МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский технологический университет «МИСИС»

На правах рукописи

Аль-Саиди Аднан Адаб К.

Метод и алгоритмы планирования маршрутов движения автономного карьерного транспорта с использованием параллельных вычислительных процедур

2.3.1. – «Системный анализ, управление и обработка информации, статистика»

Диссертация

на соискание ученой степени кандидата технических наук

Научный руководитель:

Доктор технических наук Темкин Игорь Олегович

Москва- 2024 год

Содержание

ВВЕДЕНИЕ

Актуальность работы. Автономные транспортные средства и, в частности, колесные роботы, в последние годы все более массово применяются в различных индустриальных сферах, таких как: металлургия, строительство, добыча полезных ископаемых. Как правило, их используют для решения различных производственно-логистических задач. Современное открытое горное производство – это комплекс различного технологического оборудования, снабженного средствами контроля состояния технологических агрегатов и технологической среды. Эти большие машины снабжены разнообразным компьютерным оборудованием и способны обмениваться информацией с использованием различных каналов. Наиболее подготовленными к работе в безлюдном режиме являются роботизированные большегрузные автосамосвалы, оснащенные большим парком различных сенсорных устройств. Сегодня использование роботизированной или полностью автономной карьерной техники для транспортировки горной массы является мировым промышленным трендом в технологиях открытых горных работ. Анализ результатов работы участков с использованием двух автономных самосвалов показывает повышение производительности (объемов перевозимой горной массы за определенный интервал времени) на 10-12%, что достигается за счет равномерности и непрерывности их перемещений. В случае использования большего числа самосвалов-роботов планируемая эффективность может быть еще выше за счет выбора оптимальных маршрутов и рациональных траекторий перемещения роботов, а также выбора оптимальных скоростных режимов на различных участках траекторий. Определение таких маршрутов и траекторий может позволить значительно снизить расходы топлива и степень разрушающих воздействий на узлы и агрегаты автосамосвалов. В этой связи, разработка метода маршрутизации автономных карьерных самосвалов, позволяющего учитывать различные критерии оптимальности, а также эффективных алгоритмов планирования конкретных трасс для перемещения этих машин между основными зонами

карьера с учетом технологических особенностей карьерных дорог и периодически возникающих препятствий, а также скоростных режимов роботов-самосвалов, является актуальной научной задачей.

Целью настоящего исследования является решение задачи оптимальной маршрутизации автономного карьерного транспорта на основе оперативного планирования трасс перемещения автономных самосвалов, с учетом ряда микро-факторов, влияющих на расход топлива и ударные нагрузки на узлы автосамосвалов.

Для достижения указанной цели требуется выполнить следующие **задачи**:

- Провести анализ существующих моделей и алгоритмов, обеспечивающих оптимизацию функционирования горно-ранспортного комплекса (ГТК) на основе различных критериев оптимальности.
- Осуществить аналитический обзор основных подходов к прогнозированию характеристик и оптимизации транспортных систем, включая нейросетевые модели и параллельные вычислительные алгоритмы.
- Разработать процедуру построения оценочных функций, позволяющих сравнить различные потенциальные трассы перемещения автономного самосвала, на основе мультифрактальной («плиточной») цифровой модели карьерных дорог.
- Сформировать критерий оптимальной маршрутизации с учетом конкретных трасс перемещения автосамосвала, учитывающий состояние дорожного полотна и фактическое расположение роботов-самосвалов в различных зонах карьера.
- Разработать алгоритмы, позволяющие вычислять опорные точки для формирования потенциальных трасс перемещения автономных самосвалов.

- Выполнить серию вычислительных экспериментов с использованием цифровой модели, содержащей данные о геометрических пространственных характеристиках и состоянии дорожного полотна карьерных дорог реального карьера, которые подтвердили бы работоспособность разработанного алгоритма.
- Разработать программный модуль, который может быть протестирован в реальных производственных условиях функционирования АСУ ГТК.

Основная идея работы заключается в учете характеристик отдельных участков технологических дорог на основе мультифрактальной цифровой модели транспортных зон карьера, что позволяет представить любой маршрут в виде набора конкретных трасс, а выбор наилучшей трассы осуществлять через поиск оптимального значения функции стоимости перемещения робота-самосвала с использованием взвешенного ациклического графа.

Основные научные положения, выносимые на защиту:

- Метод планирования маршрутов движения автономных самосвалов в карьере, который базируется на использовании мультифрактальной цифровой модели транспортно-технологических зон карьера и формировании эвристической функции стоимости перемещения по маршруту, учитывающей различные факторы, влияющие на эффективность работы карьерного транспорта.
- Модифицированный алгоритм Дейкстры определения оптимальной трассы, построенный с использованием модели разделяемой памяти и библиотеки OpenMP, отличительной особенностью которого является распараллеливание вычислительных процессов только при выполнении циклов, что позволяет существенно снизить время осуществления поиска в графах большой размерности и, следовательно, снизить на 50 - 80% время, затрачиваемое на построение или перестроение трассы для перемещения робота-самосвала.

- Алгоритм прогнозирования скорости робота-самосвала в любой точке дорожного полотна, который реализован на базе оригинальной нейронной сети с граф-конволюционной архитектурой, а результаты его работы используются при формировании функции стоимости и позволяют осуществить более полную оценку вариантов перемещения.

Научная новизна результатов исследования заключается в том, что:

- Разработан новый метод оптимизации процессов перемещения автономных самосвалов по карьерным дорогам, отличающийся тем, что позволяет планировать маршруты движения с учетом выбора конкретных трасс и скоростных режимов перемещения по ним.
- На основе цифровой модели (мультифрактальной «плиточной» структуры) транспортных зон карьера построена эмпирическая функция стоимости перемещения робота-самосвала по различным трассам карьерных дорог и разработаны оригинальные алгоритмы планирования оптимальных трасс перемещения робота - самосвала с использованием этой функции.

Теоретическое значение диссертации заключается в разработке оригинального метода оперативного планирования оптимальных трасс перемещения автономных автосамосвалов с использованием цифровых моделей карьерных дорог, позволяющего оценивать ранее не учитываемые факторы, влияющие на эффективность работы горного транспорта.

Практическая значимость диссертации заключается в том, что:

Разработанный метод позволяет более объективно оценивать эффективность работы автономного горного транспорта за счет учета ряда факторов, непосредственно сказывающихся на рабоспособности роботов-самосвалов при движении по карьерным дорогам.

Разработанные алгоритмы и соответствующие программные модули могут быть использованы в рамках существующих сегодня АСУ ГТК для оперативного планирования трасс, проходящих через вычисляемые на основе

специальных критериев опорные точки, и обеспечивать возможность построения на их основе цифровых траекторий перемещения автономного карьерного транспорта.

Обоснованность и достоверность научных положений, выводов и рекомендаций работы основывается на корректном использовании теории исследования операций, интеллектуальных методов анализа данных и методов поиска оптимальных решений, а также на значительном объеме вычислительных экспериментов, проведенных с использованием реальных данных инфраструктуры карьеров на территории РФ.

О надежности результатов свидетельствует их повторяемость в процессе тестирования разработанных программных средств.

Апробация работы. Основные положения и результаты диссертации докладывались и обсуждались на форумах и конференциях. Среди них:

- 4-й Международный симпозиум «Agent, Multi-Agent Systems and Robotics – 2021 (Малайзия).
- Международный симпозиум «4th IEEE International Youth Conference on Radio Electronics, Electrical and Power Engineering» 2022 (Москва)
- Международная научная-практическая конференция IAICT 2023 (Индонезия).
- XXXII Международный научный симпозиум в рамках «Недели горняка-2024»;

Реализация и внедрение результатов работы:

Разработанные метод и алгоритмы планируется использовать при решении ряда транспортно-логистических задач на индустриальных объектах Иракской Республики.

Публикации. По теме исследований опубликовано 5 работ. В том числе 3 статьи опубликованы в журналах, включенных в реферативную базу Scopus, 2 – в журналах рекомендованных ВАК для публикации основных результатов диссертационных работ по данной специальности.

Перечень опубликованных работ приведен в конце автореферата.

Структура и объем работы. Диссертация включает в себя введение, 4 главы, заключение, список использованных источников. Объем работы составляет 151 стр., в том числе основное содержание – 124 стр., 40 рисунков и 16 таблиц, список литературы из 133 наименований.

1. Анализ состояния проблем, связанных с тематикой диссертации

1.1. Актуальные задачи организации движения автотранспорта в карьерах

История научных исследований с целью решения проблемы повышения эффективности работы ГТК насчитывает более 70 лет. Многие видные ученые СССР и РФ, такие как В.М. Альтшуллер, Ю.И. Анистратов, В.В. Ржевский А.С. Спиваковский [1], внесли существенный вклад в проблему оптимизации планирования и эксплуатации карьерных самосвалов и экскаваторов. Эффективная работа горнодобывающих предприятий, осуществляющих добычу открытым способом, невозможна без развитого горно-транспортного комплекса (ГТК), включающего большое количество машин и оборудования различного назначения. Известно, что более 80% всех грузоперевозок на горнодобывающих предприятиях мира осуществляется при помощи большегрузных автосамосвалов, поэтому значительное внимание уделяется вопросам организации рационального и безопасного перемещения самосвалов между технологическими зонами карьера с грузом и без него. [2]. Решение данной проблемы перешло на новый уровень с начала 70-х годов прошлого века с появлением и широким внедрением средств автоматизации. С исследованиями в области автоматизации и управления ГТК связаны имена таких ученых, как: Ю.Н. Камынин, Л.Д. Певзнер, М.Г. Потапов, Ю.В. Стенин и других. В последующие десятилетия были созданы автоматизированные системы различного уровня сложности, такие как: «Карат», «Кварцит» «Пуск» и «Гранит», в которых поддержка работы диспетчера осуществлялась в режиме советчика. Среди этих отечественных систем наиболее известна система АСУ ГТК «Карьер», которая постоянно развивалась в течение 30-ти лет и сегодня внедрена на десятках горных предприятий РФ, а также в Африке и Латинской Америке.

В контексте быстрого развития робототехники, больших данных, искусственного интеллекта и технологий 5G сегодня наблюдается тенденция все более активного использования интеллектуальных беспилотных самосвалов [3,4]. Решению вопросов, связанных с организацией рациональной и безопасной работы этих мобильных роботов, посвящено значительное количество исследований в Китае, Канаде, Австралии, США ЮАР и Российской Федерации. В России хорошо известны публикации в этой области таких ученых, как К.Е. Трубецкой, С.С Кубрин, А.Ф. Клебанов, Д.А. Клебанов, Д.Н. Сиземов, И.О. Темкин [5]. Лидерами в разработке роботизированных и автономных большегрузных самосвалов являются известные компании: Komatsu (Япония), Caterpillar (США), Euclid-Hitachi (Япония), BELAZ (Белоруссия) [5-7].

По сравнению с существующими технологиями добычи и транспортировки полезных ископаемых самосвалы-роботы потенциально, при условии рациональной организации работ, могут обеспечить более высокую эффективность добычи полезных ископаемых, так как позволяют снизить эксплуатационные расходы за счет сокращения времени простоя оборудования из-за субъективных факторов [5, 8-10]. В последние 10-12 лет ряд инжиниринговых компаний, таких как Modular Mining Systems, Wenco Mining Systems, VIST Robotics, ЦИФРАs [11-14], в сотрудничестве с производителями горнодобывающей техники ведут разработки в этой области и начинают запускать системные пилотные проекты по автономному транспортированию горной массы.

Транспортировка карьерных грузов представляет собой один из ключевых производственных процессов в открытых горных разработках. Основной груз в карьере составляет горная масса, включающая как полезные ископаемые, так и пустую породу [15]. Маршрут транспортировки горной массы определяется конечными пунктами, включающими горные забои, зоны разгрузки и перегрузки. К таким пунктам относятся отвалы пустой породы или нижележащих руд, приемные бункеры погрузочных станций, дробильные,

обогатительные, агломерационные или брикетные фабрики, а также места временного или постоянного хранения полезного ископаемого. Расположение пунктов погрузки и разгрузки, а также применяемая технология горных работ, формируют топологию карьерных дорог, которая может быть описана графовой моделью. Характерные особенности карьерного транспорта включают значительные объемы перевозок (от нескольких десятков тысяч до десятков миллионов тонн ежегодно), однонаправленное движение грузов от забоев к приемным площадкам, большие уклоны на маршрутах и постоянные изменения координат пунктов погрузки и разгрузки, что вызывает регулярные изменения схемы карьерных дорог. [16]. Организация движения карьерного транспорта сталкивается с рядом проблем. К ним относятся: большие временные потери при транспортировке, необходимость минимизации расходов топлива, а также снижение затрат на техническое обслуживание и ремонт. Рассмотрим кратко основные факторы, которые оказываются причиной перечисленных проблем.

1. Возникновение очередей автосамосвалов при подъезде к пунктам погрузки и разгрузки (аналог пробок на дорогах общего назначения). Очевидно, что это приводит к длительным задержкам, увеличению расхода топлива и увеличению выбросов. Рациональная организация движения карьерного транспорта может помочь уменьшить пробки на дорогах за счет оптимизации маршрутов и минимизации количества транспортных средств на дороге.
2. Безопасность является главным приоритетом в транспортной отрасли. Выбор оптимальных маршрутов карьерного транспорта и режимов его движения может помочь существенно снизить вероятность сближения автомобилей на опасную дистанцию и движения по встречным курсам.
3. Проблемы с инфраструктурой, такие как наличие серьезных дефектов части дорожного полотна или неправильно организованные зоны погрузки-разгрузки, могут препятствовать нормальной работе большегрузных автосамосвалов. Однако, использование данных их

бортовой телеметрии может помочь решить эти проблемы путем выявления областей, которые нуждаются в ремонте.

Затраты на техническое обслуживание могут быть существенно снижены при использовании автономных мобильных устройств роботов-самосвалов, которые перемещаются по заданным маршрутам, используя специально определенные трассы (набор конкретных координат на отдельном фрагменте маршрута) и скоростные режимы, что позволяет снизить нагрузку на отдельные узлы и агрегаты автосамосвала. На сегодняшний день карьерные самосвалы являются наиболее продвинутым элементом системы управления горно-транспортным комплексом (ГТК) с точки зрения их функциональных возможностей., так как снабжены бортовой компьютерной системой и различными типами датчиков, с помощью которых осуществляется их пространственное позиционирование и идентификация состояния. То есть, сегодня карьерный самосвал – это роботизированная транспортная платформа, позволяющая регистрировать, обрабатывать и передавать для централизованного анализа огромные массивы информации о координатах и скорости перемещения, а также о состоянии дорожного полотна карьера, что позволяет диспетчерам и водителям оптимизировать режимы движения [17]. С организационно-технологических позиций транспортные процессы в карьерах следует рассматривать как многостадийный замкнутый цикл, в котором задействованы мобильные (большегрузные и условно-стационарные (экскаваторы) агенты, которые функционируют в рамках инфраструктурно - технологической среды, включающей такие объекты, как: экскаваторные забои, борта и уступы, карьерные дороги и технологический зоны различного назначения. Этот процесс включает следующие операционные стадии:

S1: перемещение от экскаваторных площадок (или перегрузочных пунктов) к зонам разгрузки;

S2: движение через зоны взвешивания и разгрузки;

S3: разгрузка (в зонах разгрузки или дробилках);

S4: движение к экскаваторным забоям (зонам погрузки);

S5: маневрирование вокруг экскаватора;

S6: операция загрузки автосамосвалов;

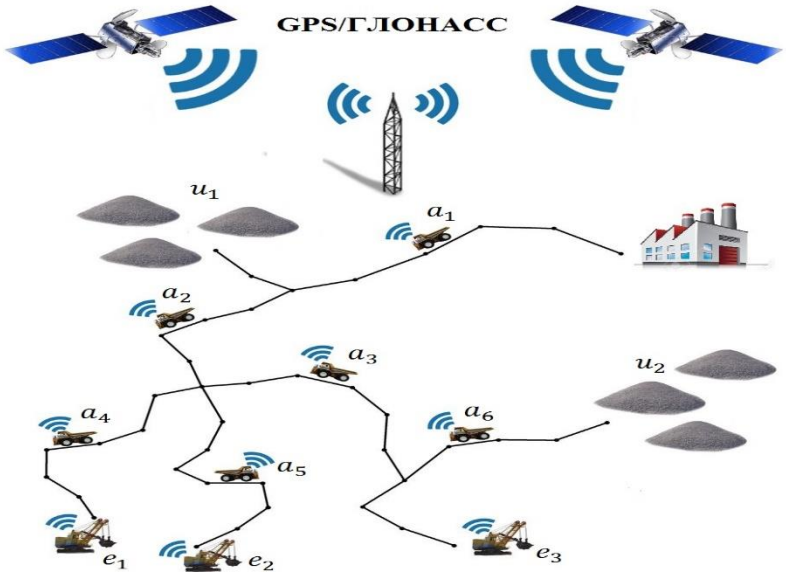


Рис. 1.1 Упрощенная схема транспортно-технологических процессов в карьере

Нужно подчеркнуть, что эффективность работы роботизированных большегрузных автосамосвалов при осуществлении транспортных операций с использованием автономных, частично-автономных или смешанных схем зависит от решения ряда задач:

1. Обеспечения оптимальной маршрутизации, то есть, такого режима движения автосамосвалов, чтобы минимизировать простои транспорта в очередях и максимизировать производительность экскаваторов.
2. Определения наиболее экономных и безопасных схем маневрирования на погрузочно-разгрузочных операциях (в том числе, с использованием заднего хода) с тем, чтобы сократить время и повысить безопасность маневров.
3. Выбор конкретной трассы движения робота и определение таких скоростных режимов, которые обеспечивали бы оптимизацию времени

прохождения трассы при минимизации аварийных рисков и расхода горюче-смазочных материалов.

Решение некоторых из перечисленных задач является предметом настоящего диссертационного исследования.

В целом, карьерный транспорт требует тщательного планирования и управления, с целью обеспечения его безопасной и эффективной работы. Важное место в этих процессах занимают различные методы и инструменты оптимизации и предиктивного анализа.

1.2. Алгоритмы оптимальной маршрутизации

1.2.1. Классические алгоритмы

Рассмотренные вопросы, в целом, связаны с хорошо известной в исследовании операций транспортной задачей. Ее целью является минимизация затрат на транспортировку товаров (горной массы) из одного места в другое [18]. Формальной базовой моделью для решения оптимизационных задач в данной сфере является ориентированный ациклический граф, хотя, учитывая, то обстоятельство, что автосамосвалы могут перемещаться по замкнутому циклу вокруг экскаваторных площадок или внутри зон разгрузки, более строго следует называть эти графы псевдографами. Поскольку в силу структурных особенностей графов, описывающих транспортно-технологическую среду карьера, а именно, разреженности матриц смежности при любом варианте описания среды, аналитическое решение задачи невозможно речь может идти об оптимизационной задаче, а именно задачи поиска пути минимальной стоимости в разреженном взвешенном графе. Существенный вклад в решение различных задач поиска оптимамодификаций транспортных задач, сводящихся к поиску оптимальных решений на графах внесли такие известные ученые, как Л. Форд, Р. Беллман, Р. Дейкстра, Д. Фалкерсон, Г. Вагнер, Дж. Левит Т. Саати и многие другие. На сегодняшний момент алгоритмам поиска

оптимальных решений в направленных графах посвящены десятки тысяч статей. В частности алгоритм Левита – алгоритм, позволяющий находить кратчайшее расстояние в графе от 1 вершины до всех остальных, работает с взвешенными ориентированными графами [19], а его асимптотическая сложность в идеальном варианте реализации равна $O(EV\hat{}2)$, где V – количество вершин в графе, а E – количество ребер.

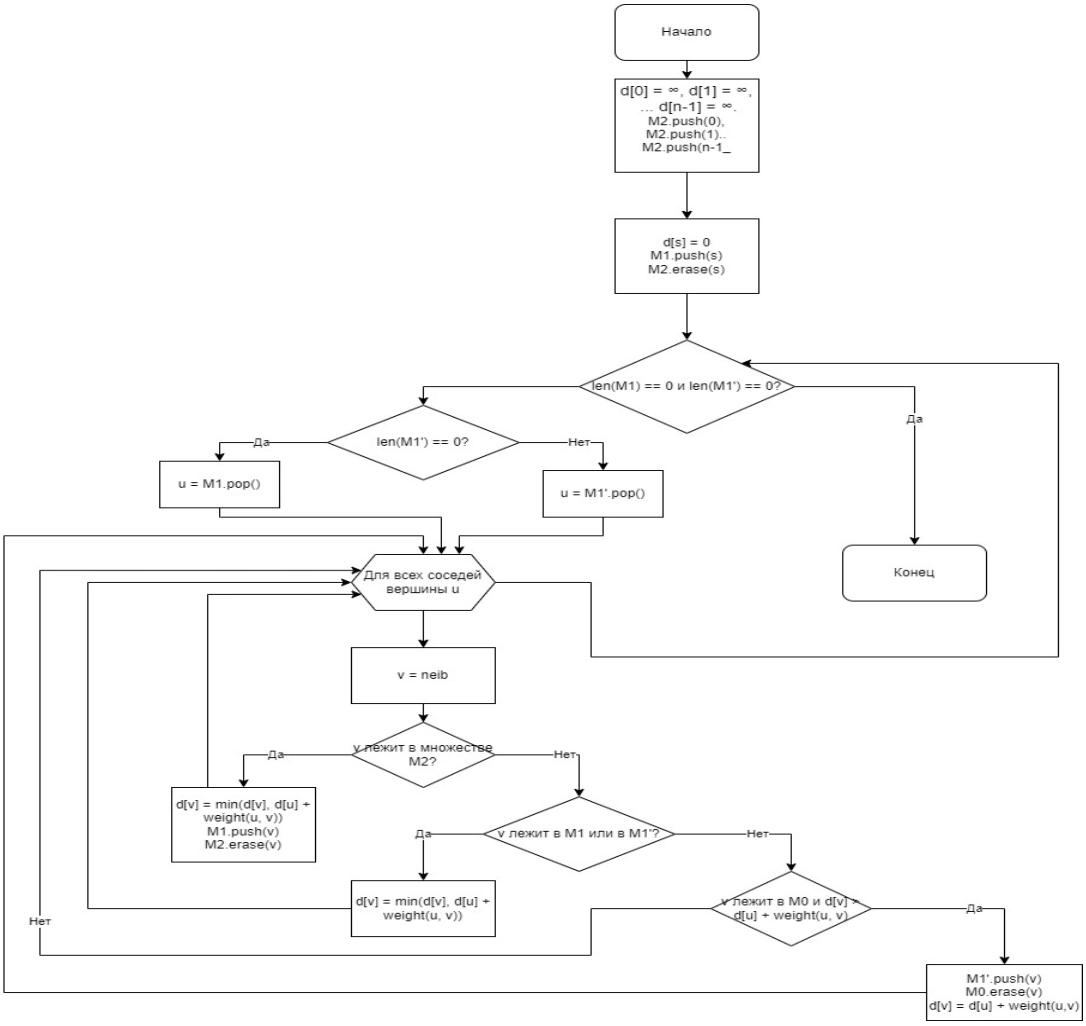


Рис. 1.2 Блок схема алгоритма Левита

Данный алгоритм позволяет восстановить последовательность вершин, которая является кратчайшим путем между двумя точками. Другим весьма популярным алгоритмом является алгоритм Дейкстры. Известно, что он работает только с графами с положительными весами ребер [20], и его асимптотическая сложность оценивается $O(E * \log(V)$.

Алгоритм с хорошей асимптотической сложностью получил признание благодаря своей простоте реализации и высокой скорости работы.

Алгоритм Дейкстры может быть оптимизирован для работы с разреженными матрицами за счет использования «Фибоначчиевых куч» [21] вместо двоичных деревьев для поддержания минимума. При этом асимптотическая сложность алгоритма Дейкстры составит $O(V * \log(V) + E)$. Одна, в случае использования матриц большой размерности, его временные характеристики могут существенно ухудшиться.

1.2.2. Оптимизационные методы в системах диспетчерского управления ГТК

В последние годы проблеме планирования в карьерах уделяется значительное внимание. Паттерсон и др. [22] построили уникальную задачу линейного программирования смешанных целых чисел для планирования работы нескольких грузовиков и использовали для ее решения алгоритм поиска табу, стремясь минимизировать энергопотребление грузовиков и экскаваторов. Чжан, Юань, Ван, Бастос, Чжан и Бао [23–28] использовали аналогичные стратегии для построения модели. Однако цели оптимизации в этих сформулированных задачах учитывали только расход топлива, игнорируя такие факторы, как потребляемое время и объем выработки. Фердус и др. [29] предложили алгоритм многокритериальной оптимизации (MOO) для планирования движения грузовых автомобилей. Zhang и др. [30] предложили основанный на декомпозиции алгоритм генетического принципа ограниченного доминирования (DBCDP-NSGA-II) для решения многокритериальной задачи интеллектуального планирования для грузовиков в открытых карьерах. Хорсанд, Хоссейнзаде и Лю [31–33] также построили многокритериальную модель планирования. Однако общее ограничение ссылок. [29–33] заключается в том, что сформулированная задача не учитывала потребности в заправке и зарядке грузовых автомобилей. Чжан и др. [34] предложили мета эвристический алгоритм поиска для решения задачи смешанно-целочисленного программирования, сформулированной для рассматриваемой схемы планирования работы нескольких грузовиков, и

экспериментально продемонстрировали, что этот подход повышает энергоэффективность транспортной системы в карьерах. Смит и др. [35] предложили дискретизированную по времени модель смешанного целочисленного программирования (MIP) для задачи планирования движения грузовиков в карьерах, а эвристика используется для быстрого создания высококачественных возможных решений. Однако предложенная модель игнорировала путь планирование мест погрузки и разгрузки. Аналогичным образом, Цзэн, Кляйн и Ли [36-38] также не рассматривали планирование пути между загрузкой и разгрузкой горшков. Большинство предыдущих моделей планирования фокусировались на одноцелевой оптимизации, особенно на потреблении энергии, и часто упускали из виду необходимость нескольких целей оптимизации. Кроме того, эти методы были сосредоточены исключительно на перевозке грузовых автомобилей без учета требований к заправке или зарядке грузовиков. Планирование пути между местами погрузки и разгрузки, важнейший аспект планирования, также в значительной степени игнорировалось в предыдущих исследованиях. В заключение этого подраздела следует отметить, что распространенные методы планирования не моделируют конкретную динамику/кинематику и другие временные ограничения работающих устройств; таким образом, нужно решать практически важную задачу планирования с учетом множества мельчайших факторов.

1.3. Обзор моделей параллельных вычислений

Параллельные вычисления представляют собой метод, при котором многочисленные вычислительные задачи или процессы выполняются одновременно. Этот подход широко применяется в сфере высокопроизводительных вычислений (High-Performance Computing, HPC) и является необходимым для моделирования и анализа сложных явлений реального мира [39].

Параллельные вычисления охватывают процесс разделения крупных задач на более мелкие, автономные и, часто, аналогичные фрагменты, которые могут быть одновременно обработаны несколькими процессорами. Эти процессоры взаимодействуют через общую память и объединяют результаты по завершении как часть единого алгоритма. Основная цель параллельных вычислений заключается в увеличении доступной вычислительной мощности для ускоренной обработки задач и приложений. Распространение и развитие параллельных вычислений в 21 веке произошли в ответ на проблему с масштабированием частоты процессоров, когда увеличение частоты приводило к увеличению энергопотребления и перегреву процессоров. Для решения этих проблем программисты и производители начали разрабатывать параллельное программное обеспечение и создавать многоядерные процессоры с улучшенной энергоэффективностью. С ростом использования многоядерных процессоров и графических ускорителей продолжает расти и важность параллельных вычислений. Графические ускорители работают в синергии с центральными процессорами, увеличивая пропускную способность данных и количество параллельных вычислений в приложениях. Благодаря использованию параллелизма, графический процессор способен выполнять больший объем работы по сравнению с центральным процессором за тот же промежуток времени. [40].

Алгоритмы и комплексные программные приложения обладают параллелизмом с различными уровнями регулярности и масштаба. Параллелизм может быть применен на различных уровнях гранулярности, таких как биты, инструкции, данные и задачи [41]. Параллелизм на уровне битов используется прозрачно для программиста за счет увеличения размера процессорного слова, что позволяет уменьшить количество инструкций, необходимых для выполнения элементарных арифметических операций и операций с битами. Аналогично параллелизм на уровне инструкций может использоваться процессорами и компиляторами практически без участия программиста. Процессоры поддерживают дублирование выполнения

инструкций с помощью таких приемов, как конвейеризация инструкций, суперскалярное и внепорядковое выполнение, а оптимизирующие компиляторы могут повысить эффективность этих приемов за счет тщательного переупорядочивания инструкций программы.

На более высоких уровнях гранулярности, таких как уровень данных, параллелизм, как правило, не может быть использован без вмешательства программиста, и основную роль играет модель программирования. Циклы в языках программирования являются основным источником параллелизма на уровне данных, поэтому этот вид параллелизма также называют параллелизмом на уровне циклов. Многие процессоры включают в себя инструкции типа "одна инструкция - несколько данных" (SIMD), которые оперируют короткими векторами, содержащими, как правило, от двух до восьми значений данных. Эти инструкции отличаются от инструкций с одной инструкцией и одними данными (SISD) [42], поскольку последние могут одновременно вычислять только один результат. Некоторые модели программирования позволяют программисту использовать SIMD-инструкции как явно, так и неявно, с помощью векторизующего компилятора. В любом случае необходимо, чтобы параллельные вычисления можно было выразить в виде операций над векторами, т.е. чтобы одна инструкция могла быть применена к нескольким значениям данных.

Другим важным способом использования параллелизма на уровне данных является многопроцессорность или многопоточность, когда параллельные вычисления распределяются между несколькими процессорами, одним процессором, Архитектуры, способные параллельно выполнять несколько потоков инструкций или их комбинаций, описываются как машины с несколькими инструкциями и несколькими данными (MIMD). Эти архитектуры могут обрабатывать несколько независимых потоков инструкций, каждый из которых оперирует с отдельными данными. Если в гнезде циклов обнаруживается параллелизм данных, нагрузка рабочих задач распределяется между обработчиками, присваивая каждому из них

определенное количество итераций для выполнения. В отличие от SIMD-инструкций, распределение работы может быть динамическим, если время обработки каждой итерации изменяется.

Наиболее грубым и наименее регулярным видом параллелизма является параллелизм на уровне задач, который почти во всем зависит от способности программиста определить части приложения, которые могут выполняться независимо друг от друга. Параллелизм на уровне задач связан с распределением как вычислений, так и данных между элементами обработки, в то время как параллелизм на уровне данных делает акцент на распределении данных между элементами обработки. Модели программирования существенно различаются по поддержке использования параллелизма на уровне задач: некоторые из них ориентированы на параллелизм на уровне задач и предоставляют достаточно гибкие абстракции для использования параллелизма данных, в то время как другие ориентированы на использование структурированного параллелизма или параллелизма на уровне циклов и обеспечивают лишь слабую поддержку нерегулярного и динамического параллелизма на уровне задач.

Кроме вышеуказанных соображений об уровне и типе параллелизма, модели параллельного программирования можно классифицировать и по абстракции памяти, которую они предоставляют программисту. Организация памяти в типичных параллельных вычислительных системах позволяет выделить две широкие архитектурные категории. В мультипроцессорах с централизованной или общей памятью память доступна всем процессорам с единым временем доступа (UMA) [44]. Процессоры обычно подключены к шине, разделяя ее пропускную способность - такая архитектура также известна как симметричный мультипроцессор (SMP). К сожалению, при увеличении числа процессоров шины и централизованная память страдают от повышенного уровня конкуренции и увеличения задержек. Такая ограниченная масштабируемость приводит к тому, что централизованная память используется только в небольших мультипроцессорах, обычно с

числом процессоров до восьми. Они также встречаются в отдельных узлах больших систем.

При второй организации память физически распределяется между процессорами. Существует две возможные политики доступа к локальной памяти. В первом случае локальная память доступна только соответствующему процессору, во втором - она может быть доступна всем процессорам, но с неравномерным временем доступа к памяти (NUMA), реализованным через глобальное адресное пространство. Первый вариант называется (чистой) организацией распределенной памяти, а второй - распределенной общей памятью, которая может быть как когерентной (ccNUMA), так и нет. Хотя масштабируемые системы ccNUMA сложны в проектировании, они создают меньшую нагрузку на программиста, поскольку репликация и когерентность прозрачно управляются аппаратными средствами [45]. Системы распределенной общей памяти, лишенные кэш-когерентности, требуют от программиста либо обеспечения согласованности представлений каждого процессора о памяти с основной памятью, либо вмешательства программного уровня, который управляет согласованностью автоматически за счет некоторого снижения производительности.

Модели программирования, применяемые для определенной платформы, тесно связаны с базовыми архитектурами памяти и коммуникаций. Модели с общей памятью позволяют программисту определять объекты, такие как переменные, доступные всем исполнителям (например, процессам или потокам), и наиболее эффективно реализуются на машинах с аппаратным обеспечением, поддерживающим согласованность памяти и кэша в системе. Явная передача сообщений, являющаяся доминирующим представителем моделей распределенной памяти, обычно используется в тех случаях, когда память нецентрализованная и не когерентна. Однако как организации с общей, так и с распределенной памятью могут поддерживать любую модель программирования, хотя и с разной степенью эффективности. Например, на архитектурах с общей памятью можно очень эффективно поддерживать

передачу сообщений, передавая указатель, а не копируя данные. Глобальная модель общей памяти также может поддерживаться на машине с распределенной памятью как аппаратное (ccNUMA), так и программное с помощью уровня времени выполнения [46], хотя получение приемлемого ускорения на широком спектре не модифицированных параллельных приложений является сложной задачей и до сих пор остается предметом активных исследований.

Наконец, для классификации моделей параллельного программирования можно рассматривать множество других различных признаков. Например, они могут быть классифицированы по области применения или по способу определения, управления или сопоставления с реальным аппаратным обеспечением объектов выполнения (рабочих). Другим признаком является парадигма программирования (процедурная, объектно-ориентированная, функциональная, потоковая и т.д.). В этой главе мы сосредоточимся на абстракции памяти, предлагаемой моделью программирования.

Модель общей памяти — это разновидность модели параллельного программирования, которая позволяет нескольким потокам или процессам обращаться к одному и тому же пространству общей памяти. В этой модели каждый поток или процесс имеет свой собственный контекст выполнения, но они могут взаимодействовать и синхронизироваться друг с другом путем чтения и записи в общие области памяти [47][48].

Существует два основных типа моделей общей памяти: симметричная многопроцессорная обработка (SMP) и неравномерный доступ к памяти (NUMA). В системах SMP каждый процессор имеет равный доступ к общей памяти, а в системах NUMA общая память физически распределена между несколькими узлами, и каждый узел имеет собственную локальную память.

Модели с общей памятью имеют ряд преимуществ перед другими моделями параллельного программирования, таких как простота программирования и высокая производительность на многоядерных процессорах [49]. Однако они также могут приводить к возникновению таких

проблем, как условия гонки и тупиковые ситуации, для предотвращения которых требуется тщательное управление. Для этого обычно используются такие примитивы синхронизации, как блокировки, семафоры и барьеры.

В целом модель общей памяти является мощным и эффективным способом распараллеливания приложений и использования преимуществ многоядерных процессоров и других архитектур параллельных вычислений. Однако она требует тщательного управления, чтобы избежать проблем, и разработчики должны тщательно балансировать между производительностью и необходимостью синхронизации и взаимодействия между потоками или процессами.

Модели с общей памятью можно классифицировать по типу используемых объектов исполнения и способу работы с ними программиста (явное или неявное создание, отображение и разделение рабочей нагрузки).

Библиотека OpenMP

Поскольку потоки являются универсальным, хотя и низкоуровневым примитивом для параллельного программирования, утверждается, что они не должны использоваться непосредственно прикладным программистом, а должны использоваться абстракции более высокого уровня, которые, возможно, отображаются на потоки базовой средой выполнения. OpenMP [50] можно рассматривать как реализацию этой философии. Это набор директив компилятора и библиотечных функций, которые используются для распараллеливания последовательного кода на языке Си/Си++ или Фортран. Важной особенностью OpenMP является поддержка инкрементального параллелизма, при котором директивы могут добавляться постепенно, начиная с последовательной программы [51].

Как и Pthreads, OpenMP является явно параллельной моделью программирования, то есть компилятор не анализирует исходный код для выявления параллелизма. Программист указывает компилятору, как следует распараллелить код, но, в отличие от Pthreads, в OpenMP потоки не являются

явным понятием, программист создает и управляет ими в основном неявно, с помощью директив более высокого уровня [52].

OpenMP поддерживает модель параллелизма fork-join. Программы начинают выполняться в одном главном потоке, который порождает дополнительные потоки по мере появления распараллеленных областей (заключенных в директиве omp parallel), т.е. вилки. Параллельные области могут быть вложенными, хотя компилятор не обязан использовать более одного уровня параллелизма. В конце крайнего параллельного региона ведущий поток объединяется со всеми рабочими потоками перед продолжением выполнения.

Параллельный регион, по сути, реплицирует задание на набор порожденных потоков. Потоки могут взаимодействовать между собой, выполняя различные части задания с помощью конструкций разделения работ. Наиболее известной из таких конструкций является директива omp for (C/C++) или omp loop (Fortran), в которой итерации смежного цикла делятся и распределяются между участвующими потоками. Это позволяет легко распараллеливать регулярные гнезда циклов и было основным преимуществом OpenMP, поскольку такие циклы широко распространены в научных кодах, таких как линейная алгебра или моделирование физических явлений на прямоугольных сетках. Начиная с версии 3.0 стандарта [53] применимость OpenMP была значительно расширена в приложениях с динамическим и нерегулярным параллелизмом (например, когда работа создается рекурсивно или содержится в циклах с неизвестным числом итераций) благодаря добавлению директивы omp task. Задачи — это блоки кода, которые помечаются программистом и могут выполняться асинхронно любым потоком. В силу своей асинхронности задачи также должны нести копию данных, с которыми они будут оперировать в момент выполнения.

Не будет преувеличением сказать, что OpenMP сегодня стал стандартом де-факто для программирования с общей памятью. Он используется для программирования мультипроцессоров и многоядерных процессоров,

независимо от того, разделяют ли они память физически или имеют организацию ccNUMA. Он также успешно реализован для ряда встраиваемых платформ [54]. Существуют даже реализации ранних версий OpenMP, ориентированные на вычислительные кластеры с использованием программных библиотек с общей виртуальной памятью (хотя и не достигшие высокой производительности).

POSIX Threads

Поток — это автономная исполнительная единица с собственным программным счетчиком и стеком выполнения, которая обычно описывается как "легкий" процесс. Обычный ("тяжелый") процесс может порождать несколько потоков; будучи автономными, потоки совместно используют код процесса, его глобальные переменные и любые данные, распределенные в куче. Интерфейс переносимых операционных систем (POSIX) предоставляет стандартный интерфейс для создания потоков и управления ими в пользовательской программе [55]. POSIX — это стандартизация интерфейса между операционной системой и приложениями в семействе операционных систем Unix. Расширения потоков POSIX.1c [56] содержат описание синтаксиса и семантики функций и типов данных, используемых для создания потоков и управления ими, и известны как Pthreads.

В параллельных приложениях несколько потоков создаются вызовом pthread_create. Поскольку скорость и последовательность выполнения различных потоков непредсказуемы и по умолчанию не упорядочены, программисты должны быть осведомлены об условиях гонки и тупиковых ситуациях. Если операции должны выполняться в определенном порядке, следует использовать синхронизацию. В качестве основного механизма синхронизации в POSIX используются переменные условий. Переменные условия предоставляют структурированное средство для ожидания (блокировки) потока до тех пор, пока другой поток не подаст сигнал о том, что определенное условие стало истинным (вызовы pthread_condition_wait / pthread_condition_signal). Расширения реального времени к POSIX [57]

определяют барьеры как дополнительный механизм синхронизации для Pthreads. Поток, вызывающий pthread_barrier_wait, блокируется до тех пор, пока все соседние потоки не выполнят тот же вызов; затем все они освобождаются для продолжения выполнения.

Pthreads предоставляет объекты mutex в качестве основного способа достижения взаимного исключения, которое обычно используется для координации доступа к ресурсу, разделяемому между несколькими потоками. Поток должен заблокировать (pthread_mutex_lock) мьютекс перед входом в критическую секцию кода и разблокировать его (pthread_mutex_unlock) сразу после выхода из нее. Pthreads также предоставляют семафоры в качестве еще одного механизма взаимного исключения.

Pthreads считаются достаточно низкоуровневой моделью программирования, которая возлагает на программиста слишком большую нагрузку. Явное управление и манипулирование сущностями выполнения иногда может предоставить полный контроль программисту-эксперту [58], но за это приходится платить повышенной сложностью и снижением производительности, поскольку большие Pthreads-программы считаются сложными для разработки, отладки и сопровождения.

Модели с распределенной памятью — это класс вычислительных моделей, предназначенных для решения задач обработки больших объемов данных и сложных вычислений за счет использования нескольких параллельно работающих вычислительных устройств [59]. Эти модели широко используются в высокопроизводительных вычислительных системах и системах параллельной обработки данных [50].

В моделях с распределенной памятью данные распределяются по нескольким блокам памяти или узлам, и каждый узел имеет собственную локальную память. Связь и координация между узлами осуществляется путем передачи сообщений, обычно с использованием интерфейса передачи сообщений (MPI). Это позволяет узлам совместно работать над общей задачей, обмениваясь информацией и промежуточными результатами.

Одной из часто используемых моделей распределенной памяти является модель интерфейса передачи сообщений - Message-Passing (MPI). В этой модели каждый узел работает независимо и имеет свою собственную память. Взаимодействие узлов осуществляется путем явной отправки и получения сообщений, что позволяет им обмениваться данными и синхронизировать свои вычисления. Эта модель особенно подходит для архитектур с распределенной памятью, таких как кластеры компьютеров или суперкомпьютеры.

Модели распределенной памяти обладают рядом преимуществ. Во-первых, они позволяют обрабатывать большие объемы данных, распределяя их по нескольким узлам и обеспечивая параллельную обработку. Это позволяет значительно сократить время выполнения вычислений по сравнению с последовательной обработкой. Во-вторых, такие модели обеспечивают масштабируемость, поскольку для обработки больших массивов данных или более сложных вычислений в систему могут быть добавлены дополнительные узлы. Наконец, модели с распределенной памятью обеспечивают отказоустойчивость, поскольку отказ одного узла не обязательно приводит к отказу всей системы [60].

Однако модели распределенной памяти сопряжены и с определенными трудностями. Явная передача сообщений и синхронизация, требуемые в этих моделях, могут приводить к накладным расходам и усложнять программирование. Для достижения эффективного параллелизма разработчикам необходимо тщательно прорабатывать схемы взаимодействия и управлять ими. Кроме того, распределение данных и балансировка нагрузки между узлами могут оказаться сложной задачей, поскольку неравномерное распределение или несбалансированная рабочая нагрузка могут привести к узким местам в производительности.

Таким образом, модели с распределенной памятью — это вычислительные модели, использующие несколько параллельно работающих вычислительных устройств для обработки больших объемов данных и

сложных вычислений. Они обладают такими преимуществами, как параллельная обработка, масштабируемость и отказоустойчивость, но при этом требуют тщательного проектирования и управления коммуникациями и распределением нагрузки. Эти модели широко используются в высокопроизводительных вычислениях для решения задач с высокой интенсивностью вычислений [50],[61].

1.4. Обзор возможности применения ИНС различных типов в задачах прогнозирования характеристик транспортных потоков

В данном разделе рассматривается несколько активно используемых сегодня нейронных сетей, которые могут более или менее успешно использоваться для решения задач прогнозирования характеристик транспортных потоков и, в частности, усредненных или индивидуальных скоростей участников движения. CNN (Convolutional Neural Network) — это высокоэффективный метод глубокого обучения распознаванию образов. Первым примером использования сверточных нейронных сетей, было их применение Янном Лекуном в распознавании рукописных данных [62]. В последующие годы CNN стала широко применяться в различных областях. В основном она используется в задачах анализа изображений и видео, например таких, как общая классификация изображений, распознавание лиц и объектов и др. Точность получаемых решений, как правило, намного выше, чем у классических нейронных сетей перестроенного типа. Примечательной чертой данной модели является возможность автоматического извлечения характеристик изображений при помощи конволюционного слоя. Этот подход сокращает необходимость в ручном извлечении характеристик и одновременно существенно повышает точность распознавания изображений. Это уменьшает зависимость людей от знаний, связанных с анализом изображений, и повышает прикладную ценность.

CNN — это разновидность нейронной сети прямой передачи данных (feed forward), образованная перекрестным суммированием конволюционного

слоя, слоя под-выборки (pooling) в различных комбинациях и полностью связанного слоя [63].

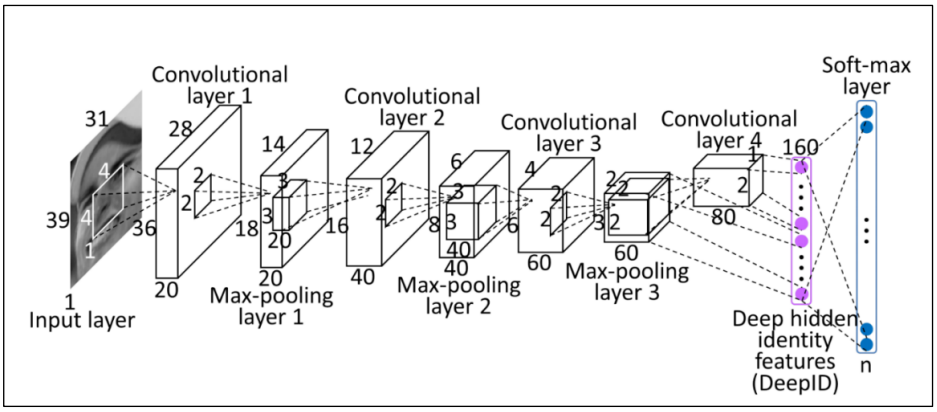


Рис. 1.3 Типичная структура конволюционной нейронной сети

Архитектура включает в себя входной слой, четыре сверточных слоя, три слоя пулинга, полно-связный слой и слой SoftMax. Как показано на схеме, CNN обладает четкой иерархической структурой. Входные данные представлены необработанными данными, такими как RGB-изображения. Принцип работы CNN заключается в извлечении глубинной информации, скрытой в исходных данных из входного слоя, путем последовательности операций, таких как свертка, пулинг и применение функций активации. Результатом являются локальные характеристики исходных данных. В случае решения задачи классификации операция SoftMax используется для получения вероятностных оценок принадлежности изображений к определенным категориям. [64]. Эта процедура позволяет преобразовать локальные характеристики в вероятностные распределения. В сверточных нейронных сетях различные операции представлены как "слои". Например, в сверточных нейронных сетях сверточный слой применяется для выполнения операции свертки, а слой пулинга используется для агрегации и снижения размерности исходных данных. Поскольку CNN может эффективно извлекать признаки изображения с помощью операций свертки, Эффективность извлечения признаков сверточных нейронных сетей (CNN) в евклидовом пространстве признается высокой в связи с временными и пространственными двумерными характеристиками транспортного потока. В предшествующих

исследованиях, посвященных прогнозированию транспортных потоков, CNN применялись преимущественно для извлечения пространственной корреляции данных о транспортных потоках. Это обусловлено неевклидовыми особенностями транспортной сети, которые затрудняют их прямое преобразование и применение к CNN в виде матриц. Поэтому исследователи часто преобразуют данные о транспортных потоках в изображения структуры транспортной сети, а затем матрицуют их с помощью сеток. Различные типы сеток могут быть использованы для представления различных областей движения. Таким образом, CNN способна извлекать и распознавать пространственные характеристики различных участков дорожного движения. Основные различия в применении CNN в области прогнозирования транспортных потоков заключаются в различных сценариях использования, улучшении моделей и разном понимании структуры транспортных сетей или данных о транспортных потоках [65].

Бао и др. [66] предложили новую сквозную модель ST-3DNet для извлечения растровых данных о дорожном движении. В своем исследовании исследователи добавили данные о времени к матричным двумерным пространственным данным, чтобы сформировать новые трехмерные пространственно-временные данные. Исследования по оптимизации и улучшению модели CNN также можно увидеть в таких исследованиях. Ею и др. [67] применили CNN для прогнозирования спроса в сфере онлайн-проката автомобилей. В исследовании предложена эффективная архитектура модели с полностью сверточной сетью и встраиванием с учетом времени для изучения сложных пространственно-временных характеристик. В исследовании эта модель используется для прогнозирования будущего спроса на такси. Модель в основном оптимизирована и улучшена на основе базовой модели CNN, с использованием объединения средних и одномерной свертки для удовлетворения актуальных потребностей исследования.

Из приведенных выше исследований видно, что CNN имеет преимущества при обработке растрированных сетевых данных. Поэтому многие исследователи применяют ее для обработки пространственных данных транспортных сетей. Однако, с точки зрения конкретных приложений, разумное представление? входных данных в евклидовом пространстве является сложным моментом для исследования. Поэтому применение модели CNN в области прогнозирования транспортных потоков имеет ограничения. Многие модели и методы хорошо работают в конкретных сценариях. Однако, большинство моделей на основе CNN не являются универсальными. Модель, которая хорошо справляется с одной задачей, имеет огромное падение производительности при применении к другим задачам.

В биологических нейронных сетях связь между нейронами не является однонаправленной. Поэтому статическая сеть - сеть с прямой передачей данных, размеры ее входа и выхода фиксированы. Поэтому при работе с различными последовательностями (текст, голос, видео) необходима нейросетевая модель со способностью к обучению в условиях, когда обучающий набор непрерывно меняется. Примером нейронной сети, способной обрабатывать данные последовательности произвольной длины, является рекуррентная нейронная сеть (RNN). Она обладает возможностью обратной связи и саморегуляции, что позволяет ей эффективно работать с последовательными данными. В настоящее время рекуррентные нейронные сети широко применяются в различных областях, включая распознавание речи, языковые модели, прогнозирование на основе последовательных данных и генерацию естественного языка [68].

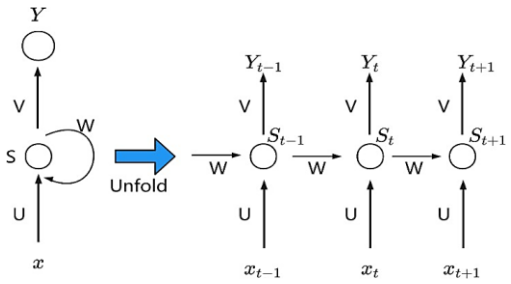


Рис. 1.4. Структура RNN

Общая структура рекуррентной нейронной сети (RNN) показана на рис.1.4 [69]. Структура в левой части является общей и может быть развернута на несколько временных шагов. Здесь X_t представляет собой входной вектор на t-м временном шаге. Y_t представляет выходной вектор на t-м временном шаге. S_t представляет вектор скрытого слоя на t-м временном шаге. U, V и W представляют матрицу параметров отображения и являются общими на каждом временном шаге. Скрытый вектор S_t каждого временного шага в RNN содержит информацию исторического временного шага. Процесс расчета модели RNN с одним скрытым слоем представлен в формулах (1) и (2), где f представляет собой нелинейную функцию активации, которая может быть такой функцией, как tanh и ReLU.

$$S_t = f\,(UX_t + WS_{t-1} + b_s) \tag{1.1}$$

$$Y_t = VS_t + b_y \tag{1.2}$$

Хотя скрытый вектор RNN может сохранять часть информации об историческом временном шаге, модель RNN имеет недостатки, такие как недостаточная долговременная память, исчезновение градиента или взрыв градиента из-за своей простой структуры. Поэтому некоторые ученые предложили вариант улучшения модели RNN модель - LSTM (Long Short-Term Memory).

Для решения проблемы неспособности RNN сохранять долговременную память Заргар и другие [70] предложили модель LSTM. На этой основе многие исследователи внесли определенные улучшения в модель с точки зрения практического применения [71]. Как показано на Рис. 1.5: LSTM добавляет трое "ворот" для управления входом, выходом и скрытым слоем информации о состоянии в RNN. Они соответствуют входным воротам, выходным воротам и воротам забывания, соответственно. Более того, для решения проблемы исчезновения градиента в RNN, LSTM также добавляет нейрон памяти для облегчения этой проблемы. Эти четыре части взаимодействуют друг с другом для определения информации памяти или информации забывания [72], [73].

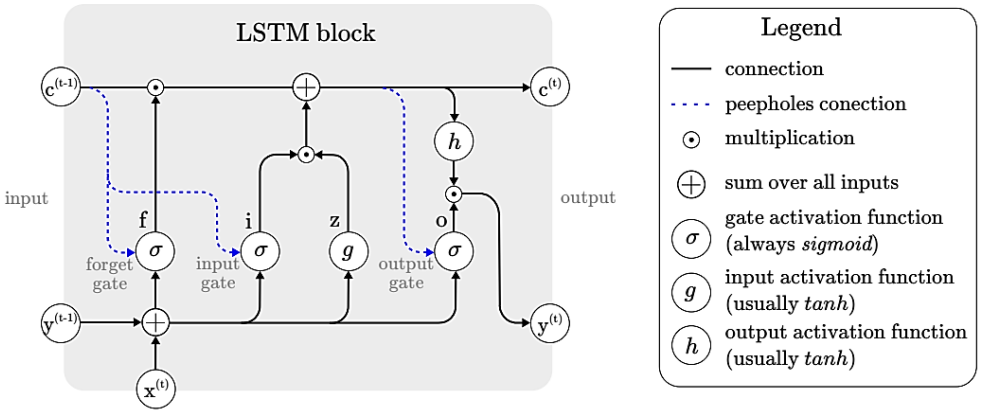


Рис. 1.5. Архитектура типичного блока LSTM

В частности, предположим, что $x_t \in \mathbb{R}^{n\times c}$ — это текущая n входная последовательность данных с размерностью c, а $H_{t\text{-}1} \in \mathbb{R}^{n\times h}$ - скрытый слой на предыдущем временном шаге, Затем входные ворота $I_t \in \mathbb{R}^{n\times h}$, ворота забывания $F_t \in \mathbb{R}^{n\times h}$ и выходные ворота $O_t \in \mathbb{R}^{n\times h}$ расчеты производятся в соответствии с следующими формулами:

$$I_t = (W_{in}x_t + W_{ih}H_{t-1} + b_i) \tag{1.3}$$

$$F_t = (W_{fn}x_t + W_{fh}H_{t-1} + b_f) \tag{1.4}$$

$$O_t = (W_{on}x_t + W_{oh}H_{t-1} + b_o) \tag{1.5}$$

Среди них W и b являются обучаемыми весовыми параметрами и параметрами смещения, соответственно. А ворота забывания и входные ворота управляют нейроном памяти $C_t \in \mathbb{R}^{n\times h}$ по формуле (6):

$$C_t = F_t \odot C_{t-1} + I_t \odot \tilde{C}_t \tag{1.6}$$

Где $\odot$ представляет собой умножение, в котором элемент соответствует умножению. $\tilde{C}_t$ представляет собой нейрон-кандидат на память. В качестве функции активации используется: $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$

$$\tilde{C}_t = \tanh\left(Wx + W_hH_{t-1} + b\right) \tag{1.7}$$

Среди них W и b являются обучаемыми весами и элементами смещения.

Информация в нейроне памяти C_t будет включена в состояние скрытого слоя H_t выходными воротами O_t:

$$H_t = O_t \odot \tanh(C_t) \tag{1.8}$$

Среди них $\odot$ представляет собой умножение Адамара элемента, соответствующей размерности.

Короче говоря, из формулы (6) можно получить, что ворота забывания F_t могут контролировать, необходимо определить, следует ли включать информацию из памяти нейрона на предыдущем временном шаге в память нейрона на текущем временном шаге. Входные ворота I_t регулируют, следует ли включать текущую входную информацию в память нейрона. Например, когда $F_t = 1$ и $I_t = 0$, то $C_t = C_{t-1}$, это означает, что информация с предыдущих временных шагов передается последовательно. Это позволяет LSTM успешно улавливать долгосрочные зависимости в последовательности и избегать проблемы исчезновения градиента, характерной для обычных рекуррентных нейронных сетей. Однако архитектура LSTM сети является достаточно сложной, что приводит к большому количеству параметров и может снижать вычислительную эффективность модели. GRU (Gate Recurrent Unit) упрощает схему трех "ворот" в LSTM до двух "ворот": ворота сброса $R_t \in \mathbb{R}^{n\times h}$ и ворота обновления $U_t \in \mathbb{R}^{n\times h}$, формула расчета выглядит следующим образом:

$$R_t = (W_{rn}x_t + W_{rh}H_{t-1} + b_r) \tag{1.9}$$

$$U_t = (W_{un}x_t + W_{uh}H_{t-1} + b_u) \tag{1.10}$$

Среди них W и b являются обучаемыми весами и элементами смещения.

GRU убирает концепцию отдельного нейрона памяти и вместо этого напрямую использует ворота обновления U_t для линейного комбинирования состояния скрытого слоя H_{t-1} предыдущего временного шага и текущего состояния скрытого слоя $\tilde{H}_t$:

$$H_t = U_t \odot H_{t-1} + (1 - U_t) \odot \tilde{H}_t \tag{1.11}$$

Состояние кандидата скрытого слоя $\tilde{H}_t$ контролируется сбросом затвора:

$$\tilde{H}_t = \tanh\left(W_{hn}x_t + R_t \odot W_{hn}H_{t-1} + b_h\right) \quad (1.12)$$

Среди них W и b - обучаемые весовые параметры и члены смещения, а $\odot$ представляет собой умножение элемента, соответствующего умножению.

В общем, историческая информация содержится только в состоянии скрытого слоя на предыдущем временном шаге. Когда ворота сброса $R_t = 0$, Состояние скрытого слоя на последнем временном шаге будет опущено. Таким образом, ворота сброса могут отражать краткосрочные зависимости в сети. Когда ворота обновления $U_t = 1$, $H_t = H_{t\text{-}1}$ означает, что вся историческая информация сохраняется, так что ворота обновления могут контролировать долгосрочную зависимость сети.

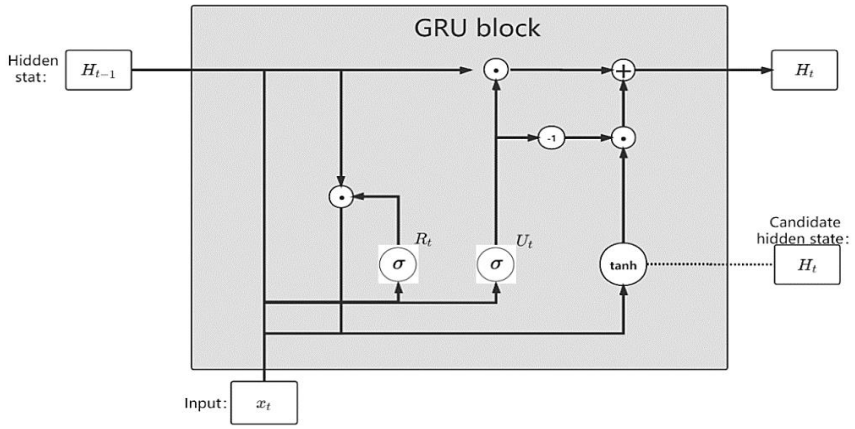


Рис. 1.6 Архитектура блока GRU

Структура базового блока GRU показана на рис. 1.6.

Из рассмотрения рекуррентных нейронных сетей (RNN) в предыдущем разделе становится ясно, что модели, такие как LSTM, GRU и другие, способны эффективно моделировать последовательности. Однако сама по себе рекуррентная нейронная сеть сталкивается с проблемами дисперсии и взрыва градиента. Модификации, такие как LSTM и GRU, лишь частично смягчают эти проблемы. Кроме того, высокая эффективность обучения рекуррентных нейронных сетей затруднена их уникальным методом

вычисления. Поскольку каждое вычисление в текущем временном шаге зависит от результата предыдущего, RNN оперирует последовательным выполнением операций, что затрудняет параллелизацию и снижает эффективность на графических процессорах. В контексте прогнозирования транспортных потоков эти недостатки ограничивают достижение желаемой производительности моделей RNN.

По сравнению с рекуррентными нейронными сетями (RNN), сверточные нейронные сети (CNN) обладают преимуществом в возможности проведения параллельных операций свертки, что способствует значительному ускорению выполнения на графических процессорах (GPU). Однако, она имеет большие недостатки при работе с временными рядами, поскольку интегрирует информацию о прошлых и будущих временных характеристиках в характеристики узла без разбора. Для решения этой проблемы Хьюэйдж и др. [74] в 2018 году предложили новую архитектуру - TCN. По сравнению с рекуррентными нейронными сетями (RNN), сверточные трансформеры (TCN) наследуют преимущества сверточных нейронных сетей (CNN) и, следовательно, обладают способностью сохранять более длинную "память" и обеспечивать более эффективный параллелизм. Это позволяет TCN эффективно использовать параллельные вычисления и значительно ускорять процесс обучения модели.

Теоретически, TCN [75] На каждом шаге свертки отбрасывается значение после определенного времени t, таким образом, выход в момент времени t в TCN зависит исключительно от входа до времени t, что обеспечивает обработку только актуальной информации и устранение неактуальных данных. Когда операция свертки не зависит от будущей информации, это характерно для TCN. Следует отметить, что TCN не является отдельной сетевой архитектурой, а скорее типом нейронной сети, разработанным на основе сверточных нейронных сетей и применяемым для решения задач, связанных с последовательными данными. Она включает в себя идею причинной свертки, свертки расширения и одномерной свертки. Среди них

причинная свертка может быть понята следующим образом: предсказание временного ряда требует, чтобы предсказание y_t в момент времени t может быть оценено только по входу x_1 - $x_{t\text{-}1}$ до момента времени t. Этот принцип напоминает идею цепей Маркова. В отличие от традиционных сверточных нейронных сетей, каузальная свертка способна анализировать только предшествующие данные и не имеет доступа к будущим данным. Поэтому проблема утечки информации решается хорошо. Одномерная каузальная свертка обычно реализуется с помощью набивка, и передний конец последовательности заполняется нулями соответствующего количества битов. Кроме того, в конце последовательности набивка не выполняется.

Расширяющая свертка заключается в добавлении весов с нулевым значением к ядру свертки для расширения рецептивного поля при том же объеме вычислений. По сравнению с традиционной сверточной нейронной сетью, хотя фактический размер ядра свертки остается прежним, размер рецептивного поля увеличивается. Поэтому преимущество расширенной свертки в том, что она увеличивает поле восприятия без потери информации за счет объединения. Это позволяет каждому выходу свертки содержать больший диапазон информации.

Таким образом, благодаря TCN, сеть может захватить долгосрочную память транспортного потока, а также гарантирует, что только данные прошлого момента будут использованы при прогнозировании будущего транспортного потока. при прогнозировании будущего транспортного потока. В этом отношении она позволяет избежать проблемы утечки информации, характерной для традиционной CNN.

В сценарии прогнозирования транспортных потоков RNN и его разновидности (LSTM или GRU) могут хорошо справиться с задачей обработки последовательных данных, поэтому они часто используются для улавливания корреляции данных о транспортных потоках во временном измерении.

Rico и др. [76] предложили новую систему прогнозирования трафика на основе множественных остаточных рекуррентных графовых нейронных сетей. В этой системе остаточные нейронные сети и GNN используются для решения проблемы взрыва и исчезновения градиента, а также для извлечения пространственных особенностей. На этой основе модель использует GRU для извлечения временных динамических особенностей высоко размерных данных. Благодаря совместному применению нескольких нейронных сетей решается задача прогнозирования транспортного потока. В исследовании Мохд Нур, et al. [77] в качестве нейронной единицы для извлечения временных характеристик использовалась LSTM. Характеристики данных на входе в LSTM извлекаются с помощью свертки графа трафика (TGC) в исходном наборе данных о транспортном потоке. Го и др. [78] предложили модель оптимизированной рекуррентной нейронной сети с преобразованием графа (OGCRNN), которая отражает временные и пространственные характеристики транспортного потока с помощью комбинации модулей GCN и GRU. Отличие от предыдущего исследования заключается в том, что в данном исследовании данные о транспортном потоке периодически оптимизируются на этапе ввода. Модель Seq2Seq + Spatial Relation, предложенная Ши и др. [79], по-прежнему использует комбинацию GCN+LSTM. Ее особенностью является то, что в соответствии с характеристиками реальных узлов дорожной сети данные в наборе данных фильтруются и отсеиваются.

Графовая нейронная сеть (GNN) представляет собой разновидность нейронной сети, специализированной на обработке графовых структур. Одним из типичных применений GNN является классификация узлов в графе. Ввиду того, поскольку GNN была впервые представлена в указанной работе, ее часто рассматривают как базис и отправную точку для дальнейших исследований в данной области. В задаче классификации узлов признак vx каждого узла v ассоциируется с истинной меткой tv. Учитывая частично маркированный граф G, цель данной задачи заключается в использовании маркированных узлов для прогнозирования меток немаркированных узлов. Алгоритм анализирует

каждый узел сети с целью достижения этой задачи, представленный d-мерным вектором hv, содержащим информацию о соседстве:

$$hv = f(xv, xco[v], hne[v], xne[v]) \quad (1.13)$$

Где $xco[v]$ представляет особенность ребра, соединенного с v. $hne[v]$ представляет вложение смежных узлов v. $xne[v]$ представляет характеристики смежных узлов v. Функция f является функцией перехода, которая отображает эти входы в d-мерное пространство. Выход GNN вычисляется путем передачи состояния hv и характеристики xv в выходную функцию g.

$$O_v = g\ (h_v, X_v) \quad (1.14)$$

Тем не менее, у оригинальной GNN существуют три ключевых ограничения:

(1) Ослабление предположения о "неподвижной точке" позволяет обучить более стабильное представление с использованием многослойного перцептрона, устраняя необходимость итеративного обновления. Это связано с тем, что в оригинальной версии GNN в различных итерациях применяются одни и те же параметры передаточной функции f, а использование различных параметров в различных слоях многослойного перцептрона (MLP) позволяет иерархически извлекать признаки.

(2) Не способна обрабатывать информацию о ребрах. Например, различные ребра в графе знаний могут отражать различные отношения между узлами.

(3) Фиксированные точки ограничивают разнообразие распределения узлов и не способствуют эффективному обучению их представлений.

Для решения вышеуказанных проблем была разработана графовая сверточная нейронная сеть (GCN).

Графовая сверточная нейронная сеть разработана для применения к графовым данным и прямого использования информации о структуре графа. GCN представляет собой архитектуру нейронной сети, способную эффективно извлекать признаки из графовых данных. К примеру, она может применяться

к социальным сетям, молекулярным структурам белков, коммуникационным сетям и другим подобным данным [80]. Аналогично конволюционной нейронной сети для извлечения признаков графов также можно использовать структуру многослойной нейронной сети. Для каждого слоя можно использовать следующую функцию отображения для вычисления:

$$H^{(l+1)} = f(H^{(l)}, A) \tag{1.15}$$

Среди них $H^{(l)} \in \mathbb{R}^{N \times d(l)}$ представляет собой выражение узлового уровня графа l -го слоя. Это $N \times d^{(l)}$-мерная матрица. N представляет собой количество узлов, а $d^{(l)}$ представляет размерность, выраженную узлами l-го слоя (размерность может быть разной в каждом слое, что определяется f и может быть гибко задано). $H^{(0)} = X$ представляет собой начальную матрицу выражения узлов 0-го слоя. Предполагая, что всего существует L-слойная сеть, $H^{(L)} = Z$ представляет матрицу выражений узлов, выводимую последним слоем.

Подобно CNN, свертка графов также использует общие веса. Однако, в отличие от CNN, вес каждого ядра представляет собой регулярную матрицу, которая назначается в соответствии с соответствующей позицией. Вес в свертке графа обычно представляет собой множество. При вычислении суммарного значения признака для узла все точки, участвующие в свертке, распределяются по нескольким различным подмножествам в соответствии с определенным правилом. Узлы в одном подмножестве принимают одинаковый вес, чтобы реализовать распределение веса. То есть, операция свертки графа взвешивает особенности каждого узла и особенности соседних узлов, а затем передается на следующий слой. Этот вид называется сверткой графа в пространственной области.

Существует значительная взаимосвязь между условиями дорожного движения и структурой дорожной сети, что привело исследователей к сформулированию модели прогнозирования транспортного потока как задачи графового моделирования. Учитывая пространственные особенности

дорожных сетей, GCN оказывается более подходящим для моделирования данных в неевклидовом пространстве по сравнению с CNN. Он также эффективнее извлекает пространственные характеристики из структуры дорожной сети. Путем применения операции свертки на пространственном графе или спектральной свертки GCN способен интегрировать информацию о соседних узлах каждого узла дорожной сети, а затем объединить ее с собственной информацией об узлах, обновляя ее до более высокоразмерного представления характеристик узлов. Обновленные данные о характеристиках содержат более богатые характеристики узлов, что позволяет нейронной сети улавливать более глубокие правила характеристик. Информация о признаках, Информация, полученная при помощи GCN на различных временных отрезках, агрегируется во временном измерении, после чего происходит дальнейшее извлечение пространственно-временной корреляции с применением нейронной сети. Сео и др. [81] предложили модель нейронной сети, называемую графовой конволюционной рекуррентной нейронной сетью (GCRN), совмещающую в себе рекуррентную сеть и операции свертки графа. Впоследствии Ли и др. [82] предложили модель DCRNN (диффузная конволюционная рекуррентная нейронная сеть), которая успешно использовала GRU и свертку графа для долгосрочного прогнозирования трафика. Ю и др. [83] Предложена модель GCN с механизмом стробирования, которая была применена к задаче прогнозирования интенсивности движения. Тизиракис и др. [84] используя метод спектрограммы GCN, авторы извлекли пространственные характеристики из исходных данных. После этого они передали их в модель CNN для дальнейшего извлечения временных характеристик. Путем суммирования пространственно-временных модулей получается более обширная информация о пространственно-временных признаках.

В области прогнозирования транспортных потоков использование графовых нейронных сетей требует учета временного измерения или временного ряда. Это означает необходимость полного внимания к сложной

пространственно-временной корреляции в сети дорожного движения. Кроме того, в процессе моделирования важно учитывать взаимосвязь между различными комбинациями моделей. Особое внимание необходимо уделить тому, чтобы избежать проблемы распространения ошибок между этапами, возникающей при пошаговом формировании результатов прогнозирования.

1.5. Выводы по результатам анализа

1. Эффективная организация движения транспортных средств в карьерах существенно сказывается на интегральных показателях работы горных предприятий, в связи с влиянием этого процесса на производительность, безопасность и общий ход работ.
2. Существующие методики оптимальной маршрутизации автосамосвалов в карьерах, способствующие рациональному распределению автосамосвалов по пунктам погрузки и разгрузки, не учитывают множество факторов, относящихся к режимам и маршрутам конкретных технологических дорог, а также специфику автономных транспортных систем.
3. На сегодняшний день существует значительное число моделей и технологий параллельных вычислений, которые позволяют существенно повысить показатели производительности известных алгоритмов поиска оптимума в пространстве состояний, таких, например, как алгоритм Дейкстры.
4. Проведенный анализ различных моделей алгоритмов, используемых для прогнозирования характеристик транспортных потоков, опирающихся на различные вычислительные формализмы, и позволил сделать вывод, что с учетом специфики задач, решаемых в данной работе, наиболее адекватным подходом представляется нейросетевой на базе граф-конволюционной архитектуры, позволяющей подчеркнуть связь между пространственными и временными структурами.

2. Разработка метода планирования маршрутов движения автономного карьерного транспорта

2.1. Концепции оптимальной маршрутизации карьерного транспорта

Традиционно, задача оптимизации транспортно-технологических процессов в карьерах связана с определением таких маршрутов перемещения автосамосвалов между ключевыми объектами инфраструктуры (зонами экскаваторной погрузки, и зонами разгрузки), которые обеспечивали бы максимальную загрузку экскаваторов и минимальное время простоя автосамосвалов. Эта статическая по своей сути задача, традиционно решаемая средствами линейного программирования и алгоритмов поиска оптимальных путей в направленных графах, сегодня в условиях использования технологий Индустрии 4.0 и роботизированных объектов ГТК должна быть существенно трансформирована. Учитывая сложность учета множества противоречивых критериев данную задачу следует рассматривать, как многоуровневую, включающую как статическую, так и динамическую части [85,86]. С операционной точки зрения можно выделить следующие основные этапы решения данной задачи:

1. Планирование маршрутов движения автосамосвалов между экскаваторами и зонами разгрузки в течение определенного временного периода (например, рабочей смены) направлено на максимизацию объемов перевозимой горной массы. В этом случае экскаваторы работают с максимальной производительностью, а самосвалы используют маршруты с минимальными затратами.
2. Определение такой траектории движения автосамосвала на маршруте, а также соответствующих ей скоростных режимов на различных ее участках, которые обеспечивали бы оптимальность некоторого комплексного критерия, учитывающего безопасность движения, расход топлива и износ основных узлов самосвала.

3. Управление движением самосвала по заданной траектории с определенной скоростью, которое обеспечивается работой навигационных систем и средств бортовой автоматики.

Хотя при транспортировке грузов автономными самосвалами возникают аналогичные проблемы, как и в классической транспортной задаче, такие как выбор конечных пунктов транспортировки и соответствующих маршрутов для каждого робота, а также определение режимов движения, использование традиционных подходов к решению транспортной задачи в системе управления автономным транспортом не приводит к эффективным результатам [87,88]. Это обусловлено рядом факторов:

1. В процессе эксплуатации системы периодически и неожиданно изменяется технологическая среда, в которой функционируют автоматизированные устройства. В частности, состояние дорожного покрытия на различных участках карьерной сети ухудшается. Кроме того, из-за сбоев в режимах работы и отказов транспортного оборудования, а также вследствие неравномерных процессов экскавации, в течение смены может изменяться схема маршрутизации.
2. В связи с особенностями технологической среды режимы перемещения роботов необходимо периодически корректировать, принимая во внимание фактическое взаимное расположение автономных агентов на карьерных трассах в конкретный момент времени. Это должно осуществляться с помощью соответствующих алгоритмов, так как при отсутствии водителей и изменении функций диспетчера другие инструменты для согласования совместной работы роботов-самосвалов отсутствуют.
3. В завершение, весь процесс функционирования транспортной системы может быть координирован частично автономной системой управления, что требует непрерывного обмена информацией между автосамосвалами и координатором.

Транспортная сеть будет представлена в виде связного графа $G\ (m_S, R)$, где $m_S = m_1 + m_2 + m_3$ (m_1 и m_2 - терминальные узлы, m_3 - внутренние узлы),

$R \subseteq m_s \times m_s$. Установленные последовательности ветвлений $r_1, r_2, \ldots, r_p$, Ранжированные таким образом, что первые и последние ветви этой последовательности инцидентны узлам $S_e \in m_1$, $S_f \in m_2$ – образуют маршруты $L_1, L_2, \ldots, L_q$ движения автономных транспортных средств, на рис. 2.1 представлен пример графа:

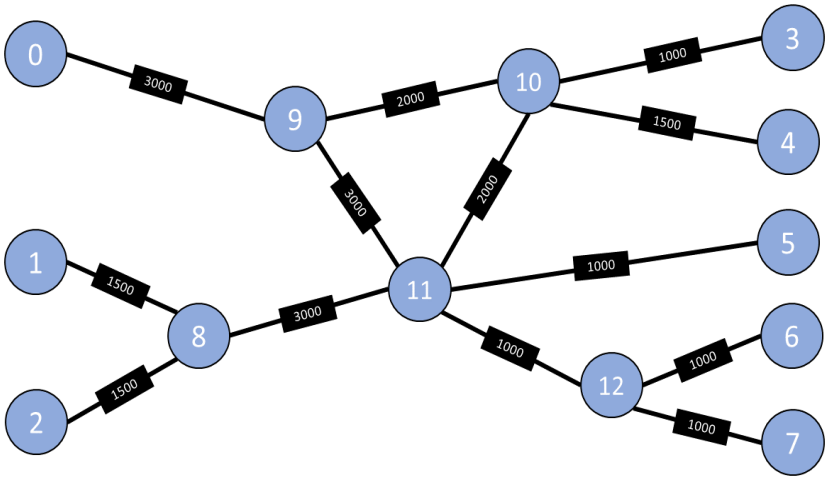


Рис. 2.1 Граф, соответствующий реальной топологии дорог в карьере.

Общее описание проблемы маршрутизации можно описать следующим образом.

1. На начальном этапе работы определяются качественные и количественные характеристики всех ветвей дороги, включая их геометрию и назначение в общей сети. В данной работе предполагается, что известно фактическое состояние дорожного полотна на всех участках транспортной сети. Каждый автономный самосвал оборудован комплексом средств контроля, такими как инклинометры, акселерометры и вибродатчики, что обеспечивает достаточный объем данных для оперативного отслеживания его местоположения и идентификации возможных нарушений дорожного полотна с привязкой к географическим координатам и конкретным фрагментам цифровой модели. Это позволяет формировать оценки качества каждого участка технологических дорог в транспортной сети карьера [85].

2. Применяемые навигационные системы обеспечивают достаточно точное определение координат $\{\widetilde{x_i}(t_k), \widetilde{y_i}(t_k), \widetilde{z_i}(t_k)\}$, i-го транспортного средства в момент t_k, $k = [0, T]$. Вопросы улучшения точности позиционирования и привязки реальной координаты к конкретной точке дорожного сегмента транспортной сети уже были подробно рассмотрены ранее и не являются предметом обсуждения в данной работе. Следовательно, предполагается наличие работоспособного механизма трансформации и проецирования координат. В дальнейшем, для упрощения записи, координаты мобильного момент времени a_i в момент t_k в пределах транспортной сети будут обозначаться как $\widetilde{x_i}[k]$.
3. В любой момент времени для каждого самосвала известны пункты назначения e_{s_1} и u_{s_2}, $S_1 = 1, m_1$, $S_2 = 1, m_2$. Здесь m_1 – представляет собой общее количество погрузочных пунктов (экскаваторов), m_2 - количество пунктов разгрузки. На начальном этапе выполняется процедура формального назначения конечных пунктов каждому агенту, основанная на общей схеме оптимизации грузоперевозок.
4. В каждый момент времени k, автономный агент a_i перемещается по определенному маршруту; заданному последовательностью вершин и ребер. $L_i[e_{S_1}, u_{S_2}]$, $S_1 = \overline{1, m_1}$, $S_2 = \overline{1, m_2}$, а весь маршрут по схеме $\{u_{S_1} - e_{S_1} - u_{S_2}\}$ образует полный цикл для i-го агента.

В ходе очередного цикла фиксируются следующие данные:

- Время, необходимое агенту для выполнения данного маршрута - $\tilde{T}_i(L_i)$;
- Количество остановок или значительных изменений скорости в течение цикла - n_i;
- средняя скорость $\bar{v}_i$ в течение цикла.

Поскольку в процессе реализации маршрутов автосамосвал может осуществлять движения по замкнутому кругу вокруг экскаваторных площадок или зон разгрузки, будет более корректным определить этот ациклический граф, как псевдограф или схему движения по карьерным дорогам (рис. 2.2).

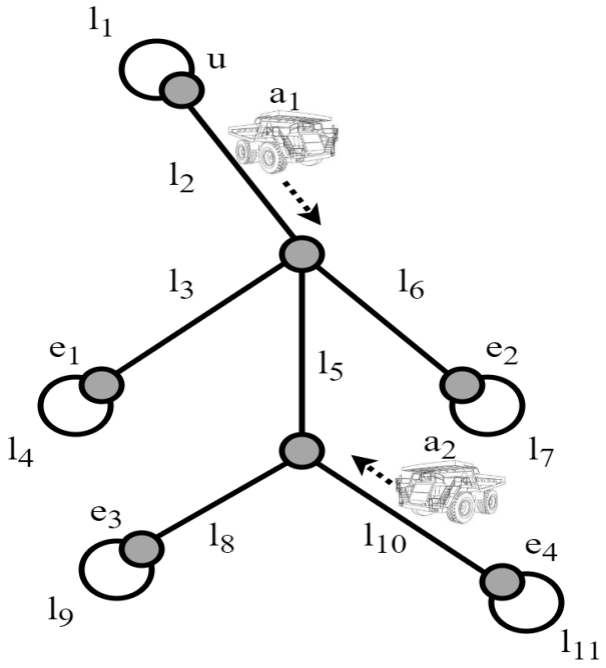


Рис. 2.2 Схема, иллюстрирующая транспортные процессы в карьере.

При выборе граничных (целевых) вершин маршрута для каждого автономного самосвала обычно применяется критерий, основанный на минимальном оценочном времени прохождения маршрута для каждого последующего агента, завершившего цикл:

$$F = \min\{\widehat{T}_i(L_j)\}, \tag{2.1}$$

где $i = 1, N,\ j = \overline{1, m_3},\ m_S = m_1 \times m_2,\ L_j$ – длина j-ого маршрута (то есть количество ребер в графах на рис. 2.1. и 2.2, а.

Предлагается оценивать время, которое требуется соответствующему агенту для прохождения маршрута, используя линейную зависимость от времени, потраченного каждым последующим агентом, который завершил цикл:

$$\widehat{T}_i(L_j) = \widehat{\propto}_{1_i} x_1(j) + \widehat{\propto}_{2_i} x_2(j) + \widehat{\propto}_{3_i} x_3(j) \tag{2.2.}$$

где $\widehat{T}_i(L_j)$ – оценка времени, затрачиваемого i-м агентом на прохождение маршрута $\widehat{T}_i(L_j)$;

x_1 – как длину соответствующего маршрута;

x_2 – как количество ключевых точек на маршруте;

x_3 – как обратную величину средней скорости на маршруте.

Коэффициенты $\widehat{\propto}_{1_i}, \widehat{\propto}_{2_i}, \widehat{\propto}_{3_i}$ в соответствующих уравнениях определяются на основе данных, полученных из наблюдений за движением автосамосвалов в предыдущих сменах. Эти наблюдения включают в себя трассы перемещения как автономных, так и управляемых водителями самосвалов.

То есть, иными словами, в алгоритме оптимальной маршрутизации каждому ребру – l_i ставится в соответствие определенный вес, который определяется длиной соответствующей дороги - участка маршрута и, возможно, комплексной оценкой сложности преодоления этой дороги.

Однако, два существенных вопроса, связанных с движением автономных самосвалов, остаются вне зоны рассмотрения. Сегодня общепризнано, что реальная эффективность использования карьерного транспорта (и не только автономного) напрямую зависит от ряда микро-факторов, таких как специфические особенности конкретной трассы перемещения самосвала, а точнее траектории перемещения автосамосвалов по дорогам, а также от скоростных режимов прохождения этих трасс. При этом возникает непростая задача учета этих микро-факторов при планировании маршрутов самосвалов. Для решения этой задачи и интегрирования этой задачи в общую задачу оптимального оперативного планирования перемещения роботов предлагается использовать подход, основанный на цифровой модели карьерных дорог.

2.2. Формирование функции стоимости перемещения робота-самосвала

Планирование является одной из ключевых задач в робототехнике. Оно заключается в нахождении последовательности осуществимых состояний между фиксированными позициями (старт, финиш) в пространстве поиска, обеспечивающей оптимальное значение некоторой функции стоимости. В нашем случае это означает, что необходимо определить последовательность опорных точек (привязанных к элементарным фрагментам дорожного

полотна), образующих конкретную трассу, по которой на основании соответствующих моделей динамики и кинематических уравнений может следовать робот, с учетом возможных препятствий на своем пути. Поиск оптимального пути непростая задача, поскольку необходимо учитывать огромное число состояний, что приводит к вычислительной сложности. В последнее время большую популярность приобрели алгоритмы, которые работают в рамках парадигмы непрерывного пространства перемещения робота. Эти алгоритмы весьма эффективны при перемещении робота в высоконагруженной динамической среде, где возможно ежесекундное изменение направления движения [89]. В предмете исследования задачи планирования движения автономных грузовых самосвалов отсутствует необходимость в постоянной корректировке их траекторий. Тем не менее, в случае возникновения такой потребности, требуется быстрое получение решения, включая ситуации, когда необходимо рассмотреть несколько автономных самосвалов одновременно. что в условиях функционирования систем управления также может быть ограничено доступностью В основе концепции, на которую опирается разрабатываемый нами метод, лежит устоявшаяся методология планирования горных работ, заключающаяся в использовании блочных 3D моделей, позволяющих описывать взаимодействие добычного оборудования с геологической структурой месторождения. Данная методология была ранее адаптирована для описания поверхностной инфраструктуры карьера и легла в основу моделирования транспортно-технологических процессов [85]. Основными элементами инфраструктуры на поверхностном уровне карьера являются карьерные дороги и различные технологические зоны. Цифровая модель этой поверхности состоит из набора правильных многоугольников, таких как квадраты или шестиугольники, с изменяемой длиной сторон, которые привязаны к координатной сетке. Каждому из этих многоугольников сопоставляется набор физико-технических и пространственно-технологических характеристик [90].

Таким образом, цифровая модель представляет собой формальную систему, содержащую следующие основные компоненты:

- Граф дорожного движения, охватывающий основные зоны карьера;
- Семейство атомарных элементов, представляющих физические характеристики дорожного покрытия, качество трассы, продольные и поперечные уклоны и другие аспекты;
- Структурная матрица атомарных элементов.
- Система мониторинга и оперативного обновления текущих условий движения (включая ремонтные работы, позиционирование самосвалов и прочее).
- Базовая версия цифровой модели состояла из трех типов атомарных элементов: правильных треугольников, квадратов и правильных шестиугольников (см. рис.2.3)

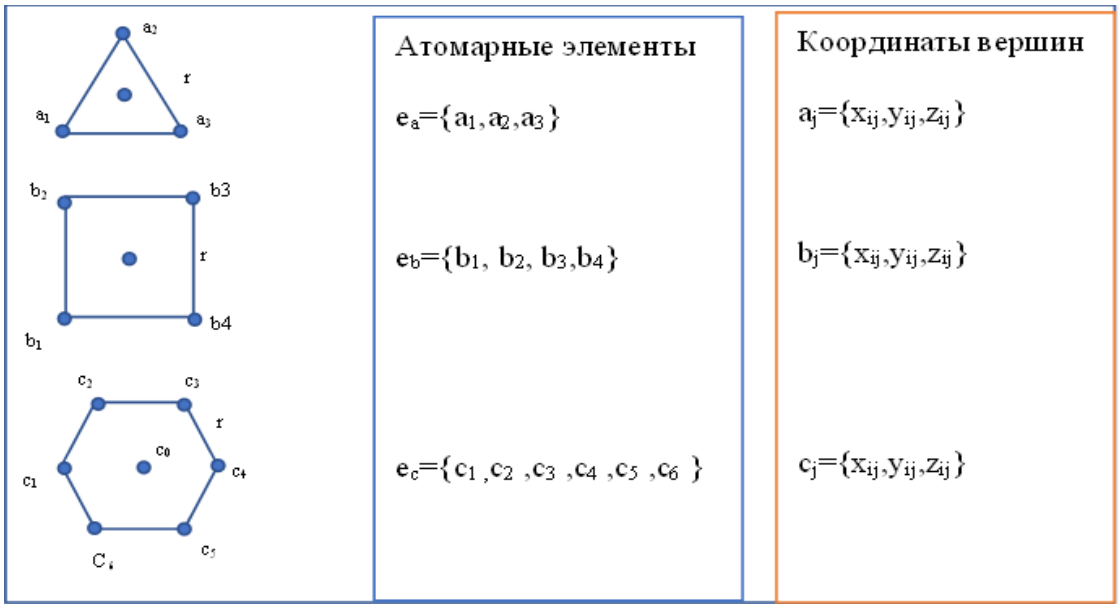


Рис. 2.3 Атомарные элементы, с учетом их геометрических особенностей

Известно, что формирование подобной модели осуществляется на основе интеграции маркшейдерских данных, которые могут быть получены в том числе и с помощью БПЛА, результатов аэро-космического мониторинга и данных бортовой телеметрии большегрузных самосвалов, модель представляет собой совокупность элементарных фрагментов, известных как

атомарные элементы, имеющие геометрическую форму правильных многоугольников и определяемые их координатами $\{x_{ij}, y_{ij}, z_{ij}\}, i = 0,1, \dots, m; j = 0,1, \dots, n$, где m и n – размерность структурной матрицы фрагментов; x_{ij}, y_{ij}, z_{ij} – координаты центров этих атомарных элементов в плоскостях x, y и z, которые в дальнейшем используются в качестве опорных точек траектории автономного самосвала. В данной работе, исходя из решаемых задач, в качестве атомарных элементов, использовались либо прямоугольные элементы размером (5x10) метров, что примерно соответствует габаритным размерам типового большегрузного самосвала грузоподъемностью 200 тонн, либо квадратные примитивы размером (1м x 1м).. В последнем случае повышается время вычислений и слегка усложняется процесс вычисления опорных точек, однако и точность расчетов может быть увеличена (рис.2.4)

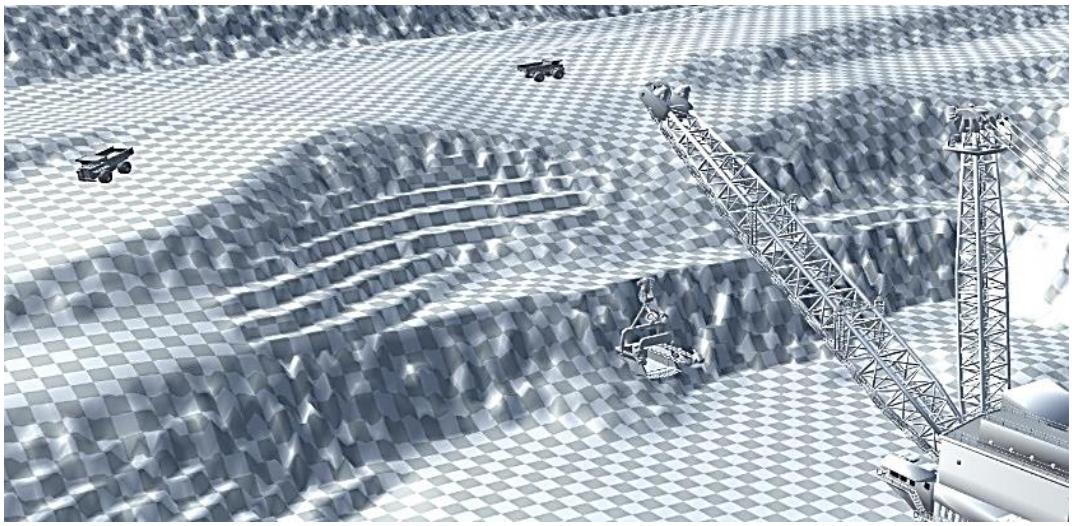
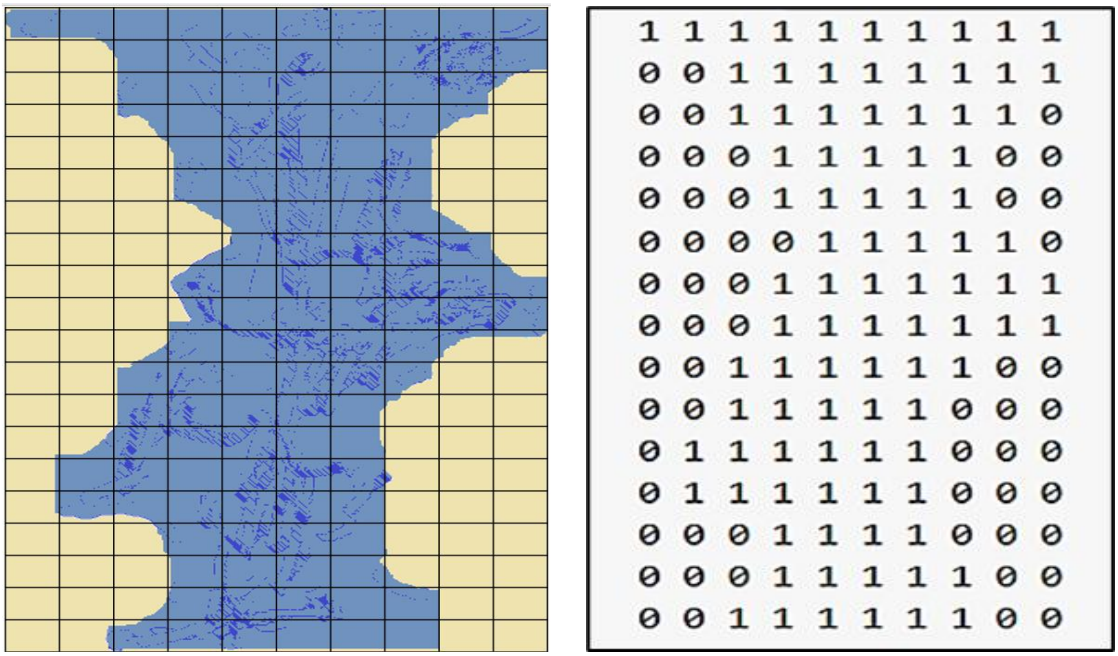


Рис. 2.4 Иллюстрация термина «цифровая модель» транспортно-технологической среды карьера.

В качестве основного компонента цифровой модели выступает структурная матрица, по которой определяются смежные фрагменты между элементами, в которой ненулевые элементы означают, что движение в этой части карьерной дороги возможно. (рис. 2.5).



а б

Рис. 2.5. (а) Цифровое представление фрагмента дороги; (б) соответствующая матрица смежности.

Кроме того, каждому фрагменту ставится в соответствие набор параметров, в котором наряду с пространственными используются также некоторые технологические характеристики. Эти параметры используются для вычисления количественной характеристики – рейтинга – C_{ij} каждого фрагмента и играют ключевую роль в процессе вычисления стоимостных функций, которые соответствуют весам графа, ребра которого связывают потенциальные опорные точки траектории автономного самосвала.

$$C_{ij} = a_1 * d_1 + a_2\, d_2 + \cdots + a_k * d_k, \qquad (2.3)$$

где $d_1, d_2, \ldots, d_k$ – совокупность коэффициентов, описывающих степень пригодности соответствующего фрагмента дорожного полотна для движения робота-самосвала, которые определяются экспертно-статистическими методами на этапе построения модели. К таким факторам, например, относятся следующие:

- погодные условия (дождь, снег, отсутствие осадков, гололед);

- состояние дорожного покрытия, на наличие деформаций (выбоины, колеи, гребни);
- расположение фрагмента относительно границы проезжей части.

$a_1, a_2, \dots, a_k$ – весовые коэффициенты, которые в общем случае могут быть равны между собой.

Значения этих рейтинговых оценок фрагментов дорожного полотна приводятся к интервалу [0,1 - 1,0]. При этом считается, что значение 0,1 – означает, что данный фрагмент наиболее подходит для перемещения робота самосвала, а 1,0 – наименее подходящий участок для выбора трассы.

В процессе функционирования данная модель периодически корректируется, а при появлении на дороге динамических препятствий, которые делают определенный фрагмент (фрагменты) непригодным для перемещения самосвала, соответствующий элемент структурной матрицы обнуляется.

Соседними фрагментами, которые можно рассматривать как потенциальные участки дороги для перемещения автономного самосвала в соответствии со схемой, представленной на рис.2.6:

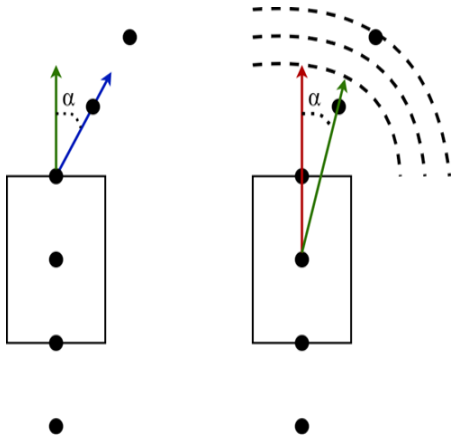


Рис.2.6 Схемы перемещения автономного самосвала через опорные точки

будем считать следующие: (Cij- > Ci,j+1), (Cij -> Ci+1,j+1), (Cij -> Ci-1,j+1).

В случае, если потенциальная трасса прокладывается в непосредственной близости от границ дорожного полотна, а также при изгибах дороги или наличие препятствий, часть смежных фрагментов исключается из

рассмотрения (элементам соответствующих столбцов и строк присваивается в зависимости от вычислительной процедуры, либо нулевое значение, либо значение заведомо большее нежели сумма всех элементов.

Учитывая, что существует только три варианта перемещения автономного самосвала: «вперед-с поворотом направо», «вперед – с поворотом налево», вперед – прямо», определим эмпирические функции стоимости следующим образом:

$$\varphi_{ij}^{l} = 1.4\left\{\left|c_{ij} - c_{i+1,j-1}\right| + \left|\frac{\tilde{v}_{ij} - \tilde{v}_{i+1,j-1}}{v_{max}}\right|\right\}$$
$$\varphi_{ij}^{r} = 1.4\left\{\left|c_{ij} - c_{i+1,j+1}\right| + \left|\frac{\tilde{v}_{ij} - \tilde{v}_{i+1,j+1}}{v_{max}}\right|\right\} \qquad (2.4)$$
$$\varphi_{ij}^{s} = \left\{\left|c_{ij} - c_{i+1,j}\right| + \left|\frac{\tilde{v}_{ij} - v_{i+1,j}}{v_{max}}\right|\right\}$$

$\varphi_{\text{i}}(l)$, $\varphi_{\text{i}}(s)$, $\varphi_{\text{i}}(r)$ где l, s, r – индексы функции стоимости перемещения соответственно: «влево», «прямо», «вправо», а $i = \overline{1, N-1}$.

Значение этих функций вычисляется на основе рейтинговых оценок фрагментов, которые отражают степень пригодности участков дороги для перемещения автосамосвала и, следовательно, они могут использоваться в качестве критерия выбора направления перемещения. Однако в представленном виде (2.4) эти функции учитывают и другой фактор, от которого зависит эффективность прохождения маршрутов автосамосвалом, а именно скорость на конкретных участках в зависимости от выбранной трассы (траектории) движения. Из реальной практики работы автотранспорта карьера приблизительно известно на каких скоростных режимах преодолевает управляемый водителем самосвал те или иные фрагменты трассы, и как это сказывается на итоговом расходе топлива. Для автономного транспорта можно для начала предположить, что и с позиций безопасности, и с позиций экономии топлива, наилучшим вариантом является равномерный характер движения. Безусловно, учитывая тот факт, что реальные трассы карьера имеют сложный профиль с большим количеством подъемов и спусков, которые могут непосредственно чередоваться, следует признать, что реальная целевая

эффективность должна вычисляться на основе более сложных формул, учитывающих и динамику движения, и непосредственную траекторию, по которой перемещается самосвал. Однако, это уже проблема, скорее автоматического управления и детального анализа целевой эффективности робота-самосвала, которая выходит за рамки настоящей работы.

Поэтому в качестве вполне рациональной гипотезы можно предположить, что наилучшим вариантом для перемещения является, такой в котором разность фиксируемых скоростей в опорных точках минимальна, так как при этом потенциально расходуются минимальные объемы топлива. Кроме того, разумным допущением выглядит и такое: «равномерное движение без торможений и ускорений обеспечивает минимальные ударные нагрузки на все механические узлы автосамосвала и, следовательно, способствует продлению срока службы этих узлов, снижая риски их аварийных выходов из строя.

В результате суммарная функция стоимости для каждого самосвала может быть записана в виде:

$$\varphi_{\Sigma} = \sum_{i=1}^{N-1} \varphi_{ij}, \text{ где } j = \{l, r, s\} \qquad (2.5)$$

Эти функции стоимости: $\varphi_{ij}(L), \varphi_{ij}(R), \varphi_{ij}(S)$, где $i = 0, N;\ j = 1, k_i, ; k_i \leq m$ выступают в качестве весов ребер графа, на основании которых формируется трасса, которая представляет собой последовательность ребер графа между терминальными вершинам x_0 и x_f, где каждое ребро графа встречается только один раз.

Введя сквозную нумерацию фрагментов:

$c_{11} \dots c_{1m}, c_{21} \dots c_{2m}, \dots c_{N1}, \dots c_{Nm}$, определим матрицу смежности W$[P, P]$, где $p = N \times m$ с элементами:

$$a_{ij} = \begin{cases} \varphi_{i,j}, & \text{если } \exists \text{ возможность перемещения между фрагментами i и j} \\ 0, & \text{если такой возможности нет} \end{cases}$$

$i, j = \overline{1, p}.$

Полученная в результате матрица будет разреженной, число ненулевых элементов которой составляет не более 1%.

Однако, именно эта матрица используется для поиска минимальной функции стоимости: $F = \varphi_{\Sigma} \to min$, а критерий оптимальности может быть сформулирован следующим образом:

$$F = \sum_{i=1}^{N-1} min\{\varphi_{i,j-1}, \varphi_{i,j}, \varphi_{i,j+1}\} \quad (2.6)$$
$$j \in \overline{1,m}$$

2.3. Метод оперативного планирования маршрутов с учетом выбора конкретной трассы перемещения автосамосвала

2.3.1. Основные этапы планирования согласованного перемещения автономных самосвалов

Как уже отмечалось ранее реализация оптимальной согласованной работы группы роботов – самосвалов является чрезвычайно сложной задачей. Мы предлагаем подходить к ее решению поэтапно, сосредоточившись на выборе оптимальных маршрутов с учетом планирования конкретных трасс для прохождения отдельных ребер (технологических дорог) транспортной сети карьера. Далее представлены основные этапы оптимальной маршрутизации автономного карьерного транспорта [91– 94].

1. Оптимизация маршрутов

На первом этапе осуществляется выбор маршрутов для перемещения автосамосвалов, которые обеспечивают максимальную загрузку экскаваторов и минимизируют время простоя транспортных средств. В процессе оптимизации маршрутов используются методы линейного программирования и алгоритмы поиска оптимальных путей в направленных графах. Эти методы позволяют решить статические задачи, такие как определение наилучшего маршрута между двумя точками с учетом заданных ограничений и критериев.

2. Динамическая адаптация

На втором этапе происходит корректировка режимов движения автосамосвалов в реальном времени, учитывая их текущее взаимное

расположение на дорожных трассах карьера и изменения в окружающей среде. Адаптация осуществляется на основе факторов, таких как деградация дорожного покрытия, изменения в работе транспортного оборудования и динамика процессов экскавации. В этом контексте используются системы частично-автономного управления, способные реагировать на изменчивость окружающей среды и обеспечивать оптимальные условия перемещения.

3. Координация и обмен информацией

На третьем этапе осуществляется координация работы транспортной системы с использованием сетевых технологий и систем обмена информацией. Постоянный обмен данными между автосамосвалами и центральной системой управления позволяет синхронизировать их работу и оперативно адаптировать маршруты к текущим условиям. Это включает мониторинг положения транспортных средств, состояния дорог и других факторов, влияющих на движение.

2.3.2. Вычисление интегральной стоимости маршрута

Для определения оптимального маршрута используется функция стоимости перемещения, которая учитывает различные параметры, такие как состояние дороги, погодные условия, скоростные режимы и прочие факторы. Стоимость перемещения определяется на основе рейтинговой оценки фрагментов дороги, которые интегрируются в общую функцию.

Комплексную функцию стоимости трассы определим следующим образом:

$$F_k = \sum_{i=1}^{P_k-1} \sum_{j=i}^{P_k} \varphi_{ij}[w_k] \rightarrow min, \quad (2.7)$$

где k – индекс ребра маршрутного графа, для которого сформирована матрица смежности W_k.

Если мы рассматриваем конкретный маршрут, как последовательность рёбер, например:

$l_2 - l_5 - l_{10}$ (рис. [39, 40] 2.2), то суммарная стоимость трассы может быть записана следующим образом:

$$F_{\Sigma} = \sum_{k=1}^{N_b} \left\{ \sum_{i=1}^{P_k - 1} \sum_{j=i}^{P_k} \varphi_{ij}[w_k] \right\} \rightarrow min, \quad (2.8)$$

где N_b – количество ребер, входящих в маршрут b.

Если теперь пронумеровать все маршруты в исходном графе $L_1, L_2, \dots L_S$, каждый из которых включает набор рёбер $l_1, l_2, \dots l_R$, то элементами маршрутной матрицы $L[S, R]$ будут

$$l_{b,r} = \begin{cases} 1, \text{если ребро } r \text{ входит в маршрут } L_b \\ 0, \text{в противном случае} \end{cases}$$

$$b = \overline{1, S}$$

Учитывая, что каждому автономному самосвалу на предварительном этапе назначается свой маршрут, вычислим маршруты минимальной стоимости (трассы минимальной стоимости), имея в виду, что матрицы смежности фракталов (атомарных элементов) для соответствующих рёбер корректируются после каждого вычисления функции стоимости. Получим следующую формулу:

$$F_{\Sigma}^{q}(L_B) = \sum_{k=1}^{R} l_{k,b} \left\{ \sum_{i=1}^{P_k - 1} \sum_{j=i}^{P_k} \varphi_{ij}([w_k]) \right\} \rightarrow min, \quad (2.9)$$

Таким образом могут быть определены трассы минимальной стоимости для всех роботов-самосвалов, участвующих в производственном процессе: $F_{\Sigma}^{1}(L_1), F_{\Sigma}^{2}(L_2), \dots F_{\Sigma}^{Q}(L_Q)$, имея в виду что каждому самосвалу назначается определённая трасса. Соотношение между величинами Q – количество автономных самосвалов и S – количество потенциальных трасс не имеет принципиального значения.

Сама процедура выбора оптимальных трасс в условиях работы группы автономных самосвалов выглядит следующим образом:

1. Случайным образом назначаются маршруты перемещения для каждого автономного самосвала.
2. Последовательно без всякого предварительного ранжирования вычисляются функции стоимости маршрутов для каждого самосвала: q = 1,2,…….Q. При этом учитывается, что матрицы смежности ребер,

через которые проходим маршрут изменяются после каждого очередного вычисления функции стоимости.

3. Определяется маршрут минимальной стоимости и за ним закрепляется этот маршрут.
4. После этого вновь пересчитываются матрицы смежности с учетом только одного первого зафиксированного маршрута (поскольку большая часть ребер может быть не задействована в первом маршруте, расчеты по остальным маршрутам производятся с исходными матрицами смежности).
5. Повторяются этапы 2,3 и 4 до тех пор, пока всем самосвалам не назначены конкретные маршруты.

2.4. Выводы по материалам главы

В данной главе рассмотрены основные вопросы оперативного планирования маршрутов карьерного транспорта.

1. Сформулированы ключевые задачи, которые требуется решить для координации действий автономного карьерного транспорта.
2. Показано, что весь набор этих задач можно условно разделить на статические и динамические.
3. Отмечается, что в настоящее время существуют достаточно эффективные концепции и работоспособные алгоритмы планирования маршрутов карьерных самосвалов (в том числе, управляемых водителями), которые обеспечивают минимизацию времени прохождения маршрутов самосвалами, а значит, в идеальном варианте и максимизацию перевезенной горной массы. При этом могут быть даже учтены ограничения, связанные с взаимным расположением этих мобильных технологических агентов на карьерных дорогах.
4. Подчеркивается, что при таком подходе не учитывается конкретный маршрут, который проходит автономный самосвал между различными

технологическими зонами. Этот маршрут может оказывать влияние на расход топлива самосвала и степень износа его узлов.

5. С целью построения оптимальной трассы преодоления конкретного маршрута для каждого самосвала предложена оригинальная функция стоимости.
6. Сформирован критерий оптимальной автономного транспорта, позволяющий учитывать микро-факторы, влияющие на комплексную эффективность перемещения самосвала по каждому элементарному фрагменту карьерной дороги.

3. Алгоритмы оптимизации функции стоимости

Как уже отмечалось, внедрение автономного транспорта в карьерных условиях представляет собой важный шаг в направлении автоматизации горнодобывающей отрасли, что требует разработки и применения инновационных подходов к маршрутизации и управлению транспортными потоками.

В данной главе основное внимание уделяется разработке и исследованию алгоритмов оптимизации, применяемых для управления маршрутами в карьерном транспорте, а также роли параллельных вычислительных моделей в повышении их эффективности и масштабируемости. Особое внимание уделяется применению алгоритма Дейкстры для определения оптимальных маршрутов и интеграции передовых методов маршрутизации с параллельными вычислительными моделями. Значительное внимание уделяется использованию библиотеки OpenMP для реализации параллельных вычислений, что способствует ускорению выполнения алгоритмов.

Цель данной главы – описать разработанные оптимизационные модели и алгоритмы, использующие параллельные вычисления для решения проблем маршрутизации, связанных с транспортировкой карьеров. Это важнейший аспект современной горнодобывающей промышленности, способный повысить эффективность и надежность транспортных процессов.

3.1. Оптимизация алгоритмов маршрутизации карьерного транспорта с использованием технологий параллельных вычислений

Оптимизационные задачи в потоковых графах играют важную роль в различных областях, таких как транспортные сети, управление цепочками поставок и маршрутизация компьютерных сетей. Эффективное распределение ресурсов, минимизация затрат и максимизация производительности в этих сетях в значительной степени зависят от эффективного решения этих

оптимизационных задач. Однако по мере увеличения размера и сложности этих задач традиционные иерархические методы сталкиваются со значительными трудностями в своевременном получении оптимальных решений [95]. Чтобы преодолеть эти трудности, необходимо разработать параллельные вычислительные модели, специально предназначенные для решения оптимизационных задач в потоковых графах и определения оптимального маршрута для карьерных дорог. Методы параллельных вычислений позволяют значительно повысить производительность и масштабируемость оптимизационных алгоритмов. Используя возможности распределенных вычислений и параллельной обработки, эти модели могут эффективно разделить проблему на более мелкие под проблемы и распределить их между несколькими вычислительными ресурсами, что позволяет одновременно искать потенциальные решения [96]. Алгоритмы планирования маршрутов играют важную роль в определении оптимального маршрута для карьерных дорог. Эти алгоритмы, такие как алгоритм Дейкстры, поиск A* и генетические алгоритмы, обеспечивают эффективные и действенные средства навигации по сложным дорожным сетям с учетом различных особенностей моделируемых объектов [97]. Разработка параллельных вычислительных моделей для решения задач оптимизации в графах потоков включает в себя различные подходы. Одним из таких подходов является адаптация традиционных алгоритмов, таких как алгоритм Дейкстры, к динамическим и зависящим от времени условиям с использованием принципов теории транспортных потоков, что привело к улучшению результатов маршрутизации [98]. Кроме того, эвристические алгоритмы и метаэвристики используются для поиска оптимальных или близких к оптимальным решений задач оптимизации маршрутов с учетом таких факторов, как интенсивность движения, состояние дорог и габариты транспортных средств [99]. Были разработаны параллельные итерационные алгоритмы для решения задач о путях в направленных графах, целью которых является получение решений с оцененной вычислительной сложностью за

конечное число итераций. Кроме того, проблема оптимизации пути с ограничениями, целью которой является максимизация результата при ограничении затрат в рамках бюджета, решается с помощью предложенного параллельного алгоритма, который обеспечивает превосходное качество решения и малое время выполнения [100].

Данная работа преследует две цели. Во-первых, это разработка инновационных моделей параллельных вычислений, специально предназначенных для решения оптимизационных задач в потоковых графах. Эти модели будут использовать преимущества параллельных вычислений, эффективно распределяя вычислительную нагрузку и позволяя быстрее находить потенциальные решения. Во-вторых, эта работа посвящена определению оптимального маршрута для карьерных дорог путем интеграции передовых алгоритмов планирования маршрута, таких как алгоритмические, с разработанными параллельными вычислительными моделями. В данной работе в качестве модели параллельных вычислений была выбрана модель общей памяти, поскольку она подходит для имеющихся компьютерных ресурсов, в качестве программной модели для выполнения этой задачи была выбрана библиотека OpenMP, а для определения кратчайшего или оптимального маршрута был выбран алгоритм Дейкстры, поскольку он подходит для этой задачи благодаря своей эффективности и точности в определении маршрута, и таким образом исследование стремится повысить эффективность и точность планирования маршрутов при управлении карьерными дорогами.

Разработанные параллельные вычислительные модели могут значительно сократить время, необходимое для решения масштабных задач, что позволит горнодобывающим организациям принимать своевременные решения и оптимизировать распределение ресурсов. Кроме того, интеграция алгоритмов планирования маршрутов с параллельными моделями обеспечивает повышенную точность и экономическую эффективность при определении

оптимальных маршрутов для карьерных дорог, что приводит к повышению эффективности перевозок и снижению эксплуатационных расходов.

Существует несколько моделей параллельных вычислений, включая:

- Модель с общей памятью предполагает использование несколькими процессорами общего пространства памяти, что позволяет им получать доступ к данным и вносить изменения в этом пространстве. Эта модель часто применяется в многоядерных процессорах и симметричных многопроцессорных системах (SMP).
- Модель распределенной памяти: Данная модель предполагает наличие нескольких процессоров, соединенных сетью и обладающих собственным пространством памяти. Коммуникация между процессорами осуществляется посредством передачи сообщений, что позволяет обмениваться данными между процессами. Такая модель широко используется в кластерах и суперкомпьютерах [50],[101].
- Модель GPU предполагает выполнение программы на GPU, специально оптимизированном для параллельной обработки. GPU включает в себя сотни или тысячи вычислительных ядер, способных выполнять инструкции программы одновременно [102].
- Гибридная модель объединяет в себе особенности моделей с общей и распределенной памятью с целью совместного использования их преимуществ. Она активно применяется в области систем высокопроизводительных вычислений (HPC) [103].

При рассмотрении моделей параллельных вычислений важно взвесить преимущества и недостатки каждой модели, чтобы определить наиболее подходящий подход для конкретного приложения. Такие факторы, как характер задачи, доступные аппаратные ресурсы, необходимая степень параллелизма и простота реализации, играют роль при принятии решения об использовании той или иной модели. В данном исследовании для распараллеливания алгоритма Дейкстры была выбрана библиотека OpenMP благодаря ее совместимости с системами с общей памятью и возможности

эффективно распределять вычислительные задачи между несколькими потоками. Воспользовавшись простотой реализации OpenMP и улучшенной производительностью на больших графах, исследователи смогли повысить эффективность планирования маршрутов для автономных транспортных средств на горных работах [50].

Модель общей памяти — это форма параллельных вычислений, при которой несколько процессоров или ядер имеют доступ к общему пространству памяти. В этой модели все процессоры могут взаимодействовать с данными, хранящимися в общей памяти, и вносить в них изменения, что позволяет им совместно решать задачи или выполнять алгоритмы [104]. Эта модель часто используется в многоядерных процессорах, симметричных многопроцессорных системах (SMP) и различных библиотеках параллельного программирования, таких как OpenMP. В модели с общей памятью каждый процессор имеет свой собственный кэш для хранения часто используемых данных, что сокращает время доступа к памяти и повышает производительность. Однако важно избегать ситуаций гонки, когда несколько процессоров пытаются одновременно получить доступ и изменить одни и те же данные, что может привести к неправильным результатам (рис. 3.1).

Чтобы реализовать модель общей памяти, программу или алгоритм необходимо разделить на более мелкие задачи, которые могут выполняться параллельно несколькими процессорами. Задачи должны быть спроектированы таким образом, чтобы минимизировать конфликты между процессорами, обращающимися к пространству общей памяти. Механизмы синхронизации, такие как блокировки или семафоры, должны использоваться для предотвращения условий гонки. Модель общей памяти может использоваться для повышения производительности многих алгоритмов и приложений, включая матричное умножение, обработку изображений и параллельную сортировку. Она часто используется в сочетании с другими моделями параллельных вычислений, такими как распределенная память или параллелизм задач, чтобы воспользоваться их преимуществами [105].

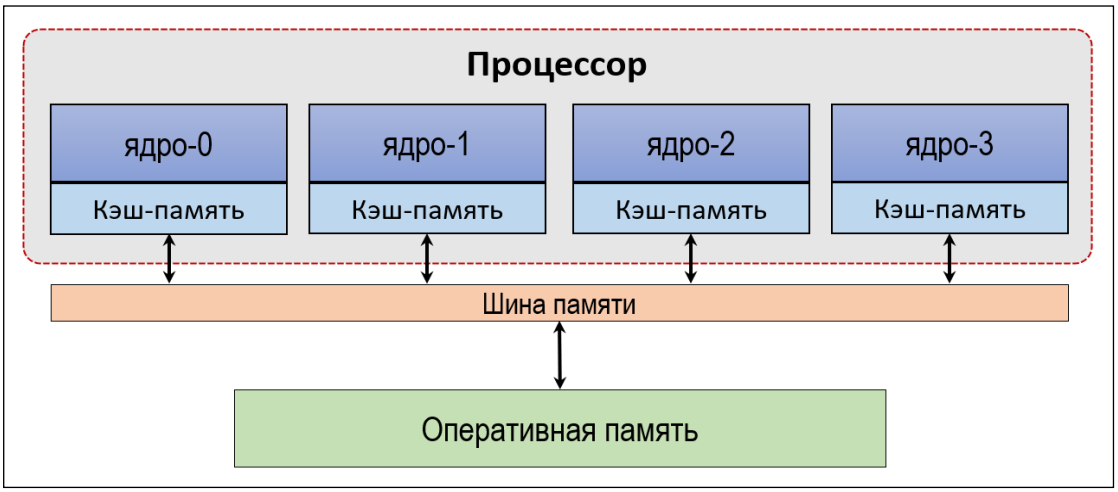


Рис. 3.1 Архитектура модели общей памяти

OpenMP (Open Multi-Processing) - это фреймворк, который широко используется в различных вычислительных областях для обеспечения параллелизма с общей памятью. Цель OpenMP - повысить производительность приложений за счет возможности параллельной обработки данных на мультипроцессорах с общей памятью. Это приводит к сокращению времени вычислений и повышению эффективности. OpenMP сыграл важную роль в оптимизации производительности широкого спектра приложений. Кроме того, OpenMP используется для выявления гонок данных в параллельных программах, что повышает производительность разработчиков и надежность системы. Это привело к значительному увеличению скорости [106].

OpenMP предоставляет интерфейс прикладного программирования (API), явно предназначенный для облегчения параллельного программирования в системах с общей памятью, включая многоядерные процессоры и SMP-системы. Его обширный набор директив, функций и переменных окружения позволяет разработчикам эффективно распараллеливать свой код. Фундаментальный принцип, лежащий в основе OpenMP, этот метод предполагает разбиение программы или алгоритма на более мелкие задачи, которые могут быть выполнены параллельно несколькими потоками. Эти потоки, будучи легковесными процессами, делят

одно и то же пространство памяти, обеспечивая тем самым эффективный доступ и модификацию общих структур данных. Используя директивы OpenMP, программисты могут указать, какие участки кода должны выполняться параллельно, и контролировать количество используемых потоков. Кроме того, OpenMP включает в себя механизмы синхронизации, такие как блокировки и барьеры, для предотвращения условий гонки и обеспечения упорядоченного выполнения потоков. Функция балансировки нагрузки в OpenMP способствует равномерному распределению работы между потоками, что значительно повышает производительность. В заключение можно сказать, что OpenMP представляет собой ценную библиотеку для параллельного программирования в системах с общей памятью, предлагая удобный и адаптируемый подход, который может заметно повысить производительность различных приложений [107]. Таким образом, OpenMP играет ключевую роль в повышении производительности и эффективности широкого спектра вычислительных задач с помощью методов параллельной обработки, на рис. 3.2 Модель Fork-Join для распараллеливания OpenMP.

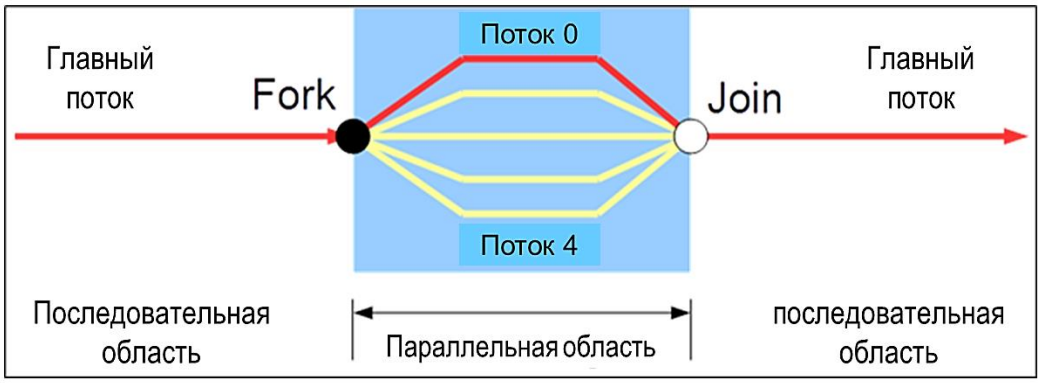


Рис. 3.2 Модель Fork-Join для распараллеливания OpenMP

Библиотека OpenMP основана на принципе параллельного программирования в системах с общей памятью. Она предоставляет набор директив, функций и переменных окружения, которые позволяют

разработчикам эффективно распараллеливать свой код. Основная концепция OpenMP заключается в разделении программы или алгоритма на более мелкие задачи, которые могут выполняться параллельно несколькими потоками. Потоки — это легковесные процессы, разделяющие одно и то же пространство памяти, что позволяет им эффективно обращаться к общим структурам данных и изменять их.

Директивы OpenMP используются для указания того, какие участки кода должны выполняться параллельно. Вставляя эти директивы соответствующим образом, программисты могут указать, какие части кода могут выполняться одновременно несколькими потоками. Эти директивы также позволяют программистам регулировать количество потоков, используемых при параллельном выполнении [108].

Механизмы синхронизации, такие как блокировки и барьеры, встроены в OpenMP для предотвращения условий гонки и обеспечения упорядоченного выполнения потоков. Блокировки используются для защиты критических участков кода, позволяя только одному потоку одновременно обращаться к этому участку. Барьеры используются для синхронизации потоков, гарантируя, что все потоки достигнут определенной точки в коде, прежде чем продолжить работу [109].

Дополнительной значимой характеристикой OpenMP является функция балансировки нагрузки. Она обеспечивает равномерное распределение операций между потоками, гарантируя, что каждый поток выполняет сопоставимый объем работы. Это свойство способствует повышению эффективности параллельного кода путем максимального использования вычислительных ресурсов, доступных для выполнения. [110].

Используя возможности библиотеки OpenMP, разработчики могут создавать эффективные и масштабируемые параллельные программы, которые могут использовать всю мощь современных многоядерных процессоров [111, 112].

3.2. Разработка и реализация параллельного алгоритма Дейкстры

Стандартный алгоритм Дейкстры — это алгоритм поиска кратчайших путей в ориентированных (потоковых) графах. Алгоритм был разработан голландским ученым-компьютерщиком Эдсгером В. Дейкстрой в 1956 году [113]. Алгоритм Дейкстры - популярный алгоритм поиска кратчайшего пути между исходной вершиной и всеми остальными вершинами во взвешенном направленном графе. Он гарантирует кратчайший путь при условии, что веса ребер неотрицательны. Приведем высокоуровневое описание стандартного алгоритма Дейкстры:

1) Инициализировать алгоритм, установив расстояние до исходной вершины равным 0, а до всех остальных вершин - бесконечности.
2) Создать пустое множество посещенных вершин.
3) Пока есть не посещенные вершины:
    - Выбрать вершину с минимальным расстоянием от источника среди не посещенных вершин. Эта вершина выбирается в качестве текущей вершины.
    - Добавить текущую вершину в набор посещенных вершин.
    - Для каждого соседа текущей вершины, который не входит в набор посещенных вершин:
        - Вычислить расстояние от источника до соседа через текущую вершину. Если оно меньше текущего расстояния, обновите расстояние соседа.
4) Когда все вершины посещены или достигнута вершина назначения, алгоритм завершает работу.

Временная сложность алгоритма Дейкстры зависит от структуры данных, используемой для очереди приоритетов. Ниже приводится разбивка временной сложности на основе различных реализаций:

- Использование несортированного списка в качестве приоритетной очереди: $O(V^2)$, где V - количество вершин в графе. На каждой итерации

алгоритм ищет вершину с наименьшим расстоянием среди всех непосещенных вершин, что занимает время O(V). Эта операция выполняется V раз, в результате чего временная сложность составляет $O(V^2)$.

- Использование отсортированного списка или двоичной кучи в качестве очереди приоритетов: O(E+V logV), где E - количество ребер в графе. На каждой итерации алгоритм извлекает из очереди приоритетов вершину с наименьшим расстоянием, что занимает время O(logV). Обновление расстояния до соседних вершин занимает в общей сложности O(E) времени. Эта операция выполняется V раз, что приводит к временной сложности O(V log V + E log V). Поскольку E может быть не более V^2, временная сложность составляет O(E + V log V).

Алгоритм Дейкстры — это широко используемый алгоритм кратчайшего пути в транспортных, телекоммуникационных и компьютерных сетях. Он определяет кратчайший путь от начального узла до всех остальных узлов графа. Однако его вычислительная сложность возрастает с увеличением размера графа, что делает его неэффективным для больших графов.

Нами предложен подход для повышения его эффективности основанный на оригинальной схеме распараллеливания процессов, который позволяет использовать его при оптимизации перевозок в карьере для поиска кратчайших маршрутов и оптимальный трасс перемещения большегрузных самосвалов.

Следует отметить, что сама идея реализация алгоритма Дейкстры в схеме параллельных вычислений не является новой. В различных исследовательских работах были предложены различные методы разработки алгоритма Дейкстры. Один из подходов предполагает использование концепции Q-детерминанта для разработки эффективных программ для численных и графовых алгоритмов, обеспечивающих полное использование ресурсов параллелизма [114]. Другой метод направлен на разработку эффективной системы навигации внутри помещений путем реализации

алгоритма Дейкстры для поиска кратчайшего пути между источником и пунктом назначения, что особенно полезно в закрытых помещениях, где отсутствует GPS [115]. Кроме того, был разработан метод оценки добычи и потребления сырья с использованием алгоритма Дейкстры для решения задач кратчайшего пути при планировании производства, что помогает определить транспортные узлы и точки продаж [116]. Кроме того, Основные проблемы при разработке алгоритма Дейкстры с использованием методов параллельных вычислений включают в себя сложность достижения значительного параллелизма без чрезмерного увеличения рабочей нагрузки [114], последовательный характер алгоритма, препятствующий эффективному распараллеливанию [117], и необходимость одновременного определения нескольких правильных вершин для улучшения параллельного выполнения [118]. Несмотря на применение таких методов распараллеливания, как OpenMP и OpenCL, значительное ускорение, которое может быть достигнуто, остается сложной задачей из-за последовательной природы алгоритма [119]. Для решения этих проблем были предложены инновационные подходы, включая использование вспомогательных потоков и транзакционной памяти, которые направлены на эффективное извлечение параллелизма и снижение накладных расходов, связанных с синхронизацией, что приводит к повышению производительности на современных многоядерных платформах [120]. В заключение следует отметить, что поиск эффективных параллельных реализаций алгоритма Дейкстры по-прежнему сталкивается с серьезными проблемами, связанными с балансом работ, фаз и эффективностью параллельного выполнения. Представим общую процедуру работы параллельного алгоритма Дейкстры:

Входные данные:

- граф: взвешенный граф, представленный матрицей смежности
- src: исходный узел для вычисления кратчайшего пути
- num_Threads: количество потоков/процессов для параллельного выполнения

Выходные данные:

- dist: массив, содержащий кратчайшие расстояния от исходного узла в графе

Вычислительная процедура:

1. Инициализировать следующие массивы и переменные:
   - dist[V]: массив для хранения кратчайших расстояний от исходного узла до каждого узла
   - sptSet[V]: массив для хранения узлов, включенных в дерево кратчайших путей
   - установить бесконечность (INF) для всех элементов dist(), кроме dist[src], который устанавливается в 0
   - установите все элементы sptSet() в false
2. Разделить итерации главного цикла между доступными потоками/процессами
   - разделить итерации первого цикла (for-i) на количество потоков.
   - разделить итерации второго цикла (for-count) на количество потоков.
3. Параллельное выполнение главного цикла:
   a. каждый поток выполняет цикл for-i независимо:
      - (инициализируем как INF и как false)
   b. каждый поток независимо выполняет цикл for-count:
      - найти вершину с минимальным значением расстояния среди еще не обработанных вершин.
      - пометить u как обработанную, установив sptSet[u] в true.
      - обновить значения расстояний до соседних вершин u:
        - для каждой вершины v в графе:
          - если v нет в sptSet и есть ребро между u и v:
            - если расстояние от u до v короче, чем текущее расстояние до v:
              - обновить dist[v] новым более коротким расстоянием.

4. Кратчайшие расстояния от исходного узла до всех остальных узлов вычислены. Вернуть массив (dist[]).

Физическое представление процесса параллельного выполнения параллельного алгоритма Дейкстры предполагает одновременную работу нескольких вычислительных устройств для определения кратчайших путей от исходного узла ко всем остальным узлам графа (см. рис. 3.3). Ниже приводится описание физического представления:

1) Разбиение графа: Граф разделяется на меньшие подграфы, присваивая каждый из них отдельному вычислительному устройству. Процесс разбиения может основываться на различных стратегиях, включая деление графа на подграфы одинакового размера или учет связности и весов ребер для обеспечения баланса в распределении рабочей нагрузки.
2) Коммуникационная сеть: между вычислительными блоками (ядрами) создается коммуникационная сеть для обмена данными и синхронизации.
3) Инициализация исходных узлов: Каждый вычислительный блок инициализирует значения расстояний для узлов в назначенном ему подграфе. Расстояние до исходного узла устанавливается равным нулю, а расстояния до всех остальных узлов - бесконечности или очень большому значению.
4) Параллельный алгоритм Дейкстры выполняется на каждом вычислительном блоке одновременно, выполняя локальные вычисления для обновления ориентировочных расстояний. Каждый модуль выбирает узел с наименьшим ориентировочным расстоянием на основе местной информации. Узел с наименьшим ориентировочным расстоянием между всеми вычислительными блоками выбирается глобально и становится следующим обрабатываемым узлом. Глобально выбранный узел передается всем вычислительным блокам, обновляя их локальную информацию на основе выбранного узла. Параллельное выполнение продолжается до тех пор, пока не будут обработаны все узлы или не будет

выполнено условие завершения, например достижение узла назначения или дальнейшее улучшение расстояний станет невозможным.

5) Реконструкция пути — это процесс, в котором кратчайшие пути от исходного узла ко всем остальным узлам восстанавливаются после завершения параллельного выполнения.

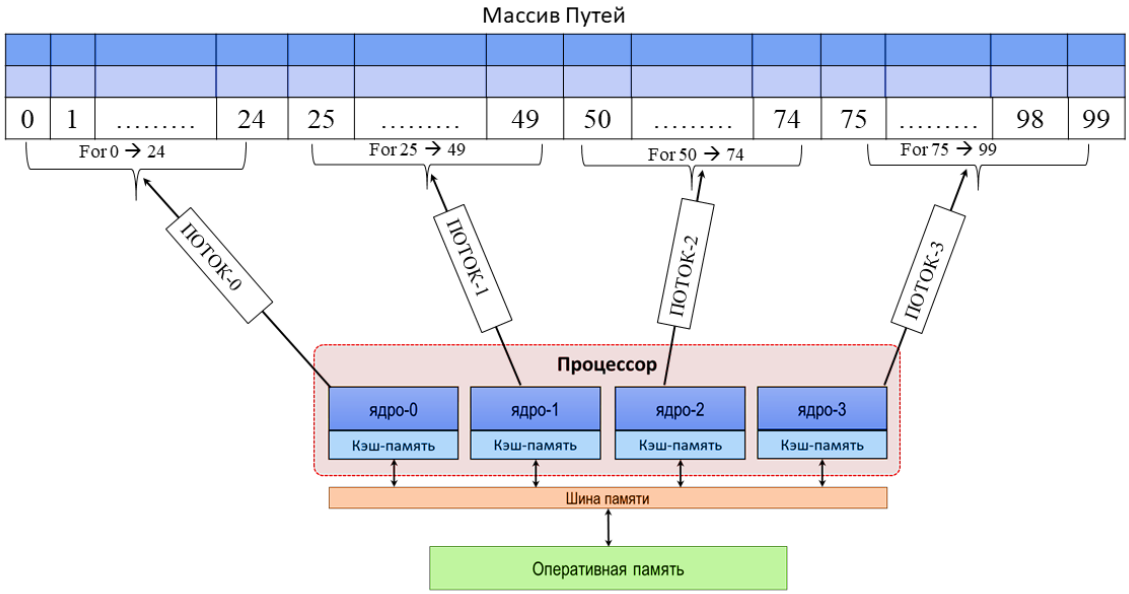


Рис. 3.3 Пример распараллеливания алгоритма Дейкстры и распределения операций группирования путей по количеству ядер

На рис. 3.4 представлена блок-схема параллельного алгоритма Дейкстры.

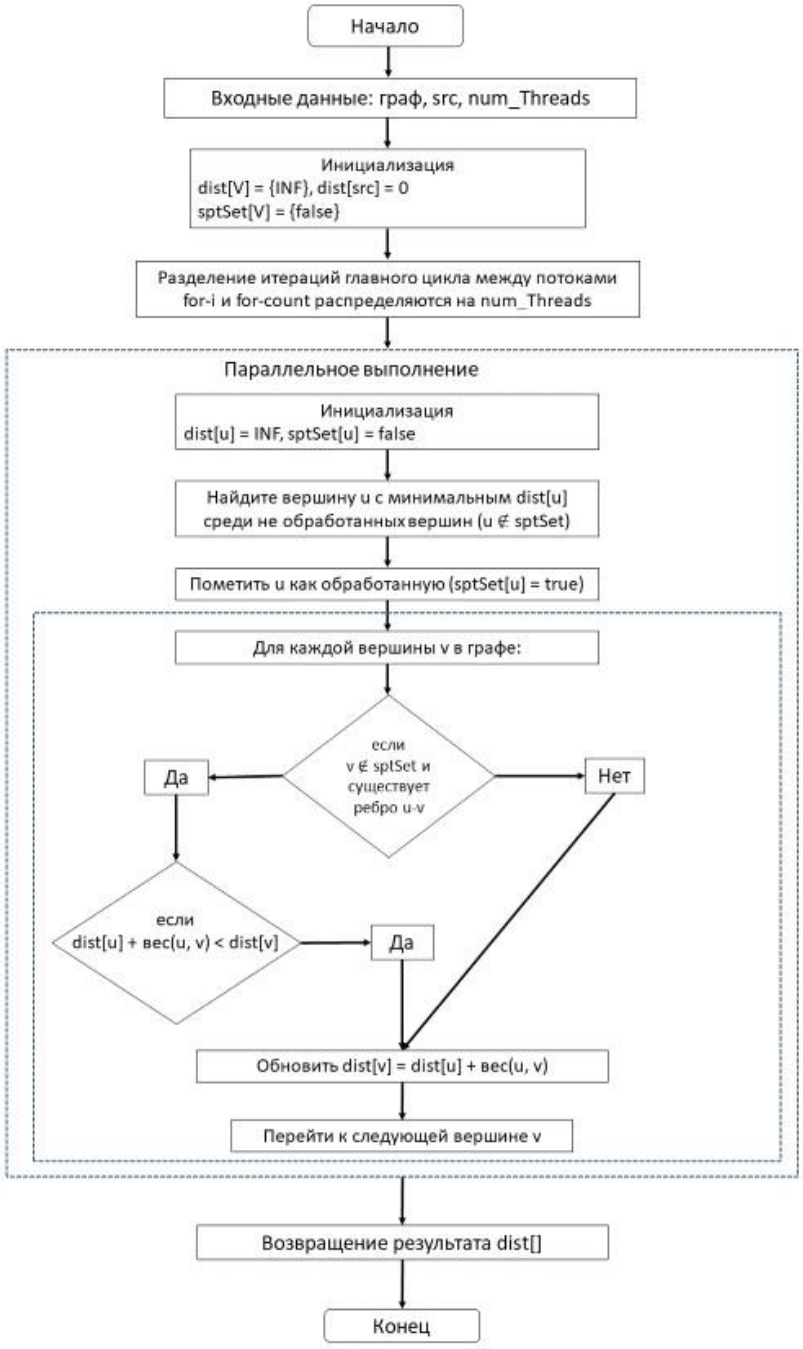


Рис.3.4 Блок-схема параллельного алгоритма Дейкстры.

Теперь рассчитаем временную сложность параллельного алгоритма Дейкстры. Для получения таблицы маршрутизации нам потребуется *O(V)* раундов итераций (пока все вершины не будут включены в кластер). В каждом раунде мы будем обновлять значения для *O(V)* вершин с помощью *P* ядер, работающих независимо друг от друга, и использовать параллельный префикс для выбора глобальной ближайшей вершины, поэтому время работы в каждом

раунде составляет $O\left(\frac{V}{P}\right) + O(\log P)$. Таким образом, общее время работы составляет:

$$T_p = O\left(\frac{V^2}{P} + V(\log P)\right), \quad (3.1)$$

где P - количество используемых ядер, а V - количество вершин. Поскольку время последовательного выполнения составляет $T_1 = O(V^2)$, ускорение и эффективность будут следующими:

Для оценки качества работы использовались классические для параллельных алгоритмов показателя качества: Ускорение и эффективность - два важных показателя производительности в параллельных вычислениях.

Ускорение-это отношение времени выполнения последовательного алгоритма к времени выполнения параллельного алгоритма, решающего ту же задачу. Оно показывает, насколько быстрее работает параллельный алгоритм по сравнению с последовательным: ускорение (S)= *T_seq* / *T_par*, где *T_seq* - время выполнения последовательного алгоритма;

T_par - время выполнения параллельного алгоритма.

Эффективность измеряет то насколько рационально используются доступные ресурсы в параллельном алгоритме.

Она представляет собой отношение ускорения, достигнутого при использовании нескольких процессоров, к количеству используемых процессоров: эффективность (E) = *S* / *P*, где *S* - ускорение; *P* - количество процессоров. Важно отметить, что достижение высокого ускорения не обязательно означает высокую эффективность. Известно, что параллельный алгоритм может достичь высокого ускорения, но иметь низкую эффективность, если ресурсы используются неэффективно. Аналогично, параллельный алгоритм может иметь высокую эффективность, но низкое ускорение, если размер задачи недостаточно велик, чтобы извлечь выгоду из параллелизма.

Разработанный алгоритм был протестирован на настольном компьютере с процессором Intel (R) Core (TM) i7-8550U с частотой 1,80 ГГц, 8 ядрами, 8 потоками и 16 ГБ оперативной памяти. В качестве языка программирования использовался C++ с библиотекой OpenMP под Microsoft Visual Basic 2022 в среде операционной системы Windows 11 с 64-битной архитектурой. Принцип работы модели общей памяти состоит в том, что алгоритм начинает выполняться последовательно с основным потоком, называемым ведущим потоком, а когда выполнение достигает параллельной области, начинается параллельное выполнение с определенным количеством потоков с помощью переменной (num_threads). переменной (num_threads) равным единице, и таким образом параллельная область выполняется с использованием только одного потока. Параллельный алгоритм выполняется с количеством потоков в диапазоне от 2 до 8. Число 8 потоков — это максимальное число потоков, равное количеству используемых ядер процессора, где каждый поток выполняется на одном ядре одновременно. После выполнения кода получаем итоговый результат – продолжительность процесса в секундах.

Вычислительные эксперименты проводились с матрицами различной размерности, соответствующими трем различным графам с количеством вершин: 500,750 и 1000. Число процессоров (ядер) варьировалось в диапазоне от 1 до 8. В таблице 3.1 представлено время в секундах выполнения операции поиска всех кратчайших путей, в зависимости от числа задействованных ядер.

Таблица 3.1

Количество вершин	Количество потоков (ядер)							
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
500	165,45	146,35	91,38	73,95	62,62	52,24	51,65	50,16
750	380,15	251,65	212,39	159,18	132,15	115,03	112,88	110,39
1000	1510,31	1340,73	866,56	654,02	544,68	461,88	421,44	375,46

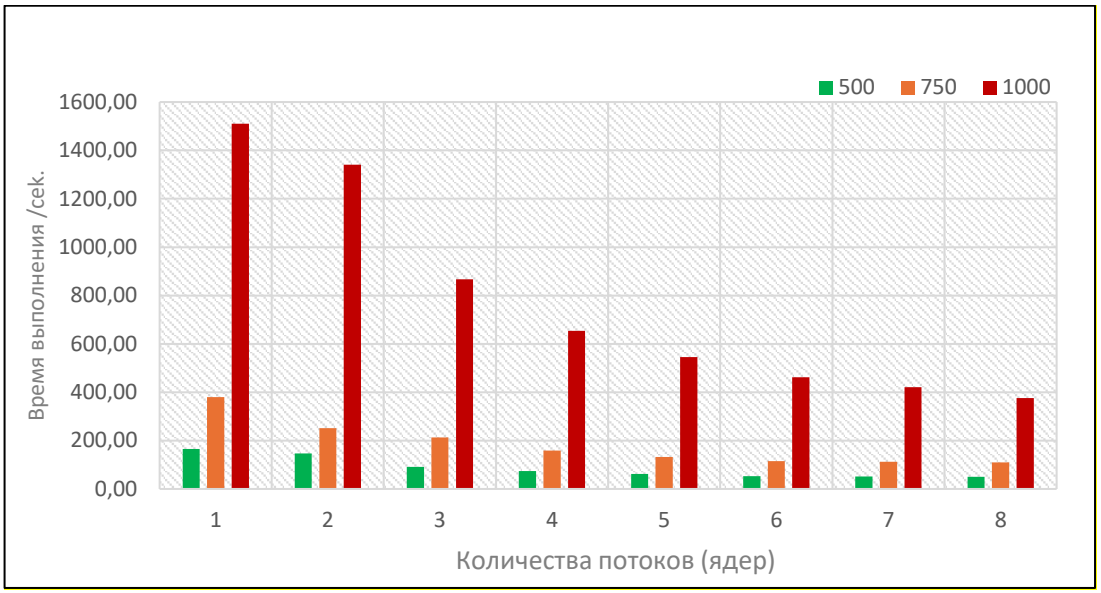


Рис.3.5 Иллюстрация эффективности использования параллельной схемы

В таблице 3.2 представлена величина ускорения по сравнению с последовательным выполнением вычислительного процесса, а рис. 3.6 наглядно иллюстрирует этот эффект:

Таблица 3.2

Количество вершин	Количество потоков /ядер							
	1	2	3	4	5	6	7	8
500	1,00	1,13	1,81	2,24	2,64	3,17	3,20	3,30
750	1,00	1,51	1,79	2,39	2,88	3,31	3,37	3,44
1000	1,00	1,13	1,74	2,31	2,77	3,27	3,58	4,02

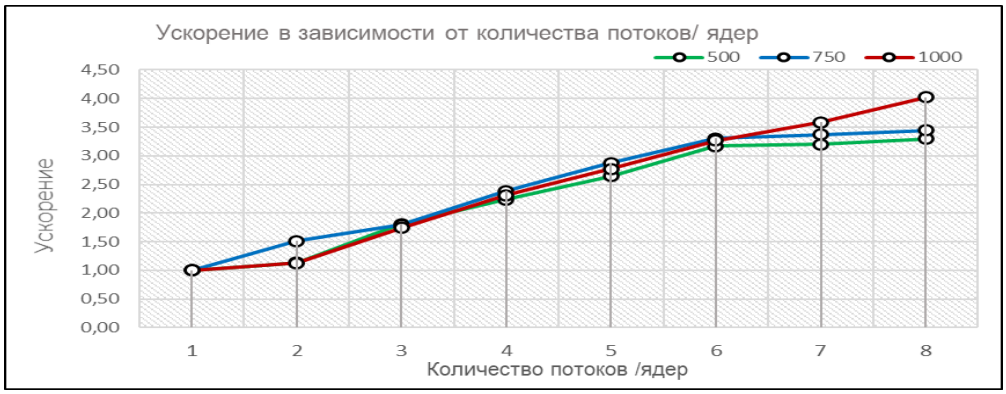


Рис. 3.6 иллюстрация ускорения вычислительного процесса

Таблица 3.3

Количество вершин	Количество потоков /ядер							
	1	2	3	4	5	6	7	8
500	1,00	0,57	0,60	0,56	0,53	0,53	0,46	0,41
750	1,00	0,76	0,60	0,60	0,58	0,55	0,48	0,43
1000	1,00	0,56	0,58	0,58	0,55	0,54	0,51	0,50

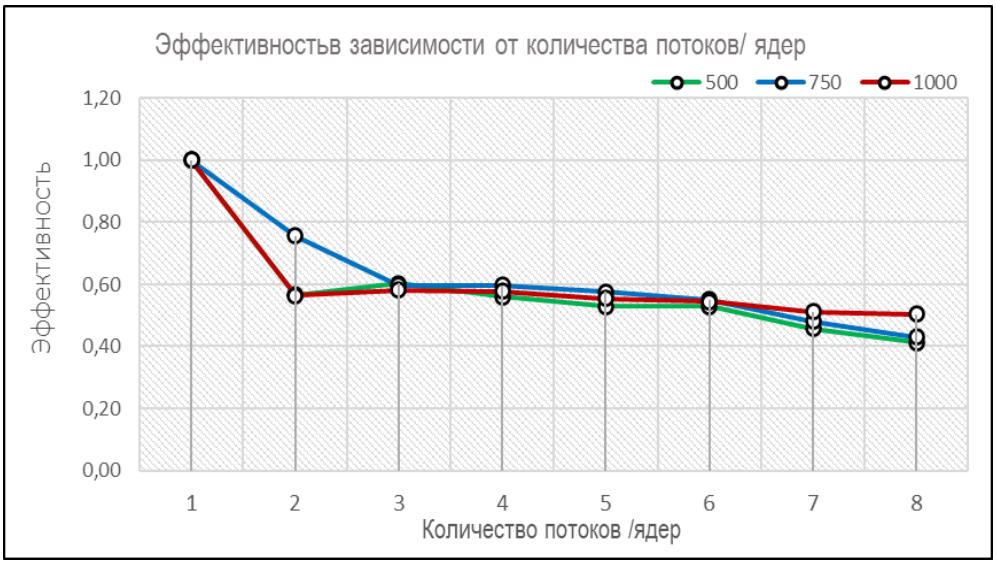


Рис. 3.7 Эффективность распараллеливания в зависимости от числа ядер

3.3. Выбор вычислительной модели для прогнозирования скорости автономного самосвала.

Как широко известно, предсказание скорости автономных карьерных самосвалов с высокой точностью, даже при движении по произвольным траекториям или фиксированным маршрутам, представляет собой значимую задачу, поскольку:

- Позволяет реализовать динамические протоколы безопасности, особенно в сценариях, когда неожиданные препятствия или изменения в

окружающей среде могут потребовать немедленной корректировки скорости транспортного средства;

- Усовершенствует маршруты, сокращает временные затраты на перемещение автосамосвалов между технологическими зонами, снижает очереди на площадках экскаваторов и обеспечивает точную синхронизацию передвижения в соответствии с ритмами других операций в транспортно-технологическом процессе.

Это связано с тем, что знание скорости на определенных временных отрезках позволяет с использованием специальных моделей прогнозировать с достаточной точностью местоположение роботов-самосвалов:

$$\hat{x}_i[k+1] = F_2\{\tilde{x}_i[k], \dots, \tilde{x}_N[k], \hat{x}_i[k+1], \dots, \hat{x}_N[k+1], \{\beta\}\} \quad (3.2)$$

Вопрос прогнозирования скорости не только отдельных мобильных объектов, но и транспортных потоков в целом давно привлекает интерес исследователей, поэтому существует значительное количество походов (моделей), прогнозирующих их скоростные характеристики:

$$\widehat{V}_i[k+1] = F_1\{\widehat{x}_i[k], \widehat{x}_i[k+1], \dots, \widehat{x}_i\,[k+l, \tilde{\bar{v}}[k]\,]\} \quad (3.3)$$

Кратко остановимся на основных подходах, к решению задачи прогнозирования скорости в транспортных потоках. Существующие модели прогнозирования транспортных потоков можно разделить на три категории: классические статистические теории и аналитические модели, традиционные методы машинного обучения и методы глубокого обучения.

Модель ARIMA часто использовалась в прошлом для прогнозирования транспортного потока на автострадах, однако точность прогнозирования была неудовлетворительной [121]. Этот метод прост, понятен, легко реализуем и имеет определенное справочное значение для прогнозирования характеристик транспортного потока. Однако он не может учесть влияние непрогнозируемых событий на транспортный поток, и поэтому его точность невысока. В 1980-х годах Стефанидис и Окутани применили модель исторического среднего (HA) и модель фильтра Калмана к системе управления городским движением [122], а также исследовали связи между участком дороги, интенсивность движения

на котором нужно спрогнозировать, и участком дороги на прилегающей территории [123]. В процессе улучшения параметров модели, исследователи внесли значительное количество корректировок для устранения ошибок прогнозирования. Некоторые ученые также прибегают к использованию традиционных моделей временных рядов для прогнозирования транспортного потока. Среди них можно выделить модель векторной авторегрессии VAR (vector autoregressive model), модель авторегрессии скользящего среднего ARMA (autoregressive moving average model), и модель авторегрессии интегрированного скользящего среднего ARIMA (autoregressive integrated moving average model), а также их вариации, такие как модель стационарности ARIMA и сезонная модель ARIMA [123-126]. Это типичные параметрические модели, которые предполагают линейную зависимость между значением временного ряда в определенный момент времени t и историческими наблюдениями, а также случайным шумом. При использовании данных моделей прогнозирование происходит путем выявления закономерностей движения транспортного потока во времени на основе предшествующих наблюдений. Эти методы часто демонстрируют хорошие результаты в случаях, когда временной ряд обладает определенной периодичностью или регулярностью. Однако в реальных условиях дорожного движения транспортные потоки подвержены влиянию множества факторов и формируются в условиях значительной степени неопределенности.

Определенные успехи были также достигнуты при использовании классических алгоритмов машинного обучения, таких как: K-Nearest Neighbours (KNN) для прогнозирования трафика. Например, Сайрус Шахаби и др. [127] вычислили географические точки с похожими условиями движения с помощью KNN. Затем объединили их для составления краткосрочных прогнозов. Основным недостатком этого метода является необходимость эвристического подбора параметра K, то есть количества ближайших соседей. Шаолун Сунь и другие [128] объединили эмпирическую декомпозицию модели EMD (Empirical Mode Decomposition) с нейронной сетью BP (Back

Propagation) для прогнозирования пешеходного потока в метро, EMD используется для разложения данных временного ряда на сезонные и трендовые компоненты и для создания новых характеристик. Кроме того, для краткосрочного прогнозирования транспортных потоков. широко используется модель Support Vector Machine (SVM) [129]. Принцип ее работы заключается в создании гиперплоскости в высоко размерном пространстве для решения нелинейной задачи классификации, что позволяет достичь более точного эффекта классификации при выборе различных базисных функций, в том числе и вейвлет-функций. Эти методы интеграции в основном моделируют нелинейность с точки зрения временного измерения последовательности. Однако они не учитывают прямую корреляцию между последовательностями транспортных потоков в пространственных измерениях. Возможности модели ограничены, и ее нелегко расширить.

Теория и методы глубоких нейронных сетей в последнее время быстро развиваются и широко применяются в различных областях. Одним из наиболее важных приложений является прогнозирование динамики процессов, в частности, характеристик транспортных потоков. Они позволяют объединять классическое прогнозирование временных рядов и традиционные методы машинного обучения для прогнозирования. Для изучения взаимосвязи последовательности движения во времени некоторые ученые используют модели прогнозирования временных рядов RNN и ее разновидности (Long-Short Term Memory (LSTM), GRU [130] для прогнозирования характеристик транспортных систем. Некоторые ученые комбинируют модели RNN с традиционными методами машинного обучения [131]. Хотя модели RNN могут отражать последовательные связи транспортного потока во временном измерении и обладают хорошей масштабируемостью, они не могут обрабатывать последовательности на основе конкретных пространственных связей между последовательностями. Чтобы учесть временную и пространственную корреляцию транспортного потока одновременно, необходимо построить инструменты прогнозирования

транспортного потока с точки зрения поиска временных и пространственных характеристик. Прогнозирование транспортного потока — это типичная пространственно-временная задача интеллектуального анализа данных.

Временные данные, Временной ряд, также известный как последовательность временных данных, представляет собой набор данных, фиксирующих определенное явление в различные моменты времени. В реальном мире данные часто имеют временную зависимость, и набор последовательных наблюдений, собранных в хронологическом порядке, образует временной ряд данных.

Пространственные данные, информация, содержащая пространственные координаты, представляет собой специфический вид данных, который обеспечивает количественное описание объектов или явлений, связанных с определенным местоположением. Примерами таких данных могут служить географические координаты и характеристики пространственного распределения объектов. Пространственно-временные данные представляют собой вариант пространственных данных, в которых также учитывается изменение во времени. Например, в случае онлайн-заказов автомобильной доставки, помимо пространственной информации, присутствуют временные атрибуты, такие как момент создания заказа. [132]. Эти данные обладают очевидными признаками пространственного распределения, а также характеристиками, такими как большие объемы информации, нелинейность и динамические изменения со временем.

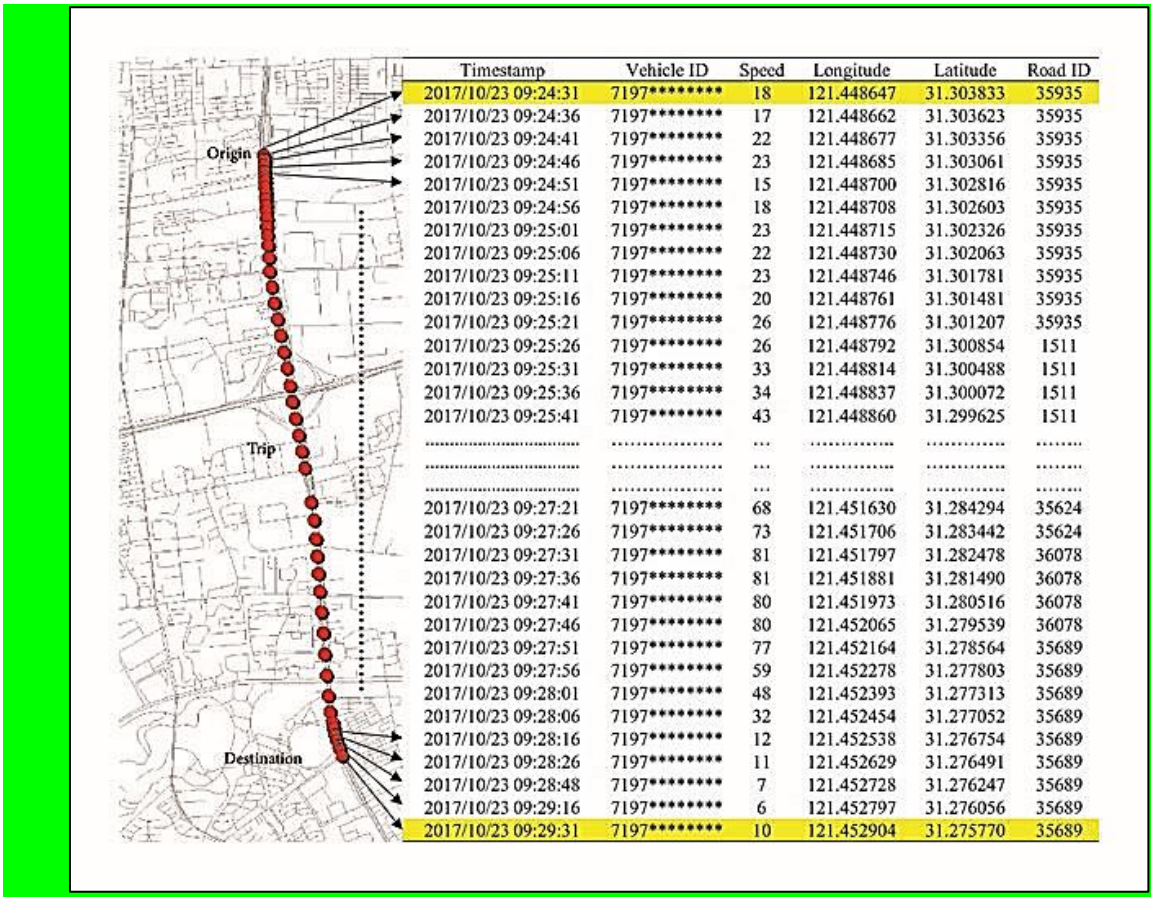
С появлением новых технологий и методов сбора данных стало возможным эффективно собирать пространственно-временные данные для исследований в различных областях. Это способствует разработке алгоритмов и моделей пространственно-временного анализа данных (STDM). Например, одной из типичных задач пространственно-временного анализа данных является прогнозирование транспортных потоков. Для решения таких задач исследователям необходимо анализировать соответствующие данные во

временных и пространственных измерениях, чтобы выявить пространственно-временные закономерности.

Приобретение пространственно-временных данных

В современном обществе накопление временных и пространственных данных связано с разнообразными формами поведения человека. Эти пространственно-временные данные представляют собой ценный источник информации о социальной жизни. Например, анализ траекторий движения пользователей с использованием мобильных устройств позволяет изучать их перемещения и привычки, а также предоставлять различные сервисы, такие как прогнозирование местоположения и рекомендации по маршрутам [133].

Траектория — Траектория представляет собой путь, пройденный объектом в пространстве в течение определенного времени. Примерами траекторий могут служить маршруты такси от места посадки до места высадки пассажиров или миграционные пути животных. Обычно данные о траекториях собираются с помощью датчиков, установленных на передвигающихся объектах, которые регулярно передают информацию о их местоположении. Например, GPS-данные могут использоваться для получения траектории движения такси. Эти данные, как правило, содержат не только информацию о местоположении объекта, но и другие характеристики, изменяющиеся во времени, такие как скорость транспортного средства или частота сердечных сокращений человека во время бега. Данные о траекториях широко применяются в различных областях, включая транспорт и экологию, как показано на Рис. 3.8 представлен пример данных о траектории движения автомобиля.



Timestamp	Vehicle ID	Speed	Longitude	Latitude	Road ID
2017/10/23 09:24:31	7197********	18	121.448647	31.303833	35935
2017/10/23 09:24:36	7197********	17	121.448662	31.303623	35935
2017/10/23 09:24:41	7197********	22	121.448677	31.303356	35935
2017/10/23 09:24:46	7197********	23	121.448685	31.303061	35935
2017/10/23 09:24:51	7197********	15	121.448700	31.302816	35935
2017/10/23 09:24:56	7197********	18	121.448708	31.302603	35935
2017/10/23 09:25:01	7197********	23	121.448715	31.302326	35935
2017/10/23 09:25:06	7197********	22	121.448730	31.302063	35935
2017/10/23 09:25:11	7197********	23	121.448746	31.301781	35935
2017/10/23 09:25:16	7197********	20	121.448761	31.301481	35935
2017/10/23 09:25:21	7197********	26	121.448776	31.301207	35935
2017/10/23 09:25:26	7197********	26	121.448792	31.300854	1511
2017/10/23 09:25:31	7197********	33	121.448814	31.300488	1511
2017/10/23 09:25:36	7197********	34	121.448837	31.300072	1511
2017/10/23 09:25:41	7197********	43	121.448860	31.299625	1511
.....	.....	...	.....	.....	.....
.....	.....	...	.....	.....	.....
.....	.....	...	.....	.....	.....
2017/10/23 09:27:21	7197********	68	121.451630	31.284294	35624
2017/10/23 09:27:26	7197********	73	121.451706	31.283442	35624
2017/10/23 09:27:31	7197********	81	121.451797	31.282478	36078
2017/10/23 09:27:36	7197********	81	121.451881	31.281490	36078
2017/10/23 09:27:41	7197********	80	121.451973	31.280516	36078
2017/10/23 09:27:46	7197********	80	121.452065	31.279539	36078
2017/10/23 09:27:51	7197********	77	121.452164	31.278564	35689
2017/10/23 09:27:56	7197********	59	121.452278	31.277803	35689
2017/10/23 09:28:01	7197********	48	121.452393	31.277313	35689
2017/10/23 09:28:06	7197********	32	121.452454	31.277052	35689
2017/10/23 09:28:16	7197********	12	121.452538	31.276754	35689
2017/10/23 09:28:26	7197********	11	121.452629	31.276491	35689
2017/10/23 09:28:48	7197********	7	121.452728	31.276247	35689
2017/10/23 09:29:16	7197********	6	121.452797	31.276056	35689
2017/10/23 09:29:31	7197********	10	121.452904	31.275770	35689

Рис. 3.8. Пример, данных о траектории движения автомобиля.

В растровых данных каждое измерение в пространственно-временной области фиксируется в определенном узле пространственно-временной сетки. Растровые данные представляют собой набор объектов, у которых имеется фиксированное местоположение, обозначаемых как $S = \{s_1, s_2, \ldots s_m\}$. Эти объекты могут быть равномерно распределены в пространстве, с постоянным расстоянием между соседними объектами. Этот процесс аналогичен распределению элементов на изображении, как показано на рис. 3.8. Они также, могут быть распределены в пространстве нерегулярно, как например, сеть датчиков пересечения, как показано на рис.3.5, Для каждого местоположения объекта растровые данные фиксируют все его наблюдения в определенные временные точки, представленные набором временных меток $T= \{t_1, t_2, \ldots t_n\}$. Интервалы между соседними временными метками могут быть как постоянными, так и переменными. Произведение коллекции объектов местоположения, временных интервалов и коллекции образует

пространственно-временную сетку, которая представляет собой комбинацию пространственных и временных данных. Эта сетка обычно представляется в виде растровых данных. Каждая ячейка (s_i, t_j) в этой сетке соответствует конкретному значению измерения в определенном пространственном местоположении и временном интервале.

Применение пространственно-временных графовых конволюционных сетей (ST-GCN) представляется нам наиболее перспективным и инновационным подходом к моделированию и прогнозированию скорости автономных самосвалов в карьерных работах в условиях, когда не все трассы и траектории, по которым возможно перемещение самосвала изучены. Вот основные причины, по которым ST-GCN являются привлекательным выбором:

1) Улавливание пространственно-временных отношений:

ST-GCN отлично справляются с захватом сложных пространственно-временных отношений, присущих траекториям АКС. Горные работы связаны с перемещением по различным местностям и средам, и ST-GCN разработаны для эффективного моделирования развивающихся отношений между транспортным средством и окружающей средой с течением времени.

2) Моделирование на основе графов:

АКС работают в сети взаимосвязанных элементов, включая местность, другие транспортные средства и операционные зоны. ST-GCN, как модели на основе графов, позволяют создать комплексное представление этих взаимодействий. Такой подход способствует более точному и целостному пониманию сложной динамики, влияющей на поведение АКС.

3) Предсказания на основе траекторий:

АКС следуют по определенным траекториям, перемещаясь по карьеру. ST-GCN особенно хорошо умеют использовать данные о траекториях, позволяя предсказывать будущие перемещения на основе исторических моделей. Это очень важно для прогнозирования скорости АКС и оптимизации их поведения в режиме реального времени.

4) Адаптация к динамичным условиям:

Карьерная среда по своей природе является динамичной, с меняющимися условиями, такими как переменчивый рельеф и меняющиеся схемы движения. ST-GCN обладают высокой степенью адаптивности, что делает их хорошо приспособленными для работы в непредсказуемых условиях горных работ. Они могут приспосабливаться к колебаниям среды, обеспечивая надежные прогнозы в различных условиях.

5) Механизм комплексного обучения:

ST-GCN используют комплексный механизм обучения, который учитывает одновременно пространственные и временные аспекты. Такая двойная направленность повышает способность модели выявлять закономерности и взаимосвязи, позволяя более точно предсказывать скорость АКС при ее изменении во времени и пространстве.

3.4. Прогнозирование скорости автономного самосвала в различных точках технологической дороги на основе GCNN - моделей.

Модель ST-GCN отражает динамические пространственно-временные характеристики транспортного потока и включает механизмы самовнушения для повышения эффективности прогнозирования. В модели используются такие методы обработки признаков, как временное косинусоидальное разложение и одноточечное кодирование, чтобы уловить периодичность и неоднородность изменений транспортного потока.

Набор данных сети карьерных дорог представляет собой реальные данные, собранные в определенных точках карьерной дороги в Сибири, Россия. Для сбора данных использовалась технология GPS, в результате чего было получено более 310 тысяч пространственных точек, которые представляют собой точки на сети карьерных дорог. Каждый образец в наборе данных содержит временную информацию, такую как дата и время считывания данных, и пространственную информацию, включая долготу, широту и высоту

над уровнем моря. В наборе данных также указана скорость движения автономных транспортных средств, как показано в таблице 3.4. Однако данные не были свободны от неоднозначных и дублирующихся записей. Поэтому был проведен этап предварительной обработки для очистки и нормализации данных. Это включало удаление дублирующихся данных и устранение несоответствий, чтобы обеспечить построение соответствующего графа набора данных. Включение в набор данных пространственной и временной информации позволяет получить полное представление о сети карьерных дорог.

Таблица 3.4.

	Название	Описание	Пример значений
1	'ID_Node'	Индекс вершин	0, 1, 2, …
2	'Data'	Дата точки	2019-11-20
3	'Time'	Время точки	10:07:34
4	'Lat'	Широта	51.260093
5	'Lon'	Долгота	37.730011
6	'Height'	Высота над уровнем моря	68.021
7	'Azimut'	горизонтальный угол, измеряемый между заранее выбранным направлением	342.20404
8	'Speed'	Значение мгновенной скорости	7.973888

Процесс обработки данных состоял из следующих этапов:

1. Извлечение признаков:

Набор данных загружается из CSV-файла, фокусируясь на столбцах 'Date_Time', 'Lat', 'Lon', 'Height', 'Azimut' и 'Speed'. Столбец 'Date_Time' преобразуется в формат времени, и из него извлекаются временные характеристики 'Hour' и 'Minute'.

2. Нормализация:

Пространственные признаки 'Lat', 'Lon', 'Height' и 'Azimuth' нормализуются с помощью стандартизации (нормализации z-score), вычитания среднего

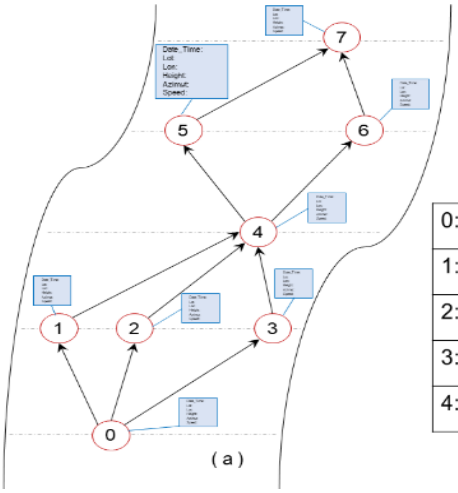
значения и деления на стандартное отклонение. Это гарантирует, что различные масштабы признаков не окажут непропорционального влияния на модель.

3. Разделение на тренировки и тесты:

Набор данных разбивается на обучающий и тестовый наборы с помощью функции scikit-learn. train_test_split

Конструирование графа осуществлялось следующим образом.

На основе теории графов граф G определяется как кортеж из множества узлов/вершин V и множества ребер/связей E: $G=(V,E)$. Каждое ребро - это пара из двух вершин, определяющая связь между ними. Генерируется матрица смежности для построения графа на основе временного порядка узлов. Процесс начинается с определения количества узлов (N) в наборе данных и инициализации матрицы смежности (adjacency_matrix) размером (NxN). Данные обрабатываются для создания "trass_matrix", которая упорядочивает узлы на основе их пространственной близости вдоль оси Y (широта). При этом инициализируется список "level_matrix" для хранения индексов узлов, расположенных в одном и том же положении по оси Y. Матрица смежности строится путем итерации по соседним уровням (строкам) "trass_matrix", обновляя соответствующие записи в матрице смежности. Итоговая матрица смежности представляет собой схему связей между узлами, основанную на их временном порядке и пространственной близости вдоль оси Y. Эта матрица может быть использована для построения неориентированного графа с узлами, представляющими точки данных, и ребрами, представляющими связи между ними. На рисунке 3.9 показаны элементы построения графа.



(а)

0:	0
1:	1, 2, 3
2:	4
3:	5, 6
4:	7

(б)

	0	1	2	3	4	5	6	7
0	0	1	1	1	0	0	0	0
1	1	0	0	0	1	0	0	0
2	1	0	0	0	1	0	0	0
3	1	0	0	0	1	0	0	0
4	0	1	1	1	0	1	1	0
5	0	0	0	0	1	0	0	1
6	0	0	0	0	1	0	0	1
7	0	0	0	0	0	1	1	0

(с)

Рис. 3.9 (а) Граф фрагмента карьерной дороги, (б) trass_matrix, (с) Матрица смежности

Модель ST-GCN обрабатывает входные признаки, используя информацию о графе с помощью матрицы смежности, затем два графовых конволюционных слоя и полностью связанный слой для получения конечного регрессионного результата. В процессе обучения модель изучает параметры конволюционных слоев и полностью связанного слоя, чтобы минимизировать функцию потерь и делать точные регрессионные прогнозы.

Пусть x обозначает входные признаки, W_{conv1} - веса первого конволюционного слоя, b_{conv1} - смещения первого конволюционного слоя, W_{conv2} - веса второго конволюционного слоя, b_{conv2} - смещения второго конволюционного слоя, W_{fc} - веса полностью связанного слоя.

1. Входной слой: Входные признаки x проходят через линейное преобразование с использованием матрицы смежности для включения информации о графе. Это достигается путем выполнения матричного умножения между входными признаками и матрицей смежности. Затем полученный тензор сжимается для удаления лишних измерений.

2. Графовые конволюционные слои: предварительно обработанный входной тензор пропускается через два конволюционных слоя (conv1 и conv2). Эти конволюционные слои работают с двумерным пространственно-временным пространством. *Первый конволюционный слой* (conv1) применяет

скрытое_размерное количество фильтров с размером ядра (1,3) по пространственному измерению, сохраняя временное измерение за счет подстановки. *Второй конволюционный слой* (conv2) применяет выходной_размер количество фильтров с размером ядра (1,3) по пространственному измерению, сохраняя при этом временное измерение за счет набивки. *Функции активации* ReLU применяются после каждого конволюционного слоя для введения нелинейности.

3. Полностью связанный слой: Выходные данные второго конволюционного слоя сглаживаются и проходят через слой с полным подключением (fc), чтобы получить конечный результат. Выход полностью связанного слоя представляет собой единичное значение, поскольку модель предназначена для задач регрессии. Обобщенная архитектура сети представлена на рис. 3.10.

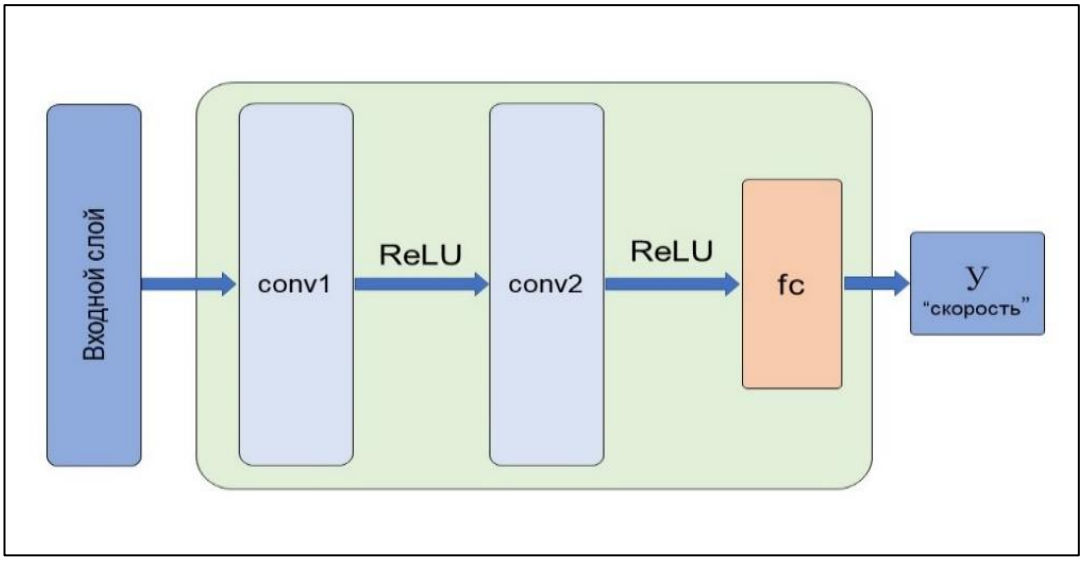


Рис. 3.10 Спроектированная архитектура GCNN

Методика обучения и оценки состоит из нескольких ключевых этапов. Вначале загружается набор данных и производится его предварительная обработка: временные характеристики извлекаются из временных меток, а пространственные характеристики нормализуются. Затем набор данных разбивается на обучающий и тестовый наборы для обучения и оценки модели. Определяется архитектура модели ST-GCN, состоящая из конволюционных

слоев, за которыми следует полностью связанный слой. Модель обучается с помощью мини-пакета стохастического градиентного спуска с функцией потерь по среднему квадрату ошибки (MSE) и оптимизатора Адама. Во время обучения производительность модели контролируется с помощью потерь при обучении и тестировании. Кроме того, для оценки точности и ошибок модели вычисляются такие метрики, как MSE, MAE и RMSE.

Процесс оценки включает в себя оценку прогностической эффективности обученной модели с помощью различных метрик. На протяжении всего процесса обучения вычисляются потери при тестировании, чтобы оценить способность модели к обобщению на невидимых данных. Кроме того, такие метрики, как MSE, MAE и RMSE, рассчитываются с использованием прогнозируемых значений скорости и фактических значений. Визуализации, включая графики потерь при обучении и тестировании в течение эпох и сравнения между фактическими и предсказанными значениями скорости, облегчают интерпретацию поведения модели и тенденций производительности.

Мы можем визуализировать производительность модели это делается путем сравнения потерь при обучении и тестировании по эпохам, чтобы визуализировать сходимость и производительность модели во время обучения, (рис. 3.11)

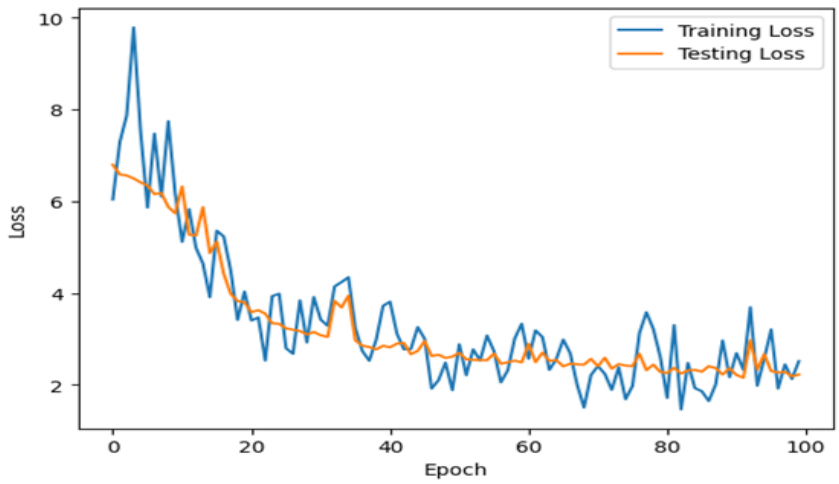


Рис. 3.11 Сравнение потерь при обучении и тестировании по эпохам.

Чтобы отследить эффективность модели, мы также представили в виде графиков оценочные показатели, включая MSE, MAE и RMSE, по эпохам (рис. 3.12).

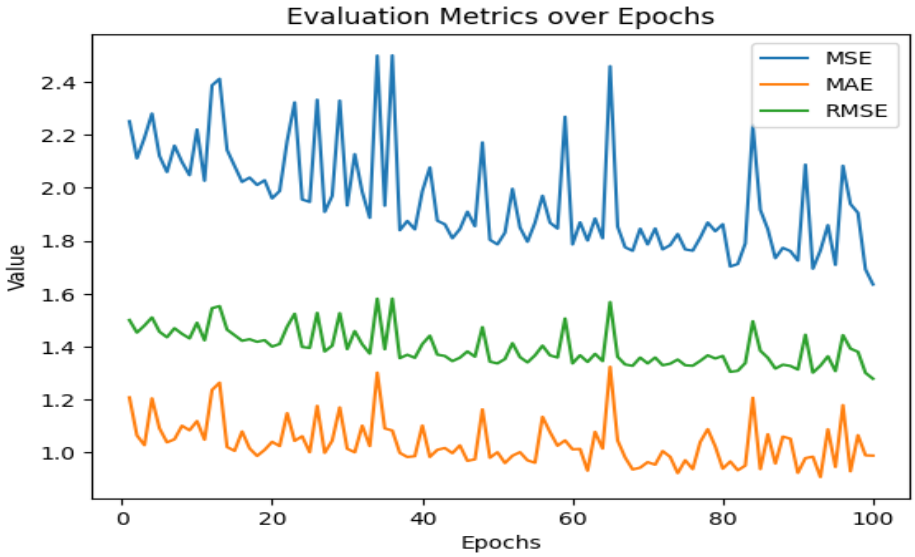


Рис. 3.12 Показатели оценки, такие как MSE, MAE и RMSE, по эпохам.

Исходя из представленных на рис. 3.12 результатов, постоянно близкие значения сравнения в течение эпох означают, что модель последовательно производит точные прогнозы без значительных колебаний или ухудшения производительности. Кроме того, на рис. 3.11 наблюдение за сходимостью потерь при обучении и тестировании в течение эпох указывает на обучаемость модели. Потери при обучении и тестировании уменьшаются и стабилизируются, что говорит о том, что модель эффективно обучается на данных и хорошо обобщает их на невидимые образцы. Кроме того, на рис. 5 показана приемлемая точность модели. Значения MSE, MAE и RMSE относительно низкие, что говорит о том, что прогнозы модели в целом близки к реальным значениям. Однако при интерпретации этих результатов важно учитывать конкретный контекст проблемной области и масштаб целевой переменной (скорости). Например, если значения скорости варьируются от 0 до 100, то RMSE в 1,49 может считаться приемлемым, в то время как если значения скорости варьируются от 0 до 50, то желательно иметь более низкий

RMSE. В таблице 3.5 представлены изменения значений перечисленных критериев в зависимости от числа эпох обучения.

Таблица 3.5

Критерий	Количество эпох						
	100	200	300	500	700	1000	2000
MSE	2.502	1.887	1.496	1.276	1.064	0.913	0.632
MAE	1.153	1.070	0.953	0.876	0.732	0.702	0.586
RMSE	1.582	1.373	1.223	1.129	1.0315	0.955	0.795

При средней скорости автосамосвала в исходной таблице, равной 16,8 км/час, ошибка обучения по RMSE не превышает 5%.

3.5. Выводы по главе

1. Проведен детальный анализ существующих схем реализации параллельного алгоритма Дейкстры. Рассмотрены особенности и возможности библиотеки OpenMP. Разработана оригинальная модификация параллельного алгоритма Дейкстры.
2. Тестирование алгоритма на полно связных графах с 500, 750 и 1000 вершин показало, что эффективность алгоритма выше в 2.5-4 раза по сравнению со стандартным алгоритмом Дейкстры и в 1,5 раз по сравнению известными алгоритмами, реализующими параллельную схему в модели с общей памятью.
3. Проанализированы различные модели (инструменты) решения задачи прогнозирования характеристик транспортных потоков и, в первую очередь, скоростей. Показано, что в случае, когда поведение мобильных объектов зависит от пространственно-временных характеристик, которые не стабильны во времени, а иногда, неизвестны, эффективным инструментом прогнозирования скорости может выступать граф-конволюционная нейронная сеть.

4. В ходе вычислительных экспериментов определена конкретная архитектура нейронной сети. Проведено обучение и тестирование сети GCNN, которое показало, что сеть обучается устойчиво с ростом числа эпох и достигает уровня около 1% ошибки. Ошибка на тестировании составила в среднем 7-8%, что показывает возможность использования модели для прогноза, особенно при средних и высоких скоростях движения. 10 км/час.

4. Результаты вычислительных экспериментов

4.1. Инструменты обработки и анализа данных

Разработанные программные модули, реализующие алгоритмы вычисления функции стоимости и построения оптимальных трасс перемещения карьерных самосвалов предполагают возможность импорта данных из различных источников, как структурированных, так и неструктурированных. Важным этапом является предварительная фильтрация данных, направленная на правильную интерпретацию и повышение качества данных.

В качестве исходных данных рассматривались данные о перемещении самосвалов по технологическим дорогам реального карьера. База данных содержит информацию о более чем 300 000 точек на местности, которая рассматривается в качестве набора данных для последующих экспериментов.

Также были использованы отдельные фрагменты цифровой модели карьера для отработки метода построения оптимальных трасс для роботов – самосвалов (рис.4.1).

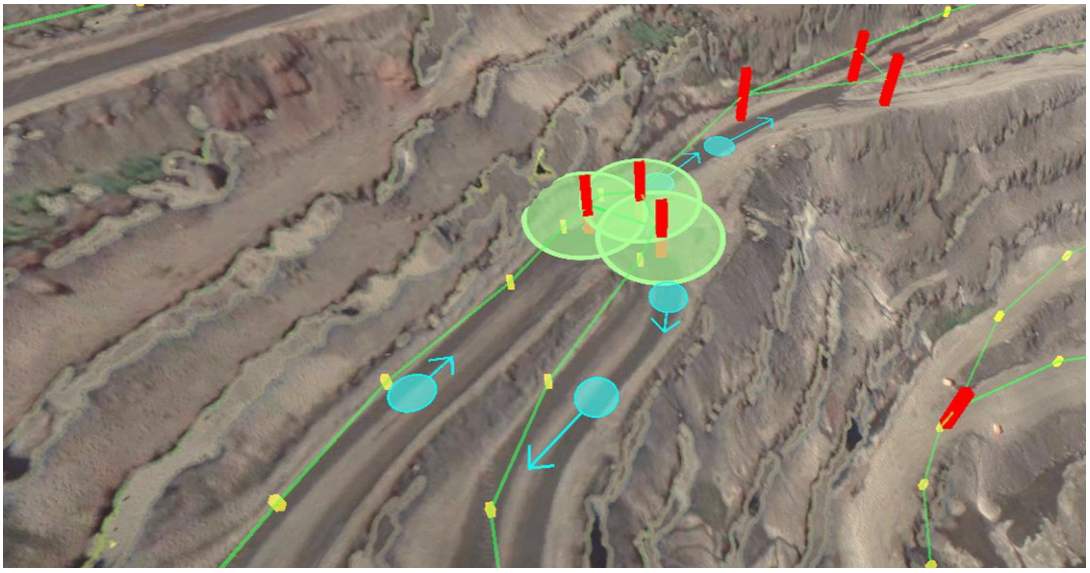


Рис. 4.1. фрагмент цифровой модели реального карьера

Процедура обработки данных включает несколько ключевых этапов, каждый из которых играет важную роль в достижении качественного анализа и интерпретации данных. Вот основные этапы этой процедуры:

1. **Импорт данных**: на этом этапе данные загружаются из различных источников, таких как базы данных, файлы, веб-сервисы или сенсоры. Важно обеспечить корректное извлечение данных для последующей обработки.
2. **Парсинг данных**: Извлеченные данные преобразуются в удобный для анализа формат. Это может включать преобразование типов данных, разбиение текстов на структурированные части и другие операции.
3. **Предварительная фильтрация данных**: на этом этапе из данных удаляются шумы, ошибки и некорректные записи. Также могут применяться методы для заполнения пропущенных данных и нормализации значений.
4. **Обеспечение качества данных**: Проверка и обеспечение качества данных являются критически важными для получения достоверных результатов анализа. Это включает валидацию данных, проверку на дубликаты, аномалии и другие несоответствия.
5. **Визуализация данных**: Важный этап, который позволяет исследовать данные и выявлять скрытые закономерности. Графики, диаграммы и другие визуальные представления помогают лучше понять структуру данных и результаты анализа.
6. **Анализ данных**: на этом этапе применяются различные методы и алгоритмы для анализа данных. Это может включать статистический анализ, машинное обучение, кластеризацию, поиск аномалий и другие подходы.
7. **Интерпретация и использование результатов**: Полученные результаты анализа интерпретируются и используются для принятия решений, оптимизации процессов или дальнейших исследований.

Для обработки и анализа данных были использованы различные инструменты и технологии, включая:

- **Языки программирования**: Excel, Python (Pandas, PyTorch, NumPy, SciPy, Scikit-learn, Matplotlib и Seaborn), R, SQL, и т.д.
- **Платформы и базы данных**: Google colap**,** Jupyter Notebook, Apache Hadoop, Apache Spark, MongoDB, PostgreSQL и др.
- **Облачные сервисы**: AWS, Google Cloud, Microsoft Azure.

Рис. 4.2 Процедура обработки данных.

4.2. Общая схема проведения вычислительных экспериментов

Процессы вычислений для отработки метода построения оптимальных трасс с использованием разработанных алгоритмов состоит из следующих основных этапов:

1. Выбор упрощенной схемы карьерных дорог, для более наглядного представления понятия маршрут, как это сделано, например, на рис. 2.2. Построение на основе статистических данных и экспертной ручной

корректировки ручногоОцифровка атомных элементов: после импорта данных и для того, чтобы регламентировать доступ к каждому элементу, атомные элементы оцифровываются, как показано на рисунке 4.3(а).

2. Моделирование транспортной инфраструктуры:

- Граф представление: транспортная сеть карьера представлена в виде графа, где узлы (вершины) соответствуют основным точкам (например, местам погрузки и разгрузки, перекресткам), а ребра соответствуют участкам дорог между этими точками, как показано на рис. 4.3(с).
- Веса ребер и элементарных элементов: каждому ребру присваивается вес, который отражает стоимость прохождения данного участка дороги. эта стоимость может учитывать различные факторы, такие как риск аварии, расход топлива и механическая нагрузка на автомобиль. А также присвоить вес каждому атомарному элементу как показано на рис. 4.3(б).
- Матрица смежности: создать матрицу смежности для определения ребер между атомарными элементами и формирования структуры цифровой модели, размерность матрицы (n*n), где n - количество атомарных элементов (вершин), элементы матрицы (0,1), где 0 означает отсутствие ребра между двумя элементами, а 1 - наличие ребра между двумя элементами, соответствующими индексу матрицы, как показано на рис. 4.4 (с).

3. Определение стоимости прохождения маршрута:

- Целевая функция: разрабатывается дискретная функция стоимости, основанная на цифровых моделях дорожной инфраструктуры. Эта функция включает такие компоненты, как длина дороги, состояние дорожного покрытия, уклоны, наличие поворотов и выбоин, а также другие характеристики.
- Анализ данных: используются реальные данные из карьера для калибровки и валидации модели, что позволяет точно оценивать стоимость маршрутов.

Суммируя вышесказанное, перечислим последовательность рабочих процедур:

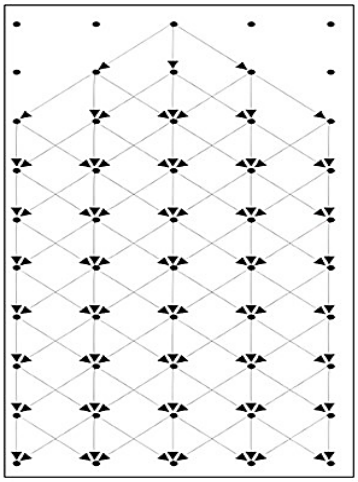
1. Выбор упрощенной схемы карьерных дорог, для более наглядного представления понятия маршрут.
2. Построение на основе статистических данных и экспертной ручной корректировки мульти-фрактальных цифровых моделей карьерных дорог, в соответствии с упрощенной топологической схемой.
3. Построение соответствующей структурной матрицы и ациклического графа (рис. 4.3), на основе которого осуществляется построение оптимальных трасс.

a0,0	a0,1	a0,2	a0,3	a0,4
a1,0	a1,1	a1,2	a1,3	a1,4
a2,0	a2,1	a2,2	a2,3	a2,4
a3,0	a3,1	a3,2	a3,3	a3,4
a4,0	a4,1	a4,2	a4,3	a4,4
a5,0	a5,1	a5,2	a5,3	a5,4
a6,0	a6,1	a6,2	a6,3	a6,4
a7,0	a7,1	a7,2	a7,3	a7,4
a8,0	a8,1	a8,2	a8,3	a8,4
a9,0	a9,1	a9,2	a9,3	a9,4

(а)

0.4	0.1	0.9	1.0	0.8
0.3	0.2	0.5	1.0	0.7
0.9	0.7	0.3	0.1	0.8
0.3	1.0	0.7	0.2	0.4
0.7	0.9	0.1	0.9	0.1
0.6	0.3	0.6	0.5	0.4
0.6	0.1	0.7	0.4	0.2
0.6	0.8	1.0	0.3	0.5
0.4	0.3	0.8	0.1	1.0
0.1	0.2	0.3	1.0	0.8

(б)



(с)

Рис. 4.3 (а) Оцифровка элементарных фрагментов (б) Фрагмент цифровой карты (в) Ациклический граф

4. Вычисление прогнозных значений скорости по всей поверхности дорожного полотна с последующей привязкой с центру соответствующих фрагментов (рис.4.4).
5. Вычисление дискретных функций стоимости на базе цифровых моделей, учитывающих рейтинговую оценку качества дорожного полотна в смежных фрагментах и потенциальные скоростные режимы.
6. С использованием параллельного алгоритма Дейкстры на основе интегральных функций стоимости осуществляется построение

оптимальной трассы для одного робота самосвала или последовательно для нескольких мобильных объектов, с учетом динамического перестроения цифровых моделей карьерных дорог.

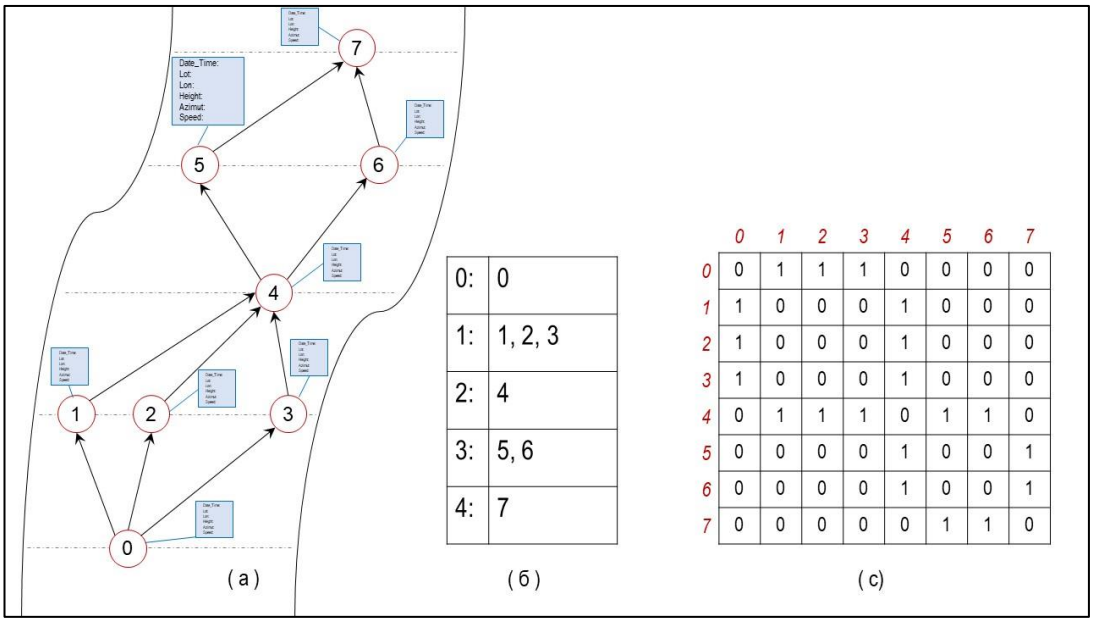


Рис. 4.4 (а) Фрагмент карьерной дороги, (б) Список атомных элементов, (с) Матрица смежности

4.3. Результаты работы алгоритма прогнозирования скорости.

В качестве ядра алгоритма прогнозирования скорости выступают нейросетевая модель (ST-GCN). Алгоритм оценивался на предмет его эффективности в прогнозировании скорости путем сравнения потерь при обучении нейросетевой модели и потерь при тестировании в течение 100 эпох. Тестовый набор данных составлял 20 % от общего числа образцов. В таблице 4.1 приведены требования для эксперимента 5.

Таблица 4.1

Требования	Детали
Процессор	Intel (R) Core (TM) i7-8550U, 1.80 GHz, 8 cores, 8 threads
Память	16 GB RAM

Операционная система	Windows 11, 64-bit architecture
Язык программирования	Python
Библиотека параллелизма	NetworkX, Pandas, PyTorch, NumPy,DGL
Среда разработки	Google colap
Измерения	MSE, MAE и RMSE, в единицах измерения
Экспериментальная установка	Desktop computer with above specifications

После обучения модели можно провести сравнение для оценки точности прогнозов модели путем построения графиков фактических и ожидаемых скоростей для их визуального сравнения. В таблице 4.2 и на рис. 4.5 показано сравнение фактических и ожидаемых скоростей в зависимости от количества эпох.

Таблица 4.2

No	**Lat**	**Lon**	**Скорость (реальная), км/час**	**Скорость (прогнозная), км/час**	**Ошибка,%** $\frac{\lvert y-\hat{y}\rvert}{y}$ *100%
1	0.782826	-0.902025	1.749111	1.486150	15.033956 %
2	-0.877050	0.964041	5.813222	4.828748	16.935084 %
3	1.150575	-0.236792	9.054221	7.788389	13.980575 %
4	0.935285	-0.270118	8.179666	7.970336	2.559151 %
5	-0.193942	1.586494	1.440444	1.227833	14.760110 %
6	-0.269474	0.703870	7.768110	7.532421	3.034063 %
7	-0.103545	0.412378	2.057778	1.915077	6.934702 %
8	-0.470909	0.606215	6.996444	6.081789	13.073139 %
9	0.549110	-0.031988	4.887222	4.510302	7.712352 %
10	0.313084	0.110418	5.607444	4.873058	13.096620 %
11	0.448801	-0.100507	7.305110	6.504719	10.956593 %
12	0.837093	-0.113224	6.224777	5.841027	6.164879 %
13	0.504416	-0.143216	7.253666	6.583998	9.232136 %
14	-0.034267	0.443232	4.938666	5.009165	1.427482 %
15	0.260452	-2.566743	7.407999	6.870983	7.249140 %
16	-0.296897	0.729057	6.842111	6.522726	4.667930 %
17	-2.670430	-2.037634	6.739222	6.805934	0.989918 %
18	0.078646	1.067617	6.018999	4.713325	21.692550 %
19	-1.186539	-2.337686	6.739222	7.067701	4.874142 %
20	0.915560	-1.042609	4.681444	3.375338	27.899649 %
21	0.590051	-0.059581	7.922444	7.343453	7.308229 %
22	-3.202521	-1.796586	6.018999	5.653193	6.077522 %

23	-0.747731	0.885154	5.041555	4.786451	5.060031 %
24	1.122576	-1.116056	4.681444	4.265652	8.881701 %
25	-0.007085	1.022625	4.527111	4.958230	9.523056 %
26	0.401269	0.116598	3.446777	3.006692	12.768015 %
27	0.047904	1.172820	5.710333	4.244087	25.677061 %
28	0.291579	1.513157	4.064111	3.386454	16.674174 %
29	1.063257	-0.211618	9.414333	9.012770	4.265441 %
30	-0.261296	0.767052	5.401666	5.127311	5.079080 %

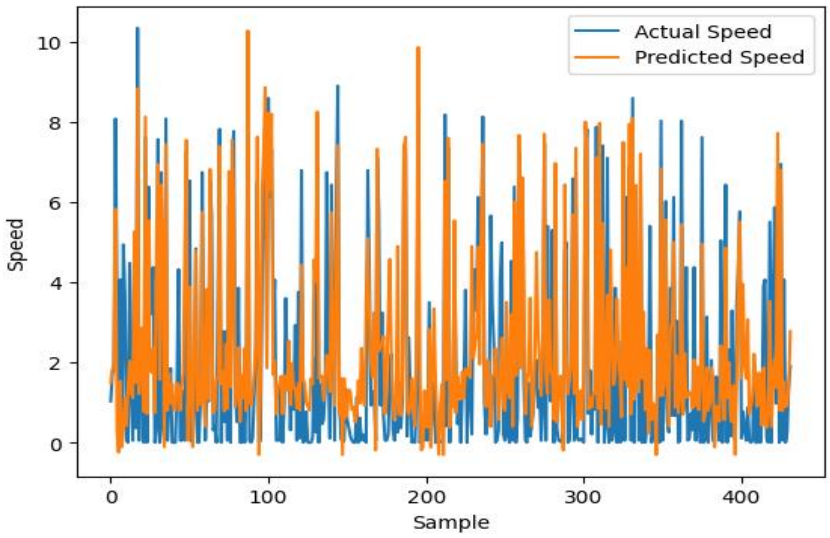


Рис. 4.5 показывает сравнение фактических и ожидаемых скоростей

Из таблицы 4.1 видно, что точность прогнозирования весьма приемля и даже на низких скоростях (менее 10 км/час) не превышает в среднем 7-8%. Сравнение потерь при обучении и тестировании по эпохам также демонстрирует устойчивость и работоспособность модели (рис. 4.6.).

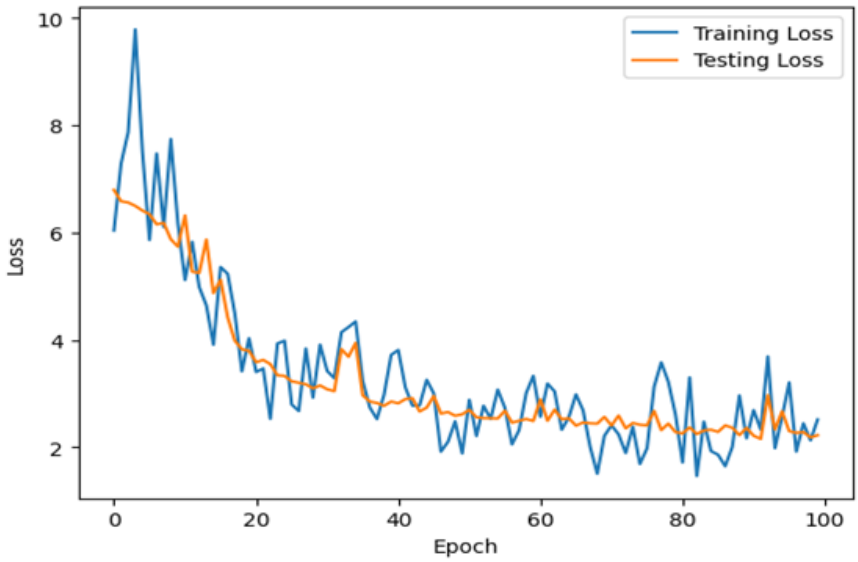


Рис. 4.6 сравнение потерь при обучении и тестировании по эпохам.

Модель Spatio-Temporal Graph Convolutional Network (ST-GCN) была оценена на основе прогнозирования скорости автономного самосвала, что показало впечатляющие результаты. Основные выводы были сделаны путем сравнения фактических и прогнозируемых скоростей, а также анализа кривых потерь при обучении и тестировании в течение 100 эпох.

Анализ точности прогнозов

Визуализация результатов продемонстрировала тесное соответствие между фактическими и прогнозируемыми кривыми скорости, что указывает на высокую точность модели ST-GCN в прогнозировании скорости в диапазоне от 0 км/ч до 12 км/ч. Близкое расположение кривых подтверждает, что модель эффективно улавливает пространственно-временные зависимости в данных, что позволяет ей делать надежные и точные прогнозы. Небольшое расхождение между фактическими и прогнозируемыми значениями свидетельствует о способности модели хорошо обобщать информацию на тестовом наборе данных.

Важность точности прогнозов в реальных приложениях

Для реальных приложений, таких как автономное вождение или спортивная аналитика, точность прогнозов скорости имеет критическое значение. Хотя незначительное занижение скорости может показаться несущественным, в критических приложениях такие расхождения могут привести к серьезным последствиям. Поэтому устранение любых предвзятостей в прогнозах модели является важным шагом к повышению её надежности и применимости в практических сценариях.

Сходимость кривых потерь

Кривые потерь при обучении и тестировании, визуализированные в течение 100 эпох, показывают последовательное снижение и близкое совпадение, что указывает на успешное обучение модели. Этот факт свидетельствует о том, что модель не страдает от переобучения. Переобучение было бы очевидно при значительном расхождении между кривыми потерь при обучении и тестировании, где потери при обучении продолжали бы

уменьшаться, а потери при тестировании увеличивались или достигали плато. Близкое совпадение этих кривых демонстрирует надежный процесс обучения и хорошо отрегулированную модель.

Метрики оценки точности

Для оценки точности прогнозирования модели использовались несколько показателей:

1. **Средняя квадратичная ошибка (MSE)**: MSE равен 1,62, что показывает среднюю квадратичную разницу между прогнозируемыми и фактическими значениями скорости. Это значение указывает на способность модели минимизировать ошибки в своих прогнозах.
2. **Среднеквадратичная ошибка (RMSE)**: RMSE равен 1,24, что является показателем средней величины ошибок в прогнозах модели. Более низкое значение RMSE по сравнению с MSE указывает на то, что ошибки в целом находятся в разумном диапазоне относительно масштаба значений скорости.
3. **Средняя абсолютная ошибка (MAE)**: MAE равен 1,2, что представляет собой среднюю абсолютную разницу между прогнозируемыми и фактическими значениями скорости. Низкое значение MAE подтверждает, что прогнозы модели в среднем относительно близки к фактическим значениям.

Модель ST-GCN продемонстрировала высокую точность и надежность в прогнозировании скорости автономного самосвала, что подтверждается близким совпадением фактических и прогнозируемых значений, а также сходимостью кривых потерь при обучении и тестировании. Метрики оценки точности, такие как MSE, RMSE и MAE, также указывают на способность модели минимизировать ошибки. Эти результаты свидетельствуют о том, что модель хорошо подходит для реальных приложений, где точное прогнозирование скорости имеет решающее значение.

4.4. Исследование метода оптимального планирования трасс с использованием параллельного алгоритма Дейкстры.

Вычисление стоимости оптимальных маршрутов на основе значений атомных весов составляющих элементов маршрута и использования дискретной функции стоимости, элементы моделировались (1000 элементов) на основе цифровой модели, а маршрут определялся с помощью алгоритма Дейкстры.

Описание эксперимента:

Для проведения эксперимента по вычислению стоимости оптимальных маршрутов в карьерных условиях с использованием дискретной функции стоимости и алгоритма Дейкстры, необходимо выполнить следующие шаги:

1.Описание цифровой модели

Для примера а был рассмотрен участок карьерной дороги размером [50м х 1400м]. Был выбран минимальный масштаб цифровой модели с квадратными атомарными элементами «плитками» (1м х 1м). Функции стоимости вычислялись двумя различными способами:

-в первом случае использовалась грубая оценка функции стоимости, когда функция принимала только три значения: 0,1 - «идеальный вариант перемещения», 0,5 – «удовлетворительно», 1,0 – «перемещение запрещено», и при этом не учитывались скоростные режимы;

-во втором случае использовалась цифровая модель с детализированными оценками.

На рис. 4.7 представлены трассы, построенные с помощью разработанного алгоритма для участка с максимальной шириной 50 м и длиной 200 м. Видно, что трасса два имеет более сложный контур, однако суммарное значение функции стоимости в этом случае оказалось существенно меньше: F1 = 88,4 против F2 = 69,7.

Очевидно, что в реальных условиях самосвал не может двигаться непосредственно через опорные точки, определенные трассой 2. Однако,

использование сглаженной по этим точкам траектории обеспечит реальное снижение ударных нагрузок и, с большой вероятностью, позитивно скажется на экономии топлива.

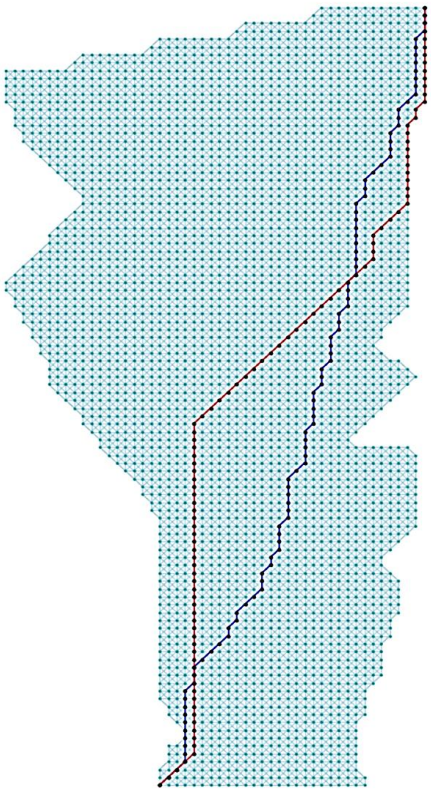


Рис. 4.7. Два варианта трассы: красный цвет: грубая цифровая модель; синий цвет: точная цифровая модель.

Для моделирования и планирования маршрутов использовался язык программирования Python и библиотека NetworkX, которая предоставляет инструменты для работы с графами.

Далее представлено краткое описание вычислительного эксперимента:

Импорт необходимых библиотек:

```
import networkx as nx
import matplotlib.pyplot as plt
```

Создание графа и добавление вершин и ребер:

```
G = nx.Graph()
# Добавление вершин (например, 1000 атомарных элементов)
for i in range(1000):
```

```
    G.add_node(i)
# Добавление ребер с весами
for i in range(999):
    G.add_edge(i, i+1, weight=atomic_weights[i])
```

Поиск оптимального маршрута с использованием алгоритма Дейкстры:

```
start_node = 0
end_node = 999
shortest_path = nx.dijkstra_path(G, source=start_node, target=end_node)
shortest_path_length = nx.dijkstra_path_length(G, source=start_node,
target=end_node)
print("Кратчайший путь:", shortest_path)
print("Длина кратчайшего пути:", shortest_path_length)
```

Визуализация графа и маршрута:

```
pos = nx.spring_layout(G)
nx.draw(G, pos, with_labels=True, node_size=50)
path_edges = list(zip(shortest_path, shortest_path[1:]))
nx.draw_networkx_nodes(G, pos, nodelist=shortest_path, node_color='r')
nx.draw_networkx_edges(G, pos, edgelist=path_edges, edge_color='r',
width=2)
plt.show()
```

При реализации параллельного алгоритма Дейкстры на графах с различным количеством вершин, число которых в данном случае зависит от размера моделируемого участка дороги участка дороги (10 тысяч, 20 тысяч, 40 тысяч, 60 тысяч и 70 тысяч) время выполнения в зависимости от количества ядер существенно менялось, однако, оставалось в пределах допустимого (таблица 4.3) .

Таблица 4.3

Количество вершин	Первая оптимальная трасса				Вторая оптимальная трасса			
	Количество ядер				Количество ядер			
	1	2	4	8	1	2	4	8
10000	0,470	0,244	0,170	0,109	0,417	0,338	0,154	0,112
20000	1,569	0,858	0,562	0,394	1,651	1,014	0,676	0,558

40000	6,306	3,572	2,230	1,664	6,225	3,629	2,320	1,845
60000	16,485	8,939	6,601	5,004	16,897	9,244	6,421	5,497
80000	33,543	17,433	11,370	8,934	39,328	19,565	14,432	10,283

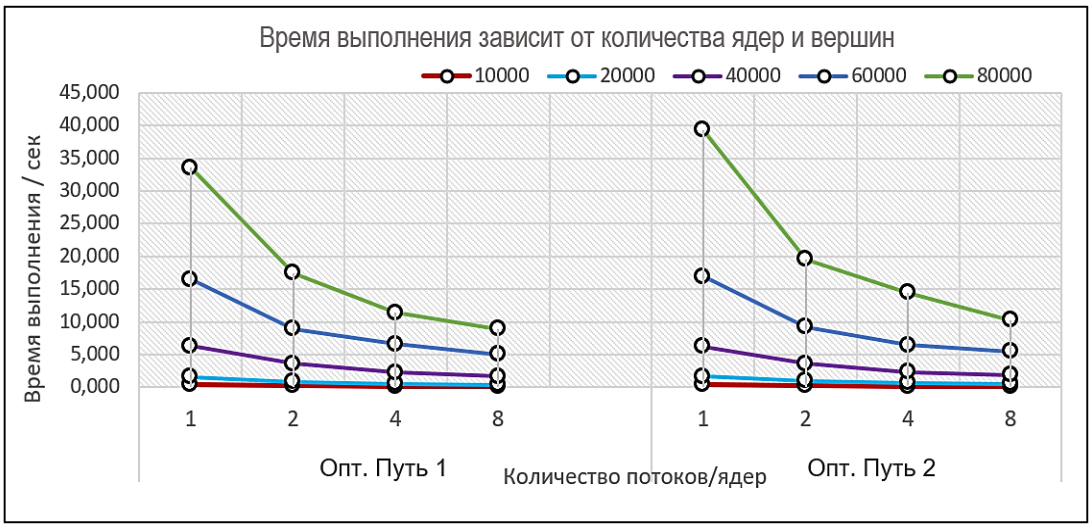


Рис. 4.8 Время поиска оптимальной трассы в зависимости от числа ребер

График и таблица наглядно иллюстрируют, что даже при работе с ациклическим графом, содержащим 60 - 70 тысяч вершин, что соответствует при использовании минимального масштаба реальному маршруту в карьере длиной от 2-х до 5-ти километров, использование параллельного режима с разделением памяти между 4 или 8 ядрами позволяет построить или перестроить маршрут в течение 5 – 15 секунд, что вполне приемлемо с точки зрения безопасности перемещения автономных самосвалов.

Эксперимент показал, что использование алгоритма Дейкстры в сочетании с цифровой моделью карьерных дорог и атомными весами элементов маршрута позволяет эффективно определять оптимальные маршруты для автономных карьерных транспортных средств, движущихся навстречу друг другу. Это помогает улучшить производительность, снизить расход топлива и сократить затраты на обслуживание.

Следует отметить, что использование параллельного алгоритма Дейкстры позволяет получить хорошие результаты при использовании стандартных

вычислительных средств. В экспериментах использовался настольный компьютер, используемый для тестирования, оснащен процессором Intel Core i7-8550U (1,80 ГГц, 8 ядер, 8 потоков) и 16 ГБ оперативной памяти. Используемая модель общей памяти обеспечивает последовательное начало выполнения алгоритма с основного (главного) потока. Когда достигается параллельная область, выполнение продолжается параллельно с использованием заданного количества потоков (контролируется переменной num_threads). Значение num_threads, равное единице, обеспечивает последовательное выполнение, а значения от 2 до 8 – параллельное. Производительность измеряется временем выполнения в секундах. В таблице 4.4 приведена сводная информация по эксперименту.

Таблица 4.4

Требования	Детали
Процессор	Intel (R) Core (TM) i7-8550U, 1.80 GHz, 8 cores, 8 threads
Память	16 GB RAM
Операционная система	Windows 11, 64-bit architecture
Язык программирования	C++
Библиотека параллелизма	OpenMP
Среда разработки	Microsoft Visual Studio 2022
Конфигурация выполнения	Number of threads: 1 (sequential), 2 to 8 (parallel)
Измерения	Execution time in seconds
Экспериментальная установка	Desktop computer with above specifications

В Таблице 4.5 представлено наилучшее время выполнения для каждого количества потоков для последовательного выполнения алгоритма Дейкстры (один поток) и параллельно с библиотеками OpenMP, отмечая, что результаты были выбраны в несколько раз лучше, чем выполнение кода. Код выполняется

в одной и той же системной среде, а источник входных данных генерируется функцией Random. В данной работе для входной матрицы смежности используется в общей сложности три набора данных. То есть, для графа используются 500 вершин, 750 вершин и 1000 вершин. После выполнения кода мы можем получить результат, который представляет собой продолжительность в секундах. Для параллельных вычислений OpenMP для выполнения кода используется 1, 2, 3,4,5,6,7 and 8 процессоров (количество вершин должно быть больше количества процессоров).

Таблица 4.5

Количество вершин	Количество ядер							
	1	2	3	4	5	6	7	8
500	165,5	146,4	91,4	74,0	62,6	52,2	51,7	50,2
750	380,2	251,7	212,4	159,2	132,2	115,0	112,9	110,4
1000	1510,3	1340,7	866,56	654,0	544,7	461,9	421,4	375,5

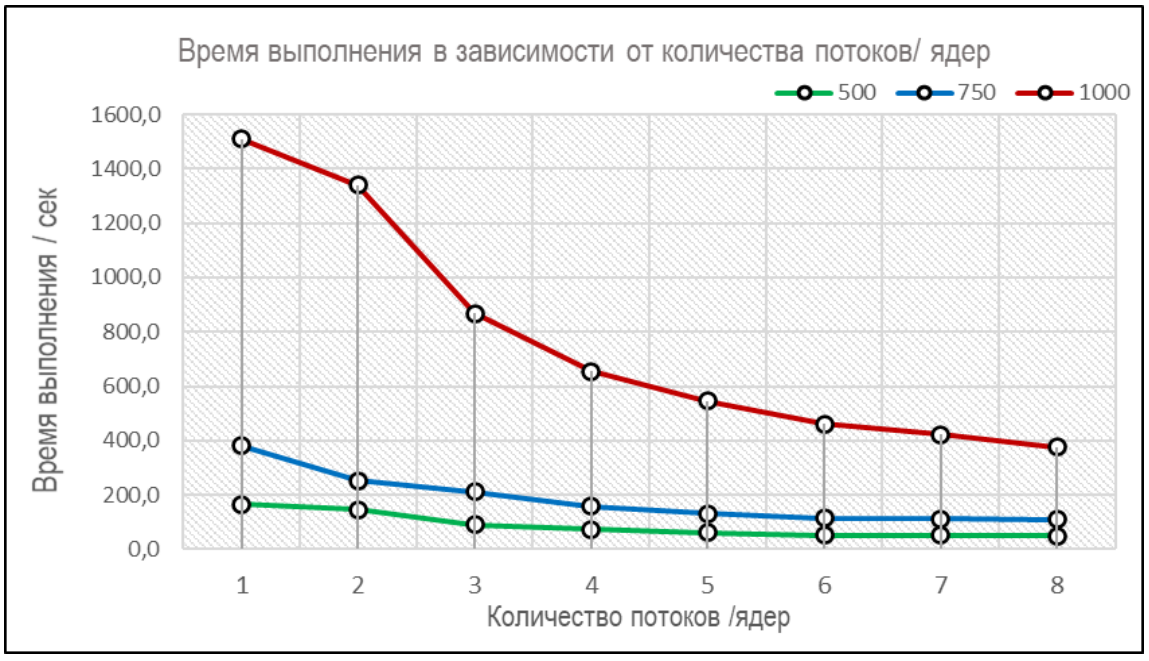


Рис. 4.9 – Время выполнения в зависимости от количества ядер

В Таблице 4.6 представлено ускорение для каждого количества ядер

Таблица 4.6

Количество вершин	Количество ядер							
	1	2	3	4	5	6	7	8
500	1,00	1,13	1,81	2,24	2,64	3,17	3,20	3,30
750	1,00	1,51	1,79	2,39	2,88	3,31	3,37	3,44
1000	1,00	1,13	1,74	2,31	2,77	3,27	3,58	4,02

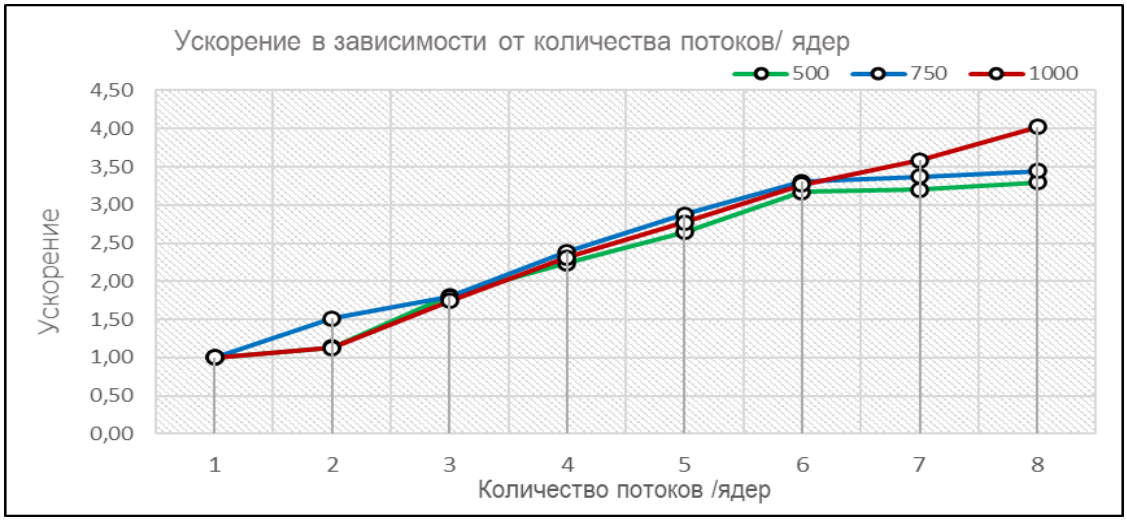


Рис. 4.10 Ускорение в зависимости от количества ядер.

В таблице 4.7 представлена оценка эффективности для каждого количества ядер.

Таблица 4.7

Количество вершин	Количество ядер							
	1	2	3	4	5	6	7	8
500	1,00	0,57	0,60	0,56	0,53	0,53	0,46	0,41
750	1,00	0,76	0,60	0,60	0,58	0,55	0,48	0,43
1000	1,00	0,56	0,58	0,58	0,55	0,54	0,51	0,50

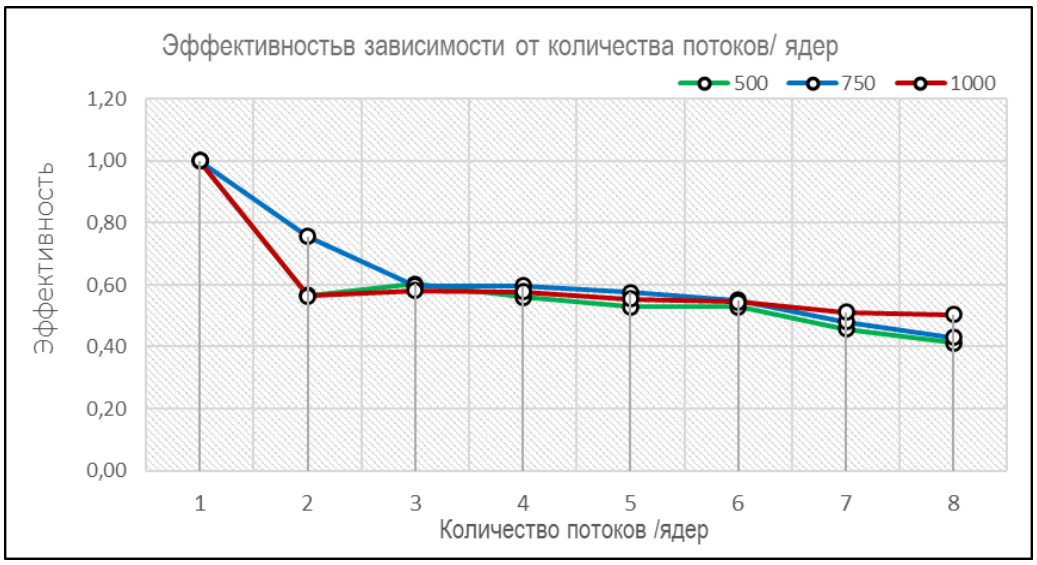


Рис. 4.11 Эффективность в зависимости от количества потоков.

При реализации параллельного алгоритма Дейкстры с использованием модели разделяемой памяти и OpenMP и проведении экспериментов с использованием больших графов. Было измерено время выполнения последовательной и параллельной версий алгоритма и проведено сравнение их производительности.

Результаты эксперимента показали, что параллельная версия алгоритма, использующая общую память и OpenMP, превзошла последовательную версию для больших графов. Сравнивая результаты, мы обнаруживаем, что время параллельного выполнения двух потоков на двух ядрах сокращается примерно на (22%, 44%, 22%) от времени последовательного выполнения когда количество узлов равно (500, 750, 1000) соответственно и на (70%, 72%, 76%) при выполнении восемью потоками на восьми ядрах когда количество узлов равно (500, 750, 1000) соответственно, как показано на рис. 4.9., 4.10

Для оценки эффективности параллельного алгоритма Дейкстры на реальных данных использовался реальный набор данных, представляющий информацию об одном из карьеров Сибири - России. Рассматривалась часть поверхности карьерной дороги, для моделирования которой использовалась цифровая модель. Количество компонентов модели составляет 70 000 точек (узлов), с соответствующими количественными значениями для каждого

начального участка. В процессе формирования графовой модели используются пространственная матрица (часть цифровой модели), информационная матрица смежности и сама матрица смежности размерности [V, V].

В таблице 4.8 показано время выполнения для разного количества параллельных потоков. В случае одного процессора речь идет о последовательном выполнении алгоритма Дейкстры. Параллельное выполнение осуществляется с помощью библиотеки OpenMP. Код реализуется в том же системном окружении. На рис. 4.10 показано время выполнения для каждого количества ядер.

Таблица 4.8

Количество потоков	Количества ядер							
	1	2	3	4	5	6	7	8
Время выполнения/сек	26,124	16,249	12,171	10,257	8,895	7,485	6,815	5,733

Рис. 4.12 - Время выполнения для каждого количества ядер.

В соответствии со схемой параллельной работы алгоритма Дейкстры, рассчитать ускорение и эффективность в каждом вычислительном

эксперименте. В таблице 4.9 приведены значения ускорения для каждого случая, а ускорение для всех случаев можно увидеть на рис. 4.11.

Таблица 4.9

Количество потоков	Количества ядер							
	1	2	3	4	5	6	7	8
Ускорение	1,608	2,146	2,547	2,937	3,490	3,833	4,557	1,608

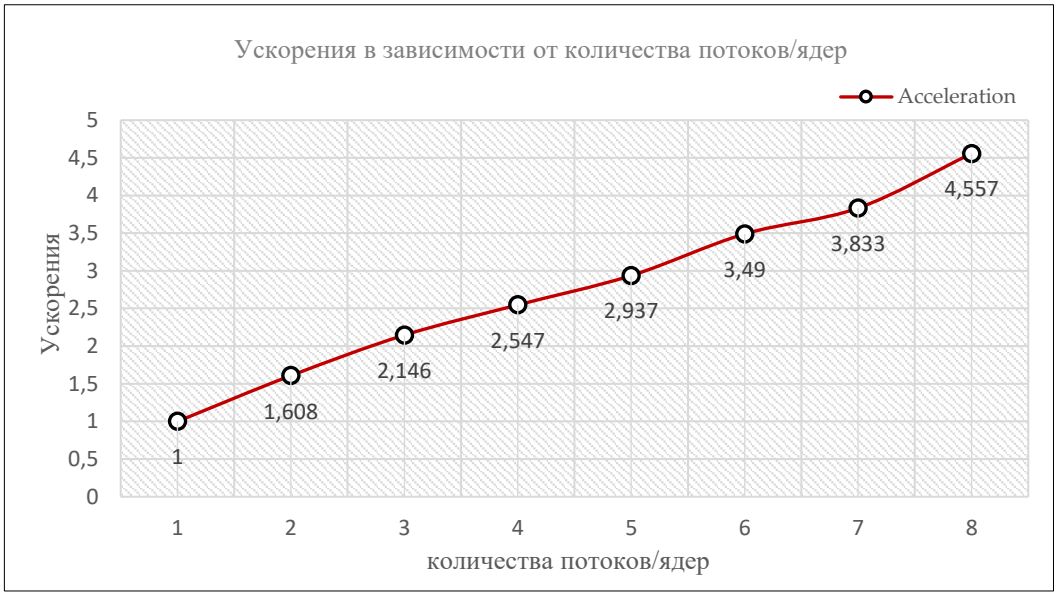


Рис. 4.13 - Ускорения для каждого количества потоков/ядер

Основываясь на значениях ускорения, можно рассчитать эффективность алгоритма, как показано в таблице 4.10 и на рис. 4.14.

Таблица 4.10

Количество потоков	Количества ядер							
	1	2	3	4	5	6	7	8
Эффективность	1,608	0,804	0,715	0,637	0,587	0,582	0,548	0,570

Рис. 4.14- Эффективность для каждого количества ядер

При реализации параллельного алгоритма Дейкстры с использованием модели разделяемой памяти и OpenMP и проведении экспериментов с использованием больших графов. Было измерено время выполнения последовательной и параллельной версий алгоритма и проведено сравнение их производительности. Результаты эксперимента, приведенные в таблице 4.8, показали, что параллельная версия алгоритма, использующая общую память и OpenMP, превзошла последовательную версию для больших графов. Сравнивая результаты, что время параллельного выполнения двух потоков на двух ядрах сокращается примерно на 38% от времени последовательного выполнения и на 80% при выполнении восемью потоками на восьми ядрах, как показано на рис. 4.12. Результаты, приведенные в таблице 4.9, показали достижение повышенной производительности процессора в целом, но во всех случаях в разных пропорциях, как показано на рис. 4.13, и это считается хорошим показателем полной производительности процессора. В таблице 4.10 и на рис. 4.14 результаты показывают эффективность алгоритма для всех случаев, и можно наблюдать изменение эффективности алгоритма по мере увеличения ускорения процессора. Достижение наивысшей эффективности алгоритма и высочайшей производительности процессора может быть достигнуто при использовании принципа параллельных вычислений только в

идеальных случаях. Согласно полученным результатам, эксперимент соответствует требованиям к времени выполнения алгоритма и эффективности, а также приемлемому уровню производительности процессора. Количество процессорных ядер напрямую влияет на то, насколько быстро выполняется алгоритм. Тип ядер, количество ядер, другие запущенные приложения, фоновые процессы и т.д. — все это может повлиять на скорость выполнения кода.

Дальнейшие эксперименты с использованием цифровых моделей проводились с целью проверки пригодности метода для решения более сложных задач планирования трасс для нескольких роботов-самосвалов. На примере участка карьерной дороги с большим числом «препятствий» - зон, перемещение по которым невозможно, были в последовательном режиме построены трассы для двух самосвалов, которые движутся навстречу друг-другу. Поскольку был выбран достаточно короткий участок, время планирования оказалось весьма незначительным (таблица 4.11).

Таблица 4.11

Маршрут	Стоимости оптимальных маршрутов (F)	Время планирования /сек
Первый опт. маршрут	12.9246117974981	0.6807565689086914
Второй опт. маршрут	15.5967477524977	0.4769802093505859

Однако сами построенные трассы наглядно иллюстрируют возможность использования динамически подстраиваемой цифровой модели для группового планирования перемещения (рис. 15).

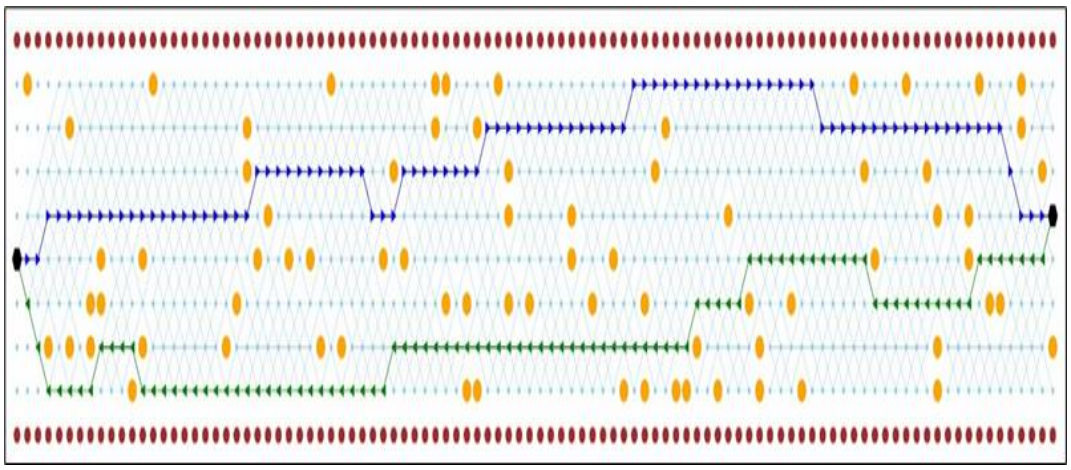


Рис. 4.15 Последовательно построенные оптимальные трассы для перемещения роботов-самосвалов навстречу друг-другу.

Разработанный метод был также протестирован на задаче планирования нескольких параллельных трасс внутри одного маршрута. Проведенные вычислительные эксперименты подтвердили возможность использования метода для этой залачи. Был рассмотрен реальный маршрут, включающий ребра $l_2 – l_5 – l_{10}$ (рис.2.2). Алгоритм находит одну за одной оптимальные, то есть соответствующие минимумам функций стоимости трассы. После нахождения каждой очередной трассы соответствующая матрица смежности корректировалась: из не удалялись ребра, которые были частью уже проложенной трассы, а также соответствующие инцидентные вершины. Очевидно, что формальную оценку предельного количества правильно построенных трасс (рис. 4.16) с помощью разработанного алгоритма, получить достаточно трудно, так как, эта величина напрямую зависит от размерности и содержимого структурной матрицы цифровой модели.

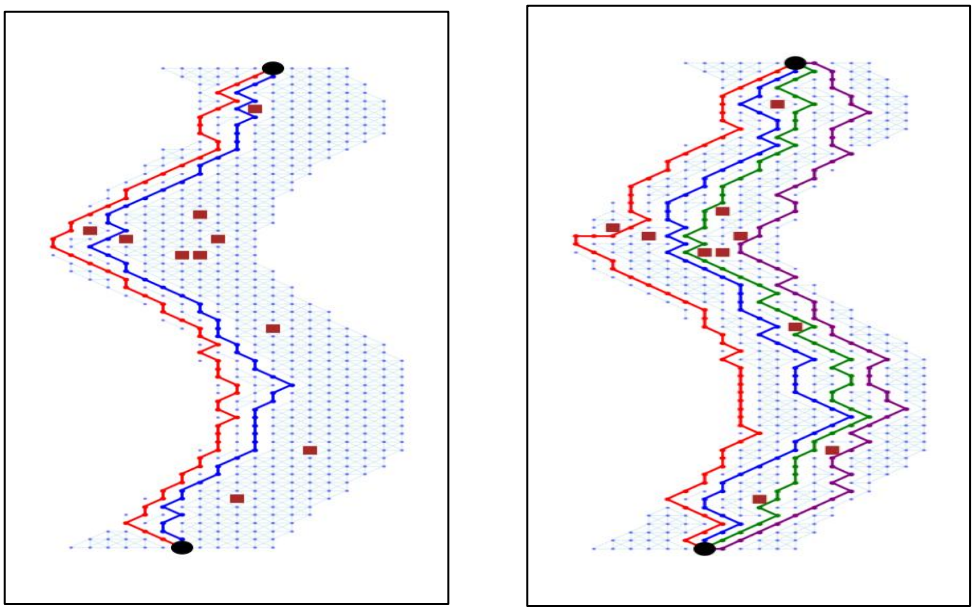


Рис. 4.16 Иллюстрация использования метода для одновременного планирования нескольких параллельных трасс.

4.5. Выводы по результатам вычислительных экспериментов

Проведенные эксперименты с использованием цифровых мультифрактальных моделей реальных карьерных дорог позволили получить количественные оценки работы алгоритмов и метода в целом и сформулировать ряд выводов:

1. Использование в качестве модели описания поля возможных скоростей граф-конволюционной сети с подобранными характеристиками заданной архитектуры позволили построить алгоритм прогноза на основании обработки большого количества траекторий автосамосвалов, обеспечивающий точность прогнозирования с ошибкой в среднем не выше 7-8%. При этом с увеличением средней скорости движения самосвалов этот показатель снижается.
2. Модифицированный параллельный алгоритм Дейкстры обеспечивает серьезное снижение в 2-4 раза в зависимости от структуры пространства состояний (степень разреженности матрицы смежности) время поиска оптимального пути, что позволяет использовать его в задаче оперативного планирования трасс перемещения автономных самосвалов.

3. Использование разработанных алгоритмов позволяет эффективно реализовать метод оптимальной маршрутизации на основе построения трасс минимальной стоимости не только для единичного робота – самосвала, но и для группы таких машин, даже в случае перемещения их по одним и тем же карьерным дорогам.

ЗАКЛЮЧЕНИЕ

1. На основе проведенного анализа современного состояния вопроса организации работы ГТК, а также существующих сегодня алгоритмов и систем управления утверждается, что проведенное диссертационное исследование является актуальным, так как:
   - цифровая трансформация и интеллектуализация всех производственных процессов сегодня является основным трендом развития горных предприятий;
   - наиболее отчетливо эти процессы наблюдаются в организации транспортных процессов при ведении открытых горных работ, где значительные успехи достигнуты в разработке автономных карьерных самосвалов;
   - сегодня существует широкий спектр моделей и инструментов для разработки эффективных алгоритмов оптимального планирования поведения индустриальных колесных роботов;
   - отсутствуют очевидные и работоспособные решения в сфере планирования трасс и траекторий для перемещения нескольких автономных карьерных самосвалов по технологическим дорогам с учетом необходимости обеспечения их согласованной и безопасной работы.
2. Показано, что цифровые мульти фрактальные модели, транспортный инфраструктуры карьера, основанные на разбиении всей дорожной поверхности карьера на множество однородных элементарных фрагментов (атомарных элементов - блоков) фиксированной формы, могут быть взяты в качестве основы для решения вопросов оптимального планирования перемещения автономных самосвалов по карьерным дорогам. При этом вводится понятие трассы, как множества опорных точек (центров атомарных элементов)

3. Разработана процедура построения эмпирической функции стоимости, на основе которой можно проводить сравнительную оценку трасс, по которым могут перемещаться автономные самосвалы. Данная функция позволяет выбирать конкретное направление движения робота, строится из расчета минимизации потерь, связанных с движением по некомфортным (ямы, рытвины, колеи, камни) участкам дорожного полотна и максимальной стабилизации их скоростных режимов и является основой для формирования интегрального критерия выбора оптимальной трассы.
4. На основании детального анализа современных подходов к оптимизации движения колесных роботов в динамической среде, отмечены специфические особенности, отличающие работу карьерных роботов-самосвалов: с одной стороны, они функционирую в достаточно регулярной среде, где не требуется ежесекундно принимать решения о смене направления движения, с другой стороны, периодически и достаточно внезапно возникающие нештатные ситуации требуют принятие решений в рамках жестких технологических регламентов с достаточной оперативностью. В этой связи выбор алгоритма Дейкстры в качестве инструмента реализации предложенного метода объясняется его многократно доказанной надежностью.
5. Наряду с этим подчеркивается, что в случае поиска решений в графах большой размерности алгоритм Дейкстры сталкивается с определенными проблемами в части скорости поиска оптимальных решений. В этой связи была разработана оригинальная модификация параллельного алгоритма Дейкстры по схеме с общей памятью в рамках библиотеки OpenMP, для поиска оптимальных путей (трасс) в ациклических графах большой размерности.
6. Предлагается при формировании функции стоимости учитывать не только характеристики дорожного полотна, но и возможные скоростные режимы перемещения между отдельными фрагментами. С этой целью разработан алгоритм прогнозирования скорости робота-самосвала в любой точке

дорожного полотна, который реализован на базе оригинальной нейронной сети с граф-конволюционной архитектурой.

7. Разработанный метод построения оптимальных маршрутов перемещения автономных самосвалов с учетом конкретных трасс движения по технологическим дорогам, включающий процедуры построения функции стоимости, а также два оригинальных алгоритма, реализован в виде программных модулей, которые при незначительной технической настройке могут быть использованы в задачах управления ГТК, а также при решении различных индустриальных логистических задач.

СПИСОК ЛИТЕРАТУРЫ

[1] Ржевский В.В. Открытые горные работы. Издание:Недра, Москва, 1985 г., 509 стр., УДК: [622.271.3] (075.8).

[2] Анистратов К.Ю., Анистратов Ю.И., Щадов М.И. Справочник по открытым горным работам. Горное дело, Москва, 2010 г., 700 стр., УДК: 622.271 (035), ISBN: 978-5-904463-01-4

[3] Спиваковский, А.О., Дьячков В.К. Транспортирующие машины: Учеб. пособие для машиностроительных вузов. – 3-е изд., перераб. – М.: Машиностроение, 1983. – 487 с.

[4] Трубецкой К.Н., Кулешов А.А., Клебанов А.Ф., Владимиров Д.Я. Современные системы управления горно-транспортными комплексами / Под редакцией акад. РАН К.Н. Трубецкого. СПб.: Наука 2007.

[5] Temkin I., Deryabin S., Konov I. Soft computing models in an intellectual open-pit mines transport control system // Procedia Computer Science, 2017, Vol. 120, P. 411–416; DOI: 10.1016/j.procs.2017.11.257.

[6] Zhang, B.; Li, D.; Ba, T.; Jiang, D.; Qiu, X.; Wang, T. Obstacle Detection and Tracking of Unmanned Mine Car Based on 3D Lidar. In Proceedings of the 2022 International Conference on Cyber-Physical Social Intelligence (ICCSI), Nanjing, China, 18–21 November 2022; pp. 222–228.

[7] Трубецкой К. Н., Клебанов А. Ф., А.Д.Рубан «Принципы построения и основные этапы научно-технической реализации проекта «Интеллектуальный карьер». II международная научно-практическая конференция «Техгормет-21 век». Тема конференции - «Карьерная техника для открытых горных работ: новые разработки и эффективные решения».

[8] Д.А.Клебанов, М.А.Макеев. Роботизированные технологии добычи полезных ископаемых рождаются в недрах инновационного цента Сколково. Журнал "Горная Промышленность" №4 2012, стр.132

[9] Y. Voronov, A. Voronov, D. Makhambayev Current State and Development Prospects of Autonomous Haulage at Surface Mines. E3S Web of

Conferences 2020, 174, 01028, https://doi.org/10.1051/e3sconf/202017401028.

[10] Vladimirov, D.Ya.; Klebanov, A.F.; Kuznetsov, I.V. Digital transformation of surface mining amd new generation of open-pit equipment. Russian Mining Industry 2020, 6, 10-12, https://doi.org/10.30686/1609-9192-2020-6-10-12.

[11] Fang, Y.; Peng, X. Micro-Factors-Aware Scheduling of Multiple Autonomous Trucks in Open-Pit Mining via Enhanced Metaheuristics. Electronics 2023, 12, 3793, https://doi.org/10.3390/electronics12183793.

[12] Tang, J.; Zhao, B.; Yang, C.; Zhou, C.; Chen, S.; Ni, X.; Ai, Y. An Architecture and Key Technologies of Autonomous Truck Dispatching System in Open-pit Mines. In Proceedings of the 2022 International Conference on Cyber-Physical Social Intelligence (ICCSI), Nanjing, China, 18–21 November 2022; pp. 122–127.

[13] Темкин И.О., Клебанов Д.А. Интеллектуальные системы управления горнотранспортными комплексами: современное состояние, задачи и механизмы решения // ГИАБ – М., 2014. – с. 257-266.

[14] Temkin I.O., Kulyanitsa A.L., Kubrin S.S. Application of intellectual system for robotic coal plough machine. Сборник «MINER'S WEEK - 2015 REPORTS OF THE XXIII INTERNATIONAL SCIENTIFIC SYMPOSIUM»

[15] Khussan, Bolatkhan, Arstanbek Abdiev, Marat Bitimbayev, Sergey Kuzmin, Sayat Issagulov, and Azamat Matayev. "Mining of Mineral Deposits." (2022).

[16] Mnzool, Mohammed, Hamad Almujibah, Mudthir Bakri, Ahmed Gaafar, Adil AM Elhassan, and Ehab Gomaa. "Optimization of cycle time for loading and hauling trucks in open-pit mining." Mining of Mineral Deposits 18, no. 1 (2024): 18-26.

[17] An, Xing, Celimuge Wu, Yangfei Lin, Min Lin, Tsutomu Yoshinaga, and Yusheng Ji. "Multi-robot systems and cooperative object transport: Communications, platforms, and challenges." IEEE Open Journal of the Computer Society 4 (2023): 23-36.

[18] Akbari, Foad, Jaber Valizadeh, and Ashkan Hafezalkotob. "Robust cooperative planning of relief logistics operations under demand uncertainty: a case study on a possible earthquake in Tehran." International Journal of Systems Science: Operations & Logistics 9, no. 3 (2022): 405-428.

[19] Kostenko, V. V., V. A. Golubtsov, R. V. Pank, and A. O. Shmidt. "Optimization of regional transport networks based on a mathematical model of passenger preferences." In Journal of Physics: Conference Series, vol. 2131, no. 3, p. 032101. IOP Publishing, 2021.

[20] AbuSalim, Samah WG, Rosziati Ibrahim, Mohd Zainuri Saringat, Sapiee Jamel, and Jahari Abdul Wahab. "Comparative analysis between dijkstra and bellman-ford algorithms in shortest path optimization." In IOP Conference Series: Materials Science and Engineering, vol. 917, no. 1, p. 012077. IOP Publishing, 2020.

[21] Andreiana, Andrei-Daniel, Costin Bădică, and Eugen Ganea. "An experimental comparison of implementations of dijkstra's single source shortest path algorithm using different priority queues data structures." In 2020 24th International Conference on System Theory, Control and Computing (ICSTCC), pp. 124-129. IEEE, 2020.

[22] Paterson, Bob. The Somme 1916—The Butte de Warlencourt: Martinpuich & Le Sars. Pen and Sword Military, 2022.

[23] Cao, Yuan, Zixuan Zhang, Fanglin Cheng, and Shuai Su. "Trajectory optimization for high-speed trains via a mixed integer linear programming approach." IEEE Transactions on Intelligent Transportation Systems 23, no. 10 (2022): 17666-17676.

[24] Yuan, Qiumeng, Shengzheng Wang, Jiansen Zhao, Tsung-Hsuan Hsieh, Zhen Sun, and Bin Liu. "Uncertainty-informed ship voyage optimization approach for exploiting safety, energy saving and low carbon routes." Ocean Engineering 266 (2022): 112887.

[25] Wang, Kai, Jianhang Wang, Lianzhong Huang, Yupeng Yuan, Guitao Wu, Hui Xing, Zhongyi Wang, Zhuang Wang, and Xiaoli Jiang. "A comprehensive review on the prediction of ship energy consumption and pollution gas emissions." Ocean Engineering 266 (2022): 112826.

[26] Bastos, Tomás Miguel Bernardo. "Task Allocation Algorithms for a Cooperative Drone Parcel Delivery System." (2020).

[27] Cao, Yuan, Zixuan Zhang, Fanglin Cheng, and Shuai Su. "Trajectory optimization for high-speed trains via a mixed integer linear programming approach." IEEE Transactions on Intelligent Transportation Systems 23, no. 10 (2022): 17666-17676.

[28] Bao, Junjiang, Xiang He, Yuanyuan Deng, Ning Zhang, Xiaopeng Zhang, Baigang An, and Gaohong He. "Parametric analysis and multi-objective optimization of a new combined system of liquid carbon dioxide energy storage and liquid natural gas cold energy power generation." Journal of Cleaner Production 363 (2022): 132591.

[29] Ferdous, Jannatul, Farid Bensebaa, Abbas S. Milani, Kasun Hewage, Pankaj Bhowmik, and Nathan Pelletier. "Development of a Generic Decision Tree for the Integration of Multi-Criteria Decision-Making (MCDM) and Multi-Objective Optimization (MOO) Methods under Uncertainty to Facilitate Sustainability Assessment: A Methodical Review." Sustainability 16, no. 7 (2024): 2684.

[30] Zhang, Sai, Caiwu Lu, Song Jiang, Lu Shan, and Neal Naixue Xiong. "An unmanned intelligent transportation scheduling system for open-pit mine vehicles based on 5G and big data." IEEE Access 8 (2020): 135524-135539.

[31] Khorsand, Reihaneh, and Mohammadreza Ramezanpour. "An energy-efficient task-scheduling algorithm based on a multi-criteria decision-making method in cloud computing." International Journal of Communication Systems 33, no. 9 (2020): e4379.

[32] Hosseinzadeh, Mehdi, Hawkar Kamaran Hama, Marwan Yassin Ghafour, Mohammad Masdari, Omed Hassan Ahmed, and Hemn Khezri. "Service

selection using multi-criteria decision making: a comprehensive overview." Journal of Network and Systems Management 28 (2020): 1639-1693.

[33] Liu, Li, Caiwu Lu, Fengjun Xiao, Ruimin Liu, and Neal Naixue Xiong. "A practical, integrated multi-criteria decision-making scheme for choosing cloud services in cloud systems." IEEE Access 9 (2021): 88391-88404.

[34] Zhang, Huizhen, Fan Liu, Liang Ma, and Ziying Zhang. "A hybrid heuristic based on a particle swarm algorithm to solve the capacitated location-routing problem with fuzzy demands." IEEE access 8 (2020): 153671-153691.

[35] Smith, Amanda, Jeff Linderoth, and James Luedtke. "Optimization-based dispatching policies for open-pit mining." Optimization and Engineering 22, no. 3 (2021): 1347-1387.

[36] Zhang, Sai, Caiwu Lu, Song Jiang, Lu Shan, and Neal Naixue Xiong. "An unmanned intelligent transportation scheduling system for open-pit mine vehicles based on 5G and big data." IEEE Access 8 (2020): 135524-135539.

[37] Klein, Tom Lorenz, and Clemens Thielen. "Improving Patient Transport in Hospitals: A Literature Review of Operations Research Methods." arXiv preprint arXiv:2404.03282 (2024).

[38] Le, Thi Diem Chau, Duy Duc Nguyen, Judit Oláh, and Miklós Pakurár. "Optimal vehicle route schedules in picking up and delivering cargo containers considering time windows in logistics distribution networks: A case study." Production Engineering Archives 26, no. 4 (2020): 174-184.

[39] Reed, Daniel, Dennis Gannon, and Jack Dongarra. "Reinventing high performance computing: challenges and opportunities." arXiv preprint arXiv:2203.02544 (2022).

[40] Zeng, Qunsong, Yuqing Du, Kaibin Huang, and Kin K. Leung. "Energy-efficient resource management for federated edge learning with CPU-GPU heterogeneous computing." IEEE Transactions on Wireless Communications 20, no. 12 (2021): 7947-7962.

[41] Posluns, Gilead, Yan Zhu, Guowei Zhang, and Mark C. Jeffrey. "A scalable architecture for reprioritizing ordered parallelism." In Proceedings of the 49th

Annual International Symposium on Computer Architecture, pp. 437-453. 2022.

[42] Mustafa, Dheya, Ruba Alkhasawneh, Fadi Obeidat, and Ahmed S. Shatnawi. "MIMD Programs Execution Support on SIMD Machines: A Holistic Survey." IEEE Access (2024).

[43] Mustafa, Dheya, Ruba Alkhasawneh, Fadi Obeidat, and Ahmed S. Shatnawi. "MIMD Programs Execution Support on SIMD Machines: A Holistic Survey." IEEE Access (2024).

[44] Godi, Prasanna Kumar, Battula Tirumala Krishna, and Pushpa Kotipalli. "Design optimisation of multiplier-free parallel pipelined FFT on field programmable gate array." IET Circuits, Devices & Systems 14, no. 7 (2020): 995-1000.

[45] Blueman, Daniel J., Foivos Zakkak, and Christos Kotselidis. "NUMAscope: Capturing and Visualizing Hardware Metrics on Large ccNUMA Systems." arXiv preprint arXiv:2111.11836 (2021).

[46] Ciesko, Jan, David Poliakoff, Daisy S. Hollman, Christian C. Trott, and Damien Lebrun-Grandié. "Towards generic parallel programming in computer science education with Kokkos." In *2020 IEEE/ACM Workshop on Education for High-Performance Computing (EduHPC)*, pp. 35-42. IEEE, 2020.

[47] Raynal, Michel, and Gadi Taubenfeld. "Mutual exclusion in fully anonymous shared memory systems." Information Processing Letters 158 (2020): 105938.

[48] Kang, Hongbo, Phillip B. Gibbons, Guy E. Blelloch, Laxman Dhulipala, Yan Gu, and Charles McGuffey. "The processing-in-memory model." In Proceedings of the 33rd ACM Symposium on Parallelism in Algorithms and Architectures, pp. 295-306. 2021.

[49] Ali, Ahsan, Hongwei Wang, Ali Ahmad Bodla, and Waseem Bahadur. "A moderated mediation model linking transactive memory system and social media with shared leadership and team innovation." Scandinavian Journal of Psychology 62, no. 4 (2021): 625-637.

[50] Al-Saeedi, Adnan Adhab K., Shamaeva Olga Yurievna, and Teaba W. Khairi. "Improving the efficiency of sparse matrix class processing by using the SPM-CSR parallel algorithm and OpenMP technology." In 2022 4th International Youth Conference on Radio Electronics, Electrical and Power Engineering (REEPE), pp. 1-4. IEEE, 2022.

[51] Elwasif, Wael. "Experimental Characterization of OpenMP Offloading Memory Operations and Unified Shared Memory Support." In International Workshop on OpenMP, pp. 210-225. Cham: Springer Nature Switzerland, 2023.

[52] Zeng, Guang. "Performance analysis of parallel programming models for C++." In Journal of Physics: Conference Series, vol. 2646, no. 1, p. 012027. IOP Publishing, 2023.

[53] Zhang, Tianyi, Shahrzad Shirzad, Bibek Wagle, Adrian S. Lemoine, Patrick Diehl, and Hartmut Kaiser. "Supporting OpenMP 5.0 Tasks in hpxMP--A study of an OpenMP implementation within Task Based Runtime Systems." arXiv preprint arXiv:2002.07970 (2020).

[54] Ozen, Guray, and Michael Wolfe. "Performant portable openMP." In Proceedings of the 31st ACM SIGPLAN International Conference on Compiler Construction, pp. 156-168. 2022.

[55] Volosatova, T. M., I. A. Kiselev, and S. V. Knyazeva. "MULTITHREADING IN POSIX STANDARD." East European Scientific Journal 3, no. 12 (76) (2021): 37-39.

[56] Certner, Olivier. "rtprio (2) and POSIX (. 1b) priorities, and their FreeBSD implementation: A deep dive (and sweep)."

[57] Castelló, Adrián, Rafael Mayo Gual, Sangmin Seo, Pavan Balaji, Enrique S. Quintana-Orti, and Antonio J. Pena. "Analysis of threading libraries for high performance computing." IEEE Transactions on Computers 69, no. 9 (2020): 1279-1292.

[58] Zeng, Guang. "Performance analysis of parallel programming models for C++." In Journal of Physics: Conference Series, vol. 2646, no. 1, p. 012027. IOP Publishing, 2023.

[59] McClelland, James L., and David E. Rumelhart. "Distributed memory and the representation of general and specific information." In Connectionist Psychology, pp. 75-106. Psychology Press, 2020.

[60] Schwitanski, Simon, Felix Tomski, Joachim Protze, Christian Terboven, and Matthias S. Müller. "An on-the-fly method to exchange vector clocks in distributed-memory programs." In 2022 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), pp. 530-540. IEEE, 2022.

[61] Velarde Martínez, Apolinar. "Parallelization of array method with hybrid programming: OpenMP and MPI." Applied Sciences 12, no. 15 (2022): 7706.

[62] Kayumov, Zufar, Dmitrii Tumakov, and Sergey Mosin. "Hierarchical convolutional neural network for handwritten digits recognition." Procedia Computer Science 171 (2020): 1927-1934.

[63] Hashan, Antor Mahamudul, Ebenezer Agbozo, Adnan Adhab K. Al-Saeedi, Srishti Saha, Abdullah Haidari, and Mohammad Naser Fazel Rabi. "Brain Tumor Detection in MRI Images Using Image Processing Techniques." In 2021 4th International Symposium on Agents, Multi-Agent Systems and Robotics (ISAMSR), pp. 24-28. IEEE, 2021.

[64] Hashan, Antor Mahamudul, K. Al-Saeedi Adnan Adhab, Rizu Md Rakib Ul Islam, Kumar Avinash, and Subhankar Dey. "Automated Human Facial Emotion Recognition System Using Depthwise Separable Convolutional Neural Network." In 2023 IEEE International Conference on Industry 4.0, Artificial Intelligence, and Communications Technology (IAICT), pp. 113-117. IEEE, 2023.

[65] Bhatt, Dulari, Chirag Patel, Hardik Talsania, Jigar Patel, Rasmika Vaghela, Sharnil Pandya, Kirit Modi, and Hemant Ghayvat. "CNN variants for

computer vision: History, architecture, application, challenges and future scope." Electronics 10, no. 20 (2021): 2470.

[66] Bao, Yin-Xin, Quan Shi, Qin-Qin Shen, and Yang Cao. "Spatial-temporal 3D residual correlation network for urban traffic status prediction." Symmetry 14, no. 1 (2021): 33.

[67] Yu, Daben, Zongping Li, Qinglun Zhong, Yi Ai, and Wei Chen. "Demand Management of Station-Based Car Sharing System Based on Deep Learning Forecasting." Journal of Advanced Transportation 2020, no. 1 (2020): 8935857.

[68] Saon, George, Zoltán Tüske, Daniel Bolanos, and Brian Kingsbury. "Advancing RNN transducer technology for speech recognition." In ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 5654-5658. IEEE, 2021.

[69] Liu, Shuijing, Peixin Chang, Weihang Liang, Neeloy Chakraborty, and Katherine Driggs-Campbell. "Decentralized structural-rnn for robot crowd navigation with deep reinforcement learning." In 2021 IEEE international conference on robotics and automation (ICRA), pp. 3517-3524. IEEE, 2021.

[70] Zargar, S. "Introduction to sequence learning models: RNN, LSTM, GRU." Department of Mechanical and Aerospace Engineering, North Carolina State University (2021).

[71] Zhao, Jingyu, Feiqing Huang, Jia Lv, Yanjie Duan, Zhen Qin, Guodong Li, and Guangjian Tian. "Do RNN and LSTM have long memory?." In International Conference on Machine Learning, pp. 11365-11375. PMLR, 2020.

[72] Zhang, Zufan, Hao Luo, Chun Wang, Chenquan Gan, and Yong Xiang. "Automatic modulation classification using CNN-LSTM based dual-stream structure." IEEE Transactions on Vehicular Technology 69, no. 11 (2020): 13521-13531.

[73] Chen, Weifeng, Fei Zheng, Shanping Gao, and Kai Hu. "An LSTM with differential structure and its application in action recognition." Mathematical Problems in Engineering 2022, no. 1 (2022): 7316396.

[74] Hewage, Pradeep, Ardhendu Behera, Marcello Trovati, Ella Pereira, Morteza Ghahremani, Francesco Palmieri, and Yonghuai Liu. "Temporal convolutional neural (TCN) network for an effective weather forecasting using time-series data from the local weather station." Soft Computing 24 (2020): 16453-16482.

[75] Zhang, Rongji, Feng Sun, Ziwen Song, Xiaolin Wang, Yingcui Du, and Shulong Dong. "Short-term traffic flow forecasting model based on GA-TCN." Journal of Advanced Transportation 2021, no. 1 (2021): 1338607.

[76] Rico, João, José Barateiro, and Arlindo Oliveira. "Graph neural networks for traffic forecasting." arXiv preprint arXiv:2104.13096 (2021).

[77] Mohd Noor, Mohd Halim, Sen Yan Tan, and Mohd Nadhir Ab Wahab. "Deep temporal conv-lstm for activity recognition." Neural Processing Letters 54, no. 5 (2022): 4027-4049.

[78] Guo, Kan, Yongli Hu, Zhen Qian, Hao Liu, Ke Zhang, Yanfeng Sun, Junbin Gao, and Baocai Yin. "Optimized graph convolution recurrent neural network for traffic prediction." IEEE Transactions on Intelligent Transportation Systems 22, no. 2 (2020): 1138-1149.

[79] Shi, Yong, Yunong Wang, Yi Qu, and Zhensong Chen. "Integrated gcn-lstm stock prices movement prediction based on knowledge-incorporated graphs construction." International Journal of Machine Learning and Cybernetics 15, no. 1 (2024): 161-176.

[80] Wei, Xin, Ruixuan Yu, and Jian Sun. "View-gcn: View-based graph convolutional network for 3d shape analysis." In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 1850-1859. 2020.

[81] Seo, Sungyong, Zijun Cui, Sam Griesemer, Joshua Hikida, and Yan Liu. "Physics-aware Causal Graph Network for Spatiotemporal Modeling." (2023).

[82] Li, Fuxian, Jie Feng, Huan Yan, Guangyin Jin, Fan Yang, Funing Sun, Depeng Jin, and Yong Li. "Dynamic graph convolutional recurrent network for traffic prediction: Benchmark and solution." ACM Transactions on Knowledge Discovery from Data 17, no. 1 (2023): 1-21.

[83] Wang, Yuhu, Shen Fang, Chunxia Zhang, Shiming Xiang, and Chunhong Pan. "TVGCN: Time-variant graph convolutional network for traffic forecasting." Neurocomputing 471 (2022): 118-129.

[84] Tzirakis, Panagiotis, Anurag Kumar, and Jacob Donley. "Multi-channel speech enhancement using graph neural networks." In ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 3415-3419. IEEE, 2021.

[85] Temkin, Igor, Alexander Myaskov, Sergey Deryabin, Iliya Konov, and Alexander Ivannikov. "Design of a digital 3D model of transport–technological environment of open-pit mines based on the common use of telemetric and geospatial information." Sensors 21, no. 18 (2021): 6277.

[86] Drori, Iddo, Anant Kharkar, William R. Sickinger, Brandon Kates, Qiang Ma, Suwen Ge, Eden Dolev, Brenda Dietrich, David P. Williamson, and Madeleine Udell. "Learning to solve combinatorial optimization problems on real-world graphs in linear time." In 2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA), pp. 19-24. IEEE, 2020.2

[87] Kuutti, Sampo, Richard Bowden, Yaochu Jin, Phil Barber, and Saber Fallah. "A survey of deep learning applications to autonomous vehicle control." IEEE Transactions on Intelligent Transportation Systems 22, no. 2 (2020): 712-733.

[88] Bathla, Gourav, Kishor Bhadane, Rahul Kumar Singh, Rajneesh Kumar, Rajanikanth Aluvalu, Rajalakshmi Krishnamurthi, Adarsh Kumar, R. N. Thakur, and Shakila Basheer. "Autonomous vehicles and intelligent

automation: Applications, challenges, and opportunities." Mobile Information Systems 2022, no. 1 (2022): 7632892.

[89] Sizemov, D.N., Temkin, I.O., Deryabin, S.A. and Vladimirov, D.Y., 2021. On some aspects of increasing the target productivity of unmanned mine dump trucs. Eurasian Mining, 36(2), pp.68-73.2

[90] Темкин И.О., Куляница А.Л., Дерябин С.А.Вычислительные модели взаимодействия автономных мобильных агентов транспортного комплекса горных предприятий, Научно-технический журнал «Информация и космос», №2, 2017, с.61-65.

[91] Zhao, Ziyu, and Lin Bi. "A new challenge: Path planning for autonomous truck of open-pit mines in the last transport section." Applied Sciences 10, no. 18 (2020): 6622.

[92] Клебанов Д.А., Макеев М.А. Роботизированные технологии добычи полезных ископаемых рождаются в недрах инновационного центра Сколково. Горная промышленность, №4(104), 2012, с.2-4.

[93] Starkov, Alexander V., Andrey A. Emelyanov, Lyubov A. Grishantseva, Ksenia I. Zhukovskaya, Alexander A. Morozov, and Alexey A. Trishin. "Methodology for managing the flows of target information in the remote sensing space system Part 1. Task formalization." RUDN Journal of Engineering Research 22, no. 1 (2021): 54-64.

[94] Timofeeva, Galina Adol'fovna. "Probabilistic solutions of conditional optimization problems." (2020): 198-212.

[95] Li, Wenhua, Xingyi Yao, Tao Zhang, Rui Wang, and Ling Wang. "Hierarchy ranking method for multimodal multiobjective optimization with local Pareto fronts." IEEE Transactions on Evolutionary Computation 27, no. 1 (2022): 98-110.

[96] Harada, Tomohiro, and Enrique Alba. "Parallel genetic algorithms: a useful survey." ACM Computing Surveys (CSUR) 53, no. 4 (2020): 1-39.

[97] Candra, Ade, Mohammad Andri Budiman, and Kevin Hartanto. "Dijkstra's and a-star in finding the shortest path: a tutorial." In 2020 International

Conference on Data Science, Artificial Intelligence, and Business Analytics (DATABIA), pp. 28-32. IEEE, 2020.

[98] Majeed, Abdul, and Ibtisam Rauf. "Graph theory: A comprehensive survey about graph theory applications in computer science and social networks." Inventions 5, no. 1 (2020): 10.

[99] Аль-Руфаи, Фаиз Муса, Борис Анатольевич Якимович, Владимир Владиславович Кувшинов, Аднан К. Аль-Саиди, and Д. Ф. Бордан. "Моделирование и анализ работы системы накопления пьезоэлектрической энергии при помощи программной среды Matlab/Simulink." Интеллектуальные системы в производстве 20, no. 3 (2022): 24-33.

[100] Dutta, Kousik Kumar, Ankita Dewan, and Venkata MV Gunturi. "A Multi-Threading Algorithm for Constrained Path Optimization Problem on Road Networks." In International Conference on Web Information Systems Engineering, pp. 110-118. Cham: Springer International Publishing, 2022.

[101] Heersmink, Richard. "Preserving narrative identity for dementia patients: embodiment, active environments, and distributed memory." Neuroethics 15, no. 1 (2022): 8.

[102] Narayanan, Deepak, Mohammad Shoeybi, Jared Casper, Patrick LeGresley, Mostofa Patwary, Vijay Korthikanti, Dmitri Vainbrand et al. "Efficient large-scale language model training on gpu clusters using megatron-lm." In Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, pp. 1-15. 2021.

[103] Fan, Yuping, Zhiling Lan, Paul Rich, William Allcock, and Michael E. Papka. "Hybrid workload scheduling on HPC systems." In 2022 IEEE International Parallel and Distributed Processing Symposium (IPDPS), pp. 470-480. IEEE, 2022.

[104] Trabes, Guillermo German, Gabriel A. Wainer, and Veronica Gil-Costa. "A Parallel Algorithm to Accelerate DEVS Simulations in Shared Memory

Architectures." IEEE Transactions on Parallel and Distributed Systems 34, no. 5 (2023): 1609-1620.

[105] Bak, Seonmyeong, Colleen Bertoni, Swen Boehm, Reuben Budiardja, Barbara M. Chapman, Johannes Doerfert, Markus Eisenbach et al. "OpenMP application experiences: Porting to accelerated nodes." Parallel Computing 109 (2022): 102856.

[106] Bora, Utpal, Shraiysh Vaishay, Saurabh Joshi, and Ramakrishna Upadrasta. "OpenMP aware MHP analysis for improved static data-race detection." In 2021 IEEE/ACM 7th Workshop on the LLVM Compiler Infrastructure in HPC (LLVM-HPC), pp. 1-11. IEEE, 2021.

[107] Neth, Brandon, Thomas RW Scogland, Alejandro Duran, and Bronis R. de Supinski. "Beyond explicit transfers: shared and managed memory in OpenMP." In OpenMP: Enabling Massive Node-Level Parallelism: 17th International Workshop on OpenMP, IWOMP 2021, Bristol, UK, September 14–16, 2021, Proceedings 17, pp. 183-194. Springer International Publishing, 2021.

[108] Kaposi, Á., Kolarovszki, Z., Kozsik, T., Zimborás, Z. and Rakyta, P., 2021. Polynomial speedup in Torontonian calculation by a scalable recursive algorithm. arXiv preprint arXiv:2109.04528.

[109] Jammer, Tim, Christian Iwainsky, and Christian Bischof. "A comparison of the scalability of openmp implementations." In Euro-Par 2020: Parallel Processing: 26th International Conference on Parallel and Distributed Computing, Warsaw, Poland, August 24–28, 2020, Proceedings 26, pp. 83-97. Springer International Publishing, 2020.

[110] Аль-Саиди, А. А., И. О. Темкин, В. И. Алтай, А. Ф. Алмунтафеки, and А. Н. Мохедхуссин. "Повышение эффективности алгоритма Дейкстры с помощью технологий параллельных вычислений с библиотекой OpenMP." Инженерный вестник Дона 8 (104) (2023): 7.

[111] Ozen, Guray, and Michael Wolfe. "Performant portable openMP." In Proceedings of the 31st ACM SIGPLAN International Conference on Compiler Construction, pp. 156-168. 2022.

[112] Naseer, Mohamed, Zayed Khaled, Islam Tharwat Abdel Halim, and Ayman Bahaa-Eldin. "Synchronization in Parallel Programming Models for Heterogeneous Many-Cores." In 2020 Sixth International Conference on Parallel, Distributed and Grid Computing (PDGC), pp. 571-576. IEEE, 2020.

[113] Candra, Ade, Mohammad Andri Budiman, and Kevin Hartanto. "Dijkstra's and a-star in finding the shortest path: a tutorial." In 2020 International Conference on Data Science, Artificial Intelligence, and Business Analytics (DATABIA), pp. 28-32. IEEE, 2020.

[114] Алеева, Валентина Николаевна, and Павел Андреевич Манатин. "ПРИМЕНЕНИЕ МЕТОДА ПРОЕКТИРОВАНИЯ Q-ЭФФЕКТИВНЫХ ПРОГРАММ ДЛЯ АЛГОРИТМА ДЕЙКСТРЫ." Вестник Южно-Уральского государственного университета. Серия: Вычислительная математика и информатика 12, no. 2 (2023): 62-77.

[115] Nayagam, R. Deiva, D. Selvathi, R. Geeta, D. Gopinath, and G. Sivakumar. "Mobile Application based Indoor Positioning and Navigational System using Dijkstra’s Algorithm." In Journal of Physics: Conference Series, vol. 2466, no. 1, p. 012007. IOP Publishing, 2023.

[116] Behún, Marcel, Dušan Knežo, Michal Cehlár, Lucia Knapčíková, and Annamária Behúnová. "Recent application of Dijkstra’s algorithm in the process of production planning." Applied Sciences 12, no. 14 (2022): 7088.

[117] Kumar, Praveen, and Anil Kumar Singh. "Mapreduce algorithm for single source shortest path problem." International Journal of Computer Network and Information Security 11, no. 3 (2020): 11.

[118] Yulianggara, Revo. "Pengembangan aplikasi informasi tempat parkir terdekat di Provinsi DKI Jakarta menggunakan location based service (lbs) dan google maps api (studi kasus: unit perparkiran dinas perhubungan DKI Jakarta)."

Bachelor's thesis, Fakultas Sains dan Teknologi UIN Syarif Hidayatullah Jakarta, 2022.

[119] Pashinska, Maria, and Iliya Bouyukliev. "About OpenMP and some combinatorial algorithms." Mathematics and Education in Mathematics 49 (2020): 167-172.

[120] Zhang, Peng, Jianbin Fang, Canqun Yang, Chun Huang, Tao Tang, and Zheng Wang. "Optimizing streaming parallelism on heterogeneous many-core architectures." IEEE Transactions on Parallel and Distributed Systems 31, no. 8 (2020): 1878-1896.

[121] Khan, Shakir, and Hela Alghulaiakh. "ARIMA model for accurate time series stocks forecasting." International Journal of Advanced Computer Science and Applications 11, no. 7 (2020).

[122] Chen, Bokui, Duo Sun, Jun Zhou, Wengfai Wong, and Zhongjun Ding. "A future intelligent traffic system with mixed autonomous vehicles and human-driven vehicles." Information Sciences 529 (2020): 59-72.

[123] Holden, K., Vector auto regression modeling and forecasting. Journal of Forecasting, 1995.14(3): p. 159-166.

[124] Hamilton, J.D., Time series analysis. 2020: Princeton university press.

[125] Chen, C., et al. Short-time traffic flow prediction with ARIMA-GARCH model. in 2011 IEEE Intelligent Vehicles Symposium (IV). 2011. IEEE.

[126] Williams, B.M. and L.A. Hoel, Modeling and forecasting vehicular traffic flow as a seasonal ARIMA process: Theoretical basis and empirical results. Journal of transportation engineering, 2003. 129(6): p. 664-672.

[127] Zheng, Z. and D. Su, Short-term traffic volume forecasting: A k-nearest neighbor approach enhanced by constrained linearly sewing principle component algorithm. Transportation Research Part C: Emerging Technologies, 2014. 43: p. 143-157.

[128] Wei, Y. and M.-C. Chen, Forecasting the short-term metro passenger flow with empirical mode decomposition and neural networks. Transportation Research Part C: Emerging Technologies, 2012. 21(1): p. 148-162.

[129] Jiang, X., L. Zhang, and X.M. Chen, Short-term forecasting of high-speed rail demand: A hybrid approach combining ensemble empirical mode decomposition and gray support vector machine with real-world applications in China. Transportation Research Part C: Emerging Technologies, 2014. 44: p. 110-127.

[130] Pan, Mingyang, Hainan Zhou, Jiayi Cao, Yisai Liu, Jiangling Hao, Shaoxi Li, and Chi-Hua Chen. "Water level prediction model based on GRU and CNN." Ieee Access 8 (2020): 60090-60100.

[131] Gao, Ya, Rong Wang, and Enmin Zhou. "Stock prediction based on optimized LSTM and GRU models." Scientific programming 2021, no. 1 (2021): 4055281.

[132] Nikparvar, Behnam, and Jean-Claude Thill. "Machine learning of spatial data." ISPRS International Journal of Geo-Information 10, no. 9 (2021): 600.

[133] Song, Yongze, Jinfeng Wang, Yong Ge, and Chengdong Xu. "An optimal parameters-based geographical detector model enhances geographic characteristics of explanatory variables for spatial heterogeneity analysis: Cases with different types of spatial data." GIScience & Remote Sensing 57, no. 5 (2020): 593-610.

Приложение 1

«Код разработки алгоритма Дейкстры с использованием параллельных вычислений и библиотеки OpenMP»

```
//The development code of Dijkstra's algorithm using parallel computing and the
OpenMP library
//Код разработки алгоритма Дейкстры с использованием параллельных вычислений и
библиотеки OpenMP
//Al-Saeedi Adnan Adhab K.,Iraq, 2024
//Аль-Саиди Аднан Адаб К.,Ирак, 2024
//Национальный исследовательский технологический университет «МИСиС»
#include <iostream>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <fstream>
#include <limits.h>
#include <conio.h>
#include <omp.h>
#include <ctime>
#include <cstdlib>
using namespace std;

// Number of vertices in the graph
#define V 70000
int INF= 99999;
int dist[V] ; // The output array. dist[i] will hold the shortest distance from src
to i
int N_Thread;

// A utility function to find the vertex with minimum distance value,
//from the set of vertices not yet included in shortest path tree
int minDistance(int dist[], bool sptSet[])
{     int min = INF; // Initialize min value
      int min_index;
            for (int v = 0; v < V; v++)
            if ((sptSet[v] == false) && (dist[v] <= min)){
                  min = dist[v];
                  min_index = v;
            }//end if
      return min_index;
} //End minDistance

// A utility function to print the constructed distance array
void printSolution(int dist[])
{     cout << "Vertex \t Distance from Source" << endl;
      for (int i = 0; i < V; i++)
            cout << i+1 << " \t\t\t\t" << dist[i] << endl;
}

// Function that implements Dijkstra's single source shortest path algorithm
//for a graph represented using adjacency matrix representation
void dijkstra(int** graph, int src)
{
      bool sptSet[V]; // sptSet[i] will be true if vertex i is included in
      //shortest path tree or shortest distance from src to i is finalized
      // Initialize all distances as INFINITE and stpSet[] as false
#pragma omp parallel for  num_threads(N_Thread)
            for (int i = 0; i < V; i++){
```

```
                dist[i] = INF;//INT_MAX
                sptSet[i] = false;
            }//for-i

            // Distance of source vertex from itself is always 0
            dist[src] = 0;

#pragma omp parallel for num_threads(N_Thread)
            // Find shortest path for all vertices
            for (int count = 0; count < V - 1; count++) {
                // Pick the minimum distance vertex from the set of vertices not
yet processed.
                //u is always equal to src in the first iteration.
                int u = minDistance(dist, sptSet);
                // Mark the picked vertex as processed
                sptSet[u] = true;
            // Update dist value of the adjacent vertices of the picked vertex.

                //cout << "N_Thread=" << N_Thread<<endl;
//#pragma omp parallel for  num_threads(N_Thread)
                for (int v = 0; v < V; v++){
                    // Update dist[v] only if is not in sptSet, there is an
edge from u to v, and total
                    // weight of path from src to v through u is smaller than
current value of dist[v]
                    if ((!sptSet[v] && graph[u][v]) && (dist[u] != INF) &&
(dist[u] + graph[u][v] < dist[v]))//INT_MAX
                        dist[v] = dist[u] + graph[u][v];
                }//End-for-v
            }//End if -count
}//End void dijkstra


int main()
{
    const int n=700, m=100;//Matrix size must be constant
// Declare matrix
    int matrix_graph[n][m];
    int matrix_n_node[n][m];
//Open data file
    ifstream iFile;
    iFile.open("D:\Dataset.txt");

//Read Data file
    for (int i = 0; i < n; i++)
        for (int j = 0; j < m; j++)
            iFile >> matrix_graph[i][j];

//Close Data file
    iFile.close();

//Print matrix_graph
    for (int i = 0; i < n; i++){
        for (int j = 0; j < m; j++)
            cout << matrix_graph[i][j]<<"  ";
        cout << endl;
    }//end for-i
//Digitization of the nodes
    int count = 0;
    for (int i = 0; i < n; i++)
        for (int j = 0; j < m; j++)
            if (matrix_graph[i][j] == 1){
```

```
                        count++;
                        matrix_n_node[i][j] = count;
                    }//end if
                    else
                        matrix_n_node[i][j] = 0;

        cout << endl << "Number of Nodes =" << count << endl;

//Print matrix_n_node
    cout << endl << "Print matrix_n_node :" << endl;
       for (int i = 0; i < n; i++){
           for (int j = 0; j < m; j++)
               cout << matrix_n_node[i][j] << "  ";
               cout << endl;
           }//end for-i

//Ganrate Adjasancy Matrix (Graph) and edges between nodes
       int** graph ;
       graph = new int*[V];
       for (int i = 0; i < V; i++)
            graph[i] = new int[V];

       for (int i = 0; i < V; i++)
           for (int j = 0; j < V; j++)
               graph[i][j] = 0;

       srand(time(0)); // Initialize random number generator.
       for (int i = 0; i < n; i++)
       for (int j = 0; j < m; j++){
           int ww = (rand() % 10) + 1;
           if (matrix_graph[i][j] == 1){
            if ((matrix_graph[i][j + 1] == 1) && (j + 1 < m)){
              graph[matrix_n_node[i][j] - 1][matrix_n_node[i][j + 1] - 1] = 1 + ww;
              graph[matrix_n_node[i][j + 1] - 1][matrix_n_node[i][j] - 1] = 1 + ww;
            }//End-if
            if ((matrix_graph[i + 1][j] == 1) && (i + 1 < n)){
              graph[matrix_n_node[i + 1][j] - 1][matrix_n_node[i][j] - 1] = 1 + ww;
              graph[matrix_n_node[i][j] - 1][matrix_n_node[i + 1][j] - 1] = 1 + ww;
              }//End-if
              //Diagnal Edge
              if ((matrix_graph[i + 1][j + 1] == 1) && (i + 1 < n) && (j + 1 < m)){
               graph[matrix_n_node[i][j] - 1][matrix_n_node[i + 1][j + 1] - 1]=1.4+ww;
               graph[matrix_n_node[i + 1][j + 1] - 1][matrix_n_node[i][j] - 1]=1.4+ww;
              }//End-if
              if ((matrix_graph[i + 1][j - 1] == 1) && (i + 1 < n) && (j - 1 >= 0)){
               graph[matrix_n_node[i][j] - 1][matrix_n_node[i + 1][j - 1] - 1]=1.4+ww;
               graph[matrix_n_node[i + 1][j - 1] - 1][matrix_n_node[i][j] - 1]=1.4+ww;
              }//End-if
              }//End-master if
       }// End for j
       //Print Matrix Graph :
              cout << endl << "Print Adjacency Matrix :" << endl;
              for (int i = 0; i < V; i++){
                    for (int j = 0; j < V; j++)
                          cout << " \t" << graph[i][j] ;
                    cout << endl;
                    }//End-for-i

       int NUM_PROCS = omp_get_num_procs();
       int NUM_THREADS = NUM_PROCS;
       cout << endl<<"In this computer" <<endl;
       cout << "Number of Processors = " << NUM_PROCS << endl;
```

```
	cout << "Number of Threads = " << NUM_THREADS << endl;
	cout << "In this example number of virtces = " << V << endl<<endl;
	cout << "Please wait to excution program ..." << endl << endl;
	double Start_time, End_time;
	for (N_Thread = 1; N_Thread <= 8; N_Thread*=2) {
		Start_time = omp_get_wtime();
		dijkstra(graph, 0);  // Function call
		End_time = omp_get_wtime();
		cout << "Execution time with " << N_Thread << " threads/core(s): " <<
End_time - Start_time << " seconds." << endl;
	}//End for N_Thread
	getch();
	int f1;
	cin >> f1;
	return 0;
}// end main program
```

Приложение 2

«Разработка Модель ST-GCN для прогнозирования скорости автономных транспортных средств на карьерных дорогах»

```
# Модель ST-GCN для прогнозирования скорости автономных транспортных средств
на карьерных дорогах
#Аль-Саиди Аднан Адаб К.,Ирак, 2024
#Национальный исследовательский технологический университет «МИСиС»
import torch
import torch.nn as nn
import torch.optim as optim
import pandas as pd
from sklearn.model_selection import train_test_split
from torch.utils.data import DataLoader, TensorDataset
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt
from sklearn.metrics import mean_squared_error, mean_absolute_error
import numpy as np
# Load dataset
data = pd.read_csv("/content/dataset-10000.csv", delimiter=';',
usecols=['Date_Time', 'Lat', 'Lon', 'Height', 'Azimut', 'Speed'])
# Preprocess the data
data = data.dropna()  # Remove all rows with NULL values from the DataFrame.
data = data.drop_duplicates()  # Remove duplicate rows based on  'Lon'
columns
data = data.drop_duplicates(['Lon'])  # Remove duplicate rows based on  'Lon'
columns
data.reset_index(drop=True, inplace=True)  # Reset the index
print(data)
# Convert Date_Time to datetime and extract temporal features
```

```
data['Date_Time'] = pd.to_datetime(data['Date_Time'])
data['Hour'] = data['Date_Time'].dt.hour
data['Minute'] = data['Date_Time'].dt.minute
# Normalize spatial features
data[['Lat', 'Lon']] = (data[['Lat', 'Lon']] - data[['Lat', 'Lon']].mean()) /
data[['Lat', 'Lon']].std()
# Split into features and target
X = data[['Hour', 'Minute', 'Lat', 'Lon']].values
y = data['Speed'].values
# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Define ST-GCN Model
class STGCN(nn.Module):
    def __init__(self, input_size, hidden_size, output_size):
        super(STGCN, self).__init__()
        self.conv1 = nn.Conv2d(1, hidden_size, kernel_size=(1, 3),
padding=(0, 1))
        self.conv2 = nn.Conv2d(hidden_size, output_size, kernel_size=(1, 3),
padding=(0, 1))
        self.fc = nn.Linear(input_size * output_size, 1)
    def forward(self, x):
        x = x.view(x.size(0), 1, -1, x.size(-1))  # Add channel dimension
        x = torch.relu(self.conv1(x))
        x = torch.relu(self.conv2(x))
        x = x.view(x.size(0), -1)
        x = self.fc(x)
        return x
# Prepare DataLoaders
train_dataset = TensorDataset(torch.tensor(X_train, dtype=torch.float32),
torch.tensor(y_train, dtype=torch.float32))
test_dataset = TensorDataset(torch.tensor(X_test, dtype=torch.float32),
torch.tensor(y_test, dtype=torch.float32))
train_loader = DataLoader(train_dataset, batch_size=64, shuffle=True)
test_loader = DataLoader(test_dataset, batch_size=64, shuffle=False)
# Instantiate the model, loss function, and optimizer
model = STGCN(input_size=X.shape[1], hidden_size=64, output_size=32)
criterion = nn.MSELoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)
# Lists to store training and testing losses
train_losses = []
test_losses = []
# Train the model
num_epochs = 1000
for epoch in range(num_epochs):
```

```
    for inputs, targets in train_loader:
        optimizer.zero_grad()
        outputs = model(inputs)
        loss = criterion(outputs, targets.view(-1, 1))
        loss.backward()
        optimizer.step()
    # Append training loss
    train_losses.append(loss.item())
    #print(f"Epoch {epoch + 1}/{num_epochs}, Loss: {loss.item()}")
    # Evaluate the model on the test set
    model.eval()
    with torch.no_grad():
       test_outputs = model(torch.tensor(X_test, dtype=torch.float32))
       test_loss = criterion(test_outputs, torch.tensor(y_test,
dtype=torch.float32).view(-1, 1))
    # Append testing loss
    test_losses.append(test_loss.item())
    #print(f"Test Loss: {test_loss.item()}")
# Pot training and testing losses
plt.plot(train_losses, label='Training Loss')
plt.plot(test_losses, label='Testing Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.show()
# Visualize actual and predicted values of speed
model.eval()
with torch.no_grad():
    predicted_speed = model(torch.tensor(X_test,
dtype=torch.float32)).numpy().flatten()

plt.plot(y_test, label='Actual Speed')
plt.plot(predicted_speed, label='Predicted Speed')
plt.xlabel('Sample')
plt.ylabel('Speed')
plt.legend()
plt.show()
#============================================================================
# Calculate metrics (MSE, MAE, RMSE)
predicted_speed = model(torch.tensor(X_test,
dtype=torch.float32)).detach().numpy().flatten()
mse = mean_squared_error(y_test, predicted_speed)
mae = mean_absolute_error(y_test, predicted_speed)
rmse = np.sqrt(mse)
print(f"MSE: {mse}")
print(f"MAE: {mae}")
```

```
print(f"RMSE: {rmse}")
# Visualize actual and predicted values of speed
plt.plot(y_test, label='Actual Speed')
plt.plot(predicted_speed, label='Predicted Speed')
plt.xlabel('Sample')
plt.ylabel('Speed')
plt.legend()
plt.show()
# Visualize metrics
metrics = ['MSE', 'MAE', 'RMSE']
values = [mse, mae, rmse]
plt.bar(metrics, values)
plt.xlabel('Metrics')
plt.ylabel('Value')
plt.title('Evaluation Metrics')
plt.show()
# Lists to store training and testing losses
train_losses = []
test_losses = []
# Lists to store metrics
mse_values = []
mae_values = []
rmse_values = []
# Train the model
num_epochs = 1000
for epoch in range(num_epochs):
    model.train()  # Set the model to training mode
    for inputs, targets in train_loader:
        optimizer.zero_grad()
        outputs = model(inputs)
        loss = criterion(outputs, targets.view(-1, 1))
        loss.backward()
        optimizer.step()

    # Append training loss
    train_losses.append(loss.item())

    model.eval()  # Set the model to evaluation mode
    with torch.no_grad():
        test_outputs = model(torch.tensor(X_test, dtype=torch.float32))
        test_loss = criterion(test_outputs, torch.tensor(y_test,
dtype=torch.float32).view(-1, 1))
    # Append testing loss
    test_losses.append(test_loss.item())
    # Calculate metrics (MSE, MAE, RMSE, MAPE)
    predicted_speed = test_outputs.detach().numpy().flatten()
```

```
    mse = mean_squared_error(y_test, predicted_speed)
    mae = mean_absolute_error(y_test, predicted_speed)
    rmse = np.sqrt(mse)
    epsilon = 1e-7  # Small epsilon value to avoid division by zero
    mape = np.mean(np.abs((y_test - predicted_speed)/(y_test + epsilon)))*100
    #mape = np.mean(np.abs((y_test - predicted_speed) / y_test)) * 100
    mse_values.append(mse)
    mae_values.append(mae)
    rmse_values.append(rmse)
    #print(f"Epoch {epoch + 1}/{num_epochs}, Loss: {loss.item()}, Test Loss:
{test_loss.item()}, MSE: {mse}, MAE: {mae}, RMSE: {rmse}, MAPE: {mape}")

# Visualize MSE, MAE, RMSE, and MAPE over epochs
epochs = range(1, num_epochs + 1)
plt.plot(epochs, mse_values, label='MSE')
plt.plot(epochs, mae_values, label='MAE')
plt.plot(epochs, rmse_values, label='RMSE')
plt.xlabel('Epochs')
plt.ylabel('Value')
plt.title('Evaluation Metrics over Epochs')
plt.legend()
plt.show()
#Insert 4444
# Lists to store training and testing losses
train_losses = []
test_losses = []

# Lists to store metrics
mse_values = []
mae_values = []
rmse_values = []
# Train the model
num_epochs = 1000
for epoch in range(num_epochs):
    model.train()  # Set the model to training mode
    epoch_train_losses = []
    for inputs, targets in train_loader:
        optimizer.zero_grad()
        outputs = model(inputs)
        loss = criterion(outputs, targets.view(-1, 1))
        loss.backward()
        optimizer.step()
        epoch_train_losses.append(loss.item())

    # Calculate average training loss for the epoch
    train_loss = np.mean(epoch_train_losses)
```

```
    train_losses.append(train_loss)

    model.eval()  # Set the model to evaluation mode
    with torch.no_grad():
        test_outputs = model(torch.tensor(X_test, dtype=torch.float32))
        test_loss = criterion(test_outputs, torch.tensor(y_test,
dtype=torch.float32).view(-1, 1))
    # Append testing loss
    test_losses.append(test_loss.item())
    # Calculate metrics (MSE, MAE, RMSE, MAPE)
    predicted_speed = test_outputs.detach().numpy().flatten()
    mse = mean_squared_error(y_test, predicted_speed)
    mae = mean_absolute_error(y_test, predicted_speed)
    rmse = np.sqrt(mse)
    epsilon = 1e-7  # Small epsilon value to avoid division by zero
    mape = np.mean(np.abs((y_test - predicted_speed)/(y_test+epsilon)))*100
    #mape = np.mean(np.abs((y_test - predicted_speed) / y_test)) * 100
    mse_values.append(mse)
    mae_values.append(mae)
    rmse_values.append(rmse)
    print(f"Epoch {epoch + 1}/{num_epochs}, Train Loss: {train_loss}, Test
Loss: {test_loss.item()}, MSE: {mse}, MAE: {mae}, RMSE: {rmse}, MAPE:
{mape}")
# Visualize training loss, testing loss, and validation loss over epochs
epochs = range(1, num_epochs + 1)
plt.plot(epochs, train_losses, label='Training Loss')
plt.plot(epochs, test_losses, label='Testing Loss')

plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.title('Training and Testing Loss over Epochs')
plt.legend()
plt.show()
```

Приложение 3

«Акты внедрения результатов диссертации»

Адрес: Ирак, Вавилон, Ул. Аль-Джамия, оф.89

Тел +9647818070538

Email: alrayyancompany1981@gmail.com

Компания "Аль Райян"

"ТОРГОВО-ТРАНСПОРТІВАЯ КОМПАНИЯ'

Общество с ограниченной ответственностью

04 /06/ 2024

А К Т

об использовании результатов научных исследований, проведенных
Аль-Саиди Аднан Адаб К.
в диссертационной работе
« **Метод и алгоритмы планирования маршрутов движения автономного карьерного транспорта с использованием параллельных вычислительных процедур»,**
представленной на соискание ученой степени кандидата технических наук по специальности «Системный анализ, управление и обработка информации, статистика»

Мы, нижеподписавшиеся, представители компания "Аль Райян", учитывая важность и потенциал кандидатской диссертации "Метод и алгоритмы планирования маршрутов движения автономного карьерного транспорта с использованием параллельных вычислительных процедур", выполненной студентом Аль-Саиди Аднан Адаб К. в рамках программы обучения по направлению Системный анализ, управление и обработка информации, Статистика на кафедре Автоматизированные системы управления университета Национальный исследовательский технологический университет «МИСиС», заявляем следующее:

1. Важность и актуальность темы кандидатской диссертации

Тема работы, посвященная разработке методов и алгоритмов планирования маршрутов движения автономного карьерного транспорта, является крайне актуальной в контексте современных вызовов в области транспортной логистики. Эффективное планирование маршрутов имеет критическое значение для повышения производительности и снижения затрат в сфере транспортных услуг.

2. **Возможности применения результатов работы в нашей компании**

Результаты и разработанные алгоритмы предоставляют ценные инструменты для оптимизации маршрутов и повышения эффективности наших автономных карьерных транспортных средств. Использование параллельных вычислительных процедур позволит нам значительно улучшить скорость и точность планирования маршрутов, что приведет к сокращению времени доставки, оптимизации расхода топлива и снижению операционных издержек.

3. **Извлечение пользы из экспериментов с программным обеспечением**

Проведение экспериментов с использованием программного обеспечения, разработанного в рамках кандидатской диссертации, позволит нам оценить эффективность новых методов и алгоритмов на практике. Мы сможем адаптировать эти результаты к конкретным условиям и требованиям нашей компании, что в свою очередь приведет к повышению нашей конкурентоспособности и улучшению обслуживания клиентов.

Мы подтверждаем, что внедрение результатов кандидатской диссертации представляет собой значимый шаг в направлении модернизации нашей транспортной компании и повышения ее эффективности.

Директор компании

Al Rayyan Company
For trade and transportation