически изменяемых сцен предполагают более высокий интеллектуальный уровень как самой системы наблюдения, позволяющий эффективно фильтровать информацию из массивного потока данных, так и добавлять, редактировать, прогнозировать недостающие данные [67, 68, 69]. Не редко объекты, расположенные в виртуальной сцене, находятся вне зоны прямой видимости камер видеонаблюдения, тем самым обнаружение и установление положения 3D-объекта усложнено. При решение данной задачи, предлагается алгоритм вычисления позиции элементов сцены с помощью двух дополнительных источников освещения [70]. Неуместное установка сенсоров наблюдения на прямых линиях, приводит к появлению в пространстве «мертвых углов», решение оптимального расположения, сводится к выделению точных закрытых форм однотипных камер наблюдения [71]. Одним из подвидов задач оптимизации размещения подобных сенсоров, является задача оптимального их размещения на границе контролируемой области. Благодаря использованию комбинаторных методов, данный подход позволяет использовать вероятностные модели локализации проникновения на охраняемую территорию [72,73]. На сегодняшний день в компьютерной графике не существует обобщенной локальной модели освещения, которая позволила бы моделировать объекты с реальными свойства без дополнительных манипуляций. В работе [74] рассмотрены существующие локальные модели освещения для построения объектов в 3D-сцене.

Для решения задачи выбора оптимальных конфигураций средств наблюдения в виртуальной динамической трёхмерной сцене недостаточно только моделирования самой сцены и расположения камер наблюдения на ней. Необходим также анализ параметров камер наблюдений, условий освещенности и доработка существующих алгоритмов размещения камер с учетом их (камер) чувствительности.

В обзоре подходов к решению задач дискретной оптимизации, основанных на генетических и эвристических алгоритмах поиска, подчёркивается, что при формализации решаемой проблемы размещения средств наблюдения описание взаимосвязи видимости узлов сцены с характеристиками источников освещения является трудоёмкой задачей и часто используется упрощённая модель функции видимости. Такие характерные ситуации требуют введения модели функции видимости, учитывающей физические свойства объектов, взаимодействие их с источниками освещения.

В исследовании приводится описание процедуры повышения адекватности функции видимости позиций для трехмерной сцены и заданной конфигурации расположения камер наблюдения, предлагается учитывать параметры сенсоров, параметры источников освещения, различные шумы, зависящие от теней виртуальных объектов, описывается структура программного пакета, в котором реализуется построение тепловой карты видимости.

2.3.1 Постановка задачи

В данной главе исследования не рассматриваются вопросы идентификации объектов наблюдения, определения траектории их движения, определения их местоположения. Мы считаем, что их совокупность задается набором возможных позиций на сцене, в частности координатами на плоской сцене. Последние определяются исходя из трехмерной модели, позволяющей определить, может ли объект наблюдаться в определенной позиции камерой, находящейся в одной из фиксированных точек (см. рис. 2.3.1).

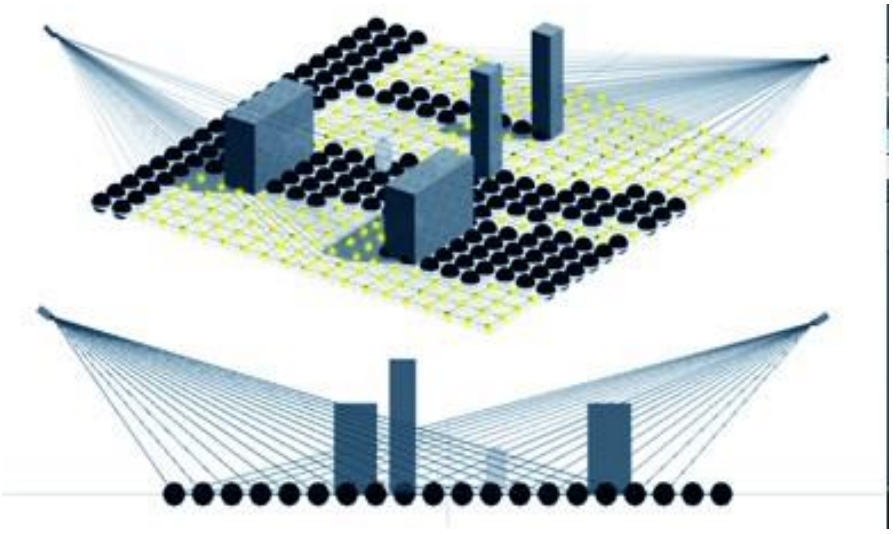


Рисунок 2.3.1 – Тестовый набор вершин

В работе [27] представлена формализация задачи оптимизации размещения камер наблюдения с точки зрения ее различных целевых постановок: максимизации уровня наблюдаемости объектов сцены, определения слепых зон, минимизации количества камер наблюдения при сохранении заданного уровня наблюдаемости. Для построения формализации предлагается использовать взвешенный ориентированный граф видимости положений объектов в трехмерной сцене $G= (V \cup A, E, f)$, основанный на сетке, наложенной на наблюдаемую поверхность (возможны 2D и 3D сетки), где

– V – множество узлов сетки, соответствующее возможным позициям объектов на сцене;
– $A=\{Ai\}$ – множества узлов сетки, в которых могут располагаться камеры, Ai соответствует возможной позиции i–й камеры;
– E – множество ориентированных ребер графа, которые соединяют вершины из множества A с вершинами из V и определяют наличие прямой видимости узла из позиции камеры;
– $f: E \rightarrow R$, функция меры видимости позиции для конкретной камеры (вес дуги графа видимости, будет определена далее).

Формально, определяем

$$\bar{a} = \{a_1, a_2, \dots, a_k\} \underset{1 \le i \le k}{\forall} a_i \in A_i \qquad (2.3.1)$$

как набор, который представляет конфигурацию датчиков. Набор всех возможных конфигураций датчиков обозначается как $\bar{A}$.

Наша задача-определить функцию наблюдаемости положения сцены с учетом параметров установленных камер. При создании системы мониторинга особое внимание уделяется следующим параметрам камер видеонаблюдения:

- чувствительность (минимальная освещенность объекта, обеспечивающая требуемое качество изображения);
- шум (степень проявления на изображении так называемого "снега", который полностью влияет на качество изображения);
- разрешение (максимальное количество строк, обнаруженных на изображении тестовой таблицы с требуемой точностью обнаружения);
- гамма-коррекция (нелинейное преобразование светосигнальной характеристики для согласования условий наблюдения и модуляционных характеристик дисплея с контрастной чувствительностью зрения).

Чувствительность камеры напрямую зависит от параметра ее светосигнальных характеристик; прямая зависимость выходного сигнала от освещенности сцены тем выше параметр, тем выше чувствительность при требуемом пороге.

Постановка задачи заключается в определении параметров камер наблюдения системы мониторинга и их влияния на степень видимости объекта на этапе: значение. Формально задача решается путем определения весовой функции узла $g\colon\ V \times \bar{A} \to R$ для графа видимости. В работах [75, 76] предложено определять значение чувствительности камеры наблюдения при минимально допустимой освещенности ПЗС-матрицы с учетом переменных величин: светимости используемого объектива, расстояния до объекта, его контрастности и некоторых других. В результате получается формула, связывающая матрицу *L0* (освещенность на объекте) с матрицей чувствительности камеры *S*:

$$S = 1 + L_0 \times \frac{k_0 \times \tau}{4 \times (1+m)^2 \times H^2} \qquad (2.3.2)$$

где: k_o-коэффициент отражения объекта (тест-таблица белого цвета); τ-коэффициент пропускания света объективом; m-отношение фокусного расстояния

объектива к расстоянию до объекта; *H*-отношение фокусного расстояния объектива к диаметру его входного зрачка.

Все эти параметры являются параметрами камер наблюдения и параметрами контролируемой сцены. Матрицу S, в свою очередь, можно рассматривать как матрицу Весов графа видимости, размерность которой определяется количеством узлов сетки, наложенных на сцену. Таким образом, функция чувствительности камеры для конкретного положения трехмерной сцены определяется тремя группами параметров: параметрами камеры наблюдения (камер), параметрами освещенности сцены (количеством источников света, направленностью, интенсивностью) и параметрами самой 3D-сцены.

В 3D-сцене освещение объекта значительно отличается от освещения реальной комнаты. Основным фактором, ограничивающим чувствительность в виртуальной сцене, является шум, зависящий от теней динамических объектов. По мере увеличения плотности и количества объектов в сцене увеличивается количество препятствий для источника света, тем самым повышая процент так называемого “снега” в выходном изображении.

Для уменьшения шума в изображении предлагается ввести переменную *N*, отражающую шум (тени динамических объектов сцены), значение которой вычисляется из следующего выражения

$$N = \frac{N_{FRONT}}{N_{BACK}}, \quad (2.3.3)$$

, где N_{FRONT}-освещение объекта со стороны камеры, для которой рассчитывается чувствительность, N_{BACK}-освещение объекта со стороны, противоположной камере. Для расчета степени видимости узлов сетки используется модифицированная формула (2.3.6), учитывающая тени на трехмерной сцене:

$$f(v, a_i) = 1 + L_0 \times \frac{k_0 \times \tau \times N}{4 \times (1+m)^2 \times H^2} \qquad (2.3.4)$$

В качестве функции видимости узла сетки для конкретной камеры [77] используется соответствующее значение матрицы *S*:

$$g(v, \bar{a}) = \frac{\sum_{i=1}^{k} f(v, a_i)}{k \times \max\limits_{v, a_i} f(v, a_i)} \qquad (2.3.5)$$

где $f(v_i,a_i)$ - значение видимости, рассчитанное для конкретной точки сцены (узла сетки), k-количество камер, с которых видна позиция V. Очевидно, что

$$\forall_{v\in V}\ \forall_{\bar{a}\in\bar{A}}\ \ 0 \le g(v,\bar{a}) \le 1 \qquad (2.3.6)$$

Тепловая карта для этой трехмерной сцены и датчика конфигурации-пара $\bar{a}$

$$HM = (V,\ \bar{a}).$$

2.3.2 Вычислительный эксперимент

Для моделирования трехмерной сцены и решения задачи оптимизации размещения сенсоров разработан программный комплекс [27], который позволяет строить и в реальном времени обновлять граф наблюдаемости, а также решать формализованную задачу (предусмотрены варианты получения оптимального решения или приближенного решения, полученного эвристическими алгоритмами).

- Инструментарий позволяет строить виртуальную сцену большой сложности и содержит следующие типы объектов:
- Цель. Представляет собой объект, расположенный на сцене, установление наблюдения за которым и является целью данного комплекса.
- Препятствия. Имитируют реальные препятствия на сцене, мешают установлению визуального контакта камеры с объектом-целью. Могут иметь геометрическую форму различной сложности.
- Средства наблюдения. Могут иметь различные алгоритмы работы, основываясь на различных функция наблюдения.

Задачу расчета чувствительности камеры в освещаемой трехмерной динамической сцене можно рассматривать как один из этапов определения эффективности конкретной конфигурации системы видеонаблюдения с использованием камер. Разработанное программное обеспечение использует формулу (2.3.5) для определения степени видимости положения сцены для заданной конфигурации камер. Рисунок. 5 показаны результаты визуализации расчета слепых зон и видимости позиций сцены.

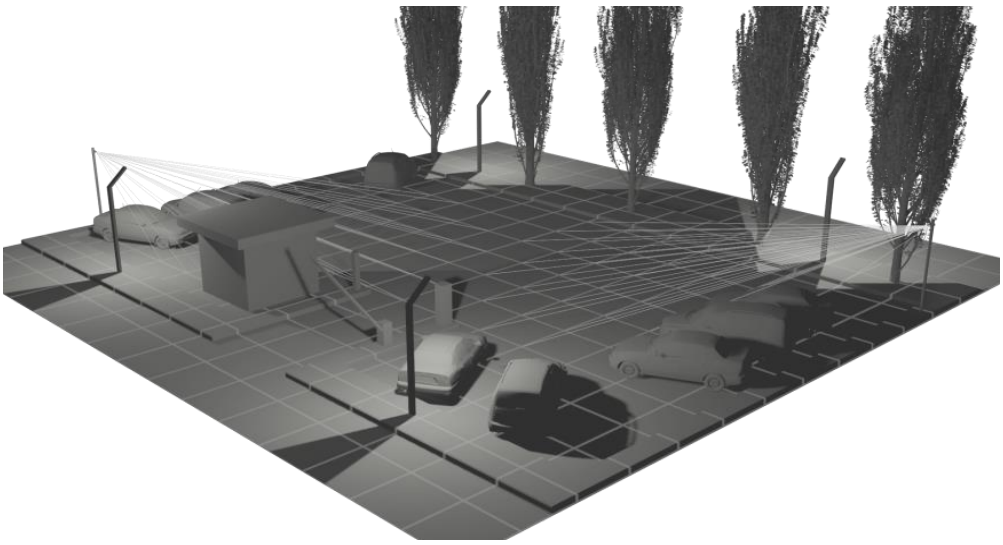


Рисунок 2.3.2 – 3D-сцена с двумя камерами наблюдения и источниками света

Оптимизация размещения камер видеонаблюдения для повышения эффективности всей системы должна решаться с более точным учетом места расположения источников света. Текущая версия программного обеспечения использует довольно простой вычисленный коэффициент N в формуле (2.3.6) для учета освещенности сцены и ее конкретных положений. Взаимное расположение источников света, их параметры также могут быть источником для оптимизации конкретной конфигурации компоновки камеры.

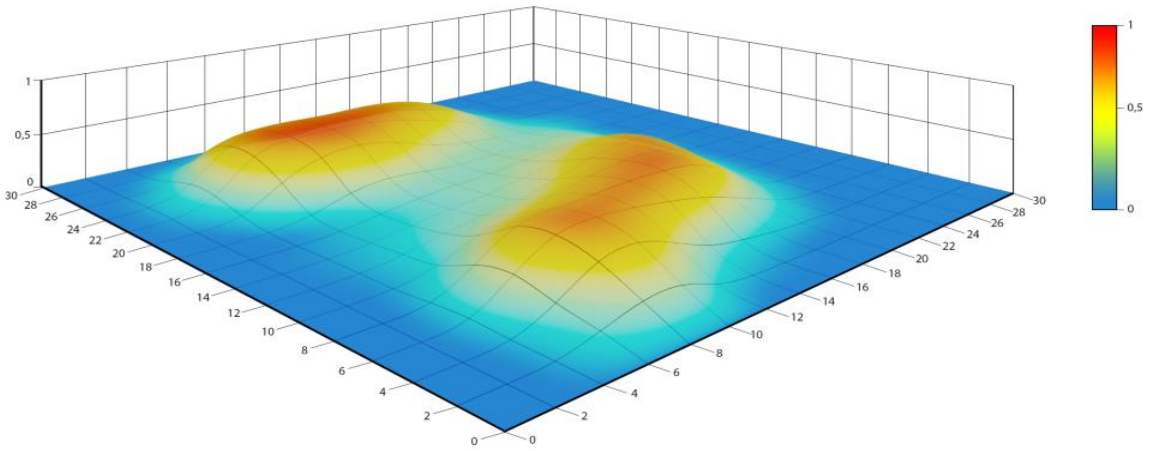


Рисунок 2.3.3 – Тепловая карта для видимости узлов 3D-поверхности

На рис. 2.3.3 показана тепловая карта видимости узлов реальной поверхности, смоделированная в разработанном программном обеспечении. Тепловая карта основана на функции чувствительности (2.3.6) двух камер наблюдения в освещенной трехмерной динамической сцене. Видимость узлов на карте отображается в

ограниченном диапазоне определяемых Формулой (1). Наиболее видимые узлы стремятся к значению 1, невидимые узлы-к значению 0.

2.4. Выводы по главе 2

Во второй главе диссертационного исследования проанализированы эвристические алгоритмы поиска с целью их применения для решения задач дискретной оптимизации размещения сенсоров наблюдения. На основе проведенного анализа в исследовании предлагается использовать модифицированный алгоритм муравьиной колонии AntMiner, разработанный R. Parpinelli [21]. Цель данного метода — получить простые правила вида <если условие, то следствие>, что позволит повысить скорость выполнения алгоритма муравьиной колонии при решении задачи в трехмерной сцене. Важно также отметить, что в существующих подходах не используются параметры источников освещения, что влияет на точность распознавания видимости узлов сцены.

В диссертационном исследования предлагается способ формализации задачи оптимального размещения камер видеонаблюдения, основанный на введении специального графа видимости, определенного для сложной трехмерной сцены с учетом физических параметров источников освещения и объектов. Представлена математическая модель, позволяющая оценить видимость узлов наложенной на сцену сетки, с учетом параметров датчиков наблюдения, источников света и их положения. На основе разработанной модели предложена визуализация степени видимости узлов сетки как тепловой карты определённой в работе функции видимости. Такая визуализация дает возможность увидеть слепые зоны для заданной конфигурации камер и качество наблюдения на различных участках сцены.

Разработанное в ходе исследования программное обеспечение позволяет в интерактивном режиме редактировать 3D-сцену, настраивать размещение и параметры камер, источников света. Конфигурация размещения камер может быть оценена как визуально, так и количественно. Программное обеспечение реализовано в среде Unity 3D.

ГЛАВА 3. АЛГОРИТМЫ РАНЖИРОВАНИЯ НА ГРАФАХ ПРИ РЕШЕНИИ ЗАДАЧ ОПТИМИЗАЦИИ РАЗМЕЩЕНИЯ СЕНСОРОВ НАБЛЮДЕНИЯ

3.1 Постановка задачи

В исследовании рассматривается задача ранжирования вершин ориентированного взвешенного графа с точки зрения использования предварительного ранжирования при решении некоторых оптимизационных задач. Постановка задачи имеет достаточно длинную историю и разнообразные подходы к ее решению. Алгоритмы ранжирования предложены в большом количестве и имеется анализ их использования и реализации. В настоящее время эта задача вновь становится актуальной в связи с тем, что появляются структурированные данные большой размерности – графы с десятками, сотнями тысяч вершин. Такого рода структурированные данные необходимо обрабатывать, анализировать их структуру. При этом важной задачей становится снижение временной сложности алгоритмов обработки за счёт распараллеливания и обработки только локальной окрестности той части графа, в которой произошли изменения. То есть решение задачи для всего графа можно в этом случае получить из имеющегося глобального за счёт его локального изменения на ограниченной окрестности вершин. Такая постановка задачи в современной литературе связана с понятием инкрементального алгоритма поиска решения, который использует для коррекции глобального решения обработку вершин, находящихся в локальной окрестности части графа, подвергшейся изменению [78, 79].

Проблема ранжирования вершин графа состоит в построении такой функции, которая определяет значения рангов всех элементов некоторого пространства. Обычно в задачах ранжирования рассматриваются данные, представляющие собой векторные пространства, то есть на вход алгоритма поступает некоторый вектор значений из евклидова пространства R^n, на выходе имеется некоторое число, которое определяет ранжирование векторов в некотором порядке. Таким образом функция ранжирования в такой постановке является функцией вида $f : R^n \to R$, которая

упорядочивает объекты евклидова пространства нужным образом. Мы говорим, что ранг объекта *v* выше ранга объекта u, если *f(v) > f(u)*.

Далее будут приведены примеры использования предварительного ранжирования вершин графа, что приводит к повышению эффективности решения уже конкретной задачи. Другой рассматриваемой задачей статьи является анализ результатов конкретных алгоритмов ранжирования на предмет типа распределения полученной совокупности рангов.

Формальная постановка задачи. Мы рассматриваем задачу построения такого отображения множества вершин ориентированного взвешенного графа в множество вещественных чисел, которое отражает доминирование одной вершины по отношению к другой. Алгоритмы такого рода используются при ранжировании сайтов в сети Интернет и имеют приложения к разнообразным задачам. В качестве определения ориентированного взвешенного графа мы используем стандартное понятие графа $G=(V, E, w)$, где V – множество вершин графа, E – множество дуг (ориентированных ребер), $w: V \rightarrow R$ – функция веса дуг. Смысловым источником такого рода задач является классическая задача топологической сортировки, которая имеет формальную постановку, опирающуюся на следующие определения. В том случае, если функция веса действует тривиальным способом, задавая вес любой дуги равным 1, мы будем опускать ее при определении графа.

Определение 1. Дан граф $G=(V, E)$, говорим, что вершина v достижима из вершины u, если $v = u$, или $(u, v) \in E$, или существует путь $p = v_1, v_2, \ldots, v_n$ в графе G такой, что $v_1 = u$ и $v_n = v$. Факт достижимости в этом случае обозначаем как $u \sim> v$. Для понятия «взвешенный граф», которое предполагает наличие функции веса для ориентированных дуг, используется такое же определение.

Задача топологической сортировки (ТС) может быть сформулирована следующим образом: топологической сортировкой графа является отображение $f : V \rightarrow R$ такое, что справедлива формула (3.1.1)

$$\forall v, u \in V(\overleftrightarrow{v} \sim > u) \rightarrow (f(v) < f(u)) \qquad (3.1.1)$$

Классическая постановка задачи о ТС накладывает серьезные ограничения на граф – он должен быть ациклическим. Заметим, что в общем случае такие графы

являются достаточно редкими и необходимо разрабатывать процедуры ранжирования пригодные для произвольных графов.

Существует тривиальное решение задачи ТС, которое может быть получено в ходе обхода графа в ширину и имеет сложность $O(n+m)$, где n – количество вершин графа, m – количество дуг. В последнее время появилось большое число публикаций, которые рассматривают различные варианты этой классической постановки. Большая часть из них посвящена построению ТС в условиях динамически изменяемого графа и учёта только локальной окрестности для этого изменения.

Под формальным определением задачи ранжирования вершин ориентированного взвешенного графа мы будем понимать задачу, описанную следующим образом. Под ранжированием понимается определение функции f, как это сделано при определении топологической сортировки, определенной выше, но имеющей следующие существенные отличия:

допускается существование таких пар вершин графа $v, u \in V$, что $f(v) = f(u)$;

отношение ранжирования строится на основе композиции отношений ранжирования для локального окружения каждой пары вершин;

снимается ограничение об отсутствии циклов в графе.

Анализируемые далее алгоритмы могут быть классифицированы в соответствии с нашими установками на два класса – локальные и глобальные. *Локальные* алгоритмы при вычислении ранга вершины учитывают структурные характеристики локальной окрестности вершины в графе. При вычислении ранга вершины в этом случае учитывается только направленность и вес дуг инцидентных данной вершине. *Глобальные* алгоритмы получают решение в результате последовательности итераций, учитывающих изменения текущего решения для всех вершин графа. Использованные далее способы вычисления рангов были адаптированы для нашей задачи и нормализованы к диапазону сравнимых значений.

3.2 Алгоритмы ранжирования в решении прикладных задач

Многие приложения (анализ структуры указателей при организации блоков памяти операционной системы, инкрементальная компиляция) сводятся к

реализации ТС ациклических ориентированных графов в условиях их динамического изменения [80]. Вычисление различных характеристик динамически изменяющихся графов представляет определённый интерес и в этом плане наибольшее значение имеют, так называемые, инкрементальные алгоритмы, которые позволяют сконструировать глобальное решение для всего графа, учитывая только локальные изменения и имеющееся решение задачи до момента изменения. Так для решения задачи построения ТС вершин графа разработаны эффективные алгоритмы динамической сортировки, которые учитывают операции добавления/удаления ребер графа и позволяют корректировать имеющееся решение сразу после изменений [81, 82].

Опишем ряд результатов, которые показывают, что задача ранжирования вершин ориентированного графа имеет большое значение для повышения эффективности оптимизационных алгоритмов. Предварительное ранжирование, индексирование данных является традиционным способом повышения скорости выполнения запросов к реляционным базам данных. Аналогичная ситуация имеет место и при работе с NoSQL базами данных (Neo4j, DEX), которые содержат информацию о динамических структурах, с большим количеством отношений различного типа [83, 84] подобных социальным, телекоммуникационным сетям, ссылочному пространству в сети Internet. Оптимизация запросов к базам данных традиционно используется в реляционных базах данных и основана на построении индексов. Один из рассматриваемых подходов использует ранжирование вершин графовой базы данных (ГБД) для инкрементальных процедур построения разбиений на множествах вершин.

Отмечается, что существуют системы, например [79, 85], MapReduce, которые имеют высокую эффективность в работе со сложноструктурированными данными, но эта эффективность снижается, когда данные имеют динамическую природу. Графовой БД называется структура подобная графу с помеченными дугами и вершинами. Обобщённая структура, которая определяет типы взаимодействия сущностей, называется схемой ГДБ см. рис.3.2.1.

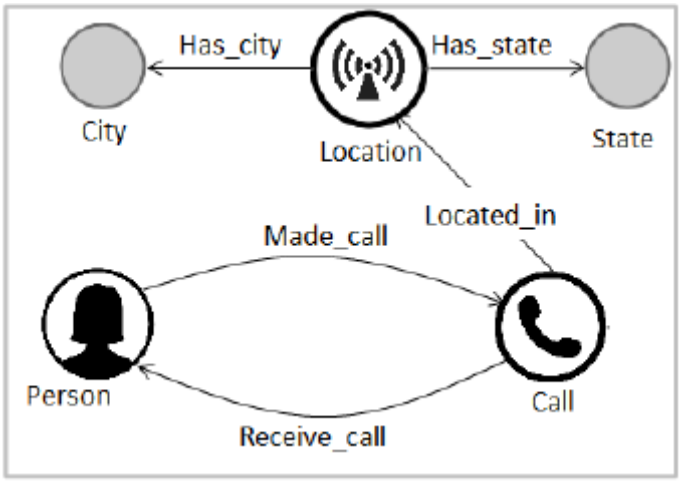


Рисунок 3.2.1 – Схема ГБД «Телефонные звонки»

Повышение скорости работы алгоритмов на основе ранжирования вершин графа предлагается и для решения классических оптимизационных задач. В работе [86, 87] предлагается муравьиный алгоритм решения обобщенной оптимизационной задачи (рассматривается задача коммивояжёра) на основе набора эвристик. На основе инкрементального определения концентрации искусственного феромона, определяются вероятности выбора следующей вершины графа при выборе продолжения промежуточного решения. В процессе работы алгоритма все вершины графа ранжируются и вероятность выбора следующей вершины зависит от её ранга.

Работа [88] содержит описание двух алгоритмов ранжирования вершин ориентированных и неориентированных графов, которые используют матрицу Лапласа. Особое внимание к этой проблеме связано с задачей обучения интеллектуальных систем принятия решений в условиях динамического изменения данных различной природы. В качестве примеров, используемых для демонстрации эффективности алгоритмов, рассматриваются два набора данных. Первый имеет отношение к компьютерной биологии и содержит иерархическую классификацию протеинов. Второй – содержит разбиение текстовых новостных статей на группы по связанным с ними тегам. Результаты вычислительного эксперимента показывают уменьшение суммарной ошибки ранжирования при использовании динамической процедуры. К особенностям рассматриваемой задачи относится то, что на структуру графов накладываются серьёзные ограничения – они должны быть *к*–

дольными, а в качестве используемых массивов для вычислительного эксперимента выбраны данные именно такой природы.

В работе [89] рассматривается оптимизационная задача поиска кратчайшего пути на местности, описываемой географической картой. Особенностью подхода, предложенного в работе, является использование в качестве математического аппарата графов, у которых множество ребер является нечётким с определением функции принадлежности ребра графу с помощью отображения

$$\mu : V^2 \rightarrow [0,1]$$

Решаемая задача связана с нечёткостью определения на географической карте объектов (дорог и полигонов) и, соответственно, расстояний между ними. Авторами предлагается переход от карты, к нечёткому графу, учитывая критерии, относящиеся к сетям дорог и полигонам, подлежащие оптимизации. Длина пути в графе вычисляется подобно обычному алгоритму для взвешенного графа (сумма весов дуг, входящих в путь, должна быть минимальной), но используются операции сложения и нахождения экстремумов для нечётких чисел. Ранжирование вершин используется при вычислении кратчайших путей в графах, соответствующих картам.

Похожий подход используется в работе [90, 91], в которой задаётся граф с нечёткими весами дуг и трапециевидной функцией принадлежности для решения задачи повышения скорости передачи данных в оптических сетях. Кратчайшие пути в этом случае ранжируются предлагаемым авторами алгоритмом. Результаты описанного вычислительного эксперимента показывают повышение производительности маршрутизации, которое связано с уменьшением времени на поиск оптимального пути и увеличением пропускной способности сети от 10 до 20%.

Ранжирование как один из этапов оптимизационного алгоритма часто используется в математическом моделировании систем видеонаблюдения. В работе [92] решается задача повторной идентификации человека в видеопотоке, который получается с нескольких камер видеонаблюдения. Полученное множество оценок видеоизображений людей преобразуется в графовую модель, для которой решается задача ранжирования, с помощью которой упрощается построение консенсусной

оценки. В [93] на основе ранжирования решается задача поиска скрытых объектов при визуальном отслеживании некоторой сцены.

3.3. Вычислительный эксперимент

Для проведения вычислительного эксперимента используются наборы данных, которые получены при анализе структурных характеристик виртуальных сообществ социальной сети ВКонтакте, схема проведённого эксперимента представлена на рис. 3.3.1.

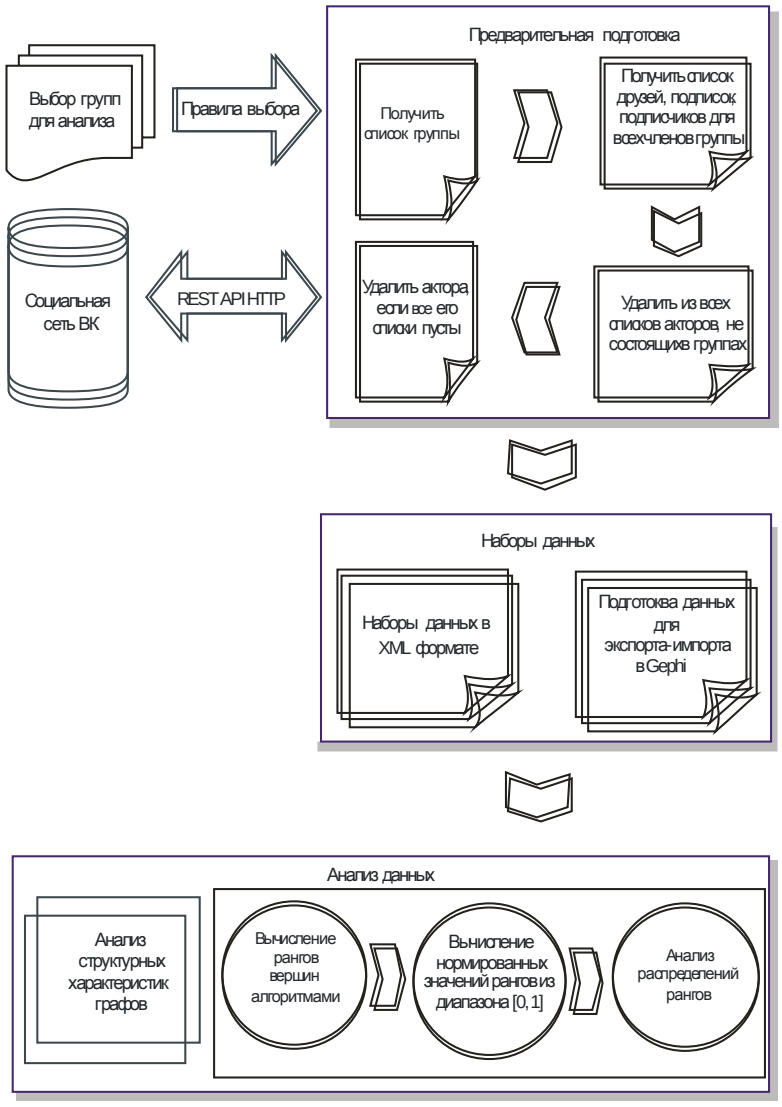


Рисунок 3.3.1 – Структура вычислительного эксперимента

Специально разработанное программное обеспечение было использовано для извлечения данных о пользователях групп, постах, комментариях и лайках к ним. Приложение вызывает методы HTTP интерфейса (REST HTTP API),

предоставляемого социальной сетью. Разработанная программа позволяет получить данные обо всех интересующих нас интеракциях пользователей в рамках одной группы. Для преобразования данных из объектной модели в XML файлы и обратно используется свободно-распространяемая библиотека JAXB. В качестве целевых выбирались группы с количеством членов порядка 10-30 тысяч.

Далее приводятся результаты анализа графов, которые получены для конкретных групп с условными названиями «Набор 1», «Набор 2», «Набор 3», имеющих от 10 до 24 тысяч участников, соответственно, вершин графов. Описание процедуры получения данных и некоторые структурные характеристики графов для соответствующих групп приведены в работе [94].

Сделаем несколько замечаний по поводу этапов подготовки и анализа данных. Последним шагом этапа предварительной подготовки данных является удаление всех изолированных вершин из полученных графов. Этот шаг позволяет “очистить” данные от вершин, которым могут быть присвоены минимальные ранги, так как они не связаны ни с одной вершиной графа. Такая процедура выглядит естественной и обоснованной.

Второе замечание касается представленных далее визуальных образов графов. В случае, когда размеры графов исчисляются десятками тысяч вершин, их визуальное представление позволяет увидеть структуру графа и структурные различия характерные для них. Здесь нужно отметить, что различные алгоритмы укладки графов могут сформировать принципиально различные визуализации, которые подчёркивают разные особенности их структурных характеристик. Для получения визуализации был использован алгоритм ForceAtlas 2, реализованный в открытой платформе для визуализации графов Gephi (https://gephi.org/), который объединяет несколько теоретических подходов к укладке. Этот алгоритм основан на минимизации энергии (вершины притягиваются или отталкиваются друг от друга в зависимости от их взаимного расположения и наличия связей), которая приписывается графу в целом, его вершинам и ребрам. Алгоритм хорошо подходит для построения изображений, подчеркивающих структуру графа, а также для визуализации подмножеств вершин с высокой степенью взаимодействия. Визуальное представление

графов приведено на рис. 3.3.2. Для каждого графа также указывается число вершин и компонент связности. Эти величины лишний раз подчёркивают различия в структурных характеристиках графов.

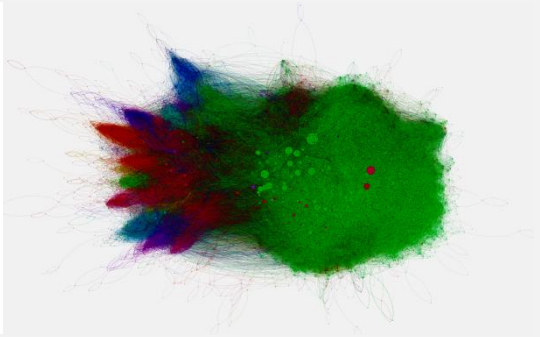
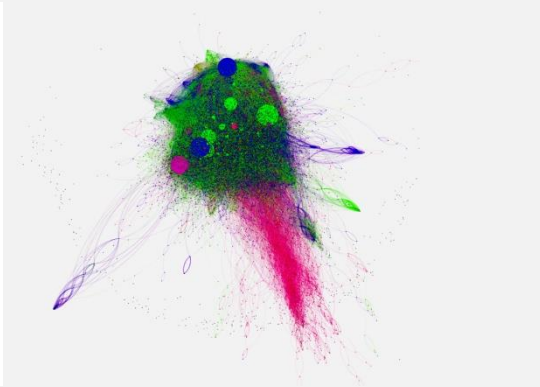
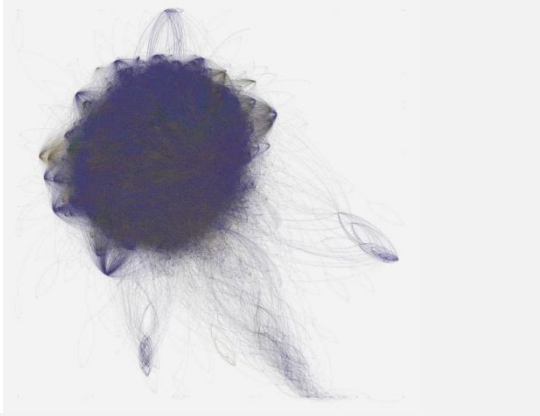
Набор 1 N = 23 651 Компонент связности: 77	
Набор 2 N = 23 979 Компонент связности: 399	
Набор 3 N =10464 Компонент связности: 121	

Рисунок 3.3.2 – Визуальное представление структуры трёх групп с использованием алгоритма “ForceAtlas 2”

Следующие далее графики на рис. 3.3.3 дают представление о виде распределения полустепеней исхода и захода для вершин соответствующего графа, представляющего первый набор данных «Набор 1». На рис. 3.3.3(а) представлено

распределение полустепеней захода, на рис. 3.3.3(b) – распределение полустепеней исхода. Для других наборов данных графики выглядят подобным образом.

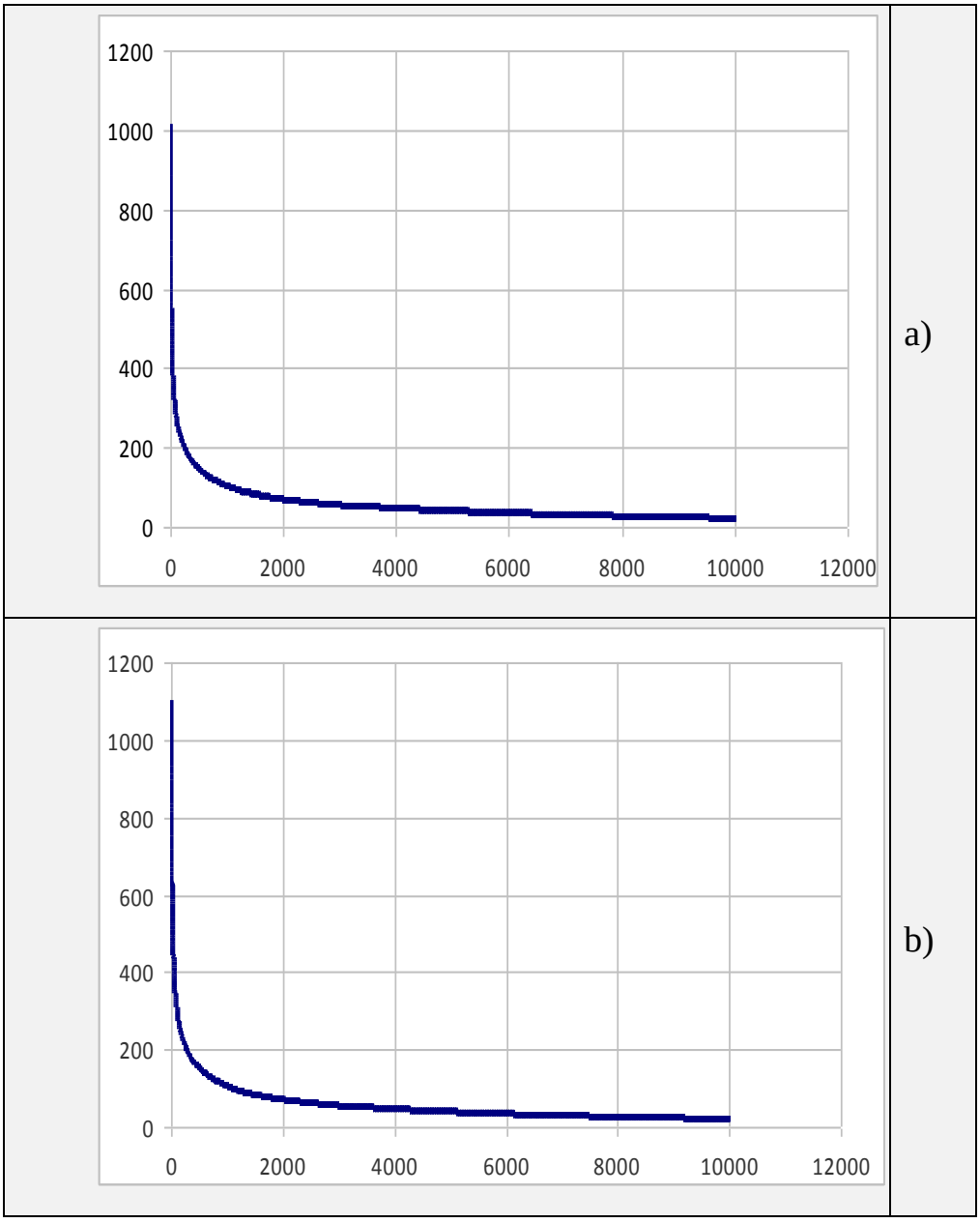


Рисунок 3.3.3 – Распределение полустепеней захода (a) и исхода (b) вершин графа «Набор 1»

3.3.1. Анализ алгоритмов ранжирования.

В качестве алгоритмов ранжирования, анализируемых далее, рассматриваются три алгоритма, которые были использованы для построения иерархий в различных ситуациях при анализе структуры ссылочного пространства глобальной сети или структуры социальных сетей. Краткое описание этих алгоритмов дано в работе [95].

В общем виде задача ранжирования, решаемая в рассматриваемых далее подходах применяется к моделям, описывающим взаимодействие объектов глобальной сети, которые представляют участников групп (вершин графа) социальной сети, либо страницы сети Интернет. Так как имеющиеся данные представляют участников групп социальной сети, будут описаны не только формулы, справедливые для определения рангов вершин, исходя из весов дуг, но и модифицированные формулы, которые учитывают такие стандартные действия участников групп как степень влияния одной входящей вершины на уровень доверия между двумя вершинами.

Первый алгоритм построения иерархии ранжирования вершин ориентированного взвешенного графа использует только характеристики их локального окружения и с точки зрения нашей терминологии является локальным [96]. Оригинальный подход решает задачу вычисления формально определенной меры доверия между пользователями микроблога Twitter и на этом основании моделирования и предсказания последовательности переходов при использовании сервиса. Степень доверия пропорциональна в этом случае количеству интеракций между двумя участниками группы (вершинами графа) и определяется только параметрами связывающих их дуг, то есть является локальной характеристикой в принятых нами обозначениях. Расчет меры доверия вершины j вершине i осуществляется по формуле (3.3.1):

$$r_{ij} = \frac{wS_{ij}+(1-w)F_{ij}}{N_i} \qquad (3.3.1)$$

, где

$S_{ij}=1$, если в дуге j имеется дуга i, и $S_{ij}=0$ в противном случае;

$F_{ij}=1$, если в дуге i имеется дуга j, и $F_{ij}=0$ в противном случае;

N_i – сумма полустепеней исхода и входа i;

w – коэффициент степени влияния S_{ij} и F_{ij} .

При проведении вычислений в работе рекомендуется принять значение w в диапазоне от 0,5 до 1. При использовании значения $w = 0,5$ приравнивается степень влияния наличия собственных подписок и подписчиков на итоговый результат.

Для учета степени влияния одной входящей вершины на уровень доверия между двумя вершинами необходимо дополнительно изменить формулу (3.3.2) расчета ранга. Для этого добавляются новые слагаемые в числитель формулы (3.3.2):

$$r_{ij} = \frac{wS_{ij}+(1-w)F_{ij}+cL_{ij}+(1-c)M_{ij}}{N_i} \quad (3.3.2)$$

где

c – коэффициент, степень влияния одной входящей вершины на уровень доверия между двумя вершинами;

L_{ij} – вес дуги (i, j);

M_{ij} – вес дуги (j, i).

В рамках проведённого вычислительного эксперимента коэффициент c принят равным 0,25.

Для вычисления ранга вершины i суммируются степени доверия ему всех вершин, которые имеют с ним интеракции. Таким образом, ранг пользователя будет рассчитываться по формуле (3.3.3):

$$r_i = \sum_{\forall j \in N_i} r_{i,j} \quad (3.3.3)$$

Далее обозначаем алгоритм ранжирования вершин, основанный на формуле (3.3.3), как ***TW***.

В работе [97] предлагается способ итеративного построения иерархии ранжирования, основанный на вероятностных характеристиках интеракций в социальных сетях. В этом случае учитываются не только интеракции между участниками сети и его локальным окружением, но и текущий усредненный уровень рангов вершин из этого окружения. То есть, на значение ранга на каждой итерации алгоритма влияют и внешние по отношению к его окружению интеракции. Следующая формула (3.3.4) используется для подсчета ранга вершины i:

$$r_i = k\frac{F_i - S_i}{N_i} + Q_i \quad (3.3.4)$$

где

k – постоянный коэффициент, значение которого в цитируемой статье принято равным 2,

F_i – множество вершин, которые ссылаются на вершинуi;

S_i – множество вершин, на которые ссылается вершина i;

N_i – множество всех вершин, с которыми взаимосвязана вершина i;

Q_i – усредненный ранг всех вершин, с которыми смежна вершина i.

Итерации алгоритма продолжаются до тех пор, пока не будет достигнута нужная точность (максимальная разница между значениями ранга вершины на двух последовательных итерациях не будет превышать наперед заданного значения ε. Если алгоритм не сходится, и разница между значениями рангов в последовательных итерациях не стремится к 0, то необходимо ограничить количество итераций. Максимальное количество итераций в нашем алгоритме было установлено в значение 500, ε = 0,1. Формальное определение сходимости будет дано далее.

Для учета степени влияния одной входящей вершины на уровень доверия между двумя вершинами добавляется дополнительное слагаемое в числителе формулы 3.3.4:

$$r_i = \frac{k(F_i - S_i) + \frac{k}{2}(M_i - L_i)}{N_i} + Q_i \qquad (3.3.5)$$

где

L_i – сумма весов дуг исходящих из вершины i,

M_i – сумма весов дуг, входящих в вершину i.

Далее алгоритм ранжирования, основанный на формуле (3.3.5), обозначается как ***FR***.

Третий, использованный в работе алгоритм ранжирования PageRank, описан в работе [98]. Итеративный алгоритм подсчитывает ранг каждой вершины, по следующей формуле:

$$r_i = c\sum\frac{r_j}{N_i} + cE_i \qquad (3.3.6)$$

где

r_i – ранг i–ой вершины,

c – коэффициент нормализации;

F_i – множество вершин, ссылающихся на вершину i.

S_i – множество вершин, на которые ссылается вершина i,

$N_i = S_i \cup F_i$ – множество всех вершин, с которыми взаимосвязана вершина I;

E_i – некоторое первоначальное значение ранга вершины. При описании алгоритма утверждается, что оптимальными значением этого параметра является 0,15.

Далее алгоритм ранжирования, основанный на формуле (3.3.6), обозначается как ***PR***.

Алгоритм ***PR*** и его модификации подробно рассматриваются в литературе, проанализированы преимущества и недостатки его использования. Но особенное внимание в последнее время уделяется практике использования модификаций ***PageRank*** для моделей данных, имеющих большую размерность. Для этого случая разработаны специальные алгоритмы, использующие только локальные фрагменты всей модели и распределённую среду вычисления [99, 100]. В работе [101] рассматривается ситуация вычисления иерархии ранжирования в случае, когда граф может содержать «определённые» и «неопределённые» дуги. Авторами применяется традиционная для таких задач терминология имеющая отношение к центральности. «Определённые» дуги являются традиционными для ориентированного графа с одной начальной и одной конечной вершинами. «Неопределённые» дуги имеют единственную вершину в качестве начальной, но множественные варианты конечной вершины, называемых «потенциальными» конечными. В качестве допустимого варианта потенциальной конечной рассматривается и вершина, которая отсутствует в заданном графе, но имеет специальное обозначение и добавлена в формальную математическую модель графа. Предложенный алгоритм ориентирован на динамический инкрементальный пересчет рангов при изменении графа в противовес статическому, когда вся совокупность рангов вычисляется заново при любом изменении. По утверждениям авторов работы достигается 16–кратное увеличение скорости построения корректной версии иерархии.

Подсчет рангов вершин графов с использованием алгоритма ***PR*** производился с помощью программы Gephi 0.82 beta. Два других индекса (***TW*** и ***FR***) рассчитывались с использованием программного обеспечения собственной разработки.

3.3.2 Сходимость алгоритма ранжирования для заданного графа.

Дадим определение понятия сходимости алгоритма ранжирования для заданного набора данных. Такое определение необходимо для корректного обсуждения временной сложности обсуждаемых алгоритмов, которая зависит не только от размерности исходных графов, но и от количества итераций, выполняемых алгоритмами. Последнее определяется либо наперёд заданным числом, либо выполнением некоторого условия, позволяющего прекратить вычисления.

Предполагаем, что алгоритм выполняет последовательное вычисление всей совокупности рангов вершин. Пусть $r^i = \left(r_1^i, r_2^i, \ldots, r_n^i\right)$ – набор рангов для вершин заданного графа *G*, полученный на *i*–ой итерации алгоритма. В качестве меры расстояния D^i между двумя последовательными наборами r^i и r^{i+1}, рассматриваемыми как вектора, удобно использовать метрику L_1 (метрика Манхэттена), определённую формулой (3.3.7):

$$D^i = d(r^i, r^{i+1}) = \sum_{j=1}^{n} \left| r_j^i - r_j^{i+1} \right| \qquad (3.3.7)$$

Более «мягким» вариантом метрики, которая предъявляет менее жёсткие условия к процедуре вычисления рангов, является метрика Чебышева, определяемая формулой (3.3.8)

$$D^i = d(r^i, r^{i+1}) = \max_{j=1,..,n} \left| r_j^i - r_j^{i+1} \right| \qquad (3.3.8)$$

Определение 2. Говорим, что алгоритм ранжирования сходится для заданного графа *G*, если

$$\lim_{i \to \infty} D^i = 0 \qquad (3.3.9)$$

Очевидно, что проблема сходимости (поточечной или равномерной) алгоритма ранжирования для заданного графа актуальна только для алгоритмов ***FR*** и ***PR***, так как они используют рекуррентную процедуру вычисления рангов.

Алгоритм ***TW*** при вычислении рангов вершин использует только характеристики их локального окружения и не требует пересчёта всей совокупности рангов при локальном изменении в графе.

Результаты вычислительного эксперимента. Диапазоны полученных рангов, которые вычисляются по представленным выше формулам, для данных различаются. При использовании алгоритма ***FR***, например, ранги могут становиться отрицательными из–за структуры формул (3.3.4), (3.3.5). По этой причине значения всех рангов были приведены к диапазону [0,1] преобразованием (3.3.10):

$$r_i^{'} = \frac{r_i - \min\limits_{i=1,N}(r_i)}{\max\limits_{i=1,N}(r_i) - \min\limits_{i=1,N}(r_i)} \qquad (3.3.10)$$

где максимум и минимум берутся по всему множеству рангов.

На рис. 3.3.4 (a, b, c) показаны гистограммы распределений рангов вершин для графа, представляющего группу «Волонтеры».

Гистограммы сформированы для совокупности рангов, полученных алгоритмами ***FR***, ***TW*** и ***PR***. Гистограммы сформированы для 3-х наборов данных.

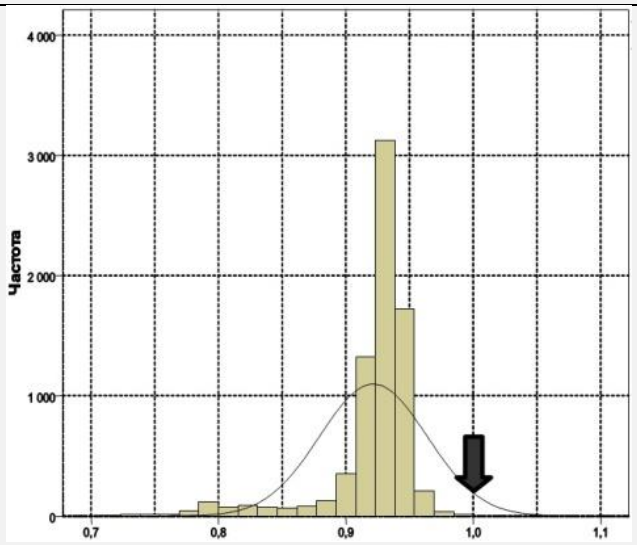
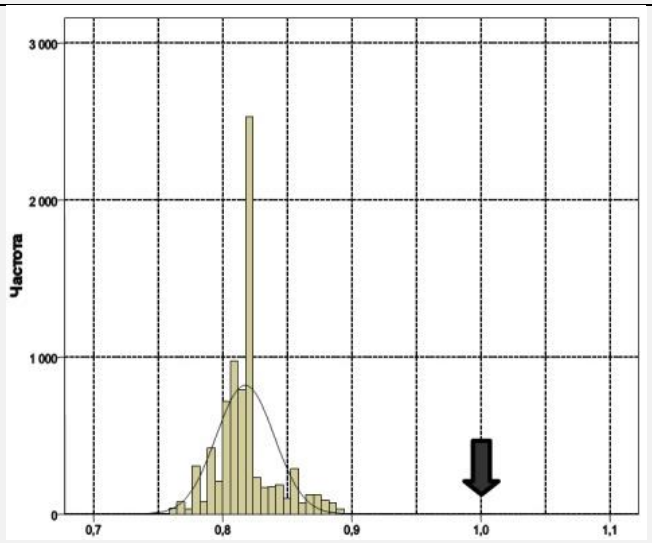
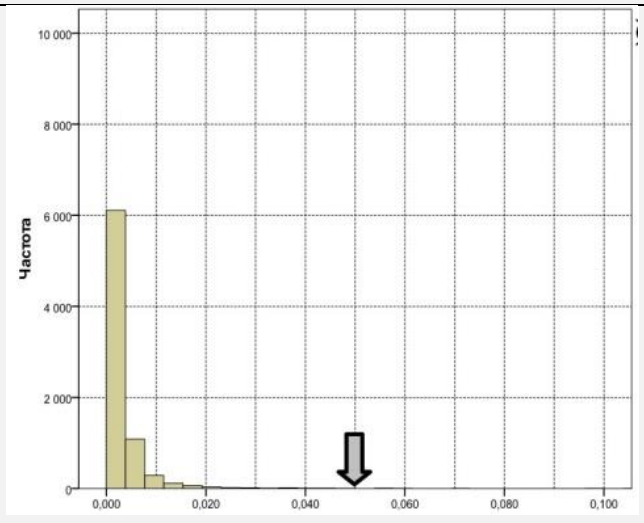
Набор 1	
	***FR*** (a)
	***TW*** (b)
	***PR*** (c)

Рисунок 3.3.4 – Гистограммы распределений рангов вершин графа «Набор 1» для FR(a), TW(b), PR(c). На a, b стрелка показывает положение 1, c – позиция значения 0,05.

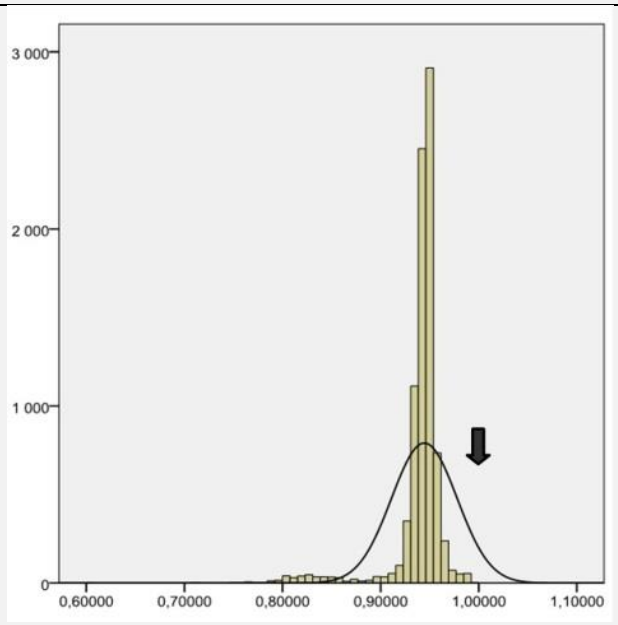
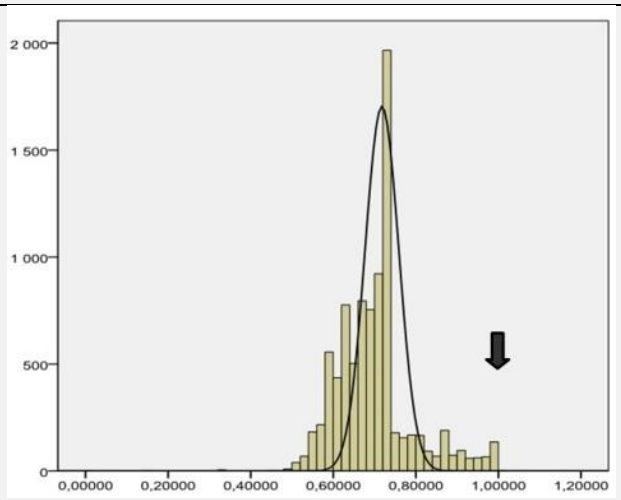
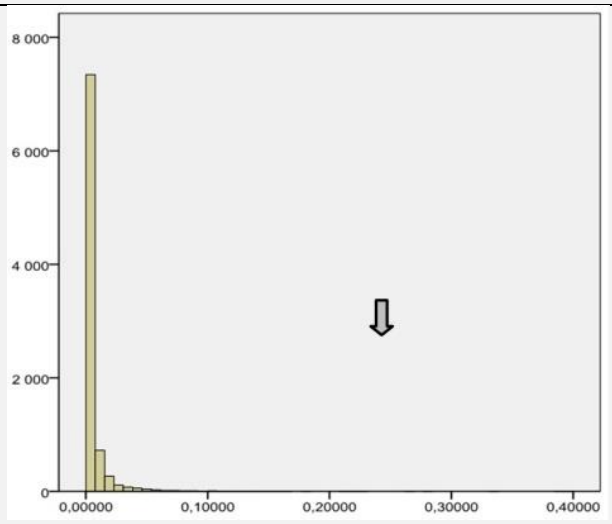
Набор 2	
	***FR*** (a)
	***TW*** (b)
	***PR*** (c)

Рисунок 3.3.5 – Гистограммы распределений для наборов данных «Набор 2» при использовании алгоритмов FR(a), TW(b), PR(c)

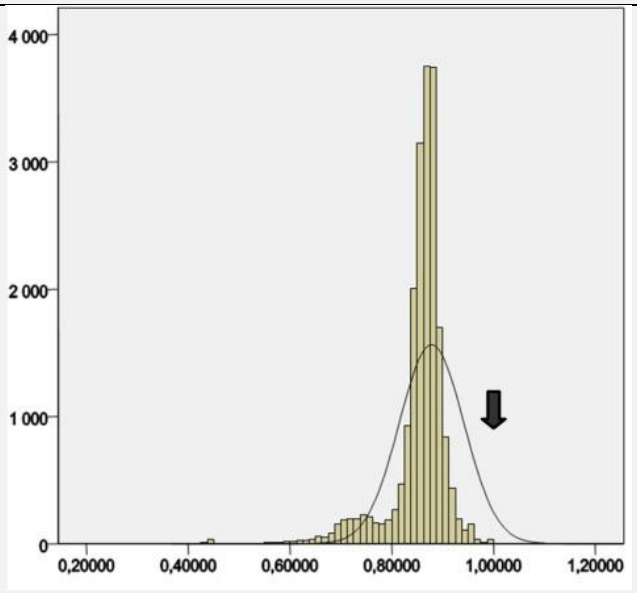
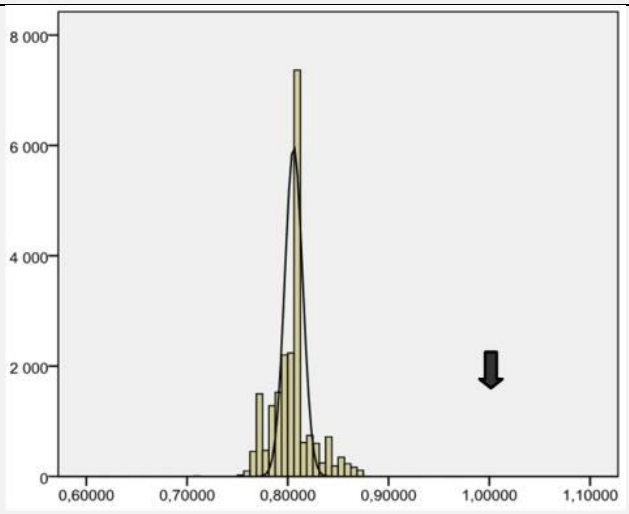
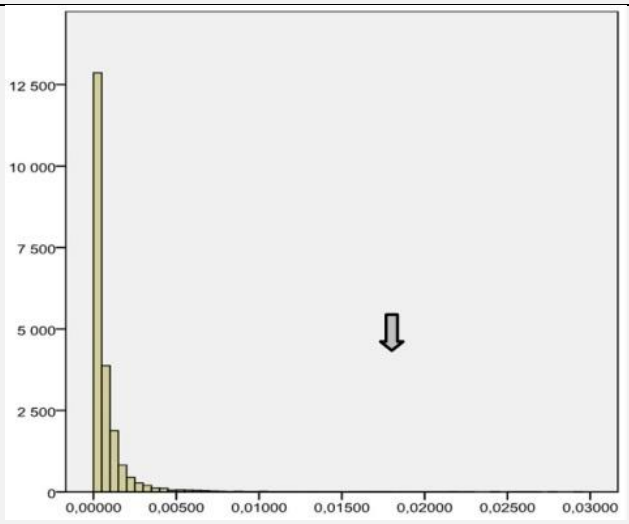
Набор 3	
	***FR*** (a)
	***TW*** (b)
	***PR*** (c)

Рисунок 3.3.6 – Гистограммы распределений для наборов данных «Набор 3» при использовании алгоритмов FR(a), TW(b), PR(c)

На графиках по оси абсцисс отложено нормализованное значение ранга вершин, на оси ординат – частота появления рангового диапазона. Для наборов данных «Набор 3» и «Набор 2» получены похожие результаты, демонстрирующие такие же закономерности в распределении полученных рангов. Гистограммы распределений для этих графов показаны на рис. 3.3.5 и 3.3.6 в уменьшенном масштабе, но позволяют подтвердить выявленные и описанные далее закономерности. Отдельно подчеркнём существенные различия в структурных характеристиках графов (количество компонент связности, статистики степенных последовательностей и некоторые другие), представляющих три набора данных, и описанных более подробно в указанной выше работе [102,103].

Распределение полученных рангов подчиняются сходным закономерностям, которые могут быть описаны следующими пунктами.

В описанных публикациях предложены решения оптимизационных задач, которые используют предварительное решение задачи ранжирования вершин. Этот подход позволяет повысить эффективность используемых алгоритмов, улучшить временные оценки используемых алгоритмов.

Имеется существенное отличие распределений рангов, генерируемых различными алгоритмами ранжирования. Если алгоритмы ***FR*** и ***TW*** в качестве результата строят ранги, имеющие распределение близкое к нормальному, то ранги, полученные алгоритмом ***PR***, имеют распределение близкое к экспоненциальному (см. рис. 3.3.4 – 3.3.6). Информация о типе распределения позволяет корректно выбирать параметрические тесты для анализа совокупности рангов.

Алгоритм ***FR*** дает совокупность рангов, распределение которых после нормирования смещено в сторону максимального значения 1, алгоритм же ***TW*** даёт меньшие значения распределения с меньшим разбросом значений. Этот вывод подтверждается данными табл. 1, которая позволяет сравнить средние значения, стандартное отклонение и скошенность распределений полученных рангов. Это различие можно объяснить тем, что алгоритм ***FR*** увеличивает ранги за счёт дополнительного слагаемого Q_i в формулах 3.3.4, 3.3.5, что приводит к сдвигу центра распределения в сторону максимального значения 1.

Таблица 3.3.1. Статистические параметры распределения рангов

		FR	***TW***	***PR***
Набор 1	Среднее	0,921	0,818	0,004
	Медиана	0,930	0,818	0,002
	Ст. отклонение	0,039	0,022	0,014
	Скошенность	-3,783	0,905	-
Набор 3	Среднее	0,854	0,804	0,001
	Медиана	0,866	0,808	0,000
	Ст. отклонение	0,057	0,021	0,002
	Скошенность	-2,697	0,553	-
Набор 2	Среднее	0,938	0,703	0,006
	Медиана	0,946	0,702	0,007
	Ст. отклонение	0,050	0,089	0,015
	Скошенность	-13,552	1,051	-

Алгоритмы ***FR*** и ***PR*** используют рекуррентную схему вычисления следующего приближения рангов, что актуализирует проблему сходимости алгоритма для заданного набора данных. Фактически, в нашем вычислительном эксперименте условием остановки процедуры вычисления рангов явилось достижение наперёд заданного максимального числа итераций. Алгоритм ***TW*** при вычислении ранга вершины использует только параметры её локального окружения и при динамическом изменении графа (добавление/удаление дуги, изменение её атрибутов) потребует только пересчета рангов двух вершин, которые соединены этой дугой. Все остальные ранги остаются неизменными. Это обстоятельство позволяет использовать эффективные инкрементальные/декрементальные процедуры вычисления рангов, сложность которых зависит только от размера локальных окружений вершин и сложности процедуры коррекции глобальной совокупности рангов.

3.4 Выводы по главе 3

В третьей главе диссертационного исследования проанализированы результаты применения алгоритмов ранжирования вершин на примере графов большой размерности (анализируемые алгоритмы обозначены в работе как TW, FR, PR). Скорость работы алгоритма ранжирования будет влиять на эффективность общей процедуры оптимизации, описанной в следующей главе. Показано, что процедуры ранжирования имеют большое прикладное значение и применяются в решении оптимизационных задач самой разнообразной природы. Проанализированные алгоритмы ранжирования имеют различную вычислительную сложность и отличаются видом статистического распределения полученных рангов. Если при решении прикладной задачи к распределению рангов предъявляются какие-либо требования (например, нормальность распределения является обязательным при использовании некоторых параметрических тестов), необходимо предварительно провести анализ результатов работы алгоритма, определить статистические характеристики полученного множества рангов.

При использовании предварительного ранжирования вершин графа в случае большой размерности модели исходных данных важным моментом является возможность использования инкрементальных алгоритмов, которые минимизируют сложность обработки данных, как правило, учитывают при построении глобального решения только локальные изменения, не требуют пересчёта всего множества рангов. В этом смысле алгоритм ***TW***, вычисляющий ранги только по данным локальной окрестности вершин, выгодно отличается от алгоритмов ***FR*** и ***PR***, которые требуют пересчёта всей совокупности рангов даже при локальном изменении графа. Это обстоятельство не позволяет использовать последние два алгоритма в случае, когда размерность модели является большой и необходимо получить решение в рамках жёстких временных ограничений. Распределённая модель вычислений смягчает это требование, но не устраняет его принципиально.

Алгоритмы ***FR*** и ***PR*** в проведённом вычислительном эксперименте выдавали совокупность рангов после предельного, наперед заданного числа итераций. В

исследовании предложено определение сходимости алгоритмов ранжирования, оставляя за скобками теоретический вопрос о критериях сходимости, определении нетривиальных структурных характеристик графов, наличие которых гарантирует достижение заданной точности вычисления рангов при выбранной мере расстояния, существование предела рекуррентной последовательности. Этот же вопрос возникает и по поводу структуры формул, используемых для последовательного вычисления рангов.

Результатом проведённого анализа является рекомендация выбора алгоритма ранжирования Lumbreras A. (**TW)** на предварительном этапе работы алгоритма оптимизации размещения средств наблюдения в трехмерной сцене. Данный алгоритм используется далее для предварительного ранжирования узлов трехмерной сцены, что позволит существенно сократить время работы всего алгоритма.

ГЛАВА 4. ТРЁХФАЗНЫЙ АЛГОРИТМ ОПТИМИЗАЦИИ ВЫБОРА РАЗМЕЩЕНИЯ СЕНСОРОВ НАБЛЮДЕНИЯ В РАЗРАБОТАННОМ КОНФИГУРАТОРЕ ПРОМЫШЛЕННЫХ ПОМЕЩЕНИЙ И ОБЪЕКТОВ ИНФРАСТРУКТУРЫ

Важной задачей при создании алгоритма и программного обеспечения автоматизированного проектирования комплексной системы видеонаблюдения является задача оптимизации размещения средств наблюдения на выделанной области для распознавания объекта за которым установлено наблюдение. Поскольку изображение практически любого объекта, передающееся через камеру наблюдения, зависит от многих факторов: его ориентации по отношению к камере, освещенности, параметров регистратора, статических и динамических параметров самого объекта, параметров сенсоров наблюдения, то изображение сложно формализовать и представить в виде определенной математической модели. Поэтому методы оптимизации размещения сенсоров наблюдения в трехмерной сцене существенно зависят от решаемых задач и редко поддаются обобщению, большинство этих методов являются нелинейными, что влияет на необходимость увеличения вычислительной мощности ЭВМ и сложности алгоритмов для обработки полученного по техническим каналам изображения [104]. Данное исследование посвящено оптимизации размещения сенсоров наблюдения в трехмерной сцене на основе реальной и реализации полученного алгоритма в виде программного приложения, работающего в режиме реального времени за приемлемое время.

В четвертой главе исследования описывается применение предложенных алгоритмов и трёх фаз(этапов) построения решения в разработанном программном комплексе оптимизации размещения камер наблюдения с максимизацией видимости выделенных зон или наблюдаемых объектов.

Оптимизация выбора размещения камер наблюдения является актуальной и важной задачей в оборонно-промышленной сфере для технических объектов, промышленных предприятий и критически важных объектов (АЭС, ГРЭС, ГЭС и т.п). Для решения данной задачи разработан программный комплекс «Конфигуратор

промышленных помещений и объектов инфраструктуры», предназначен для использования организациями, оказывающими услуги проектирования и установку систем видеонаблюдения и позволяет выполнять поддержку принятия решений за приемлемое время, создавать 3D-сцены на основе реального инфраструктурного объекта, а также оптимизировать выбор размещения на нём сенсоров наблюдения.

4.1 Конфигуратор промышленных помещений и объектов инфраструктуры

Разработанный программный комплекс «Конфигуратор промышленных помещений и объектов инфраструктуры», предназначен для использования организациями, оказывающими услуги проектирования и установки систем видеонаблюдения, для оптимизации выбора размещения сенсоров наблюдения. Полученные с применением эвристических алгоритмов поиска и предварительном ранжировании результаты, способствуют повышению эффективности выбора оптимальной конфигурации сенсоров наблюдения на наблюдаемом объекте. Таким образом, использование разработанного программного комплекса может способствовать поддержанию необходимого уровня безопасности на технических объектах, промышленных предприятиях и критически важных объектах.

Структура программного комплекса. Для моделирования трехмерной сцены и решения задачи оптимизации размещения сенсоров разработан программный комплекс, который позволяет строить и в реальном времени обновлять граф наблюдаемости, а также решать формализованную задачу (предусмотрены варианты получения оптимального решения или приближенного решения, полученного эвристическими алгоритмами).

- Инструментарий позволяет строить сложную виртуальную сцену большой размерности на основе реальных параметров, промешенных предприятия и объектов инфраструктуры и содержит следующие типы объектов:

- Цель. Представляет собой выделенная область или объект, расположенный на сцене, установление наблюдения за которым и является целью данного комплекса.

- Препятствия. Имитируют реальные препятствия на сцене, мешают установлению визуального контакта камеры с объектом-целью. Могут быть в виде геометрических трехмерных объектов, либо в виде поверхностей.

- Средства наблюдения. Могут иметь различные алгоритмы работы, основываясь на различных функция наблюдения.

- Источники освещения. Представляют собой объекты имитирующие источники освещения с учетом физических свойств реальных источников света.

Все классы, используемые программным обеспечением представлены в виде структуры на рис. 4.1.1.

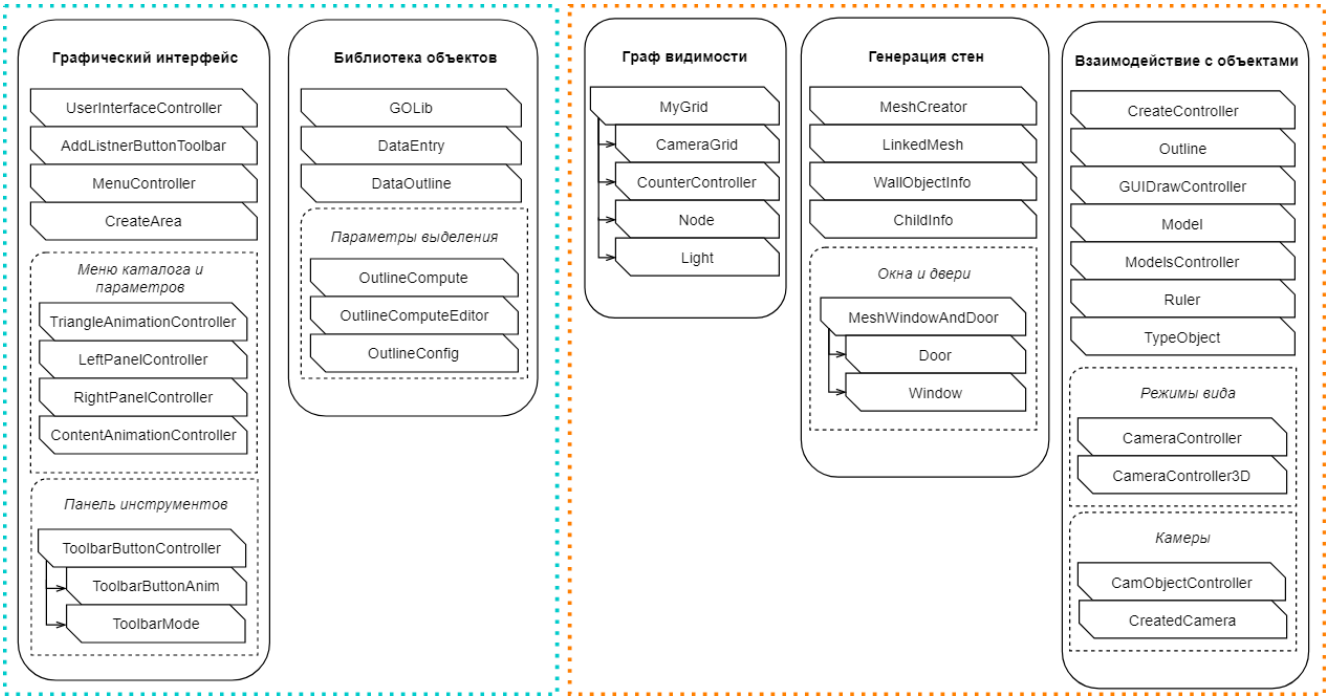


Рисунок 4.1.1 – Структура конфигуратора промышленных предприятий и объектов инфраструктуры

Программный комплекс содержит в себе несколько групп классов, опишем некоторые наиболее важные:

- Классы поддержки работоспособности графического интерфейса пользователя, состоящие из управляющих классов, классов меню параметров объектов и классов панели управления

- UserInterfaceController – отвечает за анимацию раскрытия. Содержит вызываемые методы по клику на кнопку пункта меню, по одному на каждую

конфигурацию дочерних объектов соответственно вариантам класса ContentAnimationController.

- CreateArea – отвечает за интерактивность рабочей области.
- LeftPanelController – работает с меню каталога, содержит метод, вызывающий анимацию выделения выбранного объекта.
- ContentAnimationController – реализует анимацию раскрытия пунктов в левом и правом меню. Существует в двух вариантах для следующих случаев: дочерние объекты расположены по сетке; есть единственный дочерний объект. Содержится в качестве компонента на пунктах.
- ToolbarButtonController – используется для переключения текущего инструмента в панели инструментов.
- ToolbarButtonAnim – отображает анимацию выделения инструмента.

- Классы, в которых расположены все 3D-объекты, помещаемые на виртуальную сцену

- GoLib – главный класс, который распределяет 3d-объекты по иерархии для упрощения и ускорения доступа к определённому объекту.
- OutlineConfig – компонент, отвечающий за выбор режима отображения обводки, появляющейся при выделении объекта. Выбранный режим Walls взаимодействует с компонентом Outline и изменяет просчёт геометрии для данного объекта.
- Outline – главный скрипт, добавляемый как компонент на все выделяемые трехмерные объекты, содержит визуальные параметры обводки.
- OutlineCompute – вспомогательный класс для выполнения вычисления контура обводки заранее, что позволяет оптимизировать работу конфигуратора, избегая выполнение данных вычислений в реальном времени. Подлежит временному добавлению на префабы объектов, удаляется после завершения вычислений.
- OutlineComputeEditor – вспомогательный класс, добавляет графические формы для работы класса OutlineCompute в среде Unity Editor.

- Классы для описания и построения сетки сцены. Сетка сцены предназначена для последующего формирования графа всей сцены и представляет собой набор узлов, расположение в пространстве которых зависит от способа построения сетки. На текущий момент реализована сетка «таблица», узлы которой в каждой из трех плоскостей равноудалены от своих соседей, а также сетка «проекция», в которой узлы спроецированы на поверхность сложного объекта (данная сетка предназначена для использования на неровном ландшафте).

  - MyGrid – основной класс для отрисовки сетки графа. Содержит ссылки на префабы, визуализирующие сетку и сферические узлы.

  - CameraCrid – формирует визуализацию лучей графа видимости от узлов камер к узлам сетки.

  - CounterController – вычисляет количество видимых узлов.

- Классы для генерации трёхмерной сетки в реальном времени, учитывающие реальные размеры с чертежей. Размеры возможно задавать при проектировании промышленных помещений и объектов инфраструктуры

  - MeshCreator и связанный с ним LinkedMesh, используемые для создания стен по необходимым параметрам и принимающие во внимание наличие отверстий для окон и дверей при перерисовке сетки. MeshCreator – компонент, входными параметрами которого являются минимальный угол между двумя соседними стенами и материал. Данные параметры могут быть переопределены компонентом CreateController. В поле T_parent компонента CreateController передаётся объект с компонентом Transform, данный объект будет являться родительским для создаваемых объектов.

  - MeshWindowAndDoor, а также Door и Window, для создания дверей и окон в уже имеющихся на сцене стенах.

- Классы взаимодействия с 3d-объектами, размещенными в виртуальной сцене

  - CreateController – компонент, созданный вручную, использующий скрипт с таким же названием. Для корректной работы компонент требует задать параметры. В поле Prefab Mesh Creator помещается объект или заранее созданный

шаблон объекта, сохраненный в папке проекта. Шаблоны объектов (префабы) используются для быстрого создания копий. Такой подход позволяет сократить количество объектов на сцене при запуске приложения, что повысит скорость работы и уменьшит объем занимаемой памяти.

- GUIDrawController – отображает прямоугольную область выделения, появляющуюся по перемещению мыши при зажатой левой кнопке.

- Model – класс, который в качестве компонента содержат все объекты из каталога, помещаемые на сцену.

- ModelsController – главный класс-контроллер, обеспечивающий правильное взаимодействие пользователя с объектами всех типов.

- Ruler – реализует работу инструмента "Линейка".

- CameraController – отвечает за возможность управления главной видовой камерой в 2D-режиме. Данный метод возвращает значение от -1 до 1, знак данного значения зависит от направления вращения колёсика мыши.

- CameraController3D – отвечает за изменение вида в трёхмерном режиме, позволяющий настроить скорость вращения камеры, скорость перемещения и чувствительность приближения с помощью изменения параметров Speed Cam Rotation, Speed Cam Move и Scroll Sensitivity соответственно.

- CamObjectController – отвечает за взаимодействие пользователя с объектами-камерами, в том числе за возможность их поворота вдоль двух осей.

- Главной особенностью разработанного программного комплекса является не только создание трехмерных объектов на основе реальных, но возможность построения решения оптимизации размещения средств наблюдения, далее представлены основные классы, связывающие наполненную виртуальную сцену с графом наблюдаемости узлов выделенной области (см. на рис. 4.1.2).

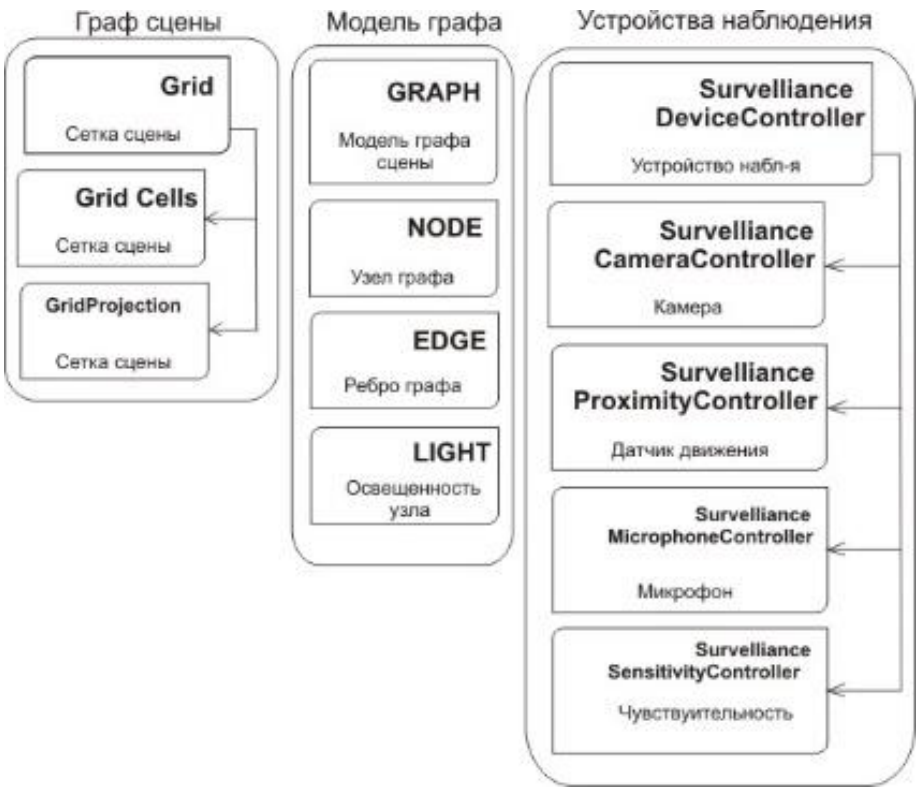


Рисунок 4.1.2 – Структура классов связывающих виртуальную сцену с графом видимости трёхмерной сцены

- Класс Scene позволяет строить необходимую нам сцену, а также управлять объектами на ней.

- Класс Scene_Object – главный родительский класс для всех объектов сцены. Позволяет производить «привязку» объекта к узлам сетки (определять к какому из узлов принадлежит объект).

- SurvellianceDeviceController. Базовый класс для всех сенсоров. Объекты класса имеют общие для всех видов сенсоров характеристики – пространственного положение, методы получения узлов сетки, которые «наблюдаются» данным устройством. Наследниками этого класса являются: SurveillanceCameraController, SurveillanceMicrophoneController, SurveillanceProximityController и SurveillanceSensitivityController, которые описывают логику работы сенсоров, соответственно, камеры, микрофона, датчика перемещения и датчика чувствительности. Все сенсоры имеют характеристики реальных устройств, учитывающиеся при определении наблюдаемости узлов этим сенсором.

- Классы, использующиеся для построения графа видимости.

- Класс Graph описывает граф видимости и используется при построении модели сцены.

- Классы Node, Edge, Light используются для описания узлов и ребер соответственно. Узлы хранят собственное положение в пространстве и список смежных ребер, имеющих определенный вес.

Классы логики предназначены для проведения вычислений и описывают алгоритм оптимизации для решения поставленной задачи определения оптимального расположения камер в пространстве.

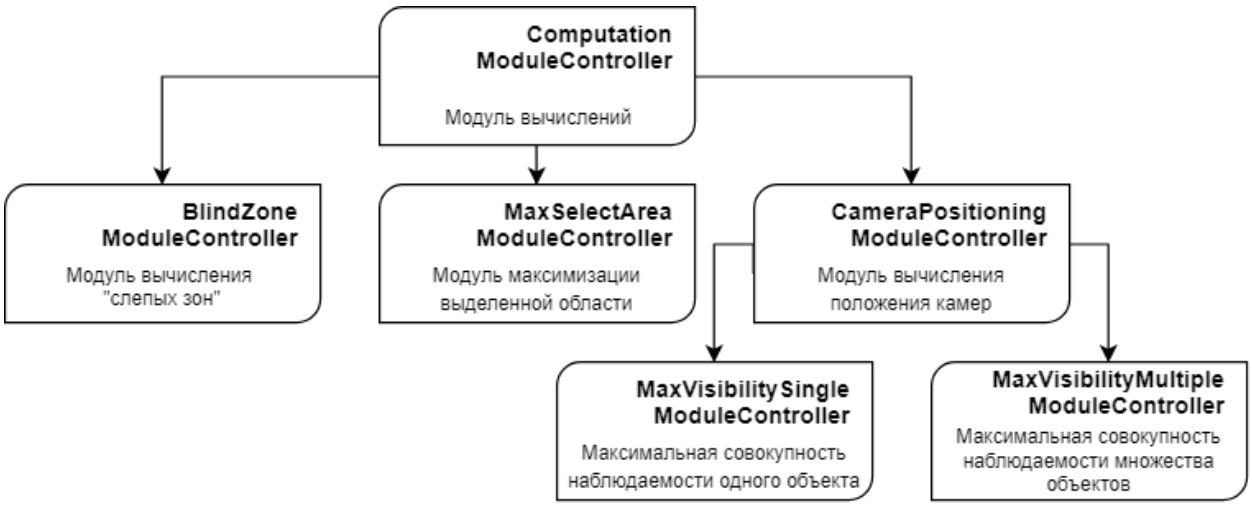


Рисунок 4.1.3 – Несколько модулей решения задачи оптимизации размещения средств наблюдения в трехмерной сцене

- Computation_Center содержит алгоритмы вычисления параметров и свойств графа (объект класса Graph).

- Статический класс Evaluation_Function содержит оценочные функции видимости объекта, которые используются при формировании графа, а именно при определении весов ребер, соединяющих сенсоры и цели.

- Класс Graph_Builder позволяет производить построение взвешенного ориентированного графа с применением имеющейся сетки сцены (Grid) и объектов сцены (Scene Object).

Алгоритмы решения задачи оптимизации размещения сенсоров наблюдения с различными критериями, например, вычислений «слепой зоны» для данной

конфигурации сенсоров размещаются в наследниках класса ComputationModuleController (Рис. 4.1.3). Данные классы также предназначены для вывода на экран интерфейса управления этими модулями.

Классы визуализации (Рис. 4.1.4) исполняют единственную функцию – создания графического представления имеющейся сцены, а также ее модели.

Рисунок 4.1.4 – Классы логики и визуализации

- Главный класс Scene UI для управления графическим представлением сцены и обеспечением взаимодействия с пользователем.
- Класс отрисовки GridRenderer, позволяющий создавать видимое представление графа и сетки сцены. Для отображения узлов и связей между ними, данный класс использует вспомогательный класс Node Connection Renderer.
- Вывод на экран визуального представления связи камеры с сеткой или графом осуществляется посредством класса SurveillanceDeviceRenderer, отображающего связи между камерой и видимыми узлами сетки сцены.

Разработанное программное обеспечение конфигуратора промышленных помещений и объектов инфраструктуры выполнено на платформе Unity, модуль конфигурации объектов и построение графа видимости узлов сцены реализованы с помощью языка программирования C# (см. приложение 1), графический интерфейс пользователя разработан в Adobe Photoshop, 3D-модели для библиотеки встраиваемых объектов смоделированы с помощью Autodesk 3Ds Max.

Программный комплекс позволяет оптимизировать расположение сенсоров в трехмерной сцене с различными критериями оптимизации, описанными в первой главе. Например, с целью минимизации «слепых зон» или максимизации покрытия выделенной области. При вычислении областей наблюдения сенсоров учитываются установленные параметры средств наблюдения (в случае камер – их угол обзора, дальность видимости, фокусное расстояние), что позволяет добиться возможности тонкой настройки и высокой точности расчетов.

4.2 Трёхфазный алгоритм оптимизации выбора размещения сенсоров наблюдения

Оптимизация размещения камер наблюдения на технических объектах, промышленных предприятиях, объектах инфраструктуры и критически важных объектах (АЭС, ГРЭС, ГЭС и т.п) является сложной задачей, а также необходимой для эффективного развертывания крупномасштабных сетей наблюдения. Существующие подходы к решению задачи размещения сенсоров наблюдения ограничены:

- количеством сенсоров наблюдения,
- планировкой объектов инфраструктуры,
- большой вычислительной сложностью,
- только одним критерием оптимизации,
- возможность контроля выделенной области,
- использованием источников освещения с реальными физическими свойствами.

По результатам диссертационного исследования описывается применение предложенных алгоритмов и трёх фаз построения решения в разработанном программном комплексе оптимизации размещения камер наблюдения с учетом существующих проблематик и возможностью решения различных задач оптимизации размещения средств наблюдения.

Перечислим различные задачи, которые будут отличаться как выбором алгоритма, так и сложностью их решения. Каждая из таких задач предполагает нахождение некоторого множества положений камер, которое является оптимальным с точки зрения критерия видимости. Опишем неформальною задачу, которую необходимо решить и приводить для нее формальную постановку в терминах графа видимости, определенного выше. Перечислим эти задачи.

1. Неформальная постановка задачи: определить положение камер, при котором достигается максимальная совокупная наблюдаемость одного объекта.

Формализация: Дана вершина v∈V. Определить конфигурацию камер, при которой видимость объекта является максимальной, то есть необходимо найти

$$\max_{\bar{a}\in\bar{A}} F(v,\bar{a}) \qquad (4.2.1)$$

2. Неформальная постановка задачи: определить положение камер, при котором достигается максимальная совокупная наблюдаемость множества объектов.

Формализация: Дана конфигурация объектов на сцене $\bar{v}\subseteq V$. Найти

$$\max_{\bar{a}\in\bar{A}} \sum_{v\in\bar{v}} F(v,\bar{a}) \qquad (4.2.2)$$

3. Неформальная постановка задачи: определить «слепые зоны» для текущей конфигурации камер.

Формализация: найти такую конфигурацию объектов $\bar{v}\subseteq V$ при которых выполняется

$$\forall \bar{a}\in\bar{A}\,(\sum_{v\in\bar{v}} F(v,\bar{a})=0) \qquad (4.2.3)$$

4. Неформальная постановка задачи: минимизировать количество камер наблюдения.

Формализация: определить минимальный набор {a1,a2,…,am} конфигураций камер $\bar{a}$, для всех вершин удовлетворяющих выражению.

$$\forall v\in\bar{v}\ \forall a_i\in\bar{a}\ (f(v,a_i)>0) \qquad (4.2.4)$$

Разработанный Трехфазный алгоритм оптимизации выбора размещения сенсоров системы наблюдения на технических объектах, промышленных предприятиях и объектах инфраструктуры позволяет решать данную задачу по различным критериям оптимизации, представленным выше и работает в 3 этапа(фазы).

Этап I

Создание 3D-сцены на основе реальной площадки (закрытого/открытого типа) согласно чертежам объекта и построение графа видимости узлов сетки(сцены) с

учетом физических свойств источников освещения, описанными в первой и второй главе исследования.

Построение графа видимости положений объектов в виртуальной сцене $G= (V \cup A, E, f)$ Формула (1.3.1), основанный на сетке, наложенной на наблюдаемую поверхность (возможны 2D и 3D сетки), где

– V – множество узлов сетки, соответствующее возможным позициям объектов на сцене;

– $A=\{Ai\}$ – множества узлов сетки, в которых могут располагаться камеры, Ai соответствует возможной позиции i–й камеры;

– E – множество ориентированных ребер графа, которые соединяют вершины из множества A с вершинами из V и определяют наличие прямой видимости узла из позиции камеры;

– $f: E \rightarrow R$, функция меры видимости позиции для конкретной камеры (вес дуги графа видимости, будет определена далее).

Формально, определяем

$$\bar{a} = \{a_1, a_2, \dots, a_k\} \underset{1 \le i \le k}{\forall} a_i \in A_i \qquad (2.3.3)$$

как набор, который представляет конфигурацию датчиков. Набор всех возможных конфигураций датчиков обозначается как $\bar{A}$.

Для расчета степени видимости узлов сетки используется модифицированная формула (2.3.6), учитывающая тени на трехмерной сцене:

$$f(v, a_i) = 1 + L_0 \times \frac{k_0 \times \tau \times N}{4 \times (1+m)^2 \times H^2} \qquad (2.3.4)$$

В качестве функции видимости узла сетки для конкретной камеры [77] используется соответствующее значение матрицы S **Формула** (2.3.2):

$$g(v, \bar{a}) = \frac{\sum_{i=1}^{k} f(v, a_i)}{k \times \max\limits_{v,\, a_i} f(v, a_i)} \qquad (2.3.5)$$

где $f(v_i, a_i)$ - значение видимости, рассчитанное для конкретной точки сцены (узла сетки), k-количество камер, с которых видна позиция V. Очевидно, что

$$\underset{v \in V}{\forall} \underset{\bar{a} \in \bar{A}}{\forall} \; 0 \le g(v, \bar{a}) \le 1 \qquad (2.3.6)$$

Этап II

На втором этапе работы трехфазного алгоритма происходит предварительное ранжирование узлов сетки(сцены) локально-ограниченным алгоритмом ранжирования Lumbreras A. (TW) см. формулу (3.3.2) для определения видимых узлов сетки(сцены), описанным в третьей главе.

Расчет ранга видимого узла сетки для алгоритма TW осуществляется по формуле

$$r_{ij} = \frac{wS_{ij}+(1-w)C_{ij}+cL_{ij}+(1-c)M_{ij}}{N_i}, \text{ где} \qquad (3.3.2)$$

S_{ij}=1, если в дуге j имеется дуга i, и S_{ij}=0 в противном случае; C_{ij}=1, если в дуге i имеется дуга j, и C_{ij}=0 в противном случае; N_i – сумма весов входящих дуг i; w – коэффициент степени влияния S_{ij} и C_{ij}, c – коэффициент, степень влияния одной входящей вершины на уровень доверия между двумя вершинами; L_{ij} – вес дуги (i, j); M_{ij} – вес дуги (j, i).

Этап III

Решение задачи оптимизации размещения сенсоров наблюдения по одному из выбранных критериев оптимизации на основе эвристического алгоритма муравьиной колонии (R. Parpinelli) см. формулу 2.2.4 с учетом параметров функции видимости, блок схема работы алгоритма муравьиной колонии представлена на рис.4.2.1.

$$P_{ij}(t) = \frac{\tau_{ij}(t) * \eta_{ij}}{\sum_i^a \sum_j^{b_i} \tau_{ij}(t) * \eta_{ij}, \forall i \in I} \qquad (2.2.4)$$

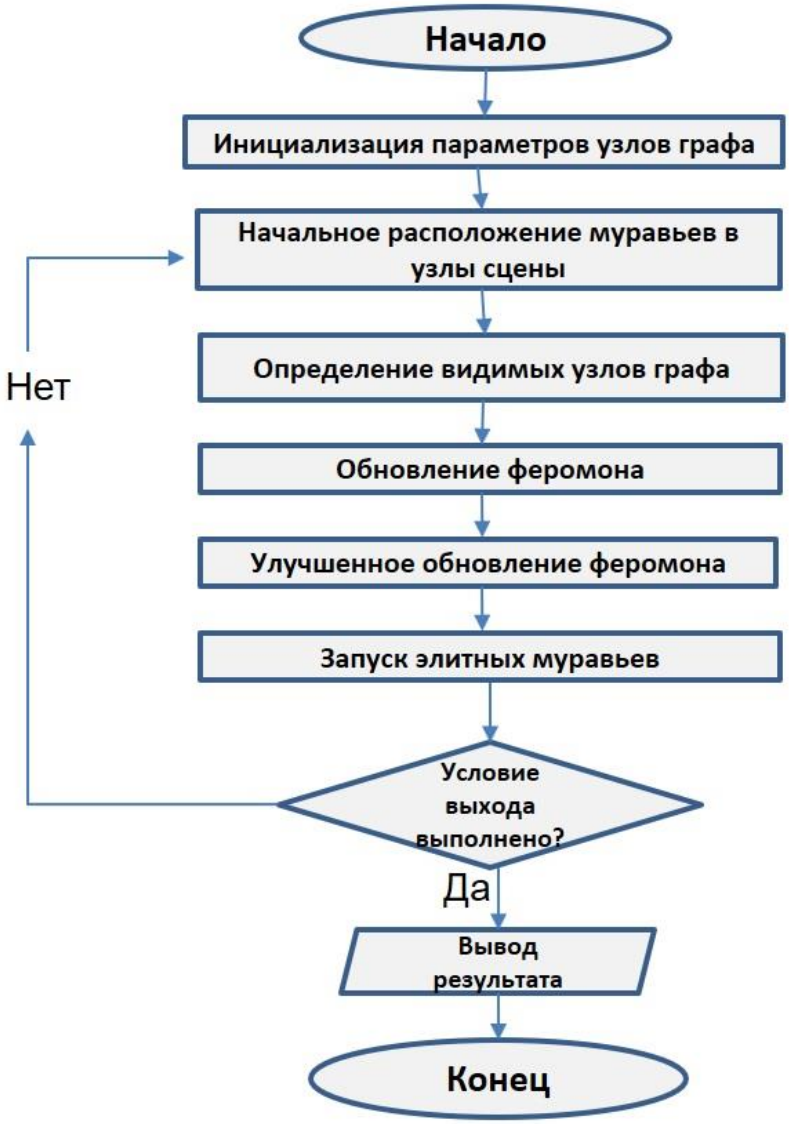


Рисунок 4.2.1 – Блок схема алгоритма муравьиной колонии

По результатам выполнения 3-х этапов работы трёхфазного алгоритма, строится граф видимости с наилучшим соотношение видимости узлов сетки и времени просчета виртуальной сцены. В рамках вычислительного эксперимента (представленного далее), сравнения работы трех алгоритмов: полного перебора(ПП), муравьиной колонии(МК) и разработанного трехфазного алгоритма(ТАО), было доказано сокращение времени просчета 3D-сцены в ходе решения задачи оптимизации размещения камер наблюдения по одному выбранному критерию оптимизации из предложенного ранее перечня.

4.3 Вычислительный эксперимент оптимизации размещения сенсоров наблюдения с целью максимизации видимости выделенной области

Для апробации разработанного трехфазного алгоритма, была построена 3D-сцена на основе реальной площадки производственного здания государственной районной электростанции (ГРЭС) с расположенным внутри здания производственным оборудованием. В производственном корпусе ГРЭС была выделена зона первого этажа здание, на котором расположено основное (критически важное) оборудование, за которой необходимо вести наблюдение. В рамка вычислительного эксперимента для контроля выделенной области были установлены 33 камеры наблюдения (рис. 4.3.1), в качестве препятствий выступало производственное оборудование и различные геометрические конструкции, расположенные над оборудованием. Критерием оптимизации при данной постановке является максимизация видимости выделенной области на критически важном объекте.

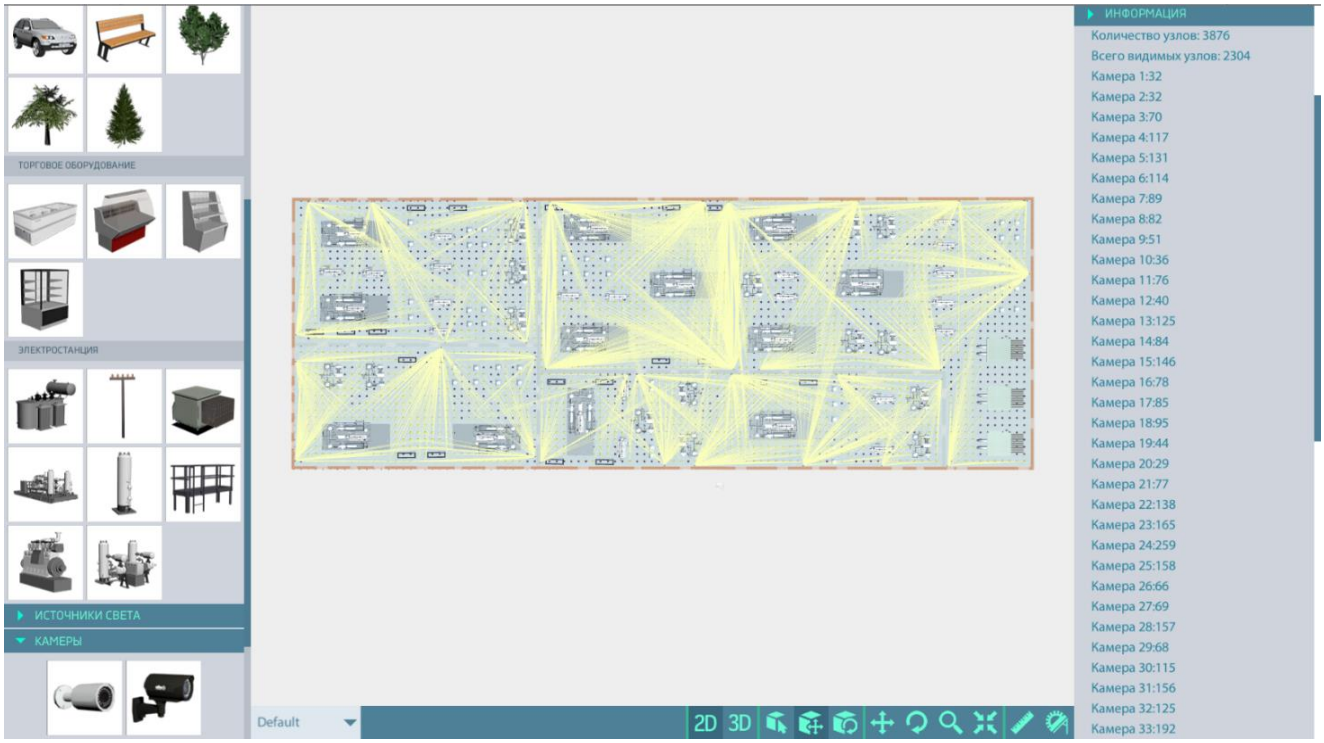


Рисунок 4.3.1 – Производственный корпус ГРЭС с расположенными оборудованием и камерами наблюдение. Вид сверху

При решении задачи оптимизации размещения камер наблюдения с максимизацией видимости выделенных зон, на наблюдаемую зону, произведено наложение сетки с шагом 1,5 м, общая площадь, первого этажа производственного корпуса

равна 4800 м^2. После построения и наложения сетки на выделанную область здания, автоматически строится граф видимости узлами сенсоров наблюдения узлов сетки(рис.4.3.2) с выводом общего количества узлов сцены 3876.

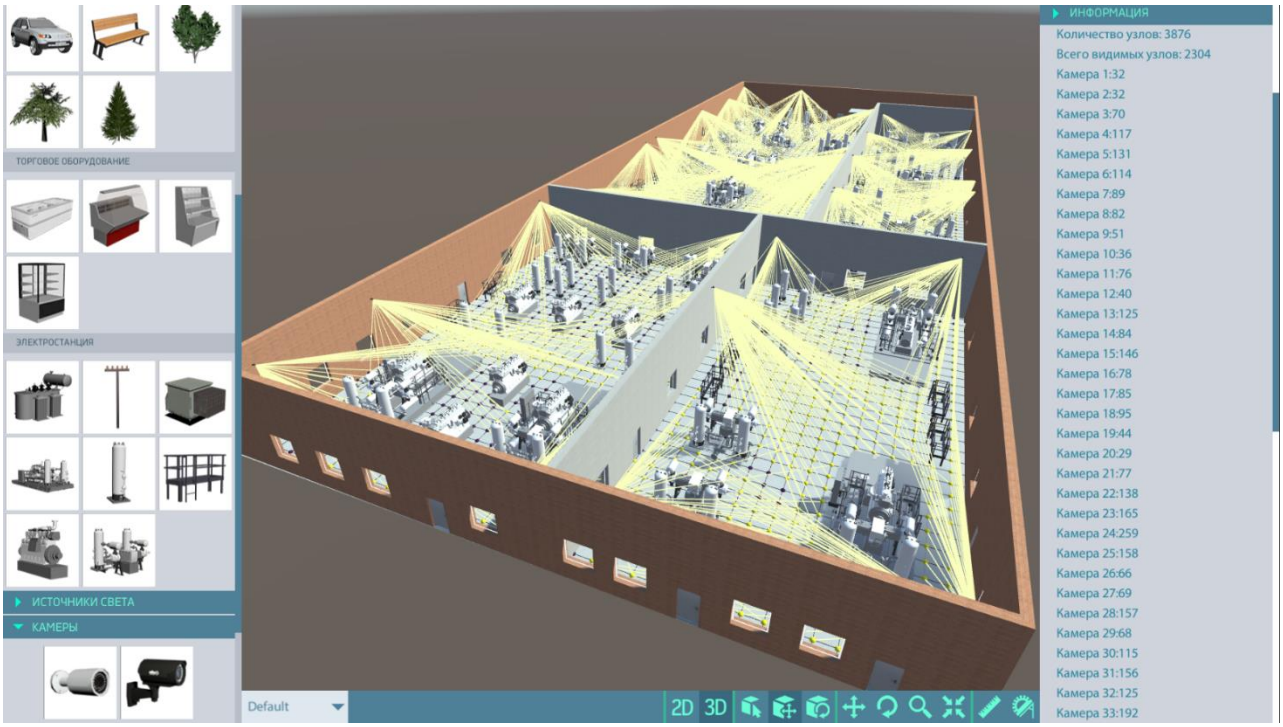


Рисунок 4.3.2 – Производственный корпус ГРЭС с расположенными оборудованием и камерами наблюдение. Вид с перспективы

В рамках анализа решений задачи дискретной оптимизации размещения сенсоров наблюдения в трехмерной сцене, в первой главе исследования, отмечено, что при расположении сенсоров наблюдения на технических объектах большой размерности, используются алгоритмы аппроксимации и эвристические алгоритмы поиска, но на графах большой размерности, возникающие при уменьшении шага сетки, менее 2 кв. метров, возрастает вычислительная мощность, что приводит к невозможности проведения эксперимента.

При решении отмеченной проблемы, предлагается, в качестве узлов сетки трехмерной сцены использовать низкополигональные (имеющие низкое количество полигонов) сферические объекты (рис.4.3.3), тем самым уменьшая общее количество полигов сцены и соответственно, уменьшается вычислительная мощность программного обеспечения, позволяя регулировать шаг сетки менее 2 кв. метров.

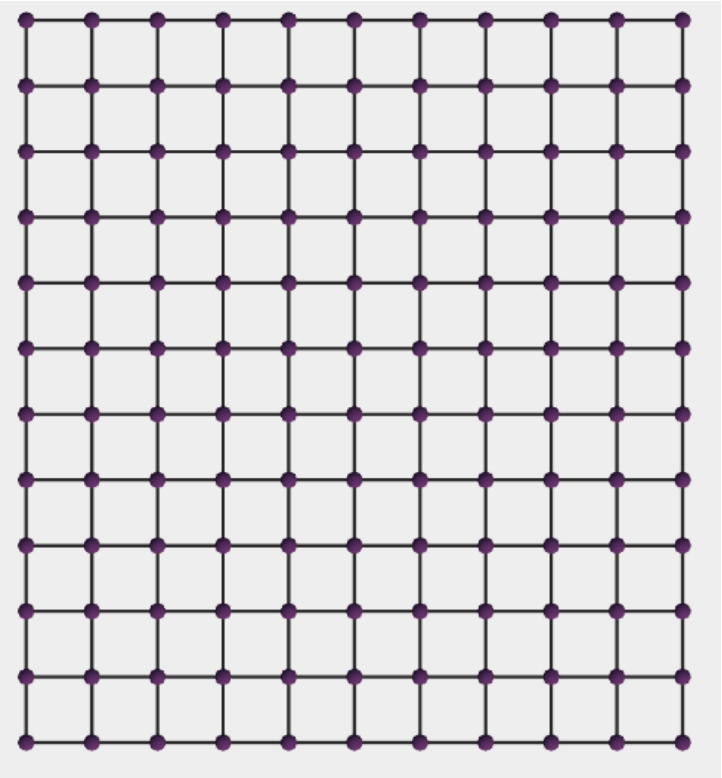


Рисунок 4.3.3 – Отображение сетки (узлов сцены) с помощью низкополигональных сферических объектов

В рамках исследования проведен вычислительный эксперимент по сравнению работы трех алгоритмов поиска оптимального расположения сенсоров наблюдения для одной конфигурации сенсоров

- классический алгоритм полного перебора (ПП),
- классический эвристический алгоритма «Муравьиной колонии» (МК),
- разработанный в рамках диссертационного исследования трехфазный алгоритма оптимизации размещения сенсоров наблюдения (ТАО).

При решении поставленной задачи, разработанный конфигуратор промышленных помещений и объектов инфраструктуры позволяет произвести сравнение выполнения выбранных алгоритмов с возможность определения видимых узлов и время выполнения, как для каждой камеры отдельно, так и совокупную видимость узлов сцены (см. рис. 4.3.4 – 4.3.6).

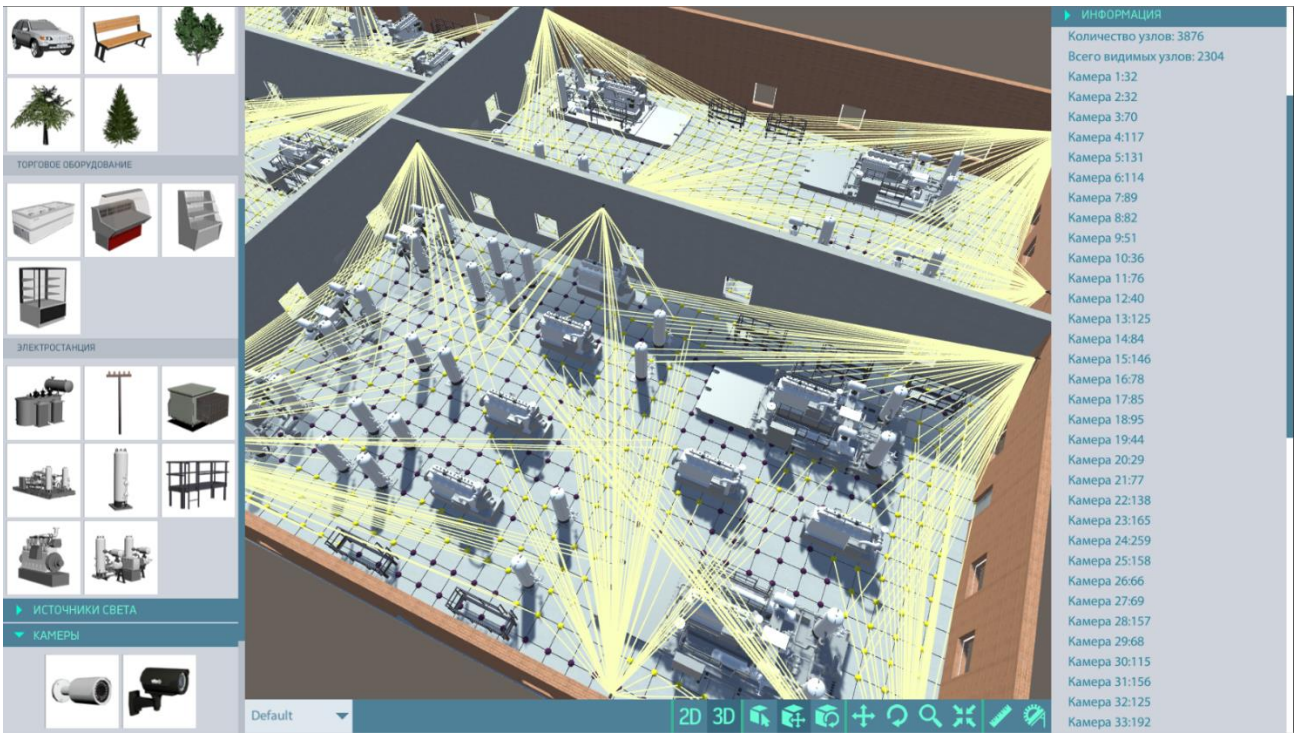


Рисунок 4.3.4 – Результаты выполнения классического алгоритма полного перебора для одной конфигурации(позиции) сенсоров наблюдения (ПП)

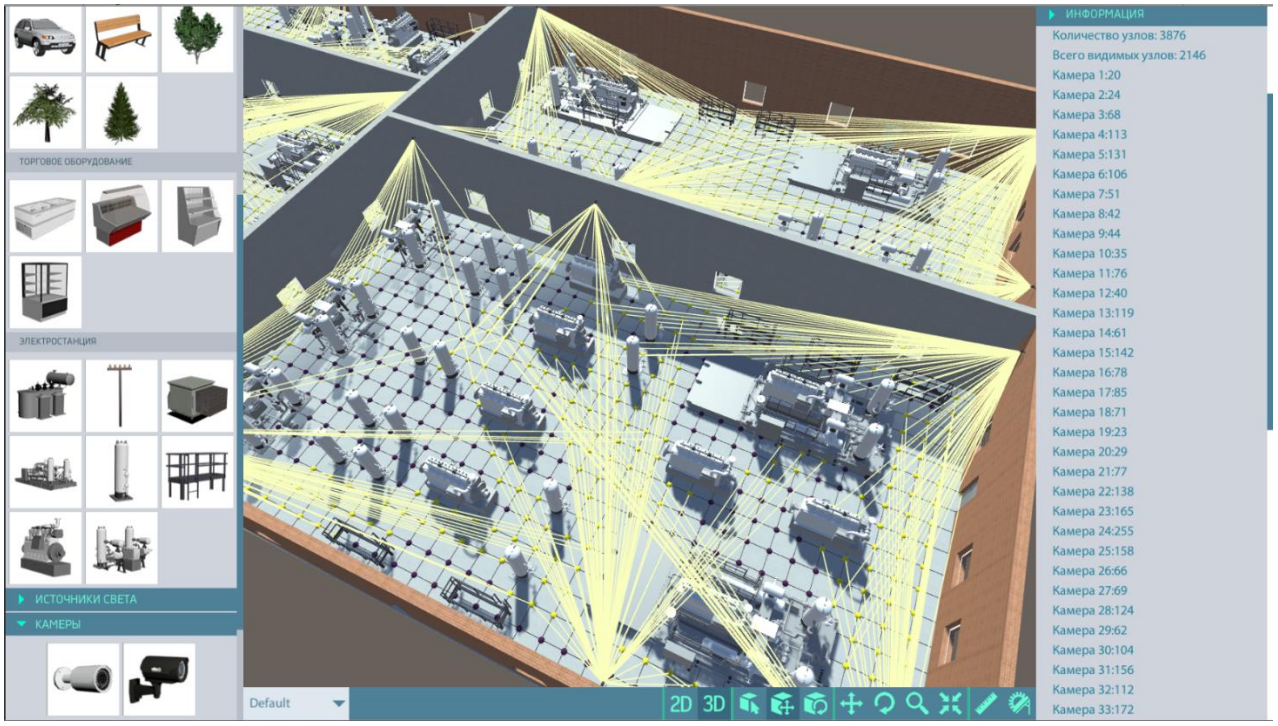


Рисунок 4.3.5 – Результаты выполнения классического алгоритма муравьиной колонии для одной конфигурации(позиции) сенсоров наблюдения (МК)

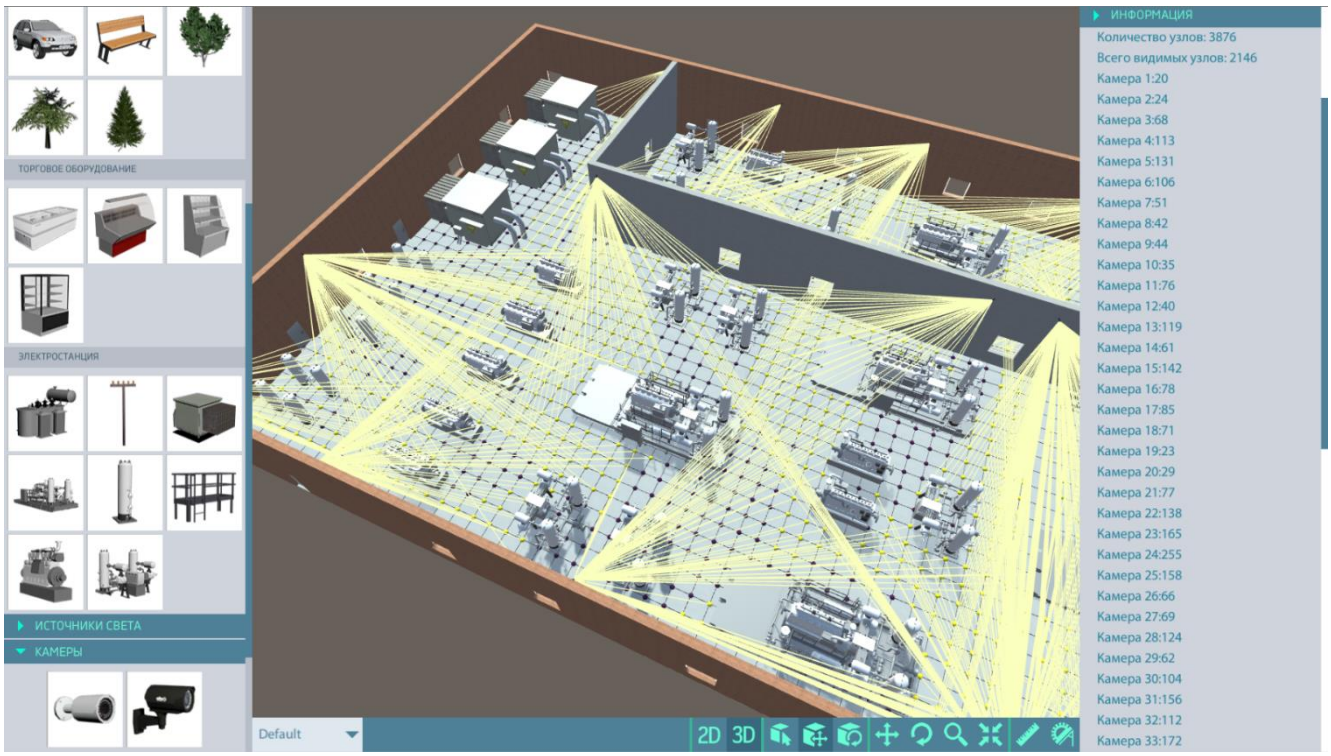


Рисунок 4.3.6 – Результаты работы разработанного трехфазного алгоритма оптимизации расположения сенсоров наблюдения для одной конфигурации(позиции) (ТАО)

Результаты вычислительного эксперимента по оптимизации размещения сенсоров наблюдения для одной конфигурации сенсоров на техническим (критически важном) объекте – производственном корпусе ГРЭС, представлены в виде сводной таблице (см. табл. 4.3.1) с распределением видимых узлов сцены и времени выполнения алгоритмов, как для каждой камеры отдельно, так для совокупного множества сенсоров наблюдения.

Таблица 1 – Результаты выполнения сравнения выполнения 3-х алгоритмов в разработанном программном комплексе «Конфигуратор промышленных помещений и объектов инфраструктуры» для одной конфигурации(позиции) сенсоров наблюдения

Площадка	Вычислительное время (миллисекунды)			Коэффициент охвата узлов графа (%)			
Производственное здание ГРЭС	ПП	МК	**ТАО**	ПП	МК	**ТАО**	Общее кол-во узлов сцены
Камера 1	31,724	20,620	**14,275**	0,83% 32	0,52% 20	**0,52% 20**	3876 100%
Камера 2	31,724	20,620	**14,275**	0,83% 32	0,62% 24	**0,62% 24**	3876 100%
Камера 3	48,785	31,7102	**21,953**	1,81%	1,75%	**1,75%**	3876

				70	68	**68**	100%
Камера 4	72,844	47,348	**32,779**	3,02% 117	2,92% 113	**2,92% 113**	3876 100%
Камера 5	76,648	49,821	**34,491**	3,38% 131	3,38% 131	**3,38% 131**	3876 100%
Камера 6	71,455	46,445	**32,154**	2,94% 114	2,73% 106	**2,73% 106**	3876 100%
Камера 7	62,124	40,380	**27,955**	2,30% 89	1,32% 51	**1,32% 51**	3876 100%
Камера 8	60,487	39,316	**27,219**	2,12% 82	1,08% 42	**1,08% 42**	3876 100%
Камера 9	53,874	35,018	**24,243**	1,32% 51	1,14% 44	**1,14% 44**	3876 100%
Камера 10	33,478	21,760	**15,065**	0,93% 36	0,90% 35	**0,90% 35**	3876 100%
Камера 11	51,754	33,640	**23,289**	1,96% 76	1,96% 76	**1,96% 76**	3876 100%
Камера 12	35,874	23,318	**16,143**	1,03% 40	1,03% 40	**1,03% 40**	3876 100%
Камера 13	75,141	48,841	**33,8134**	3,22% 125	3,07% 119	**3,07% 119**	3876 100%
Камера 14	61,152	39,748	**27,518**	2,17% 84	1,57% 61	**1,57% 61**	3876 100%
Камера 15	83,541	54,301	**37,593**	3,77% 146	3,66% 142	**3,66% 142**	3876 100%
Камера 16	52,574	34,173	**23,658**	2,01% 78	2,01% 78	**2,01% 78**	3876 100%
Камера 17	61,542	40,002	**27,693**	2,19% 85	2,19% 85	**2,19% 85**	3876 100%
Камера 18	65,874	42,818	**29,643**	2,45% 95	1,83% 71	**1,83% 71**	3876 100%
Камера 19	36,412	23,667	**16,385**	1,14% 44	0,59% 23	**0,59% 23**	3876 100%
Камера 20	30,784	20,009	**13,852**	0,75% 29	0,75% 29	**0,75% 29**	3876 100%
Камера 21	51,987	33,791	**23,394**	1,99% 77	1,99% 77	**1,99% 77**	3876 100%
Камера 22	79,541	51,701	**35,793**	3,56% 138	3,56% 138	**3,56% 138**	3876 100%
Камера 23	91,423	59,424	**41,140**	4,26% 165	4,26% 165	**4,26% 165**	3876 100%
Камера 24	168,421	109,473	**75,789**	6,68% 259	6,58% 255	**6,58% 255**	3876 100%
Камера 25	89,874	58,418	**40,443**	4,08% 158	4,08% 158	**4,08% 158**	3876 100%
Камера 26	47,252	30,713	**21,263**	1,70% 66	1,70% 66	**1,70% 66**	3876 100%
Камера 27	48,122	31,279	**21,654**	1,78% 69	1,78% 69	**1,78% 69**	3876 100%
Камера 28	90,111	58,572	**40,549**	4,05% 157	3,20% 124	**3,20% 124**	3876 100%

Камера 29	47,785	31,060	**21,503**	1,75% 68	1,60% 62	**1,60% 62**	3876 100%
Камера 30	76,214	49,539	**34,296**	2,97% 115	2,68% 104	**2,68% 104**	3876 100%
Камера 31	89,758	58,342	**40,391**	4,02% 156	4,02% 156	**4,02% 156**	3876 100%
Камера 32	75,141	48,841	**33,813**	3,22% 125	2,89% 112	**2,89% 112**	3876 100%
Камера 33	78,586	51,080	**35,363**	4,95% 192	4,44% 172	**4,44% 172**	3876 100%
Камера 1-33	2475,145	1608,844	**1113,815**	59,34% 2304	55,37% 2146	**55,37% 2146**	3876 100%

По результатам вычислительно эксперимента отметим, что наилучшую совокупную видимость узлов сетки для одной конфигурации сенсоров наблюдения показал алгоритм ПП, алгоритмы МК и предложенный ТАО, уступают в порядке 4-5%. Однако, при потере незначительно числа узлов графа видимости, получилось существенно ускорить процесс вычисления алгоритма в сцене. Так, разработанный ТАО выбора размещения сенсоров системы наблюдения на технических объектах, просчитал видимые узлы графа в 2,5 раза быстрее (алгоритма ПП).

Приведённые в таблице данные соответствуют обсчёту одной конфигурации расположения камер. Для вычисления оптимальной конфигурации необходимо выполнить полный перебор всех конфигураций. В нашем случае число таких конфигураций равно 3^{33}, что является следствием экспоненциальной сложности задачи. Временные характеристики вычислительного эксперимента отображены на рис. 4.3.7.

Предложенный алгоритм ТАО, на определение наилучших позиций для 33-х камер наблюдения затратил чуть больше 1,5 минут. Алгоритму МК для выбора наилучшей позиции для каждой из камер понадобилось порядка 6 минут. Алгоритму ПП необходимо перебрать все варианты размещения 33 камер наблюдения в 3 позициях, что приводит к экспоненциальному росту времени вычисления и возможно только для небольшого количества камер.

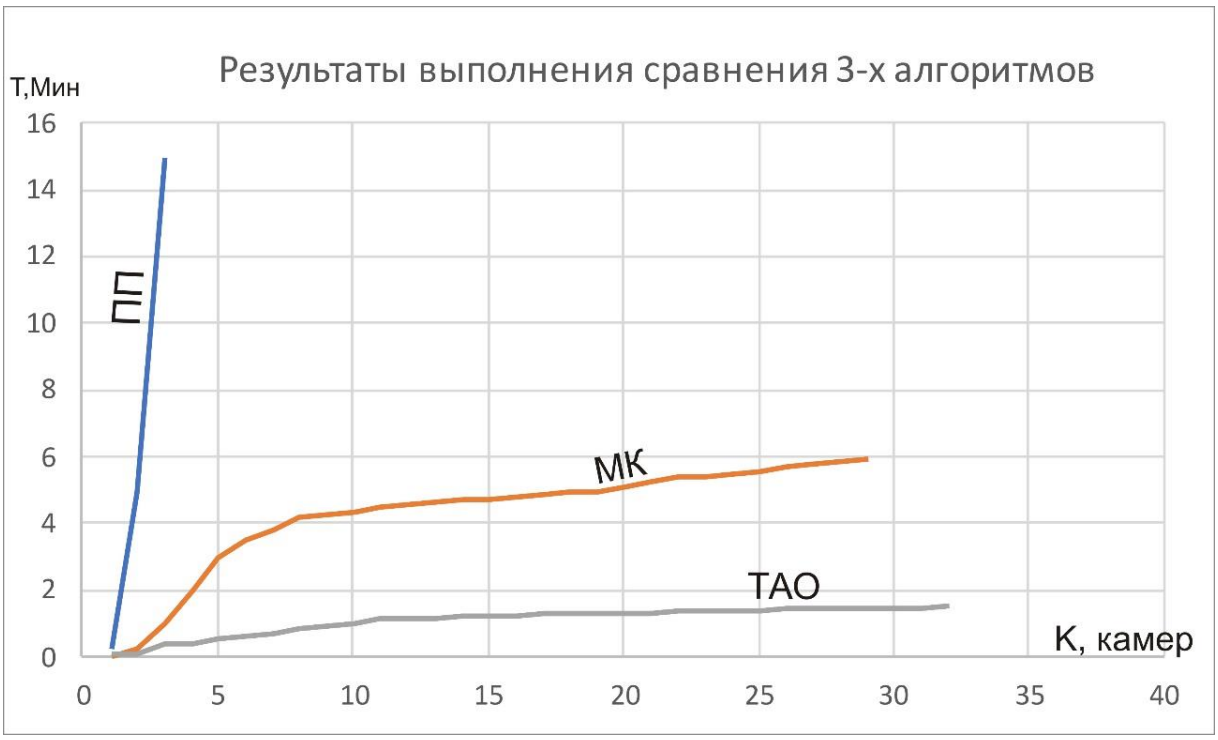


Рисунок 4.3.7 – Результаты работы разработанного трехфазного алгоритма оптимизации расположения сенсоров наблюдения для 3-х конфигураций(позиций) сенсоров наблюдения

Было доказано сокращение времени просчета 3D-сцены в ходе решения задачи дискретной оптимизации размещения камер наблюдения на техническом (критически важном) объекте – производственный корпус ГРЭС, с максимизацией видимости выделенных зон. Программный комплекс «Конфигуратор промышленных помещений и объектов инфраструктуры», предназначен для использования организациями, оказывающими услуги проектирования и установки системы видеонаблюдения. Полученные с применением трехфазного алгоритма оптимизации размещения сенсоров наблюдения(ТАО) (основанного на эвристических алгоритмах поиска и предварительном ранжировании) результаты, способствует повышению эффективности выбора оптимальной конфигурации сенсоров наблюдения на наблюдаемом объекте. Таким образом, использование разработанного программного комплекса может способствовать поддержанию необходимого уровня безопасности инфраструктурных технических объектах.

4.4 Выводы по главе 4

В настоящее время в теории и практике оптимизации выбора размещения средств наблюдения на технических объектах большую актуальность приобрела проблема построения адекватных математических моделей и алгоритмов, позволяющих за приемлемое время осуществить выбор оптимального расположения сенсоров наблюдения с учетом физических параметров источников освещения и объектов сцены.

В четвертой главе диссертационного исследования в качестве решения задачи поддержки принятия решения по оптимизации средств системы наблюдения на технических объектах, предлагается разработанный комплекс алгоритмов оптимизации размещения сенсоров в трехмерной сцене с несколькими критериями оптимизации (максимизация покрытия выделенной области сцены, минимизации «слепых зон» и некоторые другие, формализованные в исследовании). Такой подход, по сравнению с существующими, позволяет учесть большее количество параметров технического объекта, решить задачу оптимизации выбора размещения сенсоров системы наблюдения на промышленных предприятиях в условиях, когда традиционные методы поиска принципиально не могут быть реализованы за приемлемое время. На основании предложенных моделей разработан трёхфазный алгоритм решения задачи максимизации видимости выделенных зон или наблюдаемых объектов с заданными ограничениями на основе эвристического алгоритма муравьиной колонии, предварительного ранжирования вершин графа видимости и вычисления предложенной функции видимости объекта и сцены целиком.

Разработанные алгоритмы и методы, а также их реализация в программном комплексе, могут быть использованы при решении задач размещения камер видеонаблюдения на технических инфраструктурных объектах с различными критериями оптимизации.

Разработанный программный комплекс «Конфигуратор промышленных помещений и объектов инфраструктуры», предназначен для использования

организациями, оказывающими услуги проектирования и установки систем видеонаблюдения, для оптимизации выбора размещения сенсоров наблюдения. Полученные с применением эвристических алгоритмов поиска и предварительном ранжировании результаты, способствуют повышению эффективности выбора оптимальной конфигурации сенсоров наблюдения на наблюдаемом объекте.

В рамках исследования проведен вычислительный эксперимент по сравнению работы трех алгоритмов: алгоритма полного перебора (ПП), эвристического алгоритма «Муравьиной колонии» (МК) и разработанного трехфазного алгоритма (ТАО) при решении задачи размещения камер в трёхмерной сцене. По результатам вычислительно эксперимента была доказана эффективность предложенного трёхфазного алгоритма поиска оптимальной конфигурации камер, сокращение времени просчета 3D-сцены на примере производственного корпуса ГРЭС.

ЗАКЛЮЧЕНИЕ

В диссертационной работе предложены и исследованы формальные модели и алгоритмы дискретной оптимизации размещения средств наблюдения в трехмерной сцене. В результате проведенных исследований получены следующие теоретические и практические результаты.

1. Предложена, обоснована и исследована математическая модель видимости сложной трехмерной сцены, соответствующая её реальным параметрам и конфигурации сенсоров. Такой подход, по сравнению с существующими. принципиально расширяет возможности определения видимости узлов выделенной области на технических объектах.

2. Разработана и реализована функция видимости объекта и сцены целиком для определённой конфигурации и параметров камер наблюдения, которая в отличии от существующих решений представляет не только геометрические характеристики объектов на сцене, но также учитывает их физические свойства.

3. Разработан комплекс алгоритмов оптимизации размещения сенсоров в трехмерной сцене с несколькими критериями оптимизации (минимизации «слепых зон», максимизации покрытия выделенной области сцены и некоторые другие). Такой подход, по сравнению с существующими, позволяет решить задачу дискретной оптимизации выбора размещения сенсоров системы видеонаблюдения на технических объектах в условиях, когда традиционные методы поиска принципиально не могут быть реализованы в реальном времени и требуют существенных вычислительных затрат.

4. Предложен трёхфазный алгоритм решения задачи максимизации видимости выделенных зон или наблюдаемых объектов на технических объектах с заданными ограничениями на основе эвристического алгоритма муравьиной колонии, предварительного ранжирования вершин графа видимости и вычислении предложенной функции видимости объекта и сцены целиком.

5. Разработан программный комплекс «Конфигуратор промышленных помещений и объектов инфраструктуры», предназначенный для использования

организациями, оказывающими услуги проектирования и установки систем видеонаблюдения.

Таким образом, достигнута цель работы: оптимизация выбора размещения сенсоров системы наблюдения на промышленных предприятиях с целью определения положения камер, при котором достигается максимальная совокупная видимость множества объектов на выделенной области.

Максимизация совокупной видимости множества объектов на выделенной области за счет оптимизации выбора размещения сенсоров системы наблюдения на промышленных предприятиях (технических или распределённых объектах).

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Об утверждении Стратегии развития информационного общества в Российской Федерации на 2017–2030 годы и на перспективу до 2035 года [Электронный ресурс] // Правительство России [сайт]. URL: http://government.ru/docs/8024/ (дата обращения: 15.04.2019).
2. Пархачев, В.В. Способ определения оптимальной конфигурации системы видеомониторинга леса / Пархачев В.В., Громазин О.А., Филимонов А.В., Шишалов И.С. // [Электронный ресурс] – Режим доступа: http://www.findpatent.ru/patent/256/2561925.html – 25.09.2015 г.
3. Топольский Н.Г. Анализ этапов развития концепции «Безопасный город» [Текст] / Н.Г. Топольский, С.А. Качанов, А.А. Рыженко// Технологии техносферной безопасности. Вып. 1 (65). 2016. С. 1-9.
4. Давидюк Н.В. Автоматизация процедуры подбора технических средств обнаружения системы физической защиты объектов // Вестник АГТУ. Сборник научных трудов Астраханского государственного технического университета. Сер. Управление, вычислительная техника и информатика. 2009. №1. С.98-100
5. Ahn. J.-W. Two-Phase Algorithm for Optimal Camera Placement / Jun-Woo Ahn, Tai-Woo Chang,Sung-Hee Lee, and Yong Won Seo // Scientific Programming. Vol. 2016. 16 pages.
6. Hengel V. Automatic camera placement for large scale surveillance networks, / A. Van Den Hengel, R. Hill, B. Ward et al. // In Proceedings of the Workshop on Applications of Computer Vision (WACV '09). Snowbird. Utah. USA. December 2009. pp. 1–6.
7. Dong. M. Composite Differential Evolution with Modified Oracle Penalty Method for Constrained Optimization Problems / Minggang Dong, Ning Wang, Xiaohui Cheng, Chuanxian Jiang // Mathematical Problems in Engineering Vol. 2014. 15 pages.
8. Schlüter M. The oracle penalty method / M. Schlüter and M. Gerdts // Journal of Global Optimization. Vol. 47. no. 2. pp. 293–325. 2010.

9. Civera J. Camera Self-Calibration for Sequential Bayesian structure from motion / J. Civera, D. R. Bueno, A. J. Davison and J. M. M. Montiel // Proceedings of IEEE International Conference on Robotics and Automation. ICRA'09. 2009. P. 3411 – 3416.

10. El-Attar M. Robust Multistage Algorithm for Camera Self-Calibration Dealing with Varying Intrinsic Parameters / El-Attar, M. Karim, H. Tairi, S. Ionita. A // Journal of Theoretical and Applied Information Technology. 2011. No 32. P. 46 – 54.

11. Fiore L. Optimal Camera Placement with Adaptation to Dynamic Scenes / L. Fiore, G. Somasundaram, A. Drenner and N. Papanikolopoulos // Proceedings of IEEE International Conference on Robotics and Automation, 2009. P. 956-961.

12. Zhao J. Approximate techniques in solving optimal camera placement problems / J. Zhao, R. Yoshida, S.-C. S. Cheung, and D. Haws // International Journal of Distributed Sensor Networks. vol. 9. no. 11. Article ID 241913. 2013.

13. Hanel M. Optimal Camera Placement to measure Distances Conservativly Regarding Static and Dynamic Obstacles / M. Hanel, S. Kuhn, D. Henrich, J. Pannek and L. Gr¨une // International Journal of Sensor Networks. 2012. Vol. 12. P. 25-36.

14. Holt R. Summary of results on optimal camera placement for boundary monitoring / Holt R., Hong M., Martini R., Mukherjee I., Netravali R. and Wang J., // Proceedings of SPIE the international society for optical engineering. 2007. Vol. 6570. P. 657005.

15. Liu J. Automatic Camera Calibration and Scene Reconstruction with Scale-Invariant Features / J. Liu and R. Hubbold // Proceedings of the Second international conference on Advances in Visual Computing. ISVC'06. 2006. Vol. 1. P. 558-568.

16. Gonzalez-Barbosa J.-J. Optimal camera placement for total coverage / J.-J. Gonzalez-Barbosa, T. García-Ramírez, J. Salas, J.-B. Hurtado-Ramos, J.-D. Rico-Jiménez // In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '09). pp. 844–848. IEEE Press, Kobe, Japan. May 2009.

17. Hörster E. Approximating optimal visual sensor placement / E. Hörster and R. Lienhart // In Proceedings of the IEEE International Conference on Multimedia and Expo (ICME '06). pp. 1257–1260. July 2006.

18. Zhang M. Differential evolution with dynamic stochastic selection for constrained optimization / M. Zhang, W. Luo, X. Wang // Information Sciences. Vol. 178, Is. 15. 1 Aug. 2008. pp. 3043-3074

19. Fang Z. Hybrid approximation for minimum-cost target coverage in wireless sensor networks / Z. Fang and J. Wang // Optimization Letters. vol. 4, no. 3. pp. 371–381. 2010.

20. Schlüter M. An extended ant colony optimization algorithm for integrated process and control system design / M. Schlüter, J. A. Egea, L. T. Antelo, A. A. Alonso, and J. R. Banga // Industrial and Engineering Chemistry Research. vol. 48, no. 14. pp. 6723–6738. 2009.

21. Parpinelli R. S. An ant colony algorithm for classification rule discovery. / R. S. Parpinelli, H. S. Lopes, A. A. Freitas //In H. A. Abbass, R. A. Sarker, & C. S. Newton (Eds.), Data Mining: A Heuristic Approach. 2002. pp. 191–208.

22. Lee K. A secure framework of the surveillance video network integrating heterogeneous video formats and protocols / K. Lee, K. Yim, M. A. Mikki // Computers and Mathematics with Applications. vol. 63. no. 2. pp. 525–535. 2012.

23. Razmjooy N. A real-time mathematical computer method for potato inspection using machine vision / N. Razmjooy, B. S. Mousavi, F. Soleymani // Computers and Mathematics with Applications. vol. 63. no. 1. pp. 268–279. 2012.

24. Staff D. I. D. DARPA's VIRAT: Video Search, with a Twist, Defense Industry Daily, 2010.

25. Zhao J. Optimal visual sensor network configuration / J. Zhao, S.-C. S. Cheung, and T. Nguyen // In Multi-Camera Networks: Principles and Applications, H. Aghajan and A. Cavallaro, Eds. pp. 139–162. Academic Press, New York, NY, USA. 2009.

26. Morsly Y. Particle swarm optimization inspired probability algorithm for optimal camera network placement / Y. Morsly, N. Aouf, M. S. Djouadi, and M. Richardson // IEEE Sensors Journal. vol. 12, no. 5. pp. 1402–1412. 2012.

27. Королёв М.С. Оптимизация размещения средств наблюдения различной конфигурации в динамической трехмерной сцене с целью минимизации количества

«слепых зон» [Текст] / В.В. Печенкин, М.С. Королёв // Компьютерная оптика. 2017. Т. 41, № 2. С. 245-253. DOI: 10.18287/2412-6179-2017-41-2-245-253

28. Cole R. Visibility problems for polyhedral terrains / R. Cole and M. Sharir // Journal of Symbolic Computation. vol. 7, no. 1. pp. 11–30. 1989.

29. Моисеев Н.Н. Элементы теории оптимальных систем [Текст] / Н.Н. Моисеев // Наука. 1975. С. 528.

30. Бразовская Н.В. Методы оптимизации [Текст] / Н.В. Бразовская // АлтГТУ. 2000. С. 120. УДК 22.183.4.Б871

31. Гилл Ф. Практическая оптимизация [Текст] / Ф. Гилл, У. Мюррей, М. Райм // Пер. с англ. МИР. 1985. С. 321.

32. Гладков Л.А. Генетические алгоритмы [Текст] / Л.А. Гладков, В.В. Курейчик, В.М. Курейчик // М.: Физмат-лит, 2010.

33. Курейчик В.В. Обзор и анализ методов и моделей, инспирированных природными системами[Текст] / В.В. Курейчик, В.В. Курейчик // Известия ЮФУ. Технические науки. 2013. № 2 (13). С. 10-22.

34. Винюков И.А. Линейное программирование / И.А. Винюков, В.Ю. Попов, С.В. Пчелинцев // М.: Финакадемия, 2009.

35. Гончаренко В.М. Математические модели и методы исследования операций. Руководство к решению задач. // М.: Финакадемия, 2006.

36. Bodor R. Multi-Camera Positioning to Optimize Task Observability / R. Bodor, P. Schrater and N. Papanikolopoulos // Department of Computer Science and Engineering University of Minnesota. 2005. P. 552-557.

37. Zhao J. Approximate Techniques in Solving Optimal Camera Placement Problems / J. Zhao, D. Haws, R.Yoshida, S. Cheung // International Journal of Distributed Sensor Networks 2013(1):1705-1712 · November 2011

38. Kim H. Dynamic 3D Scene Reconstruction in Outdoor Environments / H. Kim, M. Sarim, T. Takai, J.-Y. Guillemaut and A. Hilton // Proceedings of International Symposium on 3D Data Processing, Visualization and Transmission. 2010. 3DPVT. P.613-626.

39. Ажмухамедов И.М. Формализация задачи размещения элементов охранной системы в контроллируемой зоне // Вестник Астраханского государственного технического университета. 2008. №1. С. 77-79.

40. Beeck T. G. Real-Time Vision-Based Pedestrian Detection in a Truck's Blind Spot Zone Using a Warping Window Approach // Proceedings of 9th International Conference, ICINCO 2012. P. 251-264.

41. Cardarelli E. Vision-Based Blind Spot Monitoring // Handbook of Intelligent Vehicles. 2012. Vol. 1. P. 1071-1087

42. Печенкин В.В. Проектный подход к формированию IT– компетенций технических специалистов в рамках разработки программного комплекса для оптимизации размещения камер наблюдения за объектами/ Печенкин В.В., Лепесткин Д.А. // Современные методы преподавания для студентов инженерных направлений: монография под общ. Редакцией О.Н. Долининой. Саратов, Сарат. гос. техн. ун–т. 2014 184 с. С. 156– 168.

43. Печенкин В.В. Построение модели управления в сложных динамических технических системах / В.В. Печенкин, Д.С. Решетников // Проблемы управления в социально-экономических и технических системах. Сборник научных статей по материалам X Всероссийской научной конференции 10-11 апреля 2014 года. Саратовский гос. тех. Университет, 2014. С. 16-19

44. Siti Kamaliah Y. Optimal Camera Placement for 3D Environment / Siti Kamaliah Mohd Yusoff, Abas Md Said, Idris Ismail // Second International Conference, ICSECS 2011, Kuantan, Pahang, Malaysia, June 27-29, 2011, Proceedings, Part II. pp. 448-459. URL: https://www.researchgate.net/publication/220868726_Optimal_Camera_Placement_for_3D_Environment

45. Avital S. Optimal Camera Placement [Текст] // Electrical Engineering and Computer Sciences University of California at Berkeley. 2012.

46. Camacho-Vallejo J.-F. A genetic algorithm for the bi-level topological design of local area networks / J.-F. Camacho-Vallejo, J. Mar-Ortiz, F. López-Ramos, and R. P. Rodríguez // PLoS ONE. Vol. 10, no. 6. Article ID e0128067. 2015.

47. Schlüter M. The oracle penalty method / M. Schlüter and M. Gerdts // Journal of Global Optimization. vol. 47, no. 2. pp. 293–325. 2010.

48. Галимов Р.Р. Оптимальное размещение камер наблюдения в распределенной системе видеонаблюдения / Р.Р. Галимов, А.Ю. Кричинин // Всетник ИЖГТУ им. М.Т. Калашникова. Т. 21, № 3. 2018. с. 192 – 197

49. Кручинин А.Ю. Оптимизация распознования объектов со сложной конфигураецией на основе приоритетного планирования со старением / А.Ю. Кручинин, Р.Р. Галимов // Вестник компьютерных и информационных технологий. № 7 (169). 2018. с. 30-38

50. Suri B. Implementing Ant colony optimization for test case selection and prioritization / B. Suri and S. Singhal // International Journal on Computer Science and Engineering. Vol. 3, no. 5. pp. 1924–1932. 2011.

51. Storn R. System design by constraint adaptation and differential evolution // IEEE Transactions on Evolutionary Computation. Vol. 3. Apr. 1999. pp. 22

52. Storn R. Differential evolution–A simple and efficient heuristic for global optimization over continuous spaces / R. Storn, K. Price, // J. Global Opt. 1997. Vol. 11 pp. 341-359

53. Чураков М. Маравьиные алгоритмы / М. Чураков, А. Якушев // Дискретная математика: алгоритмы. 2006. с. 1-15

54. Liu C.L. Improved ant colony genetic optimization algorithm and its application // Journal of Computer Applications. Vol. 33, no. 11. pp. 3111–3113. 2013.

55. МакКоннелл Дж. Основы современных алгоритмов // Техносфера. 2004. с. 368.

56. Штовба С.Д. Муравьиные алгоритмы // Exponenta Pro. Математика в приложениях. №4. 2003. с.70-75.

57. Dorigo M. Ottimizzazione, apprendimento automatico, ed algoritmi basati su metafora naturale (Optimization, Learning, and Natural Algorithms) // Doctorate in Systems and Information Electronic Engineering. Politecnico di Milano. 1992.

58. Dorigo M. Ant colony optimization / M. Dorigo and M. Birattari // in Encyclopedia of Machine Learning. Springer. 2010. pp. 36–39

59. Tsai C. W.. Ant colony optimization with dual pheromone tables for clustering / C. W. Tsai, K. C. Hu, M. C. Chiang, and C. S. Yang // In Proceedings of the IEEE International Conference on Fuzzy Systems. pp. 2916–2921. June 2011.

60. Colorni A. Distributed Optimization by Ant Colonies / A. Colorni, M. Dorigo, V. Maniezzo // Proceedings of the First European Conference on Artificial Life, Paris, France, Elsevier Publishing. pp. 134-142, 1991.

61. Dorigo M. The Ant System: Optimization by a colony of cooperating agents / M. Dorigo, V. Maniezzo, A. Colorni // IEEE Transactions on Systems, Man, and Cybernetics-Part B, 26, 1, pp. 29-41. 1996.

62. Gambardella L. M. Ant-Q: A Reinforcement Learning Approach to the Traveling Salesman Problem / L. M. Gambardella, M. Dorigo // Twelfth International Conference on Machine Learning, Morgan Kaufmann. pp. 252-260. 1995.

63. Dorigo M. Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem / M. Dorigo, L. M. Gambardella // IEEE Transactions on Evolutionary Computation. Vol. 1, 1. pp. 53-66. 1997.

64. Stützle T. MAX-MIN Ant System and local search for the traveling salesman problem / T. Stützle, H. Hoos // IEEE International Conference on Evolutionary Computation. pp. 309-314. 1997.

65. Bullnheimer B. A new rank based version of the Ant System. A computational study / Bernd Bullnheimer, Richard F. Hartl, Christine Strauß // Adaptive Information Systems and Modelling in Economics and Management Science, 1, 1997 г.

66. Tsai C. F. ACODF: a novel data clustering approach for data mining in large databases / C. F. Tsai, C. W. Tsai, H. C. Wu, and T. Yang // Journal of Systems and Software. Vol. 73, no. 1. pp. 133–145. 2004.

67. Stützle T. Parameter Adaptation in Ant Colony Optimization / T. Stützle, M. López-Ibáñez, P. Pellegrini, M. Maur, M. de Oca, M. Birattari, Michael Maur, M. Dorigo // Technical Report, IRIDIA, Université Libre de Bruxelles. 2010.

68. Локтев Д.А. Моделирование комплексной системы видеомониторинга внутри здания. Часть 1. Алгоритм размещения видеокамер и его программная реализация // М.: МГСУ, 2012. С. 84-92.

69. Levin Y. A heuristic method for large-scale multi-facility location problems / Y. Levin and A. Ben-Israel // Computers & Operations Research. Vol. 31, no. 2. pp. 257–272. 2004.

70. Lisin A.V. Control of uncovered cctv sites with additional light sources // Computer Optics 2017; pp. 123-217

71. Hsieh Y. C. An immune based two-phase approach for the multiple-type surveillance camera location problem. / Y. C. Hsieh, Y. C. Lee, P. S. You // Expert Systems with Applications, 38. 2011. pp. 5416 – 5422.

72. Магауенов Р.Г. Системы охранной сигнализации: основы теории и принципы построения / Учеб. пособие: Горячия линия. 2004. с. 367

73. Kritter J. On the optimal placement of cameras for surveillance and the underlying set cover problem / J. Kritter, M. Brévilliers, J. Lepagnot, L. Idoumghar // Accepted for publication in Applied Soft Computing, Elsevier. 2018. (JCR 2017 impact factor: 3.907, CiteScore 2017: 4.81, Scimago SJR 2017: 1.199)

74. Хлебников В.В. Моделирование реалистичных изображений объектов, используя различные алгоритмы расчета освещенности / В.В. Хлебников, А.А. Юров // Весник ТГУ. Т. 15. Вып. 2. 2010.

75. Никитин В.В. Телевидение в системах физической защиты. / В.В. Никитин, А.К. Цыцулин // СПб.: Изд-во СПбГЭТУ «ЛЭТИ», 2001. 135 с.

76. Королёв М.С. Разработка оптимальных конфигураций средств наблюдения на основе виртуального моделирования 3D сцены и параметров чувствительности камеры [Текст] / В.В. Печенкин, М.С. Королёв // Проблемы управления в социально-экономических и технических системах: сб. науч. ст. по материалам XII всерос. науч. конф., Саратов, 7-8 апр. 2016. С. 46-50.

77. Korolev M. Analysis of three-dimensional scene visual characteristics based on virtual modeling and parameters of surveillance sensors [Текст] / V. Pechenkin, M. Korolev, K. Kuznetsova, D. Piminov // Studies in Systems, Decision and Control, Vol. 199. 2019. pp. 552-562. DOI: 10.1007/978-3-030-12072-6_45.

78. Кочетов Ю.А. Гибридный алгоритм локального поиска для задачи маршрутизации разнородного ограниченного автопарка / Ю. А. Кочетов, А. В. Хмелев // Дискретный анализ и исследование операций. 2015. Т. 22. № 5. стр. 5–29.

79. Кривошеин Д.Ю. Алгоритмы пересчёта кратчайших путей в графе при изменении весов ребер / Д.Ю. Кривошеин, А.М. Марченко // Проблемы разработки перспективных микро– и наноэлектронных систем – 2012: Сб. тр. под общ. ред. академика РАН А.Л. Стемпковского. М.: ИППМ РАН. 2012. С. 263-266.

80. Demetrescu C. Dynamic graphs. / C. Demetrescu, I. Finocchi, G.F. Italiano

81. Pearce D. J. Dynamic Topological Sort Algorithm for Directed Acyclic Graphs / D. J. Pearce, H.J. Paul, A. Kelly // Journal of Experimental Algorithmics (JEA). 2006. Vol. 11. Article No. 1.7

82. Deepak A. Average-Case Analysis of Incremental Topological Ordering / A. Deepak, F. Tobias // Discrete Applied Mathematics. 28 Feb. 2010. Vol. 158. Issue 4. pp. 240–250.

83. Nicoara D. Hermes: Dynamic Partitioning for Distributed Social Network Graph Databases / D. Nicoara, S. Kamali, K. Daudjee, L. Chen // EDBT. 2015. pp. 25-36.

84. Ammar A.B. Query optimization techniques in graph Databases // International Journal of Database Management Systems (IJDMS). Aug. 2016. Vol.8. No.4. DOI: 10.5121/ijdms.2016.8401

85. Di Caro G. AntNet: distributed stigmergetic control for communications networks / G. di Caro and M. Dorigo // Journal of Artificial Intelligence Research. Vol. 9. pp. 317–365. 1998.

86. Курейчик В.В. Муравьиный алгоритм для решения оптимизационных задач с явно выраженной целевой функцией / В.В. Курейчик, М.А Жиленков // Информатика, вычислительная техника и инженерное образование. 2015. № 2. с. 1–12.

87. Gu J.-H. Improved culture ant colony optimization method for solving TSP problem / J.-H. Gu, P.-P. Fan, Q.-Z. Song, and E.-H. Liu // Computer Engineering and Applications. Vol. 46, no. 26. pp. 49–52. 2010.

88. Agarw S. Ranking on Graph Data // Proceedings of the 23rd International Conference on Machine Learning, Pittsburgh, PA. 2006. pp. 25-32. URL: citeseerx.ist.psu.edu

89. Розенберг И.Н. Использование нечетких представлений данных при определении медиан графа // Известия ТРТУ. Таганрог: ТРТУ. 2001. №4. с 64-72.

90. Adaikalam A. Fuzzy graph based shortest path ranking method for optical network / A. Adaikalam, S. Manikandan, V. Rajamani // Optical and Quantum Electronics. 2017. vol. 49. no. 9.

91. Klein A. Efficient heuristic approach with improved time complexity for QoS-aware service composition / A. Klein, F. Ishikawa, and S. Honiden // In Proceedings of the 2011 IEEE 9th International Conference on Web Services, ICWS 2011. pp. 436–443, IEEE, USA, July 2011.

92. Barman A. SHaPE: A Novel Graph Theoretic Algorithm for Making Consensus-Based Decisions in Person Re-identification Systems / A. Barman, S. K. Shah // Proceedings of the IEEE International Conference on Computer Vision (ICCV), Venice, Italy. 2018. pp. 1124-1133.

93. Roffo G. Vinciarelli A. Infinite Latent Feature Selection: A Probabilistic Latent Graph-Based Ranking Approach / Roffo G., Melzi S., Castellani U.// Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV) 2017, Venice, Italy. 2018. pp. 1407-1415.

94. Печенкин В.В. Визуализация сетевой структуры групповых отношений / В.В. Печенкин, Д.С. Решетников, В.Н. Ярская-Смирнова // 4М. Методология, методы, математическое моделирование. 2014. № 39. С. 40-61.

95. Королёв М.С. Подходы к задаче ранжирования вершин в теории графов / М.С. Королёв, Д.С. Решетников // Проблемы управления в социально-экономических и технических системах. Сбр. мат. XIII Международной научной конференции. Саратов: ИЦ "Наука", 2017. С. 138-141

96. Lumbreras A. Applying trust metrics based on user interactions to recommendation in social networks / A. Lumbreras, R. Gavaldà // ACM International Conference, 2012. pp. 1159-1164.

97. Jameson K.A. Finding an appropriate order for a hierarchy based on probabilistic dominance / K. A. Jameson, M. C. Appleby, L. C. Freeman // Animal Behaviour. 1999. Т. 57. №. 5. pp. 991-998.

98. Page L. The PageRank citation ranking: Bringing order to the Web / L. Page, S. Brin, R. Motwani // 1999. URL: google.stanford.edu/~backrub/pageranksub.ps

99. Sarma A. Das. Fast distributed pagerank computation / Sarma A. Das, A. R. Molla, G. Pandurangan, E. Upfal // Theoretical Computer Science. 2015. Vol. 561. pp. 113–121.

100. Dai L. Fully distributed PageRank computation with exponential convergence / L. Dai, N.M. Freris // arXiv:1705.09927v1 [cs.DC], 2017. pp.1-5.

101. Nathan E. Ranking in Dynamic Graphs using Exponential Centrality / E. Nathan, J. Fairbanks, D. Bader // Proceedings of 6th International Conference on Complex Networks and Their Applications. Lyon, France. Nov 2017. pp. 378–389.

102. Andziulis A. Comparison of two heuristic approaches for solving the production scheduling problem / A. Andziulis, D. Dzemydiene, R. Steponavičius, S. Jakovlev // Information Technology and Control. Vol. 40, no. 2. pp. 118–122. 2011.

103. Kuntz P. A stochastic heuristic for visualising graph clusters in a bi-dimensional space prior to partitioning / P. Kuntz, D. Snyers, and P. Layzell // Journal of Heuristics. Vol. 5, no. 3. pp. 327–351. 1999.

104. Локтев А.А. Алгоритм располнования объектов / А.А. Локтев, А.Н. Алфимцев, Д.А. Локтев // Вестник МГСУ. Вып. 5. 2012.

ПРИЛОЖЕНИЕ А

Классы программного обеспечения конфигуратора промышленных помещений и объектов инфраструктуры

```
// Класс MeshCreator
using System.Collections.Generic;
using UnityEngine;

public class MeshCreator : MonoBehaviour
{
    [SerializeField]
    float minAngle;
    [SerializeField]
    Material mat;

    CreateController createController;

    float thickness = 1;
    float height = 5;
    Vector3 currentPos;
    Vector3 targetPos;
    Vector3 startPos;
    GameObject go;
    MeshFilter mf;
    MeshRenderer mr;
    MeshCollider mc;
    Camera cam;
    bool isFirstClick;
    Vector2[] uvs;

    List<Vector3> vertex;
    List<int> triangles;

    private Vector3 prevMousePos;
    private bool isEndCreate;
    int numberPoints;

    void Awake()
    {
        numberPoints = 0;
        isEndCreate = true;
        isFirstClick = true;
        cam = Camera.main;
        vertex = new List<Vector3>();
        triangles = new List<int>();
    }

    void Start()
```

```
{
    createController = CreateController.Instance;
}

public void Create(float height, float thickness)
{
    isEndCreate = false;
    isFirstClick = true;
    this.height = height;
    this.thickness = thickness;
}

public static Vector3[] GetNormals()
{
    Vector3[] v3 = new Vector3[4];
    for (int i = 0; i < 4; i++) v3[i] = Vector3.Cross(Vector3.forward, Vector3.right).normalized;
    return v3;
}

public static Vector3[] Convert_V23_Arr(Vector2[] vect2Arr)
{
    Vector3[] v3 = new Vector3[vect2Arr.Length];
    for (int i = 0; i < vect2Arr.Length; i++) v3[i] = vect2Arr[i];
    return v3;
}

void RecalculateVertexAndTris()
{
    targetPos = cam.ScreenToWorldPoint(Input.mousePosition);
    targetPos.y = 0;

    Vector3 direct = targetPos - currentPos;
    direct.Normalize();
    float deltaX = thickness / 2 * direct.z;
    float deltaZ = thickness / 2 * direct.x;

    if (vertex.Count == numberPoints * 12)
    {
        if (numberPoints > 1)
        {

            int j = numberPoints * 12 - 24;
            Vector3 center = (vertex[j + 8] + vertex[j + 9]) / 2;
            Vector3 v1 = center - currentPos;
            Vector3 v2 = targetPos - currentPos;
            if (v2 != Vector3.zero)
            {

                v1.Normalize();
                v2.Normalize();
                Vector3 v3;
                float angle = Vector3.Angle(v1, v2);
```

```
                if (angle < minAngle)
                {
                    float deltaAngle = minAngle - angle;
                    if (Vector3.Cross(v1, v2).y > 0)
                    {
                        deltaAngle = -deltaAngle;
                    }
                    direct = targetPos - currentPos;
                    direct = new Vector3(direct.x * Mathf.Cos(Mathf.Deg2Rad * deltaAngle) - direct.z
* Mathf.Sin(Mathf.Deg2Rad * deltaAngle), 0, direct.z * Mathf.Cos(Mathf.Deg2Rad * deltaAngle) +
direct.x * Mathf.Sin(Mathf.Deg2Rad * deltaAngle));
                    targetPos = direct + currentPos;
                    v1 = center - currentPos;
                    v2 = targetPos - currentPos;
                    v1.Normalize();
                    v2.Normalize();
                    v3 = (v1 + v2).normalized * thickness / 2 / Mathf.Sin(minAngle / 2 *
Mathf.Deg2Rad);
                    direct.Normalize();
                    deltaX = thickness / 2 * direct.z;
                    deltaZ = thickness / 2 * direct.x;
                }
                else
                {
                    v3 = (v1 + v2).normalized * thickness / 2 / Mathf.Sin(Vector3.Angle(v1, v2) / 2 *
Mathf.Deg2Rad);
                }

                j += 12;
                Vector3 tmpV3 = vertex[j + 8] - currentPos;
                Vector3 inside;
                Vector3 outside;

                angle = Vector3.Angle(tmpV3, v3);

                if (Vector3.Cross(v1, v2).y > 0)
                {
                    inside = currentPos + v3;
                    outside = currentPos - v3;
                }
                else
                {
                    inside = currentPos - v3;
                    outside = currentPos + v3;
                }

                inside.y = height;
                outside.y = height;

                for (int k = 0; k < 2; k++)
                {
                    vertex[j + k * 4] = inside;
```

```
                    vertex[j + k * 4 - 10] = inside;
                    vertex[j + 1 + k * 4] = outside;
                    vertex[j + k * 4 - 9] = outside;
                }
                vertex[j + 8] = new Vector3(inside.x, 0, inside.z);
                vertex[j - 2] = new Vector3(inside.x, 0, inside.z);
                vertex[j + 9] = new Vector3(outside.x, 0, outside.z);
                vertex[j - 1] = new Vector3(outside.x, 0, outside.z);
                for (int i = 0; i < 2; i++)
                {
                    vertex[j + 2 + i * 4] = new Vector3(targetPos.x - deltaX, height, targetPos.z +
deltaZ);
                    vertex[j + 3 + i * 4] = new Vector3(targetPos.x + deltaX, height, targetPos.z -
deltaZ);
                }
                vertex[j + 10] = new Vector3(targetPos.x - deltaX, 0, targetPos.z + deltaZ);
                vertex[j + 11] = new Vector3(targetPos.x + deltaX, 0, targetPos.z - deltaZ);
            }
        }
        else
        {
            int i = vertex.Count - 12;
            for (int j = 0; j < 2; j++)
            {
                vertex[i + j * 4] = new Vector3(currentPos.x - deltaX, height, currentPos.z + deltaZ);
                vertex[i + 1 + j * 4] = new Vector3(currentPos.x + deltaX, height, currentPos.z -
deltaZ);
                vertex[i + 2 + j * 4] = new Vector3(targetPos.x - deltaX, height, targetPos.z + deltaZ);
                vertex[i + 3 + j * 4] = new Vector3(targetPos.x + deltaX, height, targetPos.z - deltaZ);
            }

            vertex[i + 8] = new Vector3(currentPos.x - deltaX, 0, currentPos.z + deltaZ);
            vertex[i + 9] = new Vector3(currentPos.x + deltaX, 0, currentPos.z - deltaZ);
            vertex[i + 10] = new Vector3(targetPos.x - deltaX, 0, targetPos.z + deltaZ);
            vertex[i + 11] = new Vector3(targetPos.x + deltaX, 0, targetPos.z - deltaZ);
        }

    }
    else
    {
        if (numberPoints > 3 && Vector3.Distance(currentPos, startPos) < thickness)
        {
            isEndCreate = true;
            createController.IsCreate = false;
            int j = (numberPoints - 2) * 12;
            Vector3 startPosEnd = (vertex[10] + vertex[11]) / 2;
            Vector3 targetPosEnd = (vertex[j + 8] + vertex[j + 9]) / 2;
            Vector3 v1 = startPosEnd - startPos;
            Vector3 v2 = targetPosEnd - startPos;
            v1.Normalize();
            v2.Normalize();
```

```
            Vector3 v3 = (v1 + v2).normalized * thickness / 2 / Mathf.Sin(Vector3.Angle(v1, v2) /
2 * Mathf.Deg2Rad);
            Vector3 inside;
            Vector3 outside;

            if (Vector3.Cross(v2, v1).y > 0)
            {
               inside = startPos + v3;
               outside = startPos - v3;
            }
            else
            {
               inside = startPos - v3;
               outside = startPos + v3;
            }
            inside.y = height;
            outside.y = height;

            for (int k = 0; k < 2; k++)
            {
               vertex[k * 4] = inside;
               vertex[j + k * 4 + 2] = inside;
               vertex[1 + k * 4] = outside;
               vertex[j + k * 4 + 3] = outside;
            }
            vertex[8] = new Vector3(inside.x, 0, inside.z);
            vertex[j + 10] = new Vector3(inside.x, 0, inside.z);
            vertex[9] = new Vector3(outside.x, 0, outside.z);
            vertex[j + 11] = new Vector3(outside.x, 0, outside.z);

            j = (numberPoints - 3) * 12;
            v1 = startPos - targetPosEnd;
            v2 = (vertex[j + 8] + vertex[j + 9]) / 2 - targetPosEnd;
            v1.Normalize();
            v2.Normalize();
            v3 = (v1 + v2).normalized * thickness / 2 / Mathf.Sin(Vector3.Angle(v1, v2) / 2 *
Mathf.Deg2Rad);

            if (Vector3.Cross(v2, v1).y > 0)
            {
               inside = targetPosEnd + v3;
               outside = targetPosEnd - v3;
            }
            else
            {
               inside = targetPosEnd - v3;
               outside = targetPosEnd + v3;
            }
            inside.y = height;
            outside.y = height;

            for (int k = 0; k < 2; k++)
```

```
                {
                    vertex[k * 4 + j + 12] = inside;
                    vertex[j + k * 4 + 2] = inside;
                    vertex[13 + k * 4 + j] = outside;
                    vertex[j + k * 4 + 3] = outside;
                }
                vertex[j + 20] = new Vector3(inside.x, 0, inside.z);
                vertex[j + 10] = new Vector3(inside.x, 0, inside.z);
                vertex[j + 21] = new Vector3(outside.x, 0, outside.z);
                vertex[j + 11] = new Vector3(outside.x, 0, outside.z);

                uvs = new Vector2[vertex.Count];
                for (int i = 0; i < (numberPoints - 1) * 12; i += 12)
                {
                    Vector3 dis = Vector3.zero;
                    int s = 1;

                    Vector3 v4 = vertex[i + 2] - vertex[i];
                    Vector3 v5 = (vertex[i + 1] + vertex[i]) / 2;

                    v4.Normalize();
                    float deltaX1 = thickness / 2 * v4.z;
                    float deltaZ1 = thickness / 2 * v4.x;

                    Vector3 v6 = new Vector3(v5.x - deltaX1, 0, v5.z + deltaZ1);

                    dis = v6 - new Vector3(vertex[i].x, 0, vertex[i].z);
                    float a = Vector3.Angle(dis, v4);
                    if (0 == Mathf.Round(a))
                    {
                        s = -1;
                    }

                    uvs[i] = new Vector2(dis.magnitude * s * 2, thickness);
                    uvs[i + 1] = new Vector2(0, 0);
                    uvs[i + 2] = new Vector2(Vector3.Distance(vertex[i], vertex[i + 2]) + dis.magnitude
* s * 2, thickness);
                    uvs[i + 3] = new Vector2(Vector3.Distance(vertex[i + 1], vertex[i + 3]), 0);
                    uvs[i + 4] = new Vector2(Vector3.Distance(vertex[i + 6], vertex[i + 4]), height);
                    uvs[i + 5] = new Vector2(-0, height);
                    uvs[i + 6] = new Vector2(0, height);
                    uvs[i + 7] = new Vector2(Vector3.Distance(vertex[i + 1], vertex[i + 7]), height);
                    uvs[i + 8] = new Vector2(Vector3.Distance(vertex[i + 8], vertex[i + 10]), 0);
                    uvs[i + 9] = new Vector2(0, 0);
                    uvs[i + 10] = new Vector2(0, 0);
                    uvs[i + 11] = new Vector2(Vector3.Distance(vertex[i + 9], vertex[i + 11]), 0);
                }

            }
            else
            {
```

```
                Vector3[] arrv3 = { new Vector3(currentPos.x - deltaX, height, currentPos.z + deltaZ),
new Vector3(currentPos.x + deltaX, height, currentPos.z - deltaZ), new Vector3(targetPos.x - deltaX,
height, targetPos.z + deltaZ), new Vector3(targetPos.x + deltaX, height, targetPos.z - deltaZ) };
                vertex.AddRange(arrv3);
                vertex.AddRange(arrv3);
                vertex.Add(new Vector3(currentPos.x - deltaX, 0, currentPos.z + deltaZ));
                vertex.Add(new Vector3(currentPos.x + deltaX, 0, currentPos.z - deltaZ));
                vertex.Add(new Vector3(targetPos.x - deltaX, 0, targetPos.z + deltaZ));
                vertex.Add(new Vector3(targetPos.x + deltaX, 0, targetPos.z - deltaZ));

                int[] tris = new[] { 2, 1, 0, 2, 3, 1, 6, 8, 10, 6, 4, 8, 5, 11, 9, 5, 7, 11 };
                for (int i = 0; i < tris.Length; i++)
                {
                    tris[i] += (numberPoints - 1) * 12;
                }
                triangles.AddRange(tris);
            }
        }
        mf.mesh.SetVertices(vertex);
        mf.mesh.triangles = triangles.ToArray();
        mf.mesh.uv = uvs;
        mf.mesh.RecalculateNormals();
    }

    void Update()
    {
        if (!isEndCreate)
        {
            if (Input.GetMouseButtonDown(0))
            {
                if (isFirstClick)
                {
                    print(isFirstClick);
                    numberPoints = 1;
                    currentPos = cam.ScreenToWorldPoint(Input.mousePosition);
                    currentPos.y = 0;
                    startPos = currentPos;
                    isFirstClick = false;
                    go = gameObject;
                    mf = go.AddComponent(typeof(MeshFilter)) as MeshFilter;
                    mr = go.AddComponent(typeof(MeshRenderer)) as MeshRenderer;
                    mc = go.AddComponent(typeof(MeshCollider)) as MeshCollider;
                    go.GetComponent<Renderer>().sharedMaterial = mat;
                    mf.mesh = new Mesh();
                    mc.sharedMesh = mf.mesh;
                    int[] tris = new[] { 2, 1, 0, 2, 3, 1, 6, 8, 10, 6, 4, 8, 5, 11, 9, 5, 7, 11 };
                }
                else
                {
                    RecalculateVertexAndTris();
                    currentPos = targetPos;
                    numberPoints++;
```

```
            }
        }
        else if (Input.GetMouseButtonDown(1))
        {
            CreateModRightClick();
        }

        if (!isFirstClick && !isEndCreate)
        {
            RecalculateVertexAndTris();
        }
    }
}

void CreateModRightClick()
{
    isEndCreate = true;
    #region Удаление последего сегмента стены
    int j = vertex.Count - 12;
    vertex.RemoveRange(j, 12);
    triangles.RemoveRange(triangles.Count - 18, 18);
    #endregion

    #region Редактирование конца предыдущего сегмента стены
    Vector3 end = (vertex[j - 1] + vertex[j - 2]) / 2;
    Vector3 start = (vertex[j - 3] + vertex[j - 4]) / 2;
    Vector3 direct = end - start;
    direct.Normalize();
    float deltaX = thickness / 2 * direct.z;
    float deltaZ = thickness / 2 * direct.x;

    vertex[j - 2] = new Vector3(end.x - deltaX, 0, end.z + deltaZ);
    vertex[j - 6] = new Vector3(end.x - deltaX, height, end.z + deltaZ);
    vertex[j - 10] = new Vector3(end.x - deltaX, height, end.z + deltaZ);
    vertex[j - 1] = new Vector3(end.x + deltaX, 0, end.z - deltaZ);
    vertex[j - 5] = new Vector3(end.x + deltaX, height, end.z - deltaZ);
    vertex[j - 9] = new Vector3(end.x + deltaX, height, end.z - deltaZ);
    #endregion

    vertex.Add(vertex[0]);
    vertex.Add(vertex[1]);
    vertex.Add(vertex[8]);
    vertex.Add(vertex[9]);

    vertex.Add(vertex[j - 10]);
    vertex.Add(vertex[j - 9]);
    vertex.Add(vertex[j - 2]);
    vertex.Add(vertex[j - 1]);

    int[] tris = new[] { 0, 3, 2, 0, 1, 3, 6, 7, 4, 7, 5, 4 };

    for (int i = 0; i < tris.Length; i++)
```

```
            {
                tris[i] += j;
            }
            triangles.AddRange(tris);



            uvs = new Vector2[vertex.Count];

            for (int i = 0; i < (numberPoints - 1) * 12; i += 12)
            {
                Vector3 dis = Vector3.zero;
                int s = 1;
                if (i != 0)
                {
                    Vector3 v1 = vertex[i + 2] - vertex[i];
                    Vector3 v2 = (vertex[i + 1] + vertex[i]) / 2;

                    v1.Normalize();
                    float deltaX1 = thickness / 2 * v1.z;
                    float deltaZ1 = thickness / 2 * v1.x;

                    Vector3 v3 = new Vector3(v2.x - deltaX1, 0, v2.z + deltaZ1);

                    dis = v3 - new Vector3(vertex[i].x, 0, vertex[i].z);
                    float a = Vector3.Angle(dis, v1);
                    if (0 == Mathf.Round(a))
                    {
                        s = -1;
                    }
                }

                uvs[i] = new Vector2(dis.magnitude * s * 2, thickness);
                print(dis.magnitude * s);
                uvs[i + 1] = new Vector2(0, 0);
                uvs[i + 2] = new Vector2(Vector3.Distance(vertex[i], vertex[i + 2]) + dis.magnitude * s *
2, thickness);
                uvs[i + 3] = new Vector2(Vector3.Distance(vertex[i + 1], vertex[i + 3]), 0);
                uvs[i + 4] = new Vector2(Vector3.Distance(vertex[i + 6], vertex[i + 4]), height);
                uvs[i + 5] = new Vector2(0, height);
                uvs[i + 6] = new Vector2(0, height);
                uvs[i + 7] = new Vector2(Vector3.Distance(vertex[i + 1], vertex[i + 7]), height);
                uvs[i + 8] = new Vector2(Vector3.Distance(vertex[i + 8], vertex[i + 10]), 0);
                uvs[i + 9] = new Vector2(0, 0);
                uvs[i + 10] = new Vector2(0, 0);
                uvs[i + 11] = new Vector2(Vector3.Distance(vertex[i + 9], vertex[i + 11]), 0);
            }

            j = (numberPoints - 1) * 12;
            uvs[j] = new Vector2(0, height);
            uvs[j + 4] = new Vector2(0, height);
            uvs[j + 1] = new Vector2(thickness, height);
```

```
            uvs[j + 5] = new Vector2(thickness, height);
            uvs[j + 2] = new Vector2(0, 0);
            uvs[j + 6] = new Vector2(0, 0);
            uvs[j + 3] = new Vector2(thickness, 0);
            uvs[j + 7] = new Vector2(thickness, 0);

            mf.mesh.triangles = triangles.ToArray();
            mf.mesh.SetVertices(vertex);
            mf.mesh.RecalculateNormals();
            mf.mesh.uv = uvs;

            createController.IsCreate = false;
        }
    }
```

// Класс MeshCreator

```
using System;
using System.Collections.Generic;
using UnityEngine;

public class ModelsController : MonoBehaviour
{
    public static ModelsController Instance { get; private set; }
    [SerializeField] SurfaceObject[] surfaceObject;
    [SerializeField] GameObject[] markingPrefabs;
    [SerializeField] GameObject[] prefabsModels;
    [SerializeField] GameObject moveArrow;
    [SerializeField] GameObject rotateIcon;
    List<GameObject> listModel = new List<GameObject>();
    List<GameObject> selectListModel = new List<GameObject>();
    List<Vector3> prevPosModel = new List<Vector3>();
    List<Vector3> prevRotateModel = new List<Vector3>();

    GameObject createdGameObject;
    GameObject go_camera;
    Transform t_createdGameObject;
    RectTransform rt_moveArrow;
    RectTransform rt_rotateIcon;
    Camera cam;
    CreateArea createArea;
    GUIDrawController guiDrawController;
    ToolbarButtonController toolbarButtonController;
    Ray ray;
    RaycastHit hit;

    Vector3 mouseStartPos;
    Vector3 sp_moveArrow;
    Vector3 sp_rotateIcon;
    Vector3 selectionStartPoint;
    Vector3 selectionEndPoint;
```

```
Vector3 startPosW;
Vector3 endPosW;
Vector2 currentPos;

Rect rect;

float minCreateSizeX;
float minCreateSizeY;

bool isStartCreateZone;

float mouseX;
float mouseY;

float selectionWidth;
float selectionHeight;

bool m_SetPos;
bool createSurface;
bool selecting;
bool startMoving;
bool startRotate;

int currIdSurface;

private void Awake()
{
   if (Instance == null)
   { // Экземпляр менеджера был найден
      Instance = this; // Задаем ссылку на экземпляр объекта
   }
   else if (Instance == this)
   { // Экземпляр объекта уже существует на сцене
      Destroy(gameObject); // Удаляем объект
   }
}

void Start()
{
   currIdSurface = -1;
   minCreateSizeX = 1;
   minCreateSizeY = 1;
   isStartCreateZone = false;
   rect = new Rect();

   rt_moveArrow = moveArrow.GetComponent<RectTransform>();
   rt_rotateIcon = rotateIcon.GetComponent<RectTransform>();

   m_SetPos = false;
   go_camera = GameObject.FindGameObjectWithTag("MainCamera");
   cam = go_camera.GetComponent<Camera>();
   createArea = CreateArea.Instance;
```

```
    guiDrawController = GUIDrawController.Instance;
    toolbarButtonController = ToolbarButtonController.Instance;
}

public void CreatePrefab(int id)
{
    createdGameObject = Instantiate(prefabsModels[id]);
    t_createdGameObject = createdGameObject.transform;
    m_SetPos = true;

    toolbarButtonController.DisableActiveButton();
}

public void CreateSurface(int id)
{
    currIdSurface = id;
    createSurface = true;
    toolbarButtonController.DisableActiveButton();
}

void SetPos()
{
    Vector3 pos = cam.ScreenToWorldPoint(Input.mousePosition);
    pos = new Vector3(pos.x, 0, pos.z);
    t_createdGameObject.position = pos;

    if (Input.GetMouseButtonDown(0))
    {
        if (!createArea.isLocated) Destroy(createdGameObject);
        else
        {
            listModel.Add(createdGameObject);
        }
        m_SetPos = false;
        toolbarButtonController.EnableActiveButton();
    }
}

void DeselectAll()
{
    foreach (GameObject unit in listModel)
    {
        unit.GetComponent<Model>().IsSelected = false;
    }
    selectListModel.Clear();
}
void Update()
{
    if (go_camera.activeSelf)
    {
        if (createSurface && createArea.isLocated && currIdSurface != -1)
        {
```

```
    if (Input.GetMouseButtonDown(0))
    {
        mouseStartPos = Input.mousePosition;
        startPosW = cam.ScreenToWorldPoint(mouseStartPos);
        mouseStartPos.y = Screen.height - mouseStartPos.y;
        isStartCreateZone = true;
        guiDrawController.StartDraw();
    }
    if (Input.GetMouseButton(0))
    {
        GetRect();
    }
    if (Input.GetMouseButtonUp(0))
    {
        GetRect();
        endPosW = cam.ScreenToWorldPoint(Input.mousePosition);

        GenerateSurface(currIdSurface);
        currIdSurface = -1;

        isStartCreateZone = false;
        createSurface = false;
        guiDrawController.StopDraw();
        toolbarButtonController.EnableActiveButton();
    }
}
else if (m_SetPos) SetPos();
else if (createArea.isLocated)
{
    if (toolbarButtonController.ToolbarMode == ToolbarMode.Select) Select();
    if (toolbarButtonController.ToolbarMode == ToolbarMode.Move) Move();
    if (toolbarButtonController.ToolbarMode == ToolbarMode.Rotate) Rotate();
}
else if (!createArea.isLocated)
{
    if (toolbarButtonController.ToolbarMode == ToolbarMode.Select)
    {
        if (moveArrow.activeSelf) moveArrow.SetActive(false);
        if (rotateIcon.activeSelf) rotateIcon.SetActive(false);
    }
    if (toolbarButtonController.ToolbarMode == ToolbarMode.Move)
    {
        HideUnhideMoveIcon();
        if (rotateIcon.activeSelf) rotateIcon.SetActive(false);
    }
    if (toolbarButtonController.ToolbarMode == ToolbarMode.Rotate)
    {
        HideUnhideRotateIcon();
        if (moveArrow.activeSelf) moveArrow.SetActive(false);
    }
}
```

```
            if (toolbarButtonController.ToolbarMode == ToolbarMode.None && selectList-
Model.Count != 0)
            {
               DeselectAll();
               if (rotateIcon.activeSelf) rotateIcon.SetActive(false);
               if (moveArrow.activeSelf) moveArrow.SetActive(false);
            }
         }
         else
         {
            DeselectAll();
            if (rotateIcon.activeSelf) rotateIcon.SetActive(false);
            if (moveArrow.activeSelf) moveArrow.SetActive(false);
         }
         if (Input.GetKeyDown(KeyCode.Delete))
         {
            if (selectListModel.Count > 0)
            {
               foreach (GameObject item in selectListModel)
               {
                  listModel.Remove(item);
                  Destroy(item);
               }
               selectListModel.Clear();
            }
         }
      }

      void GetRect()
      {
         currentPos = Input.mousePosition;
         currentPos.y = Screen.height - currentPos.y;
         Vector2 start = new Vector2(mouseStartPos.x < currentPos.x ? mouseStartPos.x : current-
Pos.x, mouseStartPos.y < currentPos.y ? mouseStartPos.y : currentPos.y);
         Vector2 sizeDrawRect = new Vector2(mouseStartPos.x > currentPos.x ? mouseStartPos.x :
currentPos.x, mouseStartPos.y > currentPos.y ? mouseStartPos.y : currentPos.y) - start;

         rect = new Rect(start, sizeDrawRect);
         guiDrawController.DrawScreenRectGUI(rect);
      }

      public void GenerateSurface(int id)
      {
         if (Mathf.Abs(startPosW.x - endPosW.x) >= minCreateSizeX || Mathf.Abs(startPosW.y -
endPosW.y) >= minCreateSizeY)
         {

            GameObject go = Instantiate(surfaceObject[id].surfacePrefab);
            Transform tr_go = go.transform;
            Vector3 tmp = (startPosW + endPosW) / 2;
            tr_go.position = new Vector3(tmp.x, surfaceObject[id].layer, tmp.z);
```

```
                tr_go.localScale = new Vector3(Mathf.Abs(startPosW.x - endPosW.x),
Mathf.Abs(startPosW.z - endPosW.z), 1);
                MeshRenderer meshRenderer = go.GetComponent<MeshRenderer>();
                meshRenderer.material.mainTextureScale = new Vector2(tr_go.localScale.x * surfaceOb-
ject[id].tileValue, tr_go.localScale.y * surfaceObject[id].tileValue);
                listModel.Add(go);
            }
        }

        private void Rotate()
        {
            SelectOneObject();
            HideUnhideRotateIcon();
        }

        private void HideUnhideRotateIcon()
        {
            if (!startRotate)
            {
                if (selectListModel.Count > 0)
                {
                    if (!rotateIcon.activeSelf) rotateIcon.SetActive(true);
                    if (selectListModel.Count == 1)
                    {
                        rt_rotateIcon.position = (Vector2)cam.WorldToScreenPoint(selectList-
Model[0].transform.position);
                    }
                    else
                    {
                        rt_rotateIcon.position = (Vector2)cam.WorldToScreenPoint(GetMiddlePositionSe-
lection());
                    }
                }
                else
                {
                    if (rotateIcon.activeSelf) rotateIcon.SetActive(false);
                }
            }
        }

        void Move()
        {
            SelectOneObject();
            HideUnhideMoveIcon();
        }

        private void HideUnhideMoveIcon()
        {
            if (!startMoving)
            {
                if (selectListModel.Count > 0)
                {
```

```
                if (!moveArrow.activeSelf) moveArrow.SetActive(true);
                if (selectListModel.Count == 1)
                {
                    rt_moveArrow.position = (Vector2)cam.WorldToScreenPoint(selectList-
Model[0].transform.position);
                }
                else
                {
                    rt_moveArrow.position = (Vector2)cam.WorldToScreenPoint(GetMiddlePositionSe-
lection());
                }
            }
            else
            {
                if (moveArrow.activeSelf) moveArrow.SetActive(false);
            }
        }
    }

    Vector3 GetMiddlePositionSelection()
    {
        Vector3 sum = new Vector3();

        foreach (GameObject selectObject in selectListModel)
        {
            sum += selectObject.transform.position;
        }

        sum /= selectListModel.Count;
        return sum;
    }

    private void SelectOneObject()
    {
        if (Input.GetMouseButtonDown(0))
        {
            ray = cam.ScreenPointToRay(Input.mousePosition);

            DeselectAll();
            if (Physics.Raycast(ray, out hit))
            {
                if (hit.collider.gameObject.tag == "Model")
                {
                    hit.collider.gameObject.GetComponent<Model>().IsSelected = true;
                    selectListModel.Add(hit.collider.gameObject);
                }
            }
        }
    }

    public void BeginDragRotate()
    {
```

```
            mouseStartPos = Input.mousePosition;

            startRotate = true;

            prevRotateModel.Clear();

            foreach (GameObject model in selectListModel)
            {
                prevRotateModel.Add(model.transform.eulerAngles);
            }
        }

        public void DragRotate()
        {
            float deltaY = Input.mousePosition.y - mouseStartPos.y;


            deltaY *= 0.4f;

            for (int i = 0; i < selectListModel.Count; i++)
            {
                selectListModel[i].transform.eulerAngles = new Vector3(prevRotateModel[i].x, prevRo-
tateModel[i].y - deltaY, prevRotateModel[i].z);
            }
        }

        public void EndDragRotate()
        {
            startRotate = false;
        }

        public void BeginDragMove()
        {
            mouseStartPos = Input.mousePosition;
            sp_moveArrow = rt_moveArrow.position;

            startMoving = true;

            prevPosModel.Clear();

            foreach (GameObject model in selectListModel)
            {
                prevPosModel.Add(model.transform.position);
            }

        }

        public void EndDragMove()
        {
            startMoving = false;
        }
```

```
    public void DragUpDownMove()
    {
        float deltaY = sp_moveArrow.y + Input.mousePosition.y - mouseStartPos.y;
        rt_moveArrow.position = new Vector3(sp_moveArrow.x, deltaY);

        float deltaPos = cam.ScreenToWorldPoint(sp_moveArrow).z - cam.Screen-
ToWorldPoint(rt_moveArrow.position).z;

        for (int i = 0; i < selectListModel.Count; i++)
        {
            selectListModel[i].transform.position = new Vector3(prevPosModel[i].x,
prevPosModel[i].y, prevPosModel[i].z - deltaPos);
        }
    }

    public void DragLeftRightMove()
    {
        float deltaX = sp_moveArrow.x + Input.mousePosition.x - mouseStartPos.x;
        rt_moveArrow.position = new Vector3(deltaX, sp_moveArrow.y);

        float deltaPos = cam.ScreenToWorldPoint(sp_moveArrow).x - cam.Screen-
ToWorldPoint(rt_moveArrow.position).x;

        for (int i = 0; i < selectListModel.Count; i++)
        {
            selectListModel[i].transform.position = new Vector3(prevPosModel[i].x - deltaPos,
prevPosModel[i].y, prevPosModel[i].z);
        }
    }

    public void Drag4SideMove()
    {
        float deltaX = sp_moveArrow.x + Input.mousePosition.x - mouseStartPos.x;
        float deltaY = sp_moveArrow.y + Input.mousePosition.y - mouseStartPos.y;
        rt_moveArrow.position = new Vector3(deltaX, deltaY);

        float deltaPosX = cam.ScreenToWorldPoint(sp_moveArrow).x - cam.Screen-
ToWorldPoint(rt_moveArrow.position).x;
        float deltaPosZ = cam.ScreenToWorldPoint(sp_moveArrow).z - cam.Screen-
ToWorldPoint(rt_moveArrow.position).z;

        for (int i = 0; i < selectListModel.Count; i++)
        {
            selectListModel[i].transform.position = new Vector3(prevPosModel[i].x - deltaPosX,
prevPosModel[i].y, prevPosModel[i].z - deltaPosZ);
        }
    }

    private void Select()
    {
        if (Input.GetMouseButtonDown(0))
        {
```

```
            selecting = true;
            mouseStartPos = Input.mousePosition;

            guiDrawController.StartDraw();
            selectionStartPoint = cam.ScreenToWorldPoint(mouseStartPos);

        }

        mouseX = Input.mousePosition.x;
        mouseY = Input.mousePosition.y;

        selectionWidth = Mathf.Abs(mouseStartPos.x - mouseX);
        selectionHeight = Mathf.Abs(mouseStartPos.y - mouseY);

        guiDrawController.DrawScreenRectGUI(new Rect(mouseX > mouseStartPos.x ? mouseStartPos.x : mouseX, Mathf.Min(Screen.height - mouseY, Screen.height - mouseStartPos.y), selectionWidth, selectionHeight));
        if (Input.GetMouseButtonUp(0))
        {
            selecting = false;
            DeselectAll();
            guiDrawController.StopDraw();
            if (mouseStartPos == Input.mousePosition)
            {
                SingleSelect();
            }
            else
            {
                MultiSelect();
            }
        }
    }

    private void MultiSelect()
    {

        selectionEndPoint = cam.ScreenToWorldPoint(Input.mousePosition);
        SelectHightlighted();
    }

    private void SelectHightlighted()
    {
        float minX = Mathf.Min(selectionStartPoint.x, selectionEndPoint.x);
        float maxX = Mathf.Max(selectionStartPoint.x, selectionEndPoint.x);
        float minZ = Mathf.Min(selectionStartPoint.z, selectionEndPoint.z);
        float maxZ = Mathf.Max(selectionStartPoint.z, selectionEndPoint.z);


        foreach (GameObject unit in listModel)
        {
            Bounds bounds = unit.transform.GetComponent<Collider>().bounds;
```

```
                if (minX < bounds.min.x && maxX > bounds.max.x && minZ < bounds.min.z && maxZ
> bounds.max.z)
                {
                    unit.GetComponent<Model>().IsSelected = true;
                    selectListModel.Add(unit);
                }
            }
        }

        private void SingleSelect()
        {
            ray = cam.ScreenPointToRay(Input.mousePosition);

            if (Physics.Raycast(ray, out hit))
            {

                if (hit.collider.gameObject.tag == "Model")
                {
                    hit.collider.gameObject.GetComponent<Model>().IsSelected = true;
                    selectListModel.Add(hit.collider.gameObject);
                }
            }
        }


    }
    [Serializable]
    public struct SurfaceObject
    {
        public GameObject surfacePrefab;
        public float layer;
        public float tileValue;
    }
```

ПРИЛОЖЕНИЕ Б

Свидетельство о государственной регистрации программы для ЭВМ

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2019612733

Оптимизация размещения средств наблюдения в трёхмерной сцене с целью минимизации слепых зон

Правообладатель: ***Федеральное государственное бюджетное образовательное учреждение высшего образования «Саратовский государственный технический университет имени Гагарина Ю.А.» (СГТУ имени Гагарина Ю.А.) (RU)***

Авторы: ***Королёв Михаил Сергеевич (RU),***
Печенкин Виталий Владимирович (RU)

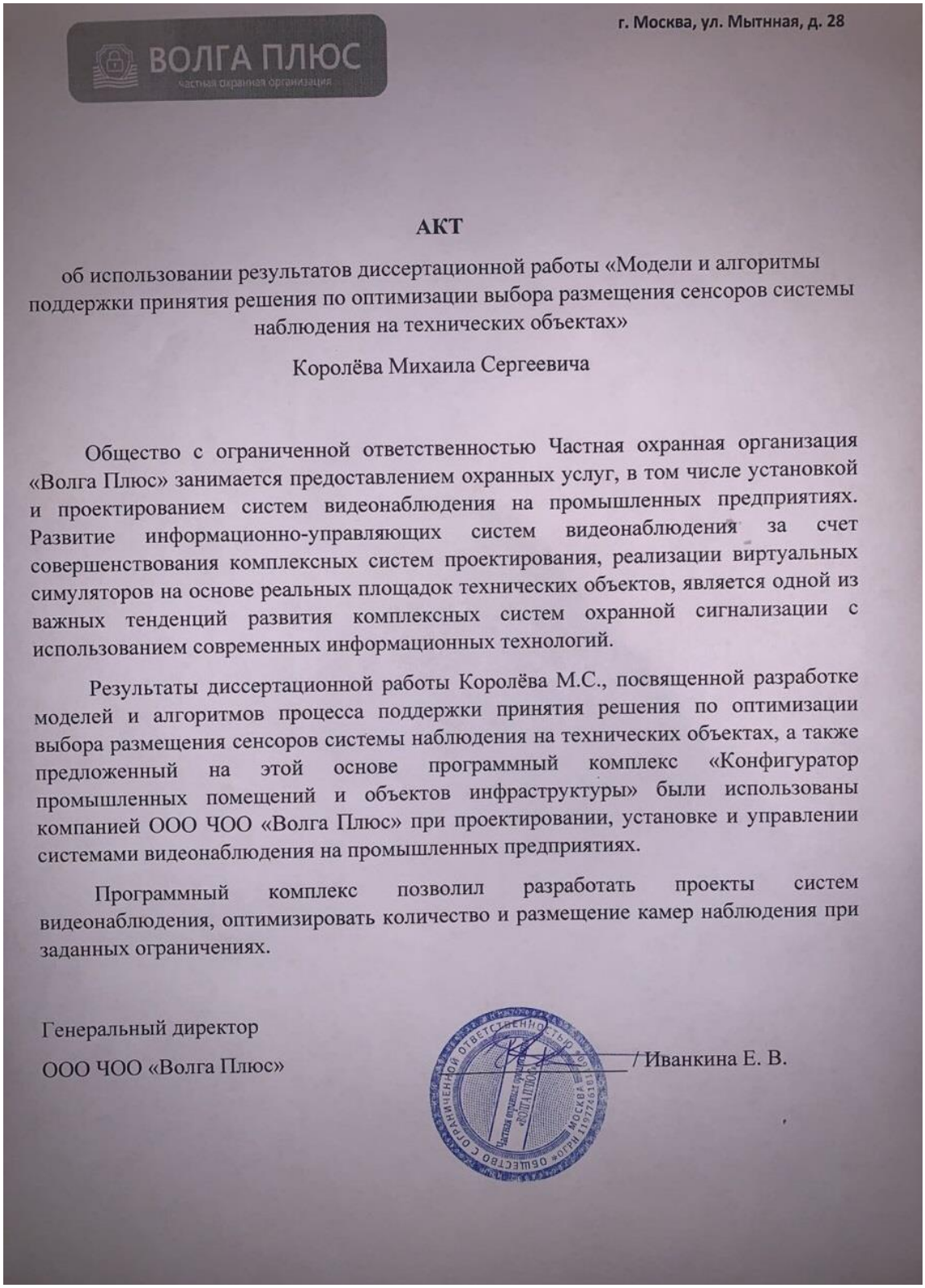
Заявка № **2019611241**
Дата поступления **12 февраля 2019 г.**
Дата государственной регистрации
в Реестре программ для ЭВМ ***26 февраля 2019 г.***

Руководитель Федеральной службы
по интеллектуальной собственности

Г.П. Ивлиев

ПРИЛОЖЕНИЕ В

Акт об использовании результатов диссертационной работы «Модели и алгоритмы поддержки принятия решения по оптимизации выбора размещения сенсоров системы наблюдения на технических объектах»



ВОЛГА ПЛЮС
частная охранная организация

г. Москва, ул. Мытная, д. 28

АКТ

об использовании результатов диссертационной работы «Модели и алгоритмы поддержки принятия решения по оптимизации выбора размещения сенсоров системы наблюдения на технических объектах»

Королёва Михаила Сергеевича

Общество с ограниченной ответственностью Частная охранная организация «Волга Плюс» занимается предоставлением охранных услуг, в том числе установкой и проектированием систем видеонаблюдения на промышленных предприятиях. Развитие информационно-управляющих систем видеонаблюдения за счет совершенствования комплексных систем проектирования, реализации виртуальных симуляторов на основе реальных площадок технических объектов, является одной из важных тенденций развития комплексных систем охранной сигнализации с использованием современных информационных технологий.

Результаты диссертационной работы Королёва М.С., посвященной разработке моделей и алгоритмов процесса поддержки принятия решения по оптимизации выбора размещения сенсоров системы наблюдения на технических объектах, а также предложенный на этой основе программный комплекс «Конфигуратор промышленных помещений и объектов инфраструктуры» были использованы компанией ООО ЧОО «Волга Плюс» при проектировании, установке и управлении системами видеонаблюдения на промышленных предприятиях.

Программный комплекс позволил разработать проекты систем видеонаблюдения, оптимизировать количество и размещение камер наблюдения при заданных ограничениях.

Генеральный директор
ООО ЧОО «Волга Плюс» / Иванкина Е. В.

ПРИЛОЖЕНИЕ Г

Акт об использовании результатов работы в учебном процессе Института прикладных информационных технологий и коммуникаций ФГБОУ ВО «Саратовский государственный технический университет имени Гагарина Ю.А.».

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования «Саратовский государственный технический университет имени Гагарина Ю.А.» (СГТУ имени Гагарина Ю.А.)

ул. Политехническая, 77, г. Саратов, 410054
Телефоны: (8452) 99-88-11;
факс (8452) 99-88-10;
(8452) 99-86-03; факс (8452) 99-86-04
E-mail: sstu_office@sstu.ru

_____№_____

На № _____

АКТ
об использовании результатов кандидатской диссертации
Королёва Михаила Сергеевича

Комиссия в составе:

1. Проректора по учебной работе, д.т.н. проф. Калгановой С.Г.
2. Директора ИнПИТ, д.т.н., заведующий кафедрой «Прикладные информационные технологии» Долининой О.Н.
3. Профессора кафедры «Прикладные информационные технологии», д.ф.–м.н. Бровко А.В.

подтверждает, что результаты диссертационной работы на соискание ученой степени кандидата технических наук Королёва М.С. в области поддержки поддержки принятия решения по оптимизации выбора размещения сенсоров системы наблюдения на технических объектах, направленная на повышение эффективности процесса исследования различных аспектов дискретной оптимизации размещения сенсоров системы видеонаблюдения, внедрены в образовательную деятельность СГТУ имени Гагарина Ю.А. и используются при проведении лекционных и практических занятий по дисциплинам «Прикладные аспекты теории графов» и «Методы обработки многомерных данных» направления подготовки 09.03.02 «Информационные системы и технологии» (бакалавриат).

Председатель комиссии:

Калганова С.Г. _____

Члены:

Долинина О.Н. _____

Бровко А.В. _____