Федеральное государственное бюджетное образовательное учреждение высшего образования «Сибирский государственный университет науки и технологий имени академика М.Ф. Решетнева»

На правах рукописи

Ахматшин Фарид Галиуллович

МОДЕЛИ И АЛГОРИТМЫ АВТОМАТИЧЕСКОЙ ГРУППИРОВКИ ОБЪЕКТОВ ДЛЯ СИСТЕМ АНАЛИЗА И ХРАНЕНИЯ ДАННЫХ НА ОСНОВЕ МЕТОДОВ СЕМЕЙСТВА *k*-СРЕДНИХ

2.3.1 – Системный анализ, управление и обработка информации, статистика

ДИССЕРТАЦИЯ

на соискание ученой степени кандидата технических наук

Научный руководитель
доктор технических наук, профессор
Казаковцев Л.А.

Красноярск – 2025

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Актуальность. Прогнозируемый объем данных, накапливаемых человечеством, с каждым годом увеличивается в геометрической прогрессии. Современное и будущее развитие технологий анализа и хранения информации связано, в том числе, с использованием методов автоматической группировки (кластеризации) данных и приближенного поиска ближайших соседей (ANN – Approximate Nearest Neighbors) как ценного источника новых знаний о структуре хранимой информации. Автоматическое получение информации о сходстве и компактности (упорядоченности и однородности) данных стимулирует процесс получения новой информации в области анализа и хранения данных.

Задачи автоматической группировки (кластеризации) применяются в быстро развивающихся сферах деятельности, например при анализе данных неразрушающих тестовых испытаний промышленной продукции, в системах архивации данных в системах хранения данных, в векторных базах данных.

Традиционные методы приближенного поиска ближайших соседей, в том числе с применением локально-чувствительного хэширования (Locality Sensitive Hashing, LSH), снижают вычислительную сложность решения NP-трудных задач автоматической группировки. В таких задачах кластерная структура упорядоченных однородных наборов данных определяется путем оптимизации целевой функции с учетом меры подобия (различия), которую выражают, в том числе, с использованием функции расстояния. При этом способы предварительной подготовки данных, в частности нормализация числовых показателей, иногда имеют решающее значение.

Способы обработки информации (например, кластеризации) исходных данных не всегда применимы напрямую. Для унификации шкал величин исходные данные стандартизируют, применяя затем расстояние Евклида или квадрат расстояния. Выявление шума в исходных данных и обнаружение «выбросов» – одно из требований к методам нормализации исходных данных; при

этом специфические задачи требуют особых способов предобработки данных. Многообразные методы кластерного анализа используют различные способы предобработки информации, комбинируя применение различных метрик, методов вычисления координат и расстояний для поиска оптимальных решений, дающих приемлемое решение для большинства практических задач.

С одной стороны, развитие систем анализа данных требует разработки адекватных методов обработки и предобработки данных (в том числе, нормализации), новых моделей и алгоритмов автоматической группировки для той или иной предметной области. С другой стороны, вследствие увеличения объемов обрабатываемых и анализируемых данных усложняется внутренняя структура систем анализа данных, включая подсистемы хранения данных, требуя применения алгоритмов машинного обучения, в том числе алгоритмов кластерного анализа, для повышения эффективности управления в системах хранения данных (дисковые массивы, векторные базы данных). При этом увеличение объема обрабатываемых данных выявляет низкую вычислительную эффективность существующих алгоритмов. Решению задач в этих областях и посвящена данная работа.

Степень разработанности темы. С предложенной Г. Штейнгаузом реализацией алгоритма С. Ллойда для задачи *k*-средних связано наиболее популярное направление развития методов автоматической группировки данных. Первоначальное название предиктора пакетирования для описания подхода к автоматической кластеризации, открытое Г. Штейнгаузом, приобрело в дальнейшем название *k*-средних, которое ввел Дж. Маккуин. Позже независимо Эдвард У. Форги опубликовал тот же метод. Дальнейшее развитие популярной алгоритмической реализации алгоритма *k*-средних связано с усовершенствованием простых шагов инициализации, классификации и разбиения на кластеры до конвергенции, из которых состоит данный алгоритм. Методы инициализации начальных центров на основе случайного выбора впервые были предложены одновременно с началом использования алгоритма *k*-средних в методах Форги, Спата, и Маккуина. Развитие данных идей продолжено

в методе инициализации *k*-means++ Д. Артуром и С. Василвицким. Дальнейшее развитие популярной алгоритмической реализации связано с модернизацией алгоритма *k*-means++.

Настоящая диссертация направлена на разработку новых алгоритмов автоматической группировки, используемых в системах анализа и хранения данных, на повышение эффективности алгоритмов кластеризации при обработке больших данных в системах автоматической группировки объектов, в том числе в составе векторной СУБД и подсистем компрессии данных в составе систем хранения данных.

Объектом диссертационного исследования являются задачи автоматической группировки объектов для систем анализа и хранения данных, **предметом исследования** – являются алгоритмы для решения данных задач.

Целью исследования является повышение эффективности (вычислительной производительности, а также повышения качества результата по внешним и внутренним критериям качества) алгоритмов автоматической группировки объектов для систем анализа и хранения данных.

Задачи, решаемые в процессе достижения поставленной цели:

1. Разработать подход к нормализации данных для предобработки входных данных, используемых в системах анализа данных результатов неразрушающих испытаний образцов промышленной продукции, комбинирующий нормализацию по допустимым значениям параметров оцениваемых значений продукции и оценке Джеймса-Штейна.

2. Разработать алгоритм кластеризации для системы анализа данных электрорадиоизделий на основе данных тестовых испытаний с использованием жадной эвристической процедуры выбора радиуса локальных концентраций по размеченным данным.

3. Разработать алгоритм кластеризации для создания индекса векторной базы данных предназначенного для построения индекса приближенного поиска ближайших соседей, как компромисс между точностью и временем вычислений,

значительно улучшающий метрику полноты в задачах приближенного поиска ближайших соседей.

4. Разработать алгоритм автоматической группировки повторяющихся фрагментов блоков данных для использования в системах хранения данных на основе алгоритма *k*-средних совместно с применением локально-чувствительного хэширования LSH.

5. Разработать процедуру инициализации центров кластеров для алгоритмов кластеризации, способную быстро находить приемлемое начальное решение при большом объеме данных.

Новые научные результаты, выносимые на защиту:

1. Предложен новый подход к нормализации данных для предобработки входных данных, используемых в системах анализа данных результатов неразрушающих испытаний образцов промышленной продукции, комбинирующий нормализацию по допустимым значениям параметров оцениваемых характеристик продукции и оценки Джеймса–Штейна.

2. Предложен новый алгоритм кластеризации системы анализа данных электрорадиоизделий на основе данных тестовых испытаний с использованием жадной эвристической процедуры выбора радиуса локальных концентраций по размеченным данным.

3. Предложен новый алгоритм кластеризации для построения индекса векторной базы данных для приближенного поиска ближайших соседей, обеспечивающий компромисс между точностью и временем вычислений, существенно улучшает метрику полноты в задачах приближенного поиска ближайших соседей.

4. Предложен новый алгоритм автоматической группировки повторяющихся фрагментов блоков данных на основе алгоритма *k*-средних совместно с локально-чувствительным хэшированием (LSH), обеспечивающий увеличение эффективности сжатия данных в системах хранения данных.

5. Предложена новая процедура инициализации центров кластеров для алгоритмов автоматической группировки больших объемов данных,

использующая вспомогательную структуру данных – массив слагаемых для вычисления суммы квадратов расстояния.

Значение для теории. Теоретическая значимость состоит в дополнении эффективных алгоритмов решения задач автоматической группировки, а также алгоритмов предобработки данных для таких задач.

Практическая ценность состоит в дополнении модельно-алгоритмического инструментария, используемого в системах анализа данных результатов тестирования образцов промышленной продукции с повышенными требованиями качества, в частности – электронной компонентной базы космического применения, и могут использоваться в соответствующих испытательных технических центрах. Кроме того, новая процедура инициализации центров кластеров для алгоритмов кластеризации, имеет универсальный характер и может применяться при обработке больших данных в любых системах автоматической группировки объектов. Новые алгоритмы кластеризации применяются для построения индекса для векторной базы данных и для разработки модели оптимального использования дискового пространства с учетом компрессии данных.

Методы исследования. Результаты прикладных и теоретических задач получены с применением методов системного анализа, исследования операций, теории размещения, теории оптимизации.

Положения, выносимые на защиту:

1. Подход к нормализации данных для предобработки входных данных, используемых в системах анализа данных результатов неразрушающих испытаний образцов промышленной продукции, комбинирующий нормализацию по допустимым значениям параметров оцениваемых характеристик продукции и оценки Джеймса–Штейна, обеспечивает повышение точности решения задачи кластеризации на 10% по индексу Рэнда, на примере задачи автоматической группировки электрорадиоизделий.

2. Алгоритм кластеризации для системы анализа данных электрорадиоизделий на основе данных тестовых испытаний с использованием

жадной эвристической процедуры выбора радиуса локальных концентраций по размеченным данным обеспечивает повышение точности (по индексу Рэнда) и скорости получения результатов автоматической группировки по сравнению с алгоритмом *k*-средних.

3. Алгоритм кластеризации для построения индекса векторной базы данных для приближенного поиска ближайших соседей обеспечивает компромисс между точностью и временем вычислений, существенно улучшает метрику полноты в задачах приближенного поиска ближайших соседей.

4. Алгоритм автоматической группировки повторяющихся фрагментов блоков данных на основе алгоритма *k*-средних совместно с локально-чувствительным хэшированием (LSH), обеспечивает увеличение эффективности сжатия данных в системах хранения данных.

5. Процедура инициализации центров кластеров для алгоритмов автоматической группировки больших объемов данных, использует вспомогательную структуру данных – массив слагаемых для вычисления суммы квадратов расстояния, обеспечивает снижает вычислительные затраты в сравнении с алгоритмом k-means++ без снижения качества получаемого начального решения.

Практическая реализация результатов: программная разработка алгоритма автоматической группировки повторяющихся фрагментов блоков данных на основе алгоритма *k*-means, совместно с локально-чувствительным хэшированием (LSH) предназначена для использования в системах хранения данных. Она реализована в разработанной модели оптимального использования дискового пространства с учетом компрессии данных в рамках работ по договору №20769 от 05.10.2023 г. Исследование по разработке нового алгоритма кластеризации, основанного на жадной агломеративной процедуре, для построения индекса для векторной базы данных выполнено по договору № ТС2024051430 от 14.06.2024г. Исследование по разработке новой процедуры инициализации центров кластеров для алгоритмов кластеризации, применяемых к

большим данных выполнено при поддержке Министерства науки и высшего образования Российской Федерации (проект FEFE-2023-0004).

Апробация работы. Основные положения и результаты диссертационной работы докладывались и обсуждались на международных семинарах и конференциях: «Решетневские чтения» (2020 г., г. Красноярск), Mathematical Optimization Theory and Operations Research (MOTOR, 2023 – 2024 гг., г. Екатеринбург, г. Омск), семинар «Математические модели принятия решений» Институт математики имени С.Л.Соболева, (2024 г., г.Новосибирск).

Публикации. Основные теоретические и практические результаты диссертации содержатся в 14 публикациях, среди которых 5 работ в ведущих рецензируемых журналах, рекомендуемых в действующем перечне ВАК, 5 – в международных изданиях, индексируемых в системах цитирования Web of Science и Scopus. Имеется свидетельство о государственной регистрации программы для ЭВМ.

1. НОРМАЛИЗАЦИЯ ДАННЫХ В СИСТЕМАХ АВТОМАТИЧЕСКОЙ ГРУППППИРОВКИ ПРОМЫШЛЕННОЙ ПРОДУКЦИИ

В разделе 1 рассмотрены вопросы нормализации данных в задаче автоматической группировки промышленной продукции на примере электрорадиоизделий (ЭРИ) по однородным производственным партиям, основанной на модели *k*-средних, а также разработке нового подхода по нормализации данных промышленной продукции, комбинирующего нормализацию по допустимым значениям параметров оцениваемых характеристик продукции и оценку Джеймса-Штейна. Результаты данного раздела были опубликованы в [1, 2].

1.1 Современные методы нормализации данных о промышленной продукции

Рассмотрим задачу автоматической группировки (выделения однородных партий) промышленной продукции на примере электрорадиоизделий (ЭРИ), предназначенных для комплектации бортовой аппаратуры космических аппаратов (КА). Высокие требования предъявляются к ЭРИ, их надежности и долговечности. Повышение требований к качеству промышленных изделий ЭРИ свидетельствует о применении современных технологии в процессе производственных испытаний. Развитие технологического процесса обеспечивает выпуск партий ЭРИ со стабильными параметрами отдельных экземпляров продукции.

Надежные экземпляры ЭРИ выделяют в специализированных тестовых центрах [3, 4]. Контроль характеристик ЭРИ в них предотвращает [5-7] отказы - скачкообразный (внезапный) или постепенный выход (дрейф) параметров изделия за установленные пределы. Исключение изменения (дрейфа) параметров в полупроводниковых приборах [5, 8] снижает интенсивность отказов выборки за счет дополнительных отбраковочных испытаний (ДОИ) у потребителя [9-10].

Обязательная оценка дрейфа параметров [4] в процессе электротермотренировки (ЭТТ) является наиболее эффективным испытанием.

В отрасли с высокими требованиями к качеству продукции контроль качества промышленной продукции требует получения наиболее точного и стабильного результата в отрасли. Точность относится к снижению доли ошибок автоматической группировки, а стабильность – к повторяемости результатов при многократном запуске алгоритма.

Анализ результатов тестовых испытаний ЭРИ производится с целью комплектации критически важных электронных узлов КА компонентной базой (ЭКБ) соответствующего качества. В ЭКБ все однотипные элементы схемы должны иметь близкие характеристики для обеспечения их согласованной работы, что наилучшим образом достигается в случае, если элементы изготовлены в рамках одной производственной партии из единой партии сырья [11]. Соотносить конкретные экземпляры изделий с выявленными группами изделий – эта задача, в результатах решения которой должна обеспечиваться стабильность и высокая точность получаемых результатов за приемлемое время. Модель *k*-средних в данной задаче хорошо зарекомендовала себя [3, 12-13], достигнута достаточно высокая точность разбиения на однородные партии.

В специализированных испытательных центрах проводятся сотни тестов для комплектации бортового оборудования космических систем высоконадежной электронной компонентной базой и анализа каждого полупроводникового устройства. Одним из требований является то, что отгружаемая партия продукции должна быть изготовлена из одной партии сырья (пластин), что не гарантируется для устройств, используемых в космической промышленности. Различные алгоритмы кластеризации реализуются на основе многомерных результатов тестирования для решения задачи обнаружения однородных партий продукции [12, 14 – 17].

Решение задачи *k*-средних [3] в многомерном пространстве состоит в том, чтобы найти координаты k точек (центров или центроидов) $x_1, \ldots, x_k$ в *M*-мерном пространстве таким образом, чтобы сумма квадратов расстояний от известных

координат точек (векторов данных) $a_1, \ldots, a_N$ до ближайшей из требуемых точек достигла своего минимума $argminF(x_1, \ldots x_k) = \sum_{i=1}^{N} min_{j \in (1,k)} \|x_j - a_i\|^2$.

Поскольку результаты измерений различных параметров имеют различный диапазон значений и разные единицы измерения, проводится нормализация данных. При этом для нормализации используются данные не только исследуемой партии изделий, разброс показаний одного отдельно взятого параметра может быть очень незначительным, но и предыдущих исследованных партий ЭРИ, для учета статистических характеристик изделий, не связанных с особенностями изготовления конкретной исследуемой партии.

В ранних исследованиях [4, 1] сравнивались различные способы нормализации для решения задачи автоматической группировки ЭРИ по однородным производственным партиям, основанной на модели *k*-средних. Производилось сравнение различных способов нормализации данных в решении задачи *k*-средних, изучались результаты точности кластеризации, и был предложен новый способ нормализации по допустимым значениям параметра многомерных данных.

Использование многомерных данных в модели автоматической группировки влияет на точность решения задачи. Между некоторыми характеристиками часто существуют явные корреляции. Задача кластеризации нормализованных данных эффективно решается с помощью алгоритма *k*-средних со специальными мерами расстояния, в которых учитываются эти зависимости.

В исходных многомерных данных могут присутствовать выбросы, которые влияют на интерпретацию полученных результатов. Для снижения влияния выбросов применяется нормализация значений исходных данных [18].

Методы нормализации оказывают большое влияние на решение задачи автоматической группировки изделий промышленной продукции по однородным производственным партиям. При этом каждый из продуктов имеет большое количество входных характеристик. Характеристики с такими диапазонами значений также оказывают значительное влияние на группировку образцов промышленной продукции. В связи с этим возникает задача выработки такого

подхода к нормализации данных в результате тестовых испытаний образцов промышленной продукции, который бы обеспечивал повышение точности решения задачи по выделению однородных партий промышленной продукции. Нашей целью является разработка адекватных методов обработки и предобработки данных (в том числе, нормализации).

1.2 Подход к нормализации на основе данных промышленной продукции

В моделях *k*-средних [17] могут применяться различные меры расстояния [19-20]. Функции расстояния и их определение играют важную роль в задаче кластеризации. В нашем исследовании мы используем квадрат расстояния Евклида и квадрат расстояния Махаланобиса [19-21]. Пусть n_j – количество точек (векторов данных) в *j*-ом кластере, координата центра *j*-ого кластера равна среднему значению нормализованных данных $x'_j = \frac{1}{n_j}\sum_{i=1}^{n_j} a'_i$ в этом кластере. Обозначим квадрат расстояния Евклида d_E, от x_j до a_i, как $d_E(x_j, a_i) = \|x_j - a_i\|^2$, а квадрат расстояния Махаланобиса d_M –как $d_M(x_j, a_i) = (x_j - a_i)^T S^{-1}(x_j - a_i)$, где *S* – ковариационная матрица.

Обозначим квадрат расстояния Евклида d_E, от x_j до a_i, как $d_E(x_j, a_i) = \|x_j - a_i\|^2$. А квадрат расстояния Махаланобиса d_M –как $d_M(x_j, a_i) = (x_j - a_i)^T S^{-1}(x_j - a_i)$, где *S* – ковариационная матрица.

В случае задач кластеризации квадратичные расстояния Евклида являются наиболее популярными [21]. Для вычисления разностей (расстояний) в нормализованном пространстве характеристик, также используем расстояние Махаланобиса для сопоставления результатов [25]. Алгоритмы автоматической группировки объектов на основе оптимизационной модели *k*-средних с мерой расстояния Махаланобиса позволяют снизить долю ошибок (повысить индекс

Рэнда) при выявлении однородных производственных партий продукции по результатам тестовых испытаний [25].

Расстояние Махаланобиса является масштабно-инвариантным [22]. Благодаря этому свойству нормализация данных не имеет значения, если применяется это расстояние. В то же время специальный метод нормализации по границам допустимых значений дрейфа показал высокую эффективность. Нормализация по интервалу значений (нормализация 0-1) или нормализация по стандартному отклонению уравнивает значимость всех параметров, неизбежно увеличивая значимость неинформативных параметров, содержащих только шум. Привязка границ параметров к границам, определяемым их физической природой, устанавливает шкалу, пропорциональную допустимым отклонениям этих параметров в условиях эксплуатации (допустимый дрейф параметров), без привязки к размаху и дисперсии этих значений в конкретной производственной партии. При переходе на расстояние Махаланобиса эти преимущества теряются. Решением проблемы может быть применение нового способа нормализации данных $(a_1,\dots a_N)$ с использованием диапазона допустимых значений h контролируемых параметров (КП) соответствующих режимов испытаний $a' = \frac{(a-a_{min})}{h}$, где a' – координаты вектора нормализованных значений измеряемого параметра.

Исследование алгоритма *k*-средних в рассматриваемой области связано с выбором методов нормализации данных, метрики расстояния и инициализации центра [23, 13, 24, 25].

Автоматическая группировка выполняется над исходными данными с применением различных мер расстояния. Однако расчет по метрике Евклида должен производиться над данными, параметры которых должны быть нормализованы, а также для исключения влияния неинформативных параметров. Исследование алгоритма автоматической группировки связано с выбором метода нормализации данных, выбором метрики расстояния и выбором метода инициализации центров.

Пусть набор данных $a_1, \ldots, a_N$ состоит из N точек. Обозначим центры, полученные после применения автоматической группировки *k*-средних $x_1, \ldots, x_j, \ldots, x_k$. Цель алгоритма автоматической группировки состоит в том, чтобы сумма квадратов расстояний от известных точек до ближайших центров достигла своего минимума:

$$\operatorname{argmin} F\left(x_1, \ldots, x_j, \ldots, x_k\right) = \sum_{i=1}^{N} \min_{j \in \{1,\ldots,k\}} \left\| x_j - a_i \right\|^2. \qquad (1.1)$$

Для минимизации суммы квадратов расстояний в алгоритме *k*-средних центры итеративно назначаются и обновляются [26, 27]. При этом алгоритм *k*-средних применяется для решения задачи автоматической группировки, сводимой к задаче минимизации целевой функции, которая эквивалентна задаче нахождения квадрата расстояния от координат точек до ближайшего центра кластера [28–30].

Предлагаемый в настоящей работе подход заключается в использовании улучшенной оценки усадки Джеймса–Штейна. Для этого уменьшим влияние неинформативных параметров, нормализованных исходных данных $a'_1, \ldots, a'_N$, для соответствующих способов нормализации $a' = \frac{a - a_{min}}{h}$, где a' – координаты вектора нормализованных значений измеряемого параметра.

Для минимизации целевой функции и улучшения нормализованных значений сжимаем значения параметров в направлении среднего значения выборки μ. Сжимаем нормализованные значения выборки путем уменьшения больших значений и увеличения меньших. Для этого используем базовую формулу оценки Джеймса-Штейна $a'' = (\quad)^+(a' - x) + x$, в формуле (1.2) используем фактор усадки $(\quad)^+$, а также значения координат точек a' в кластере относительно центра x.

Для улучшения нормализованных значений a' в кластере мы используем оценку Джеймса-Штейна для сжатия a'_i до среднего значения μ всего набора данных:

$$a''_i = \left(1 - \frac{(p-2)t\sigma^2}{\|a'_i - \mu\|^2}\right)^+ (a'_i - \mu) + \mu, \qquad (1.2)$$

где $\mu = \frac{1}{N}\sum_{i=1}^{N} a'_i$ – среднее значение выборки; $(\quad)^+$ – оценка с положительной частью, равной нулю при отрицательных значениях; σ^2 – дисперсия значений a'; t – коэффициент сокращения значения фактора усадки до не нулевого значения. Оценка Джеймса-Штейна способствует уменьшению среднего значения $\|x_j - a_i\|$ в кластере для минимизации значения целевой функции (1.1) за счет применения коэффициента t. С помощью нелинейного преобразования (1.2) мы обнуляем неинформативные параметры, используемые для повышения точности кластеризации. Предполагаем, что среди измеряемых параметров, полученных в результате тестовых испытаний, есть один или два параметра влияющих на увеличение точности кластеризации, и соответственно остальные измеряемые параметры уменьшают точности кластеризации.

Преобразованные данные a''_i используются алгоритмом *k*-средних для решения задачи разделения промышленной продукции на однородные производственные партии. Исследуем эффективность использования сокращения оценки усадки после нормализации данных перед выполнением базового алгоритма *k*-средних. В отличие от результатов исследования [30], в котором выполняется усадка центров относительно начала координат, рассматриваем процесс преобразования положения координат каждой точки относительно центра всего набора данных.

1.3 Об оценке сходства и разнообразия выборочных наборов данных

В качестве меры точности кластеризации мы используем индекс Рэнда (RI – Rand Index) [33], который определяет долю пар объектов, для которых эталонное и результирующее кластерное расщепление аналогичны. В отличие от индекса Рэнда (RI) индекс Жаккара (JI) игнорирует элементы, отсутствующие в обоих исследуемых наборах TN (истинно отрицательные). Однако оба индекса

используют для своего построения результаты разбиения на истинно положительные (TP), ложно-положительные (FP) и ложноотрицательные (FN) результаты.

1.4 Методика сравнительного исследования подходов к нормализации данных

В нашем исследовании мы использовали данные результатов испытаний, выполненных в испытательном центре для партий интегральных схем. Для контроля осуществлялась раздельная проверка принадлежности каждого замеренного контрольного параметра (КП) заданному диапазону, основанного на параметрах ужесточенных норм (УН). Партии ИС прошли дополнительные отбраковочные испытания (ДОИ) с обязательной электротермотренировкой (ЭТТ). Она проводилась в динамическом режиме продолжительное время с повышенными температурами.

Исходные данные представляют собой совокупность некоторых параметров ЭРИ, измеренных в ходе обязательных испытаний. В приводимом ниже примере выборка (смешанная партия) первоначально состояла из данных о продуктах, принадлежащих к разным однородным партиям двух типов интегральных схем 140УД25А и 140УД26А до и после ЭТТ. Общее количество ЭРИ составляет 1228 и 746 шт. для первого и второго типономинала интегральных схем (ИС) соответственно. Набор данных 140УД25А состоит из 18 однородных партий, 140УД26А – из 9 партий. Каждая партия первого типа ИС имеет в своем составе 134, 31, 360, 29, 3, 39, 49, 155, 158, 58, 12, 19, 49, 21, 57, 32, 7 и 15 устройств. Партии второго типа ИС – 53, 33, 21, 182, 123, 32, 161, 75 и 66 устройств. Изделия (устройства) в каждой партии описываются 18 входными измеряемыми параметрами. Кроме того, использовались исходные данные набора тестовых испытаний смешанной партии двух типов интегральных схем (ИС) 140УД25А и 140УД26А до и после электротермической подготовки (ЭТП). Общее количество интегральных схем равно 807 и 532 для первого и второго типа интегральных

схем. Кроме того, были сформированы дополнительные наборы данных меньшего объема в количестве 201 и 132 штуки ИС первого и второго типа соответственно.

Смешанная партия составлена из комбинации устройств, принадлежащих разным однородным партиям. При таком подходе предположительно однородную партию можно разделить на несколько гетерогенных партий. В то же время смешанная партия может быть разделена на большее количество однородных партий, чем это обычно происходит. Меньшие кластеры с большей вероятностью содержат данные одного и того же класса, т. е. уменьшается вероятность ложного отнесения объектов разных классов к одному кластеру. Доля объектов одного класса, ложно отнесенных к разным классам, не столь важна для оценки статистических характеристик однородных групп объектов.

Для выделения предположительно однородных партий с некоторой точностью использовались хорошо известные модели кластерного анализа:

– Модель DM, где с помощью меры расстояния Махаланобиса ковариационная матрица вычисляется для всей обучающей выборки. Целевая функция определяется как сумма квадратов расстояний.

– Модель DE, где *k*-среднее с расстоянием Евклида. Целевая функция определяется как сумма квадратов расстояний.

Предлагаем новый подход по нормализации данных, для чего дополнительно улучшаем средние значения точек в кластере относительно ближайших центров по формуле (1.2).

Использование преобразования Джеймса–Штейна позволит дополнительно выявлять неинформативные параметры. В наборе данных тестовых испытаний ИС, состоящем из 18 измеряемых параметров, вручную были исключены некоторые неинформативные параметры, так как они содержали данные об установочных параметрах, одинаковых для всех изделий. Неинформативными параметрами после преобразования Джеймса–Штейна считаем те, которые обнуляются в результате нелинейного преобразования (1.2). Считаем, что после применения (1.2) остаются не обнуленными информативные параметры, так как

использование не обнуленных параметров влияет на увеличение точности кластеризации.

Рассматриваемый набор данных сгруппирован с использованием нормализации по стандартному отклонению (A), для которого использовались данные от 3 до 18 параметров, причем параметры 1 и 2 являлись неинформативными и в дальнейших расчетах не учитывались. Вектор данных нормализованных по стандартному отклонению интегральных схем (ИС) 140УД25А и 140УД26А может быть представлен по формуле

$$a_{std} = (0, 0, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, a_{14}, a_{15}, a_{16}, a_{17}), \quad (1.3)$$

что показывает использование значений с a_2 до a_{17}, с коэффициентом сокращения оценки усадки Джеймса–Штейна $t = 0$.

При нормализации по значениям допустимого дрейфа (B) использовались только параметры 3–6, для которых установлены соответствующие нормы дрейфа (другие параметры были обнулены). Изменение значений параметров 3–6 до и после ЭТТ учтена путем добавления параметров 19–22 (величин дрейфа параметров ЭТТ). Вектор данных с использованием нормализации по значениям допустимого дрейфа интегральных схем (ИС) 140УД25А и 140УД26А может быть представлен как

$$a_{vpd} = (0, 0, a_2, a_3, a_4, a_5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, a_{18}, a_{19}, a_{20}, a_{21}), \quad (1.4)$$

что показывает использование значений с a_2 до a_5 и с a_{18} до a_{21}, с коэффициентом сокращения оценки усадки Джеймса–Штейна $t = 0$.

Набор данных с использованием нормализации по допустимым значениям параметров (C) был рассчитан для 3–16 параметров, для которых были установлены соответствующие нормы, значения остальных параметров были обнулены. Разница в изменении 3–6 параметров до и после ЭТТ учтена путем добавления 19–22 параметров. Вектор данных с использованием нормализации по допустимым значениям параметров интегральных схем (ИС) 140УД25А и 140УД26А может быть представлен следующем образом

$$a_{apv} = (0, 0, a_2, a_3, a_4, a_5, a_6, \dots, a_{11}, a_{12}, a_{13}, a_{14}, a_{15}, 0, 0, a_{18}, a_{19}, a_{20}, a_{21}), \quad (1.5)$$

что показывает использование значений с a_2 до a_{15} и с a_{18} до a_{21}, с коэффициентом сокращения оценки усадки Джеймса–Штейна $t = 0$.

Таблица 1.1 – Характеристики сформированных наборов данных

Обозначение набора данных	Общее кол-во	Кол-во элементов однородных партий изделий	Способ нормализации	Кол-во параметров	Кол-во информативных параметров	Тип ИС
N_1, N_{13}	807	134, 155, 158, 360	A	18	16	140УД25А
N_2, N_{14}	201	33, 39, 39, 90				
N_3, N_{15}	807	134, 155, 158, 360	B	22	8	
N_4, N_{16}	201	33, 39, 39, 90				
N_5, N_{17}	807	134, 155, 158, 360	C	22	18	
N_6, N_{18}	201	33, 39, 39, 90				
N_7, N_{19}	532	182, 123, 161, 66	A	18	16	140УД26А
N_8, N_{20}	132	45, 31, 40, 16				
N_9, N_{21}	532	182, 123, 161, 66	B	22	8	
N_{10}, N_{22}	132	45, 31, 40, 16				
N_{11}, N_{23}	532	182, 123, 161, 66	C	22	18	
N_{12}, N_{24}	132	45, 31, 40, 16				

Таким образом, сформировано 24 наборов данных $N_1, \dots, N_{24}$ (таблица 1.1) по 12 для каждого типа ИС, сгруппированных с использованием нормализации по стандартному отклонению (A), по значениям допустимого дрейфа (B) и по допустимым значениям параметров (C). Наборы данных двух типов ИС до ЭТТ для ИС 140УД25А с N_1 до N_6, и для ИС 140УД26А с N_7 до N_{12}. А также наборы данных двух типов ИС после ЭТТ для ИС 140УД25А с N_{13} до N_{18}, и для ИС 140УД26А с N_{19} до N_{24}.

Для каждого набора данных проведены эксперименты с использованием алгоритма кластеризации *k*-средних. При этом исследовали влияние коэффициента сокращения оценки усадки Джеймса–Штейна $t = (0, \dots, 100)$ на

изменение точности кластеризации для различных способов нормализации данных. С использованием нового подхода к нормализации данных, сжимаем значения параметров в направлении среднего значения выборки μ, по формуле (1.2) оценки Джеймса–Штейна с коэффициентом t.

1.5 Результаты вычислительных экспериментов

В следующем эксперименте использовались наборы данных $N_1, \dots, N_{24}$. Наборы данных обрабатывались по формуле (1.2).

Для исследования нового способа нормализации использовались изделия с 18 измеряемыми в ходе тестирования параметрами, среди которых первые два параметра не информативны и занулялись, потому что содержали данные об установочных параметрах, одинаковых для всех изделий.

Были сформированы наборы данных N_1, N_2, N_7, N_8, N_{13}, N_{14}, N_{19}, N_{20} с использованием нормализации по стандартному отклонению, для которого использовались данные с 3 по 18 параметра, параметры 1 и 2 не информативны и занулялись.

Для наборов данных N_3, N_4, N_9, N_{10}, N_{15}, N_{16}, N_{21}, N_{22} нормализация по значениям допустимого дрейфа рассчитывалась только для 3-6 параметров, для которых соответствующие нормы были установлены, остальные данные занулялись. Разность изменения 3-6 параметров до и после ЭТТ учитывались добавлением 19-22 параметра.

Для наборов данных N_5, N_6, N_{11}, N_{12}, N_{17}, N_{18}, N_{23}, N_{24} нормализация по допустимым значениям параметра рассчитывалась для 3-16 параметра, для которых соответствующие нормы были установлены, остальные данные занулялись. Разность изменения 3-6 параметров до и после ЭТТ учитывались добавлением 19-22 параметра.

Результаты были получены методом кластеризации *k*-средних с квадратом расстояния Махаланобиса и Евклида. Для каждой модели мы провели 30 экспериментов. Сравнение результатов кластеризации с различными

показателями расстояния, количества точных попаданий до ЭТТ приведены в таблицах 1.1 и 1.2, после ЭТТ в таблицах 1.3 и 1.4, представлены максимальное (*Max*), минимальное (*Min*), среднее значение (*Mean*) и стандартное отклонение (σ) для индекса Рэнда и целевой функции. Для целевой функции также рассчитываются коэффициент вариации (*V*) и коэффициент размаха (*R*).

Таблица 1.2 – Результаты сравнительных экспериментов по кластеризации полной смешанной партии (количество данных *N*=1228) ИС 140УД25А до ЭТТ

	Нормализация по стандартному отклонению		Нормализация по значениям допустимого дрейфа		Нормализация по допустимым значениям параметра	
	DM	DE	DM	DE	DM	DE
Индекс Рэнда						
Max	0,824	0,823	0,826	0,831	0,809	0,823
Min	0,799	0,814	0,786	0,795	0,786	0,810
Mean	0,812	0,819	0,813	0,817	0,800	0,818
σ	0,006	0,003	0,009	0,010	0,006	0,003
Целевая функция (справочно, сравнение значений не проводилось)						
Max	13403	18046	1331	1365	11154	9747
Min	12132	16725	653	725	9280	8952
Mean	12843	17135	1072	966	10408	9369
σ	348	290	159	206	457	174
V	0,027	0,016	0,149	0,213	0,043	0,018
R	1270	1320	678	640	1873	795

Согласно полученным значениям индекса Рэнда, результаты с использованием подхода по нормализации по допустимым значениям параметра статистически не значимы среди представленных подходов (Таблица 1.2 и 1.3) во всех сериях экспериментов. То есть, в первой серии экспериментов выявлено преимущество использования нормализации по значениям допустимого дрейфа. Однако далее в Разделе 1 будет показано преимущество (совместно с оценкой Джеймса–Штейна) использования данного подхода к нормализации по

допустимым значениям параметра по сравнению с подходом по нормализации по стандартному отклонению и по нормализации по значениям допустимого дрейфа.

Предложенный подход с нормализацией по допустимым значениям параметра в модели кластеризации *k*-средних с квадратом расстояния Махаланобиса и Евклида, был сопоставлен с подходом с использованием нормализации по стандартному отклонению и нормализации по значениям допустимого дрейфа. Учитывая более высокое среднее значение индекса Рэнда подход с нормализацией по допустимым значениям параметра, применяемый для кластеризации электрорадиоизделий по однородными производственными партиям, имеет преимущество перед традиционной нормализацией по стандартному отклонению и по значениям допустимого дрейфа.

Таблица 1.3 – Результаты сравнительных экспериментов по кластеризации полной смешанная партия (количество данных *N*=746) ИС 140УД26А до ЭТТ

	Нормализация по стандартному отклонению		Нормализация по значениям допустимого дрейфа		Нормализация по допустимым значениям параметра	
	DM	DE	DM	DE	DM	DE
Индекс Рэнда						
Max	0,824	0,795	0,878	0,902	0,782	0,792
Min	0,754	0,760	0,819	0,842	0,747	0,766
Mean	0,800	0,783	0,861	0,867	0,766	0,782
σ	0,016	0,008	0,012	0,016	0,007	0,005
Целевая функция (справочно, сравнение значений не проводилось)						
Max	9596	19658	1340	1450	8213	11246
Min	8991	17611	641	841	7499	9907
Mean	9312	18008	888	1081	7870	10467
σ	161	406	153	189	177	360
V	0,017	0,022	0,172	0,174	0,022	0,034
R	605	2047	698	608	714	1338

Сравнительный анализ эффективности нового подхода с использованием нормализации по стандартному отклонению (A), с использованием нормализации по значениям допустимого дрейфа (B), с использованием нормализации по допустимым значениям параметров (C), приведено в таблицах 1.6–1.9, где показаны максимум (*Max*), минимум (*Min*), средние значения (*Mean*), медиана (*Median*) и стандартное отклонение (*StD*) для индекса Рэнда (*RI*). Средние результаты кластеризации *k*-средних с использованием нормализации по стандартному отклонению (A), значениям допустимого дрейфа (B), а также по допустимым значениям параметров (C) с различным коэффициентом *t* сокращения оценки усадки Джеймса–Штейна для ИС 140УД25А (размер набора данных $N = 201$) представлены в таблицах 1.6–1.7 и на рисунках 1.1–1.4.

Таблица 1.4 – Результаты сравнительных экспериментов кластеризации полной смешанной партии (количество данных *N*=1228) ИС 140УД25А после ЭТТ

	Нормализация по стандартному отклонению		Нормализация по значениям допустимого дрейфа		Нормализация по допустимым значениям параметра	
	DM	DE	DM	DE	DM	DE
Индекс Рэнда						
Max	0,834	0,830	0,844	0,826	0,810	0,822
Min	0,816	0,820	0,832	0,812	0,791	0,800
Mean	0,828	0,826	0,837	0,819	0,800	0,816
σ	0,004	0,002	0,002	0,003	0,004	0,003
Целевая функция (справочно, сравнение значений не проводилось)						
Max	14093	19516	4689	18245	11941	9905
Min	12799	18478	4099	17214	9276	8940
Mean	13388	18732	4346	17654	10421	9278
σ	333	205	175	258	507	198
V	0,024	0,010	0,040	0,014	0,048	0,021
R	1294	1038	590	1031	2215	965

Таблица 1.5 – Результаты сравнительных экспериментов кластеризации полной смешанной партии (количество данных *N*=746) ИС 140УД26А после ЭТТ

	Нормализация по стандартному отклонению		Нормализация по значениям допустимого дрейфа		Нормализация по допустимым значениям параметра	
	DM	DE	DM	DE	DM	DE
Индекс Рэнда						
Max	0,818	0,799	0,848	0,806	0,839	0,798
Min	0,772	0,781	0,806	0,770	0,779	0,782
Mean	0,796	0,789	0,813	0,788	0,823	0,790
σ	0,010	0,004	0,010	0,008	0,011	0,004
Целевая функция (справочно, сравнение значений не проводилось)						
Max	9557	17047	3525	12717	10948	30240
Min	9066	15850	2899	11518	10345	29060
Mean	9314	16266	3149	12013	10602	29629
σ	117	306	176	317	151	277
V	0,012	0,018	0,056	0,026	0,014	0,009
R	490	1196	625	1199	603	1179

В рассматриваемом примере набор данных N_2 при коэффициенте сокращения оценки усадки Джеймса–Штейна $t = 20$ (Таблица 1.6) показывает наилучшее значение точности кластеризации по индексу Рэнда при минимальном значении стандартного отклонения целевой функции. Вектор данных нормализованных по стандартному отклонению и с применением сокращения оценки усадки Джеймса-Штейна может быть представлен по формуле

$$a_{std} = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, a_{16}, 0), \quad (1.6)$$

что показывает наибольшее влияния параметра 16.

Набор данных N_4 неэффективен для кластеризации с использованием нового подхода. Вектор данных нормализованных по значениям допустимого дрейфа с применением сокращения оценки усадки Джеймса-Штейна показывает уменьшение влияния значений всех параметров с коэффициентом сокращения оценки усадки Джеймса–Штейна $t > 4$. В результате применения оценки

Джеймса–Штейна совместно с нормализацией по значениям допустимого дрейфа не было выявлено информативных параметров, влияющих на увеличение точности кластеризации.

Таблица 1.6 – Результаты кластеризации для ИС 140УД25А с использованием нормализации по стандартному отклонению (A) для набора данных N_2 с коэффициентом t сокращения оценки усадки Джеймса–Штейна (JS)

t	Значение целевой функции *F* (справочно)							Индекс Рэнда (RI)				
t	*Max*	*Min*	*Mean*	*Median*	*StD*	*Var*	*Spn*	*Max*	*Min*	*Mean*	*Median*	*StD*
0	77,428	66,811	68,663	66,813	2,746	0,040	10,617	0,605	0,559	0,592	0,564	0,011
1	24,803	20,604	21,491	20,604	0,829	0,039	4,198	0,573	0,417	0,468	0,419	0,055
2	12,597	10,602	11,005	10,606	0,376	0,034	1,995	0,562	0,337	0,456	0,339	0,059
3	6,856	6,130	6,392	6,167	0,185	0,029	0,726	0,622	0,368	0,467	0,381	0,057
4	5,324	4,855	5,030	4,862	0,110	0,022	0,470	0,622	0,396	0,496	0,401	0,086
5	4,635	3,960	4,205	4,007	0,148	0,035	0,675	0,645	0,386	0,553	0,393	0,093
6	3,858	3,303	3,491	3,310	0,146	0,042	0,555	0,637	0,372	0,547	0,374	0,097
7	3,626	2,842	3,033	2,854	0,192	0,063	0,784	0,669	0,351	0,594	0,358	0,067
8	3,360	2,527	2,682	2,530	0,177	0,066	0,833	0,710	0,348	0,602	0,461	0,057
9	2,763	2,218	2,396	2,236	0,134	0,056	0,546	**0,720**	0,547	0,633	0,563	0,038
10	2,339	1,920	2,070	1,921	0,128	0,062	0,419	0,701	0,378	0,620	0,381	0,073
20	0,879	0,579	0,651	0,584	0,074	0,113	0,300	0,690	**0,569**	**0,654**	**0,587**	0,027
30	0,901	0,448	0,596	0,449	0,141	0,237	0,452	0,671	0,511	0,618	0,547	0,032
40	1,163	0,380	0,538	0,388	0,172	0,320	0,783	0,647	0,299	0,582	0,383	0,065
50	0,805	0,343	0,451	0,346	0,087	0,193	0,462	0,636	0,466	0,571	0,468	0,041
60	0,694	0,290	0,438	0,296	0,128	0,291	0,404	0,605	0,405	0,518	0,405	0,058
70	0,738	0,248	0,380	0,256	0,124	**0,326**	0,490	0,573	0,299	0,489	0,345	0,056
80	0,662	0,236	0,399	0,238	0,125	0,313	0,426	0,547	0,299	0,444	0,299	0,063
90	0,493	0,196	0,311	0,198	0,089	0,288	**0,296**	0,573	0,375	0,457	0,375	0,058
100	2,339	1,920	2,070	1,921	0,128	0,062	0,419	0,701	0,378	0,620	0,381	0,073

Набор данных N_6 при коэффициенте сокращения оценки усадки Джеймса–Штейна $t = 40$ показывает наилучшее значение точности кластеризации по индексу Рэнда и минимальное значение стандартного отклонения целевой

функции. Вектор данных нормализованных по допустимому значению параметра и с применением сокращения оценки усадки Джеймса-Штейна может быть представлен по формуле

$$a_{std} = (0, 0, 0, 0, a_4, a_5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0), \quad (1.7)$$

показывает наибольшее влияние параметров 4–5.

Таблица 1.7 – Результаты кластеризации для ИС 140УД25А с использованием нормализации по допустимому значению параметра (C) для набора данных N_6 с коэффициентом t сокращения оценки усадки Джеймса–Штейна (JS)

t	Значение целевой функции *F* (справочно)							Индекс Рэнда (RI)				
t	*Max*	*Min*	*Mean*	*Median*	*StD*	*Var*	*Spn*	*Max*	*Min*	*Mean*	*Median*	*StD*
0	46,240	40,127	41,271	40,130	1,696	0,041	6,113	0,603	**0,573**	0,593	**0,575**	0,006
1	12,072	10,582	11,037	10,628	0,304	0,028	1,490	0,554	0,409	0,452	0,411	0,034
2	5,577	4,807	5,012	4,807	0,234	0,047	0,771	0,487	0,359	0,429	0,360	0,033
3	2,655	2,295	2,360	2,296	0,064	0,027	0,360	0,581	0,397	0,478	0,400	0,044
4	1,951	1,674	1,739	1,682	0,054	0,031	0,277	0,586	0,387	0,467	0,393	0,057
5	1,674	1,387	1,487	1,390	0,069	0,046	0,287	0,616	0,386	0,484	0,388	0,064
6	1,790	1,205	1,299	1,216	0,104	0,080	0,585	0,625	0,383	0,471	0,385	0,082
7	1,498	1,061	1,149	1,067	0,086	0,075	0,437	0,614	0,380	0,480	0,392	0,081
8	1,288	0,965	1,026	0,965	0,078	0,076	0,322	0,598	0,406	0,462	0,406	0,069
9	1,137	0,864	0,955	0,870	0,064	0,067	0,273	0,596	0,370	0,463	0,371	0,080
10	1,074	0,754	0,864	0,762	0,073	0,085	0,320	0,637	0,399	0,505	0,399	0,088
20	0,577	0,374	0,441	0,377	0,056	0,127	0,202	**0,770**	0,541	0,687	0,544	0,062
30	0,518	0,360	0,406	0,361	0,045	0,110	0,158	0,764	0,530	0,677	0,539	0,056
40	0,477	0,344	0,395	0,344	0,041	0,103	0,133	0,760	0,556	**0,679**	0,559	0,044
50	0,513	0,341	0,392	0,343	0,047	0,120	0,172	0,753	0,467	0,659	0,497	0,066
60	0,494	0,324	0,375	0,324	0,044	0,117	0,170	0,747	0,467	0,666	0,507	0,052
70	0,471	0,318	0,367	0,318	0,036	0,097	0,153	0,721	0,536	0,656	0,545	0,046
80	0,422	0,313	0,355	0,313	0,028	0,079	0,108	0,744	0,521	0,655	0,548	0,039
90	0,523	0,306	0,356	0,306	0,051	0,143	0,217	0,706	0,511	0,641	0,518	0,044
100	1,074	0,754	0,864	0,762	0,073	0,085	0,320	0,637	0,399	0,505	0,399	0,088

Как видно из рисунков 1.1 и 1.3 максимальные значения точности по индексу Рэнда достигаются при минимальных значениях целевой функции.

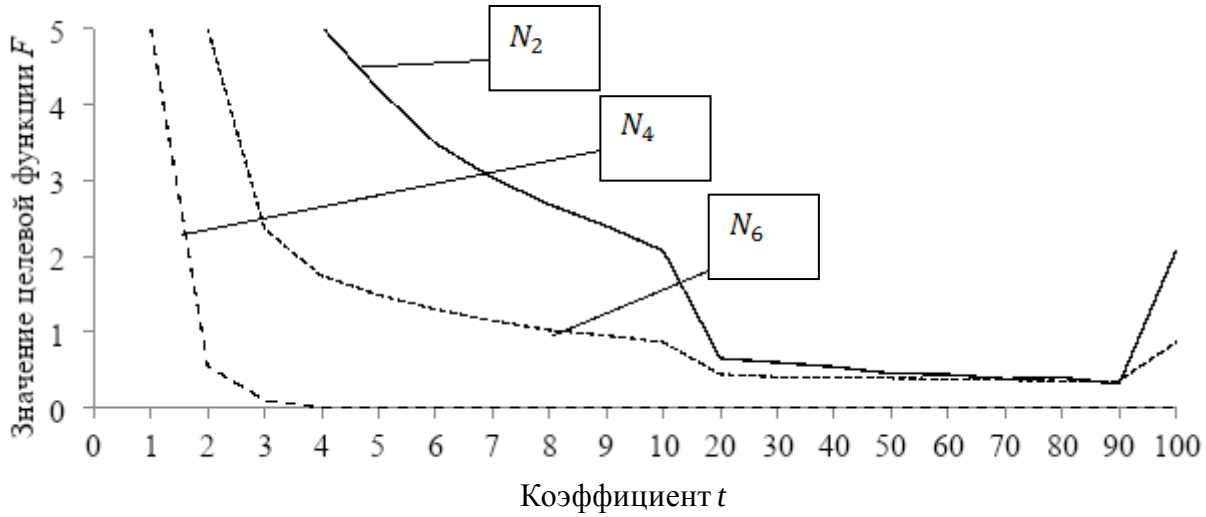

Рисунок 1.1 – Сравнение значений целевой функции F в результате кластеризации наборов данных N_2, N_4, N_6 с коэффициентом t сокращения оценки усадки Джеймса–Штейна

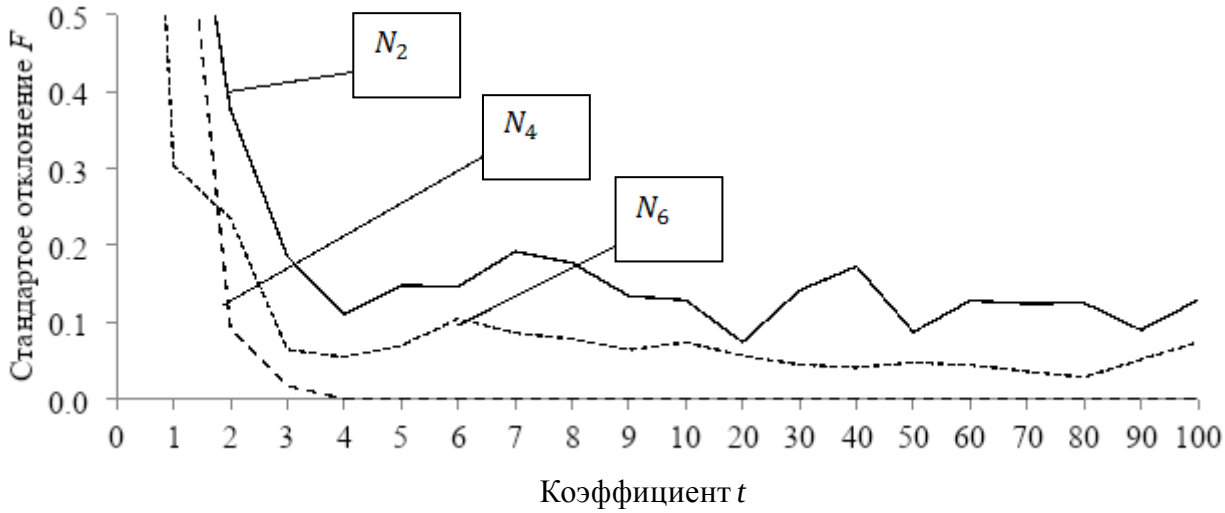

Рисунок 1.2 – Сравнение значений стандартного отклонения F в результате кластеризации наборов данных N_2, N_4, N_6 с коэффициентом t сокращения оценки усадки Джеймса–Штейна

Снижение стандартного отклонения индекса Рэнда при низком стандартном отклонении целевой функции на рисунках 1.2 и 1.4 указывает на улучшение качества кластеризации с применением нового подхода к нормализации данных по допустимым значениям параметров оцениваемых значений продукции и оценке Джеймса-Штейна.

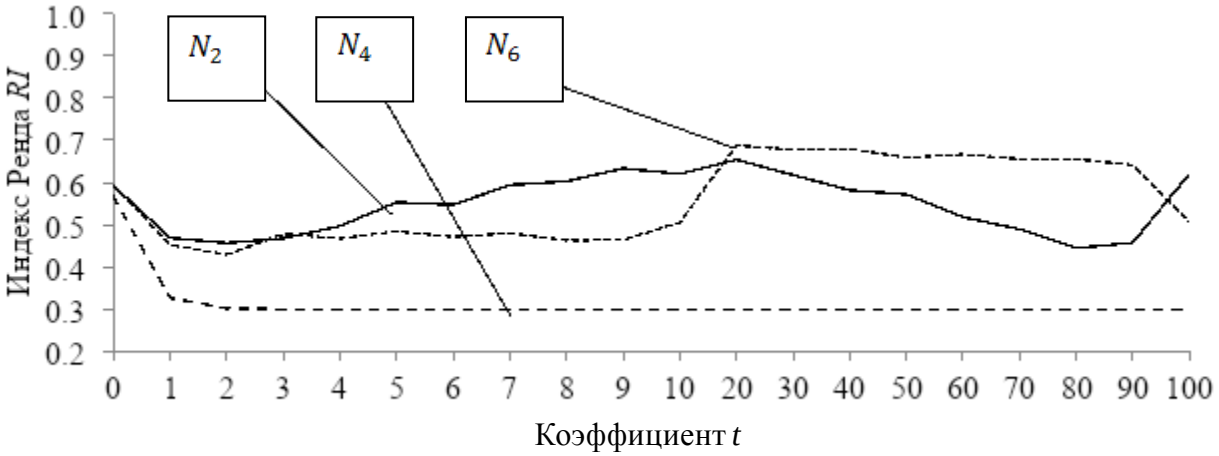

Рисунок 1.3 – Сравнение значений индекса Рэнда RI в результате кластеризации наборов данных N_2, N_4, N_6 с коэффициентом t сокращения оценки усадки Джеймса–Штейна

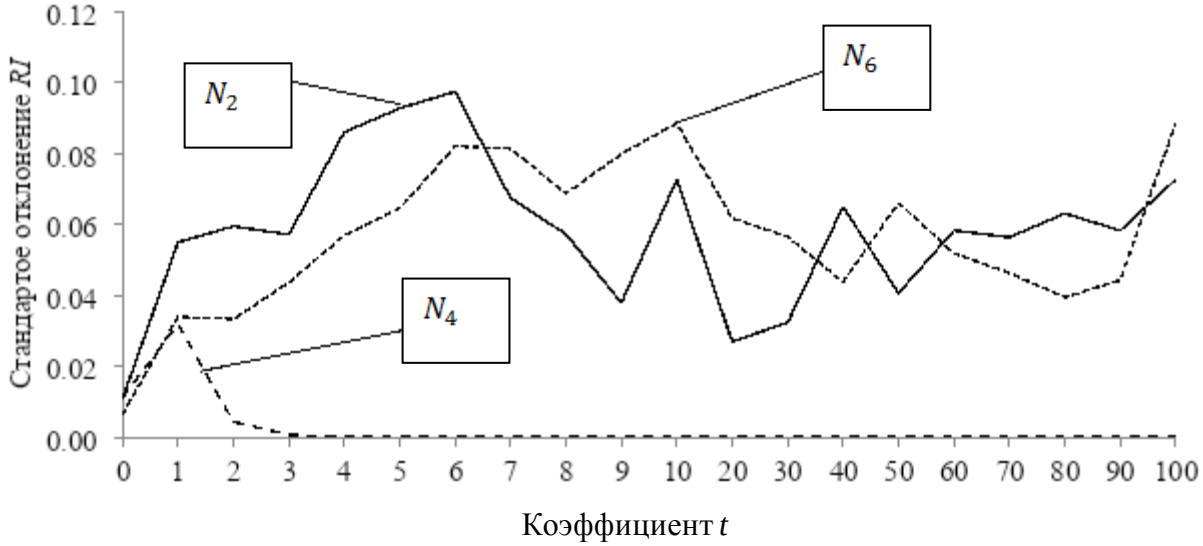

Рисунок 1.4 – Сравнение значений стандартного отклонения RI в результате кластеризации наборов данных N_2, N_4, N_6 с коэффициентом t сокращения оценки усадки Джеймса–Штейна

Таблица 1.8 – Результаты кластеризации для ИС 140УД265А с использованием нормализации по стандартному отклонению (A) для набора данных N_8 с коэффициентом t сокращения оценки усадки Джеймса–Штейна (JS)

t	Значение целевой функции F (справочно)							Индекс Рэнда (RI)				
t	*Max*	*Min*	*Mean*	*Median*	*StD*	*Var*	*Spn*	*Max*	*Min*	*Mean*	*Median*	*StD*
0	63,737	57,724	59,685	57,725	1,642	0,028	6,013	0,662	0,580	0,615	0,581	0,019
1	19,854	12,967	13,832	12,967	1,375	0,099	6,888	0,576	0,376	0,483	0,398	0,053
2	7,244	6,018	6,461	6,029	0,299	0,046	1,227	0,596	0,332	0,486	0,349	0,057
3	3,966	3,655	3,796	3,659	0,085	0,022	0,311	0,626	0,482	0,562	0,490	0,032
4	3,452	2,794	2,927	2,798	0,164	0,056	0,657	0,657	0,399	0,568	0,432	0,053
5	2,910	2,406	2,482	2,417	0,092	0,037	0,504	0,703	0,490	0,573	0,492	0,061
6	2,363	2,122	2,215	2,128	0,059	0,027	0,241	0,677	0,497	0,610	0,500	0,058
7	2,145	1,941	2,030	1,947	0,049	0,024	0,204	0,684	0,479	0,603	0,482	0,064
8	2,043	1,770	1,867	1,784	0,072	0,038	0,273	0,687	0,496	0,632	0,502	0,050
9	1,956	1,609	1,730	1,616	0,088	0,051	0,348	0,687	0,497	**0,626**	0,500	0,050
10	1,798	1,464	1,582	1,487	0,072	0,045	0,334	0,688	0,511	0,622	0,512	0,057
20	0,939	0,607	0,704	0,611	0,076	0,108	0,332	0,615	0,412	0,552	0,414	0,046
30	0,623	0,251	0,406	0,260	0,085	0,209	0,372	0,582	0,272	0,482	0,323	0,074
40	0,401	0,147	0,239	0,152	0,065	0,274	0,254	0,518	0,272	0,422	0,272	0,073
50	0,276	0,100	0,187	0,108	0,051	0,272	0,176	0,478	0,272	0,356	0,272	0,058
60	0,189	0,070	0,126	0,071	0,040	0,317	0,119	0,440	0,272	0,354	0,272	0,057
70	0,126	0,050	0,098	0,050	0,029	0,296	0,076	0,413	0,272	0,314	0,272	0,046
80	0,078	0,020	0,058	0,023	0,022	0,371	0,059	0,381	0,272	0,304	0,272	0,035
90	0,051	0,017	0,039	0,017	0,015	0,385	0,033	0,359	0,272	0,288	0,272	0,022
100	1,798	1,464	1,582	1,487	0,072	0,045	0,334	0,688	0,511	0,622	0,512	0,057

Таблица 1.9 – Результаты кластеризации для ИС 140УД26А с использованием нормализации по допустимому значению параметра (C) для набора данных N_{12} с коэффициентом t сокращения оценки усадки Джеймса–Штейна (JS)

t	Значение целевой функции *F* (справочно)							Индекс Рэнда (RI)				
t	*Max*	*Min*	*Mean*	*Median*	*StD*	*Var*	*Spn*	*Max*	*Min*	*Mean*	*Median*	*StD*
0	34,203	26,052	26,844	26,058	1,546	0,058	8,152	0,611	0,584	0,603	0,586	0,006
1	8,612	6,320	6,731	6,320	0,590	0,088	2,292	0,544	0,392	0,448	0,396	0,042
2	3,585	2,829	3,035	2,830	0,161	0,053	0,756	0,559	0,310	0,440	0,320	0,057
3	1,694	1,225	1,367	1,226	0,113	0,082	0,469	0,683	0,375	0,488	0,395	0,080
4	1,013	0,835	0,888	0,839	0,037	0,042	0,178	0,715	0,406	0,549	0,415	0,107
5	0,915	0,698	0,766	0,700	0,057	0,075	0,218	0,731	0,409	0,584	0,410	0,122
6	0,792	0,607	0,664	0,607	0,051	0,077	0,185	0,726	0,385	0,629	0,388	0,122
7	0,792	0,550	0,620	0,554	0,051	0,082	0,242	0,724	0,402	0,666	0,402	0,083
8	0,799	0,507	0,571	0,507	0,056	0,097	0,293	0,824	0,380	0,632	0,381	0,127
9	0,594	0,459	0,514	0,461	0,036	0,070	0,135	0,732	0,380	0,650	0,386	0,101
10	0,552	0,430	0,479	0,430	0,041	0,085	0,122	0,901	0,647	0,733	0,664	0,056
20	0,286	0,153	0,182	0,154	0,035	0,193	0,133	0,939	0,723	0,872	0,723	0,057
30	0,210	0,126	0,149	0,126	0,020	0,136	0,084	**0,957**	**0,788**	**0,879**	**0,793**	0,049
40	0,235	0,120	0,156	0,120	0,038	0,245	0,115	0,932	0,626	0,856	0,627	0,090
50	0,234	0,120	0,144	0,122	0,026	0,180	0,113	0,930	0,632	0,872	0,685	0,068
60	0,240	0,123	0,156	0,124	0,030	0,190	0,116	0,927	0,757	0,857	0,770	0,046
70	0,195	0,122	0,145	0,122	0,019	0,134	0,073	0,927	0,782	0,865	0,784	0,044
80	0,220	0,125	0,148	0,126	**0,018**	0,123	0,094	0,924	0,782	0,862	0,782	0,041
90	0,234	0,124	0,150	0,124	0,029	0,193	0,111	0,918	0,684	0,853	0,730	0,055
100	0,552	0,430	0,479	0,430	0,041	0,085	0,122	0,901	0,647	0,733	0,664	0,056

Для достижения максимальных значений точности по индексу Рэнда достаточно минимальных значений целевой функции как видно из рисунков 1.5 и 1.7.

Усредненные значения целевой функции модели *k*-средних с использованием нормализации по стандартному отклонению (A), значениям допустимого дрейфа (B), а также по допустимым значениям параметров (C) с

различным коэффициентом сокращения оценки усадки Джеймса–Штейна, оцененного для ИС 140УД26А (размер набора данных N = 132), представлены в таблицах 1.8–1.9 и на рисунках 1.5–1.8.

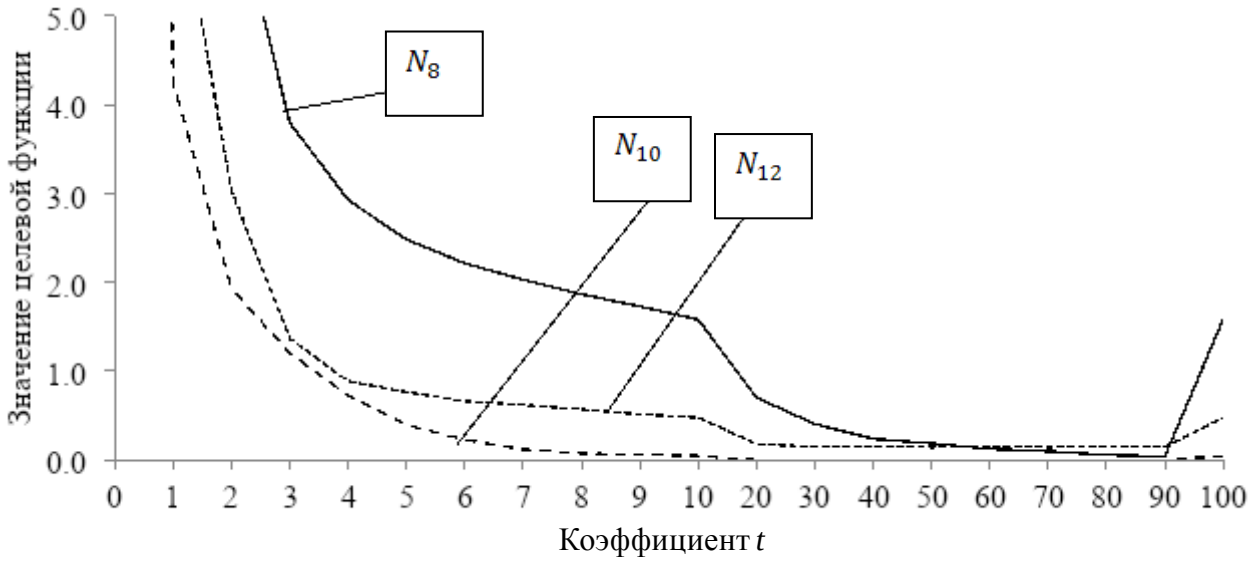

Рисунок 1.5 – Сравнение значений целевой функции F в результате кластеризации наборов данных N_8, N_{10}, N_{12} с коэффициентом t сокращения оценки усадки Джеймса–Штейна

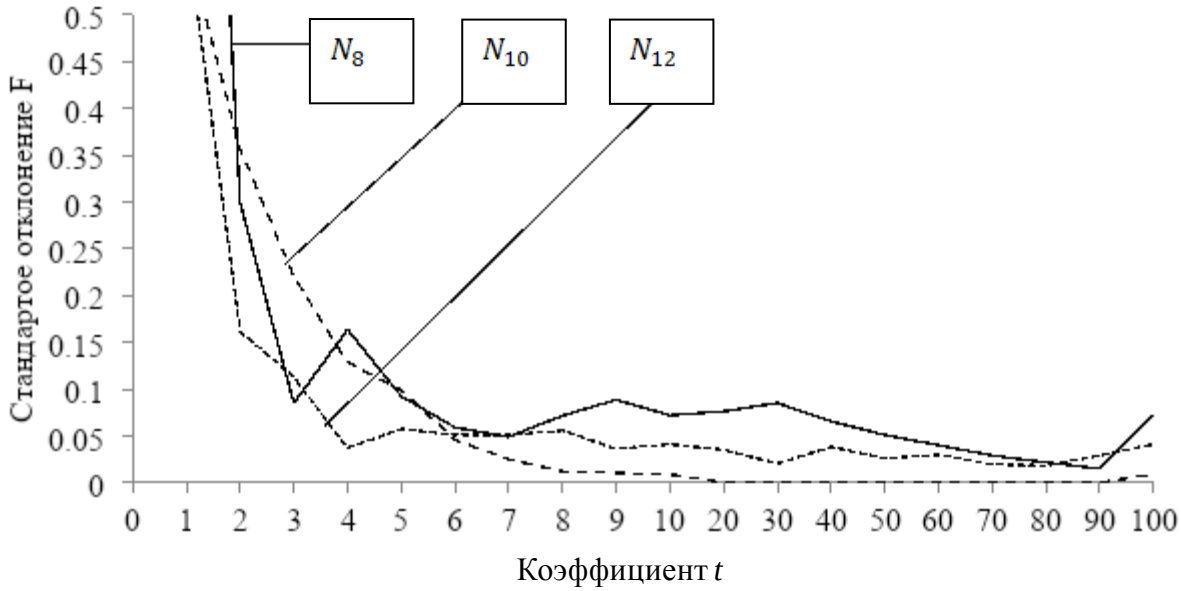

Рисунок 1.6 – Сравнение значений стандартного отклонения F в результате кластеризации наборов данных N_8, N_{10}, N_{12} с коэффициентом t сокращения оценки усадки Джеймса–Штейна

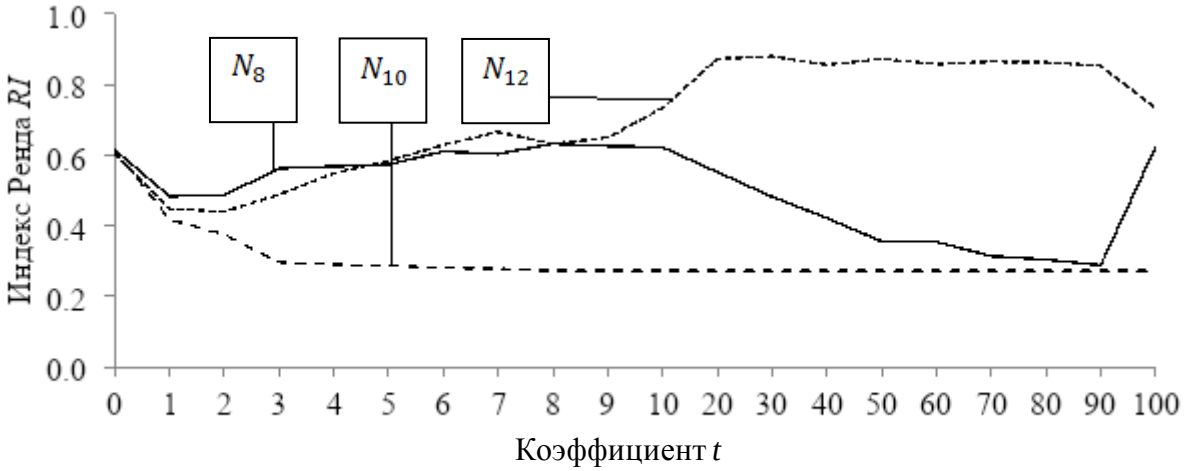

Рисунок 1.7 – Сравнение значений индекса Рэнда RI в результате кластеризации наборов данных N_8, N_{10}, N_{12} с коэффициентом t сокращения оценки усадки Джеймса–Штейна

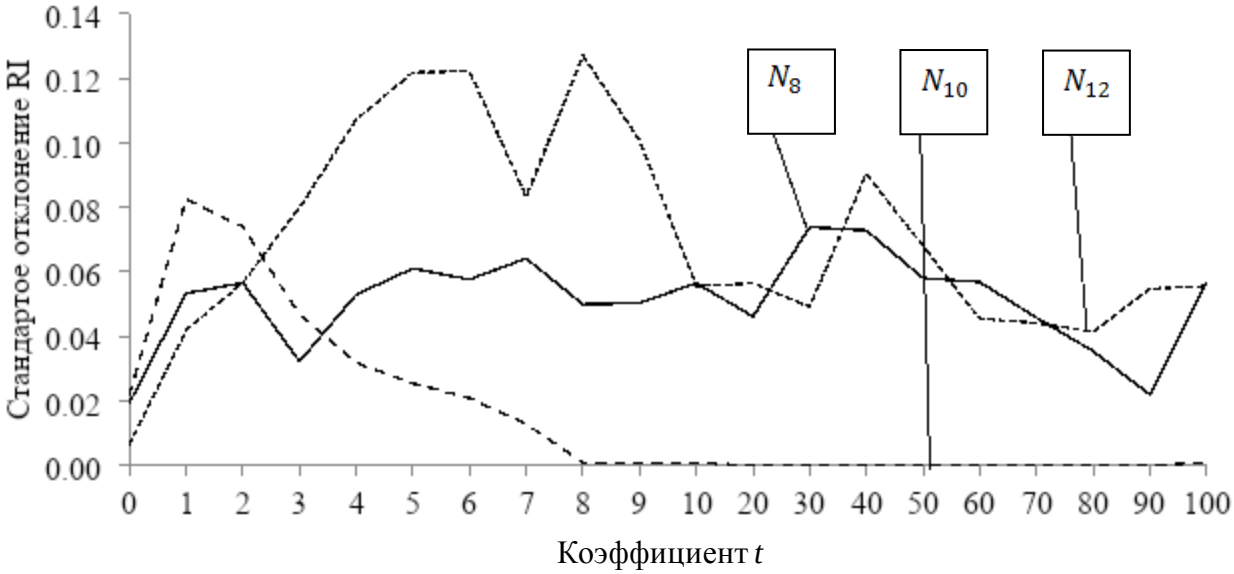

Рисунок 1.8 – Сравнение значений стандартного отклонения RI в результате кластеризации наборов данных N_8, N_{10}, N_{12} с коэффициентом t сокращения оценки усадки Джеймса–Штейна

Снижение стандартного отклонения индекса Рэнда при низком стандартном отклонении целевой функции на рисунках 1.2 и 1.4 указывает на улучшение качества кластеризации с применением нового подхода к нормализации данных по допустимым значениям параметров оцениваемых значений продукции и оценке Джеймса-Штейна. Однако увеличение стандартного отклонения RI на

рисунке 1.4 уменьшает значимость роста значений RI для наборов данных N_2, N_4, N_6 ИС 140УД25А до ЭТТ. Напротив, на рисунках 1.7 и 1.8 показано незначительное увеличение стандартного отклонения RI совместно с ростом значений RI в результате кластеризации наборов данных N_8, N_{10}, N_{12} ИС 140УД26А после электротермотренировки (ЭТТ).

Таким образом, снижение стандартного отклонения индекса Рэнда при низком стандартном отклонении целевой функции на рисунках 1.6 и 1.8 указывает на улучшение качества кластеризации с применением нового подхода к нормализации данных по допустимым значениям параметров оцениваемых значений продукции и оценке Джеймса-Штейна.

Набор данных N_8 для ИС 140УД26А с коэффициентом сокращения оценки усадки Джеймса–Штейна $t = 8$ показывает наилучшее значение точности кластеризации по индексу Рэнда и минимальное значение стандартного отклонения целевой функции. Вектор данных нормализованных по стандартному отклонению и с применением сокращения оценки усадки Джеймса-Штейна может быть представлен по формуле аналогичной (1.6), что показывает наибольшее влияние параметра 16.

Неэффективно для кластеризации с использованием нового подхода использовать набор данных N_{10} нормализованный по значениям допустимого дрейфа с применением сокращения оценки усадки Джеймса-Штейна. Вектор данных показывает уменьшение влияния значения всех параметров.

Набор данных N_{12} при коэффициенте сокращения оценки усадки Джеймса–Штейна $t = 70$ показано наилучшее значение точности кластеризации по индексу Рэнда и минимальное значение стандартного отклонения целевой функции. Вектор данных нормализованных по допустимому значению параметра и с применением сокращения оценки усадки Джеймса-Штейна может быть представлен по формуле аналогичной (1.7), что показывает наибольшие влияния параметров 4–5.

Таким образом, применение предложенного подхода к нормализации для предобработки входных данных, используемых в системах анализа данных

результатов неразрушающих испытаний образцов промышленной продукции, кроме повышения точности кластеризации, позволяет выявлять наиболее значимые контролируемые параметры продукции, что в перспективе позволит сократить затраты на испытания за счет уменьшения числа контролируемых параметров.

Результаты Раздела 1

Использование сокращения оценки усадки Джеймса–Штейна уменьшает влияние неинформативных параметров нормализованных данных промышленной продукции и повышает точность определения однотипных групп электрорадиоизделий нормализованных по допустимому значению параметра.

Исходя из результатов проведенных экспериментов, следует, что при кластеризации ЭРИ по однородным партиям для повышения точности кластеризации, предпочтительно использовать новый подход комбинирующий нормализацию с дополнительным улучшением среднего значения точек в кластере относительно центра выборки, по формуле (1.2) сокращения оценки усадки Джеймса–Штейна с коэффициентом t. Использование нормализации по допустимым значениям параметров оцениваемых характеристик продукции и сокращение оценки усадки Джеймса–Штейна обеспечивает повышение точности решения задачи кластеризации на 10% по индексу Рэнда.

С использованием нового подхода по нормализации данных, дополнительно нормализовали значения векторов данных в кластере относительно центра набора данных (сжимаем значения параметров в направлении среднего значения выборки), по формуле (1.2) сокращения оценки усадки Джеймса–Штейна с коэффициентом t.

2. АЛГОРИТМ АВТОМАТИЧЕСКОЙ ГРУППИРОВКИ С ИСПОЛЬЗОВАНИЕМ ЖАДНОЙ ЭВРИСТИЧЕСКОЙ ПРОЦЕДУРЫ ВЫБОРА РАДИУСА ЛОКАЛЬНЫХ КОНЦЕНТРАЦИЙ

В разделе 2 рассмотрены вопросы разработки алгоритма кластеризации электрорадиоизделий (решается задача разделения смешанной партии ЭРИ на однородные партии продукции) по результатам тестов с жадной эвристической процедурой выбора радиуса локальных концентраций по размеченным данным. Нами рассматривается задача поиска радиуса локальных концентраций для алгоритма кластеризации с мерой похожести на центр с заранее заданным числом кластеров. Результаты сравнивались с различными способами нормализации данных для решения задачи кластеризации методом k-средних. Результаты данного раздела были опубликованы в [34, 35].

2.1 Математическая постановка задачи автоматической группировки

Классические и наиболее распространенные алгоритмы кластеризации, в том числе алгоритм *k*-средних, ориентированы под конкретные задачи. Исследование алгоритма в данной области связано с выбором способа нормализации данных, выбором метрики расстояния и выбором способа инициализации центров. Этот алгоритм зависит от заданного заранее числа кластеров, находит все группы для разделения на кластеры одновременно. Процесс кластеризации заключается в группировке N наблюдений в k групп, где начальные центры групп задаем с помощью M-мерных векторов. Учитывая набор наблюдений $a_1, \dots, a_i, \dots, a_N$, которые разделены на k кластеров $C = \{C_1, \dots, C_j, \dots, C_k\}$, целевая функция достигает своего минимума:

$$argmin \sum_{j=1}^{k} \sum_{x \in C_j} \|a - x_j\|^2, \qquad (2.1)$$

где x_j – среднее значение наблюдений в кластере C_j. Кластеры, полученные алгоритмом *k*-средних, имеют сферическую форму.

2.2 Теоретический анализ алгоритма кластеризации электрорадиоизделий

В классической реализации алгоритма *k*-средних в варианте Ллойда [31] состоит из шагов назначения каждой точки данных ближайшему центру кластера и пересчета центров кластеров на основе сформированных кластеров. Алгоритмы семейства *k*-средних основаны на мере похожести (сходства, близости) на центр оцениваемой между точками в многомерном пространстве через расстояние Евклида. В работах [36, 37] Загоруйко Н.Г. и др. сформулированы требования локальности, нормированности и инвариантности, которым должна удовлетворять мера сходства. Кроме того, эти алгоритмы основаны на использовании гипотезы компактности, по которой точки относятся к одному кластеру, если в пространстве они расположенные рядом.

Подход, предложенный в работах Загоруйко Н.Г. основанный на использовании конкурентного сходства FRiS-функции, следует данным требованиям и гипотезе, и аналогичен в этом отношении алгоритму Ллойда. Использование FRiS-компактности в алгоритме GRAD [38] используется для выбора информативных параметров (признаков).

Работы Федосова В.И., Шкабериной Г.Ш., Голованова С.М. [3, 4, 40] и др. показывают, что, хотя в зависимости от условий производства параметры однотипных изделий различных партиях могут существенно различаться, изделия в составе одной партии имеют сходство параметров, благодаря чему партии можно разделить. В алгоритмах FRiS [39] кластеризация осуществляется с использованием расстояния от центра кластера, который назовем радиусом локальных концентраций, определяет предельный совокупный разброс значений параметров элементов кластера, т.е. предельный разброс значений параметров изделий в партии, являясь дополнительной характеристикой получаемого

результата автоматической группировки, важный для эксперта, принимающего заключение о составе исследуемой партии. При этом такой радиус в практических задачах заранее неизвестен.

Предлагается новый алгоритм автоматической группировки с автоматическим выбором радиуса поиска локальных концентраций с использованием разведочного поиска по размеченным данным, который формирует кластеры сферической формы и получает преимущество перед алгоритмом *k*-средних по скорости и точности кластеризации по индексу Рэнда.

В общей постановке задачи алгоритм автоматической группировки ищет такое разбиение *N* объектов на *k* кластеров, чтобы критерий похожести *F* на центр для всех кластеров был минимальным. Обозначим координаты центра *j*-го кластера символом x_j. Сумма расстояний $p(x_j, a_i)$ между центром сферы с радиусом R_0 и всеми n_j точками a_i этого кластера $p_j = \sum p(x_j, a_i)$, где $i = 1, \dots, n_j$, а сумма таких внутренних расстояний для всех *k* кластеров $F = \sum p_j$, $j = 1, \dots, k$.

Для нового алгоритма LEES (Localized Estimation Exploratory Search – локального оценочного исследовательского поиска) функция качества кластеризации *F* использует дополнительный критерий *f*, соответствующий единице при равенстве искомого числа кластеров *k* и найденного числа кластеров *K*

$$F = argmin\, f(K, p_j) \sum_{j=1}^{k} p_j \begin{cases} f(K) = 1, \text{ если } K = k, \\ f(K) = \infty, \text{ если } K \neq k, \\ i \epsilon C_j, \text{ если } p_j(x_j, a_i) \leq R_0 R. \end{cases} \quad (2.2)$$

где *R* – коэффициент радиуса поиска локальных концентраций, C_j – это *j*-ый кластер.

Условие присоединения точки к кластеру a_i определяем по формуле

$$p_j(x_j, a_i) \leq R_0 R. \quad (2.3)$$

Наилучшему варианту соответствует минимальное значение *F*.

Новый алгоритм LEES состоит из следующих шагов:

Алгоритм 1. LEES

Дано: Задаем точное число кластеров k сферической формы с радиусом R_0. Для вычисления расстояний между объектами применяем метрику Евклида.

Шаг 1. Случайно выбираем первый центр x_1, а следующие центры на удалении R_0 заданного радиуса с коэффициентом R от каждого центра по формуле (2.3), для улучшения инициализации центров x_j.

Шаг 2. Присваиваем последовательно точки кластеру, которому соответствует ближайший центр x_j. Для минимизации целевой функции (2.2) используем метод жадного выбора ближайшего центра.

Шаг 3. Пересчитываем новые центры для каждого кластера, как среднее арифметическое координат точек кластера.

Повторять Шаги 2 и 3 пока изменения целевой функции не стабилизируются, а пересчитанные центры кластеров не перестанут изменяться.

Для алгоритма LEES число кластеров должно быть заранее задано. Задача определения числа кластеров (однородных групп), из которых состоит набор данных (например, набор данных о результатах тестирования промышленной продукции) является отдельной сложной задачей [41]. На практике приходится находить решения для различных предполагаемых чисел кластеров, выбирая затем решение с числом кластеров, определенным с привлечением дополнительного критерия оценки качества результатов кластеризации. Таким критерием для данных об образцах промышленной продукции является критерий силуэта [3, 22, 41]. В алгоритме LEES лежит базовая процедура, в основном совпадающая с алгоритмом *k*-средних. Шаг 1 отличается тем, что улучшается инициализация центров кластеров.

2.3 Методика исследования нового алгоритма на примере автоматической группировки электрорадиоизделий

Для выделения предположительно однородных партий с некоторой точностью были использованы данные результатов испытаний [35], проведенных в испытательном центре. Эксперименты были проведены на различных типах интегральных схем, в качестве примера в диссертации приведены эксперименты для двух интегральных схем (ИС) 140УД25А и 140УД26А, принадлежащих к разным однородным партиям. Исходные данные представляют собой комбинацию некоторых параметров и состоят из 18 партий и 9 партий первой и второй ИС соответственно. Продукты (устройства) в каждой партии описываются 18 входными измеряемыми параметрами. Для первой партии ИС отобран 1, 3, 8 и 9 набор данных в количестве $N = 807$. Для второй партии ИС отобран 4, 5, 7 и 9 набор данных в количестве $N = 532$. Набор данных прореживался, отбрасывались каждые 2, 3 и 4 данные о результатах испытаний, количество составило $N = 201$ и $N = 132$ для первой и второй ИС соответственно. Отобранные данные тестовых испытаний нормализовались по стандартному отклонению и по допустимым значениям параметра. [34, 1]

2.4 Результаты вычислительных экспериментов

Сравнение алгоритма LEES с *k*-средних проводилось на системе с 8Гб оперативной памяти, процессором Intel Core i3-3220 с частотой 3,3ГГц, операционной системе Ubuntu 18.04.6 LTS. Для каждого алгоритма мы провели 30 экспериментов (запусков алгоритма). Каждый запуск продолжался 1, 10 и 60 секунд.

Средние результаты точности кластеризации 1, 3, 8 и 9 партии данных ИС 140УД25А алгоритмом *k*-средних в ограничении по времени, а также максимальное (*max*), минимальное (*min*), среднее значение (*mean*) и стандартное отклонение (*st.dev*) для индекса Рэнда и целевой функции приведены в таблицах 2.1 – 2.8. Для целевой функции также рассчитываются коэффициент вариации (*var*) и коэффициент размаха (*spn*).

Таблица 2.1 – Точность кластеризации смешанной партия (N=201) с нормализацией по стандартному отклонению

	k-средних			LEES		
Время, с.	1	10	60	1	10	60
	Индекс Рэнда					
max	0,605	0,605	0,605	0,794	0,778	0,799
min	0,55	0,542	0,563	0,605	0,641	0,656
mean	0,596	0,593	0,591	0,712	0,711	0,741
st.dev	0,01	0,014	0,01	0,052	0,039	0,04
	Целевая функция					
max	85,1	80,6	77,2	85	81	78,4
min	66,9	66,8	66,8	76,1	72,5	72,2
mean	69,4	69	69,7	81,1	77,1	75,1
st.dev	3,5	2,9	3,2	2,5	2	1,3
var	0,051	0,041	0,046	0,03	0,026	0,017
spn	18,2	13,8	10,4	9	8,5	6,2

Таблица 2.2 – Точность кластеризации смешанной партия (N=807) с нормализацией по стандартному отклонению

	k-средних			LEES		
Время, с.	1	10	60	1	10	60
	Индекс Рэнда					
max	0,6	0,599	0,6	0,76	0,777	0,781
min	0,557	0,58	0,581	0,539	0,599	0,67
mean	0,584	0,594	0,595	0,669	0,718	0,738
st.dev	0,013	0,004	0,004	0,063	0,045	0,026
	Целевая функция					
max	326,5	296,9	298,4	411,6	348,2	336,7
min	273	269,7	269,8	325	323,5	312,7
mean	296,5	274	273,8	365	335,4	324,3
st.dev	14,6	6	5,9	21,1	7	5,7
var	0,049	0,022	0,022	0,058	0,021	0,018
spn	53,5	27,3	28,6	86,5	24,8	24

Таблица 2.3 – Точность кластеризации смешанной партия (N=201) с нормализацией по допустимым значениям параметра

	k-средних			LEES		
Время, с.	1	10	60	1	10	60
	Индекс Рэнда					
max	0,603	0,602	0,602	0,809	0,82	0,803
min	0,548	0,58	0,55	0,556	0,608	0,675
mean	0,594	0,594	0,593	0,68	0,715	0,743
st.dev	0,01	0,006	0,011	0,059	0,054	0,04
	Целевая функция					
max	51,8	46,4	51,4	58,9	54,4	51,1
min	40	40	40,1	48,9	47,5	47,6
mean	42	41,7	42,2	54,6	51,2	49,6
st.dev	2,4	1,4	2,7	2,2	1,6	1
var	0,057	0,035	0,063	0,041	0,032	0,019
spn	11,8	6,4	11,3	10	6,9	3,5

Таблица 2.4 – Точность кластеризации смешанной партия (N=807) с нормализацией по допустимым значениям параметра

	k-средних			LEES		
Время, с.	1	10	60	1	10	60
	Индекс Рэнда					
max	0,601	0,6	0,6	0,784	0,783	0,784
min	0,532	0,579	0,575	0,349	0,585	0,654
mean	0,58	0,593	0,594	0,62	0,707	0,732
st.dev	0,019	0,006	0,005	0,104	0,049	0,035
	Целевая функция					
max	214	188,2	194,8	329,4	246,1	230,2
min	165,5	164	163,9	223,6	219,4	212,7
mean	184,6	170	168,6	260,2	231,5	221,8
st.dev	15,1	7,2	6,5	24,4	5,8	4
var	0,082	0,042	0,038	0,094	0,025	0,018
spn	48,4	24,2	30,9	105,8	26,7	17,5

Таблица 2.5 – Точность кластеризации смешанной партия (*N*=132) с нормализацией по стандартному отклонению

	k-средних			LEES		
Время, с.	1	10	60	1	10	60
	Индекс Рэнда					
max	0,646	0,657	0,613	0,747	0,742	0,749
min	0,588	0,597	0,57	0,538	0,626	0,671
mean	0,613	0,617	0,6	0,684	0,705	0,716
st.dev	0,016	0,016	0,013	0,046	0,03	0,023
	Целевая функция					
max	64,7	62,9	62,8	68,4	65,9	63,7
min	57,8	57,8	57,8	63,7	60,3	60
mean	59,6	59,4	60	66	63,6	61,9
st.dev	1,5	1,4	1,4	1,4	1,2	0,8
var	0,026	0,024	0,023	0,021	0,019	0,014
spn	6,8	5,2	5	4,7	5,7	3,7

Таблица 2.6 – Точность кластеризации смешанной партия (*N*=532) с нормализацией по стандартному отклонению

	k-средних			LEES		
Время, с.	1	10	60	1	10	60
	Индекс Рэнда					
max	0,654	0,655	0,654	0,738	0,728	0,733
min	0,583	0,589	0,595	0,559	0,606	0,658
mean	0,621	0,618	0,627	0,676	0,684	0,7
st.dev	0,02	0,013	0,018	0,047	0,027	0,02
	Целевая функция					
max	251,3	252	249,6	308,7	281,4	272
min	235,8	233,8	240,6	269	264,8	256,4
mean	242,7	237,8	242,2	286,9	272,7	267,1
st.dev	4,2	3,6	1,8	11,1	4,3	3,1
var	0,017	0,015	0,007	0,039	0,016	0,012
spn	15,5	18,2	8,9	39,7	16,7	15,6

Таблица 2.7 – Точность кластеризации смешанной партия (*N*=132) с нормализацией по допустимым значениям параметра

	k-средних			LEES		
Время, с.	1	10	60	1	10	60
	Индекс Рэнда					
max	0,613	0,614	0,613	0,752	0,759	0,77
min	0,576	0,567	0,57	0,623	0,614	0,663
mean	0,606	0,605	0,6	0,703	0,704	0,713
st.dev	0,01	0,01	0,013	0,031	0,035	0,025
	Целевая функция					
max	34,4	32	31,7	37,1	33,8	32,6
min	26,1	26,1	26,1	32,8	30,1	28,5
mean	27,7	27,6	28	34,4	32,2	31,1
st.dev	1,8	1,3	1,5	1,1	0,8	0,9
var	0,063	0,046	0,052	0,031	0,024	0,028
spn	8,3	5,9	5,6	4,2	3,7	4,1

Таблица 2.8 – Точность кластеризации смешанной партия (*N*=532) с нормализацией по допустимым значениям параметра

	k-средних			LEES		
Время, с.	1	10	60	1	10	60
	Индекс Рэнда					
max	0,615	0,613	0,613	0,748	0,744	0,743
min	0,556	0,574	0,573	0,545	0,656	0,644
mean	0,604	0,606	0,607	0,686	0,691	0,703
st.dev	0,013	0,012	0,009	0,042	0,02	0,023
	Целевая функция					
max	137,4	133,4	138,9	175,4	152,3	143,7
min	106,1	106,2	106	138,3	138,7	136,4
mean	113,1	111,8	111,3	153	144,9	140,1
st.dev	6,4	7,3	6,7	8,6	3	1,9
var	0,057	0,065	0,061	0,056	0,021	0,014
spn	31,3	27,3	32,9	37,2	13,6	7,3

В результатах вычисления точности кластеризации алгоритмом *k*-средних для смешанных партий ИС с различной нормализацией наблюдается уменьшение точности кластеризации по индексу Рэнда с увеличением времени выполнения алгоритма. Это предположительно связано со сходимостью к локальному минимуму и неудачным выбором центров инициализации. Такое уменьшение точности отсутствует в результатах экспериментов с применением нового алгоритма LEES, как показано в таблицах 2.1 – 2.8.

Согласно полученным значениям индекса Рэнда, подход с использованием алгоритма LEES с жадной эвристикой выбора радиуса поиска локальных концентраций продемонстрировал наилучшую точность при большем значении целевой функции в сравнении с алгоритмом *k*-средних.

Точность и скорость работы программной реализации алгоритма вполне приемлемы для решения задачи кластеризации электрорадиоизделий на основе данных тестовых испытаний. Использование жадной эвристики в алгоритме LEES показало большую эффективность по сравнению с классической реализацией алгоритма *k*-средних.

Результаты Раздела 2

Использование жадной эвристики, Шаг 1 Алгоритма LEES, для выбора радиуса поиска локальных концентраций в алгоритме кластеризации LEES с точно заданным числом кластеров имеет преимущество по скорости точной кластеризации в сравнении с алгоритмом *k*-средних, использующим жадную эвристику для выбора центроидов, при использовании нормализованных значений тестовых испытаний по стандартному отклонению и по допустимым значениям параметра.

Эффективно применение нового алгоритма LEES в области кластеризации системы анализа данных электрорадиоизделий (разделения смешанной партии ЭРИ на однородные партии продукции) на основе данных тестовых испытаний с применением задачи *k*-средних для нормализованных данных.

3. РАСШИРЕННЫЙ АЛГОРИТМ КЛАСТЕРИЗАЦИИ ДЛЯ РЕШЕНИЯ ЗАДАЧИ ПРИБЛИЖЕННОГО ПОИСКА БЛИЖАЙШЕГО СОСЕДА

В разделе 3 рассмотрены вопросы разработки нового алгоритма кластеризации, основанного на жадной агломеративной процедуре для построения индекса для векторной базы данных. В вычислительных экспериментах показано, что индекс (набор центров), сгенерированный этим алгоритмом, достигает более высокой оценки полноты по сравнению с алгоритмами *k*-средних и *k*-means++. Результаты данного раздела были опубликованы в [42].

3.1 Обзор литературы о решении задачи поиска ближайшего соседа

Современные инструменты сбора данных позволяют накапливать большие объемы многомерной информации об объекте исследования. Обработка данной информации с использованием соответствующих методов и алгоритмов интеллектуального анализа становится ценным источником знаний об объекте исследования. Поиск ближайших соседей (NNS) — это задача оптимизации, которая заключается в поиске вектора, наиболее похожего на заданный вектор (вектор запроса) в исходном наборе данных. Под сходством обычно понимают функцию несходства, которая определяет расстояние между любыми векторами и классами в метрическом пространстве R^M. Это пространство состоит из набора векторов с функцией расстояния $d(a_i, q_i)$. Чем больше значение функции расстояния, тем меньше сходство между вектором запроса и вектором из исходного набора данных. Между двумя точками расстояние порядка p определяется функцией Минковского (L^p-норма) [43]:

$$d_p(a_i, q_i) = \left(\sum_{i=1}^{M} |a_i - q_i|^p \right)^{\frac{1}{p}},$$

где a и q — векторы значений параметров, а M — размерность вектора.

Используя параметр p, определяемый исследователем, вес переменной i может быть постепенно увеличен или уменьшен. Конкретные случаи функции Минковского зависят от значения p. Например, при $p = 2$ функция вычисляет расстояние Евклида; при $p = 1$ — манхэттенское расстояние; при $p = \infty$ — расстояние Чебышева; а при $p = 0$ — расстояние Хэмминга [44]. Помимо вариаций функции в зависимости от параметра p, существуют и другие методы вычисления расстояний, такие как расстояние Махаланобиса [43,44], косинусное расстояние [44] и другие.

Задача поиска ближайшего соседа может быть описана следующим образом [44, 46]. Дан набор N точек и $d(a_i, q_i)$. как функция для вычисления расстояния между двумя точками a_i, q_i. Пара (N, d) выделяет метрическое пространство, если d имеет такие характеристики, как рефлексивность, неотрицательность, симметрия и неравенство треугольника.

В более подробной классификации задачи NNS могут быть определены в евклидовом пространстве следующим образом [44]:

Точная NNS: Дано множество S из N точек в M-мерном пространстве R^M $(S \subset R^M)$, построить структуру данных, которая при любой точке запроса $q \in R^M$ находит точку в S с наименьшим расстоянием до q .

Это определение для небольшого набора данных с низкой размерностью имеет сублинейное (или даже логарифмическое) время запроса, но для большого набора данных с высокой размерностью является экспоненциальным. Аппроксимация может уменьшить экспоненциальную сложность до полиномиального времени [45]. Приближенная NNS определяется как [44]:

Приближенная NNS: задан набор N точек в M-мерном пространстве R^M $(N \subset R^M)$, построить структуру данных, которая при любой точке запроса $q \in R^M$ сообщает о любой точке в пределах расстояния, не превышающего S раз от q top, где p — точка из S, ближайшая к q.

Существует два подхода к решению задачи поиска ближайшего соседа, которые можно в целом классифицировать как точные (линейные методы, разбиение

пространства) и приближенные (методы графа близости, методы на основе хеширования, файлы векторной аппроксимации и поиск на основе сжатия или кластеризации).

Идея линейного подхода заключается в последовательном вычислении расстояния от вектора запроса до каждого вектора в наборе данных. Временная сложность этого подхода пропорциональна количеству и размерности данных $O(MN)$, где M — размерность вектора, а N — общее количество векторов в наборе данных [47]. В результате производительность ухудшается с увеличением размерности [48]. Хотя существуют некоторые эвристики для снижения вычислительных затрат [47], этот метод плохо масштабируется в многомерных и крупномасштабных задачах.

Для решения задачи поиска по сходству в многомерных векторных пространствах были предложены различные древовидные структуры для разбиения пространства, такие как KD-деревья и их вариации [49, 54, 55], R-деревья и их вариации [50, 56, 57], B+-tree [51, 52, 53] и Cover-tree [58, 59]. Основная идея этих структур заключается в сужении пространства поиска путем разбиения его плоскостями или выровненными по осям областями. Расстояния вычисляются только для соседей, которые лежат по ту же сторону плоскости, что и вектор запроса. Такая древовидная организация данных уменьшает количество кандидатов для поиска ближайшего соседа. Однако производительность этих методов также ухудшается для многомерных и крупномасштабных данных, поскольку построение таких структур данных обычно требует значительного времени и памяти [60].

В [61] исследуют влияние размерности на поиск ближайшего соседа. По мере увеличения размерности расстояние до ближайшей точки данных приближается к расстоянию до самой дальней точки данных, что иллюстрирует «проклятие размерности».

Для решения проблемы размерности предлагается подход поиска приближенных ближайших соседей (ANN). Преимуществом ANN является его способность эффективно находить приближенные решения, что может привести к значительному ускорению и экономии памяти по сравнению с точным поиском бли-

жайших соседей. В некоторых случаях набор данных настолько велик, что полностью точный поиск нецелесообразен в разумные сроки. В других сценариях разница между точным ближайшим соседом и приближенным соседом может быть незначительной.

Во многих работах представлены методы графа близости, такие как Navigable Small Worlds (NSW) [62, 63, 64] и Hierarchical Navigable Small Worlds (HNSW) [65]. Эти методы основаны на концепции малого мира, которая относится к свойству графов, где каждый узел имеет ближние связи со своими соседями (обычно около $log(N)$ шагов), а также несколько дальних связей, которые облегчают эффективное глобальное исследование. Алгоритм HNSW создает многослойную иерархическую структуру графа для эффективной индексации и поиска ближайших соседей в многомерных пространствах. В отличие от точного метода *k*-ближайших соседей (kNN) [66], который выполняет полный поиск данных, HNSW хорошо масштабируется для больших наборов данных.

Обучение хешированию — еще один метод решения проблемы ANN [67]. При обучении хешированию данные кодируются в короткие коды, известные как хэш-коды, которые значительно короче исходных измерений данных. Хотя обучение хешированию часто эффективно для ANN, некоторая потеря информации неизбежна в процессе кодирования (т. е. при проецировании данных на пространство с меньшей размерностью). Алгоритмы ANN на основе хеширования обычно делятся на две основные категории: независимые от данных методы, такие как локально-чувствительное хеширование (LSH) [68], и зависимые от данных методы, такие как локально-сохраняющее хеширование (LPH) [69]. LSH [70] позволяет быстро извлекать похожие записи в больших базах данных. Этот подход относится к классу рандомизированных алгоритмов. Рандомизированный алгоритм не гарантирует точного ответа, но вместо этого обеспечивает высокую вероятность возврата правильного или почти правильного результата.

В [71] представлен файл векторной аппроксимации (VA-файл) для поиска сходства в многомерных векторных пространствах. VA-файл преодолевает размерное проклятие, следуя не подходу к разделению данных обычных методов ин-

дексации, а скорее подходу на основе фильтрации файлов сигнатур. Пространство, а не данные, делится на ячейки, и векторам назначаются приближения на основе ячеек, в которых они лежат. VA-файл содержит эти небольшие, закодированные в битах приближения. Сканируя сначала все самые маленькие приближения, запросы ближайшего соседа должны посещать только часть самих векторов. Таким образом, VA-файл используется как простой фильтр, во многом как файл сигнатуры является фильтром.

В [72] предложен эффективный метод индексации для многомерных баз данных с использованием подхода фильтрации, известного как метод векторного приближения (VA-файла). Этот подход эффективно поддерживает поиск ближайшего соседа и определяет расстояния между кластерами на основе разбиения на гиперплоскости. Кроме того, другие исследователи представили файл частичной векторной аппроксимации [73], который предназначен для эффективной обработки запросов частичного сходства в любом подпространстве, даже если конкретное подпространство заранее неизвестно.

Идея частичного VA-файла заключается в том, чтобы разделить VA-файл на отдельные файлы для каждого измерения и загружать только те измерения, которые необходимы для ответа на запрос. Таким образом, частичный VA-файл создается для повышения производительности запросов для систем, которым приходится справляться с широким спектром ранее неизвестных подпространств запросов. В этих экспериментах [72] продемонстрировано, что предложенный ими частичный VA-файл с новыми алгоритмами (для частичного kNN и ε-диапазона) повышает среднюю производительность запросов по сравнению с исходным VA-файлом при ответе на запросы частичного сходства. Существуют методы поиска ANN для многомерных данных, которые кодируют данные в компактные коды на основе векторного квантования [74]. Поиск на основе сжатия или кластеризации является наиболее подходящим подходом. Сначала пространство данных разлагается на декартово произведение некоторых низко мерных подпространств, а затем данные квантуются в каждом подпространстве отдельно [67]. Многие работы по-

священы изучению приблизительного поиска ближайшего соседа на основе PQ [67, 75– 80].

Методы поиска ближайшего соседа (NNS) сегодня широко используются в различных областях, таких как системы моделирования дорожного движения, где на состояние каждого транспортного средства влияют находящиеся поблизости транспортные средства, называемые соседями [52], транзакции ACM в системах баз данных [51], обработка изображений [55], распознавание образов, поиск информации [57], классификация объектов и поиск по сходству и другие. Для решения этих задач объекты преобразуются в векторную форму (структуру), которая описывает параметры объекта, и поиск ближайшего соседа выполняется в этом векторном пространстве.

В векторных базах данных, поиск включает в себя поиск данных с приблизительными соответствиями вектору запроса [81]: цель состоит в том, чтобы определить векторы данных, которые наиболее близки к вектору запроса, с помощью алгоритмов приблизительного поиска ближайшего соседа (ANN) [82]. Основным преимуществом этого подхода является его способность значительно повышать производительность поиска при небольшом ущербе точности запроса.

Алгоритмы поиска ближайшего соседа обычно следуют двухэтапному процессу. На первом этапе строится вспомогательная структура данных, известная как векторный индекс [83, 65]. На втором этапе, когда делается поисковый запрос, эта структура используется для ускорения процесса извлечения данных. Таким образом, ANN обычно включает в себя по крайней мере два алгоритма, работающих в тандеме.

Индексы баз данных также являются особенностью традиционных баз данных, но структура индексов в векторных базах данных отличается [83, 65].

Структуры индексов, используемые в векторных базах данных, могут сильно различаться. Один из простейших подходов, часто сочетаемый с другими методами для построения более сложных структур индексов в векторных базах данных, заключается в поиске главных точек (центров или центроидов) в векторном пространстве. Эти главные точки являются идеализированными представителями

векторных кластеров данных со средними значениями характеристик. Все векторы данных назначаются определенным главным точкам или кластерам. При выполнении поискового запроса сначала определяются ближайшие главные центры к вектору запроса, а затем поиск проводится в соответствующих векторных кластерах данных. Этот подход значительно ускоряет процесс извлечения данных, избегая полного сканирования векторов данных. Однако в многомерных пространствах эти центры являются лишь приблизительными представителями своих кластеров, и для достижения высокоточных результатов требуется изучить значительную часть векторов данных.

Проблема поиска центральных точек является классической проблемой размещения, которая может быть сформулирована как задача p-медианы [84 – 86] или как задача *k*-средних, предложенная Г. Штейгаузом [87, 24, 88 – 90].

Для решения обеих задач используется классический алгоритм — алгоритм *k*-средних, также известный как алгоритм Ллойда [31] (Lloyd) или алгоритм альтернативного размещения-распределения (ALA). Этот метод начинается с начального решения, состоящего из набора центров или центроидов $c_1, \ldots, c_k$, которые итеративно обновляются с помощью двух чередующихся шагов:

Шаг 1: Назначить каждую точку данных ближайшему центру кластера (центроиду), создавая новые кластеры C_j .

Шаг 2: Пересчитать центры кластеров на основе вновь сформированных кластеров.

Шаги 1 и 2 повторяются до тех пор, пока изменения внутри каждого кластера не стабилизируются и не произойдет никаких дальнейших существенных изменений.

Важно отметить, что хотя алгоритм *k*-средних применим как к задаче *k*-средних, так и к задаче *k*-медианы, вычислительная сложность для этих двух случаев существенно различается. Ключевое отличие заключается во втором шаге алгоритма.

Для задачи *k*-средних нахождение центроида одного кластера включает простое вычисление арифметического среднего каждой координаты векторов данных в этом кластере. Это относительно просто и вычислительно эффективно.

Напротив, для задачи p-медианы (в частности, для задачи 1-медианы) определение центра одного кластера требует решения задачи Вебера. Эта задача обычно решается с помощью итерационных процедур, таких как алгоритм Вайсфельда [91] или его более современных вариаций [92 – 98]. Хотя эти продвинутые алгоритмы могут улучшить скорость сходимости в определенных ситуациях, они по-прежнему в своей основе полагаются на итерационные процедуры для нахождения центра кластера с предопределенным уровнем точности.

В результате алгоритм *k*-средних получил гораздо более широкое распространение в кластерных приложениях, в том числе для построения индексов векторных баз данных, благодаря своей относительной простоте и эффективности по сравнению с более сложной задачей p-медианы.

При построении индексов векторных баз данных поиск центральных точек можно комбинировать с другими методами, такими как квантование продуктов [74]. Процесс поиска центральных точек и кластеризации данных может быть многоэтапным: изначально определяются большие кластеры векторов данных, а затем каждый из этих кластеров далее делится на более мелкие векторы данных и т. д. Однако кластеризация составляет основу большинства подходов, что приводит к гипотезе о том, что качество решений кластеризации определяет эффективность поиска в векторных базах данных.

Как упоминалось ранее, алгоритм *k*-средних требует начального набора центров кластеров, которые он затем уточняет. Однако *k*-средних является алгоритмом локального поиска и обеспечивает только локально оптимальное решение задачи *k*-средних или *k*-медианы. Более того, полученное решение сильно зависит от предоставленного начального набора центров. Например, в классической задаче BIRCH3 алгоритм *k*-средних часто дает решение, которое почти в три раза хуже с точки зрения целевой функции по сравнению с оптимальным решением [99].

Значительный прогресс был достигнут в инициализации алгоритма *k*-средних. Более ранние методы предлагали случайный выбор k векторов данных в качестве начальных центров. Напротив, алгоритм *k*-means++ [100] и его вариации [101] предлагают выбирать начальные центры с вероятностями, обратно пропорциональными расстоянию от ближайшего уже выбранного центра. Такой подход часто значительно улучшает результаты алгоритма *k*-средних, хотя фундаментальные проблемы остаются. Например, даже с этими улучшениями производительность алгоритма *k*-средних в сложной задаче BIRCH3 показывает лишь незначительное улучшение.

Более продвинутые алгоритмы для поиска соседей дают лучшие результаты. Например, алгоритмы, использующие подход соседства SWAP [102, 99], включая самый простой, j-средние [103], показали многообещающие результаты. Эти алгоритмы включают проверку всех векторов данных в качестве потенциальных центров на каждой итерации, что делает их практически неосуществимыми для очень больших наборов данных из-за их высокой вычислительной стоимости. Агломеративные подходы также продемонстрировали высокую производительность. В этих методах предлагается начальное решение с избыточным количеством центров, а затем количество центров уменьшается. Хотя эти алгоритмы также требуют больших вычислительных затрат, они применимы к наборам данных, состоящим из миллионов векторов [101]. Агломеративные процедуры изначально были введены как оператор кроссовера в эволюционных алгоритмах для решения задачи p-медианы [84] и позже были адаптированы к другим задачам размещения, включая задачу *k*-средних. Они эффективно применялись в более простых алгоритмических фреймворках, таких как поиск переменного соседства [104 – 106], эволюционные алгоритмы $1 + \lambda$ [107, 108] и эволюционные алгоритмы 1+1 [109 – 112].

Для векторных баз данных ключевой метрикой производительности является Recall@x [113, 114] – количество правильно идентифицированных ближайших соседей из x, т.е. из топ-элементов результатов поиска. Кроме того, фаза построения индекса, которая включает выполнение алгоритма кластеризации, должна быть завершена за разумное время.

Мы предлагаем алгоритм, основанный на жадной агломеративной процедуре для построения индекса для векторной базы данных, с использованием модели *k*-средних или *p*-медианной. Этот алгоритм несколько проще, чем ранее используемый эволюционный алгоритм с жадной агломеративной процедурой BasicAggl Алгоритм 3.2 с операторами кроссовера. Тем не менее, вычислительные эксперименты показывают, что индекс (набор центров), сгенерированный этим алгоритмом, достигает более высокой оценки полноты по сравнению с алгоритмами *k*-средних и *k*-means++.

3.2 Описание расширенного алгоритма кластеризации для решения задачи приближенного поиска ближайшего соседа

Алгоритмы кластеризации, использующие жадную агломеративную эвристическую процедуру, работают, изначально создавая решение с избыточным количеством центров. Затем это решение уточняется с помощью алгоритма *k*-средних. Затем центры удаляются из решения, а оставшиеся центры дополнительно улучшаются с помощью *k*-средних, повторяя эти шаги до тех пор, пока не останется желаемое количество центров. Удаление центра обычно приводит к увеличению значения целевой функции, что указывает на ухудшение решения. Жадные агломеративные алгоритмы решают эту проблему, удаляя центры, которые вызывают наименьшее увеличение целевой функции.

В [115] был предложен метод значительного ускорения жадной агломеративной процедуры: несколько центров (обычно 20-25% «лишних» центров) удаляются одновременно из промежуточного решения. Этот подход, который немного ухудшает качество конечного решения, становится необходимым для большого количества кластеров. Для больших наборов данных количество кластеров составляет не менее тысяч, что делает поочередное удаление центров крайне неэффективным с точки зрения вычислительных ресурсов.

Как и классический алгоритм *k*-средних, агломеративные процедуры требуют начального решения. Алгоритмы, включающие жадную агломеративную процедуру, отличаются в первую очередь методами генерации начальных решений. Например, генетические алгоритмы работают с популяцией решений (наборами центров). На каждой итерации выбирается пара «родительских» решений, объединяется в промежуточное решение с избытком центров, а затем уточняется с помощью агломеративной процедуры для уменьшения количества центров до желаемого количества. Таким образом, результатом агломеративной процедуры в таком генетическом алгоритме является «дочернее» решение, сгенерированное путем объединения «родительских» решений. Эту процедуру кроссинговера необходимо повторять много раз, что, учитывая высокую вычислительную стоимость одного запуска агломеративной процедуры, делает ее непрактичной для больших наборов данных.

Алгоритм 3.1 Процедура Ллойда (*k*-средних)

Требуется: Набор начальных центров $C = \{X_1, \ldots, X_k\}$. Если C не задано, то начальные центры выбираются случайным образом из набора векторов данных $\{A_1, \ldots, A_N\}$.

Повторить:

Шаг 1. Для каждого центра X_j , $j = 1, \ldots, k$, строится подмножество G_j векторов данных, для которых X_j является ближайшим центром;

Шаг 2. Для каждого подмножества G_j , $j = 1, \ldots, k$, вычислить его новый центр, решив задачу Вебера с помощью алгоритма Вайсфельда [41] на G_j (в случае задачи p-медианы) или усреднив все векторы данных в G_j (в случае задачи *k*-средних);

Пока все центры не останутся неизменными.

Более простые реализации жадной агломеративной процедуры включают алгоритмы, такие как эволюционный алгоритм 1+1, где одно решение «развивается». Второе решение генерируется случайным образом в каждой итерации и объединяется с развивающимся решением. Даже этот упрощенный

подход требует больших вычислительных затрат.Мы предлагаем более простой алгоритм, основанный на жадной агломеративной процедуре. Алгоритм генерирует два начальных решения, либо путем случайного выбора векторов данных в качестве начальных центров, либо с помощью процедуры *k*-means++. Эти решения улучшаются с помощью алгоритма *k*-средних, а затем объединяются в промежуточное решение с избыточным числом центров, которое далее уточняется с помощью жадной агломеративной процедуры.

Алгоритм 3.2 BasicAggl(C)

Требуется: Набор начальных центров $C = \{X_1, \ldots, X_K\}$, $K > k$, требуемое число k.

Шаг 1. $C \leftarrow Lloyd(C)$;

Шаг 2. Пока $|C| > k$ выполнять

Шаг 2a. Для $i = 1, \ldots, K$ выполнять

$F_i \leftarrow F(C \backslash \{X_i\})$.

Шаг 2b. Выбрать подмножество $C` \subset C$ центров r_{elim} с минимальными значениями соответствующих переменных F_i; $r_{elim} = 1 + \lfloor 0.25\ (K - k) \rfloor$.

Шаг 2c. $S \leftarrow Lloyd(C \backslash C`)$.

Вычислительные эксперименты показывают, что этот простой алгоритм достаточен для значительного улучшения качества результатов кластеризации, что приводит к улучшению метрики полноты для приближенного поиска ближайших соседей с использованием построенного индекса.

Алгоритм 3.3 Greedy (Упрощенная жадная агломеративная процедура для построения индекса векторной базы данных)

Шаг 1. Инициализируем два набора центров: C_1, C_2. Можно использовать два способа инициализации: случайным образом выбрать k векторов данных в качестве начальных центров или запустить алгоритм *k*-means++.

Шаг 2. $C_1 \leftarrow Lloyd\ (C_1)$; $C_2 \leftarrow Lloyd\ (C_2)$.

Шаг 3. Выполнить $BasicAggl(\ C_1 \cup C_2)$.

Этот алгоритм значительно проще, чем ранее используемые эволюционные алгоритмы с жадными агломеративными процедурами (например, BasicAggl с операторами кроссовера [116, 117]). Кроме того, он требует значительно меньше вычислительных ресурсов. Тем не менее, вычислительные эксперименты показывают, что этот подход существенно увеличивает метрику полноты для приблизительного поиска ближайшего соседа с использованием построенного индекса по сравнению с классическим алгоритмом *k*-средних и алгоритмом *k*-means++.

3.3 Результаты экспериментальных исследований

Для оценки эффективности поиска данных с использованием индексов, построенных с использованием различных моделей и алгоритмов кластеризации, мы использовали два классических набора данных для задачи приближенного поиска ближайших соседей (ANN): SIFT10K (10 000 векторов данных размерностью 128) и SIFT1M (1 000 000 векторов данных размерностью 128). Индексы были построены с использованием различного количества центров, причем для кластеризации использовались как *k*-средние, так и *p*-медианные формулировки. Для генерации начальных решений в классическом алгоритме *k*-средних и предлагаемом агломеративном алгоритме использовались два метода инициализации: случайный выбор векторов данных в качестве начальных центров и алгоритм *k*-means++.

В таблицах 3.1–3.2 представлены сравнительные результаты для различных алгоритмов. Каждый алгоритм был запущен 30 раз. В таблицах показаны достигнутые значения целевой функции (сумма расстояний до ближайших центров в случае задачи *p*-медианы или сумма квадратов расстояний до ближайших центров в случае задачи *k*-средних), а также достигнутые значения полноты. Тестовые наборы запросов для наборов данных были определены для расчета метрики полноты, и мы использовали конкретные тестовые наборы запросов, предоставленные разработчиками наборов данных.

Таблица 3.1 – Сравнительные результаты алгоритмов кластеризации для задачи *k*-средних и использования индекса на их основе для приближенного поиска ближайшего соседа

Алгоритм кластеризации	Метод инициализации	Достигнутое значение целевой функции, 30 запусков			Достигнутая полнота 100@100, 30 запусков алгоритма		
		максимальное (худшее)	среднее	минимальное (лучшее)	максимальное (лучшее)	среднее	максимальное (худшее)
Набор данных SIFT1M, 256 центров							
Lloyd	Random	23508792	23516733,6	23525960	95,0547	94,95983	94,8563
Lloyd	*k*-means++	23500268	23513930	23521506	95,0643	94,96042	94,8627
Greedy	Random	23475608	23481650	23520708	95,3264	95,22806	94,8898
Greedy	*k*-means++	23475712	23478027,71	23481328	95,2568	95,21683	95,1711
Набор данных SIFT1M, 1024 центров							
Lloyd	Random	21898272	21905756	21913664	98,1763	98,1094	98,0786
Lloyd	*k*-means++	21893030	21902027	21915006	98,1882	98,15466	98,1142
Greedy	Random	21771792	21774277	21776486	98,3914	98,36305	98,3296
Greedy	*k*-means++	21787168	21789126	21793534	98,365	98,33062	98,2981
Набор данных SIFT1M, 4096 центров							
Lloyd	Random	20153940	20161542	20171012	99,2685	99,24751	99,225
Lloyd	*k*-means++	20144342	20150518	20157960	99,3136	99,28448	99,2555
Greedy	Random	19796108	19810150	19824192	99,3038	99,31032	99,2966
Greedy	*k*-means++	19897040	19898170	19899276	99,2999	99,28422	99,2744
Набор данных SIFT10K, 16 центров							
Lloyd	Random	6813531,5	6816816,25	6821559	93,760002	92,77214	91,64
Lloyd	*k*-means++	6813700	6815397,9	6817533	93,29	92,788	91,56
Greedy	Случайный	6813594	6813669,357	6813709	92,839996	92,74857	92,68
Greedy	*k*-means++	6813606	6815072,214	6817917	93,339996	92,99714	92,72

Таблица 3.2 – Сравнительные результаты алгоритмов кластеризации для задачи p-медианы и использования индекса на их основе для приближенного поиска ближайшего соседа

Алгоритм кластеризации	Метод инициализации	Достигнутое значение целевой функции, 30 запусков			Достигнутая полнота 100@100, 30 запусков алгоритма		
		максимальное (худшее)	среднее	минимальное (лучшее)	максимальное (лучшее)	среднее	максимальное (худшее)
Набор данных SIFT1M, 256 центров							
Lloyd	Random	1526660,5	1526894,047	1527368	94,9902	94,91393	94,8528
Lloyd	*k*-means++	1526220	1526647,51	1526971	95,0227	94,93122	94,8067
Greedy	Random	1525145,7	1525279,839	1525429	95,2924	95,2326	95,1788
Greedy	*k*-means++	1525222	1525350,163	1525510	95,2534	95,20896	95,1509
Набор данных SIFT1M, 1024 центров							
Lloyd	Random	1472318	1472596	1473034	98,1035	98,0772	98,0313
Lloyd	*k*-means++	1472276	1472479	1472652	98,1959	98,16716	98,1285
Greedy	Random	1467882	1467984	1468104	98,3833	98,37234	98,3554
Greedy	*k*-means++	1468462	1468553	1468695	98,3617	98,33311	98,3033
Набор данных SIFT1M, 4096 центров							
Lloyd	Random	1411846	1411933	1411933	99,255	99,2498	99,2498
Lloyd	*k*-means++	1411796	1412248	1412550	99,3298	99,3002	99,2897
Greedy	Random	1399359	1399488	1399571	99,3131	99,28942	99,279
Greedy	*k*-means++	1401860	1402281	1404126	99,3122	99,29232	99,2581
Набор данных SIFT10K, 16 центров							
Lloyd	Random	411371,37	411446,0469	411559	93,470001	92,473	91,55
Lloyd	*k*-means++	411368,68	411426,9688	411477,3	93,3	92,562	91,38
Greedy	Random	411357,09	411361,8527	411370,3	92,82	92,73143	92,57
Greedy	*k*-means++	411362,21	411403,8348	411412,3	92,889999	92,82286	92,74

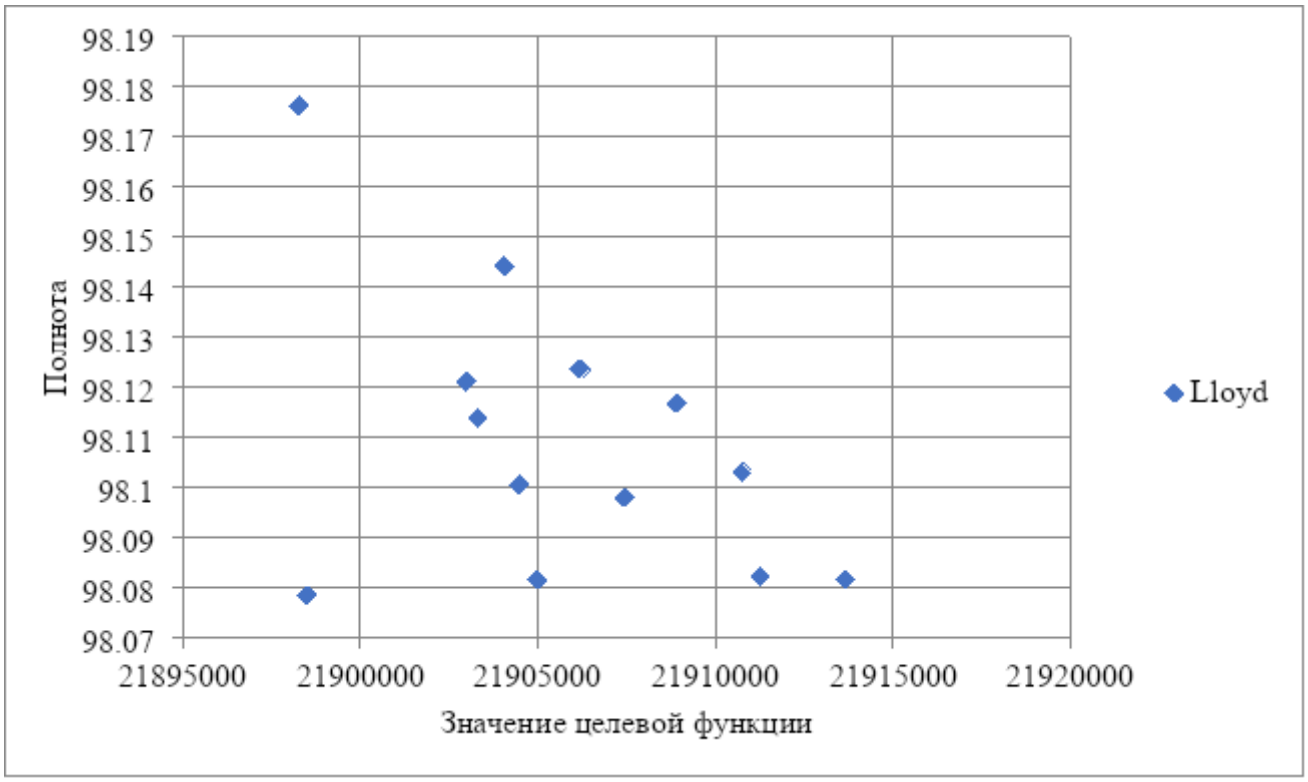

Рисунок 3.1 – Зависимость полноты поиска ближайшего соседа на основе индекса, построенного алгоритмом Ллойда для задачи *k*-средних, от достигнутого значения целевой функции *k*-средних для набора данных SIFT1M с разделением на 1024 центра

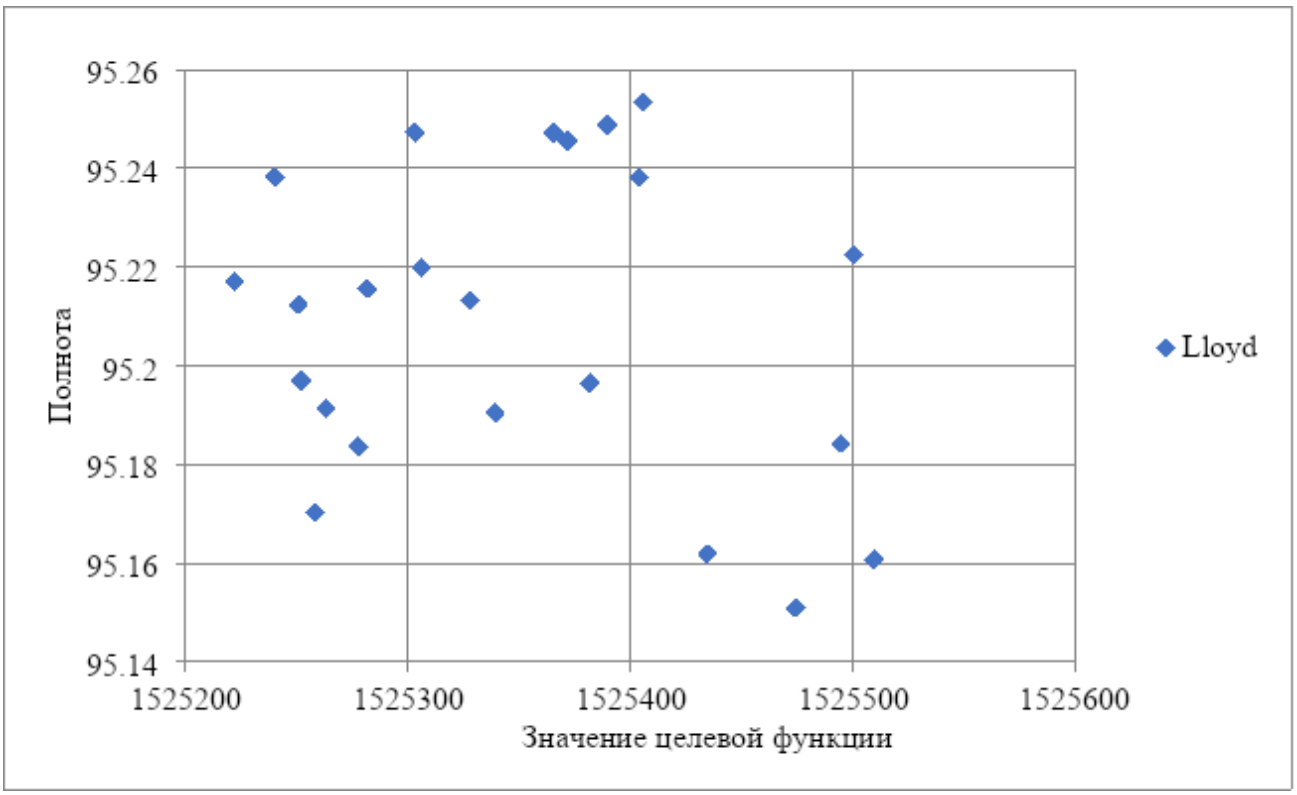

Рисунок 3.2 – Зависимость полноты поиска ближайшего соседа на основе индекса, построенного алгоритмом Ллойда для задачи p-медианы, от достигнутого значения целевой функции p-медианы для набора данных SIFT1M с разделением на 256 центра

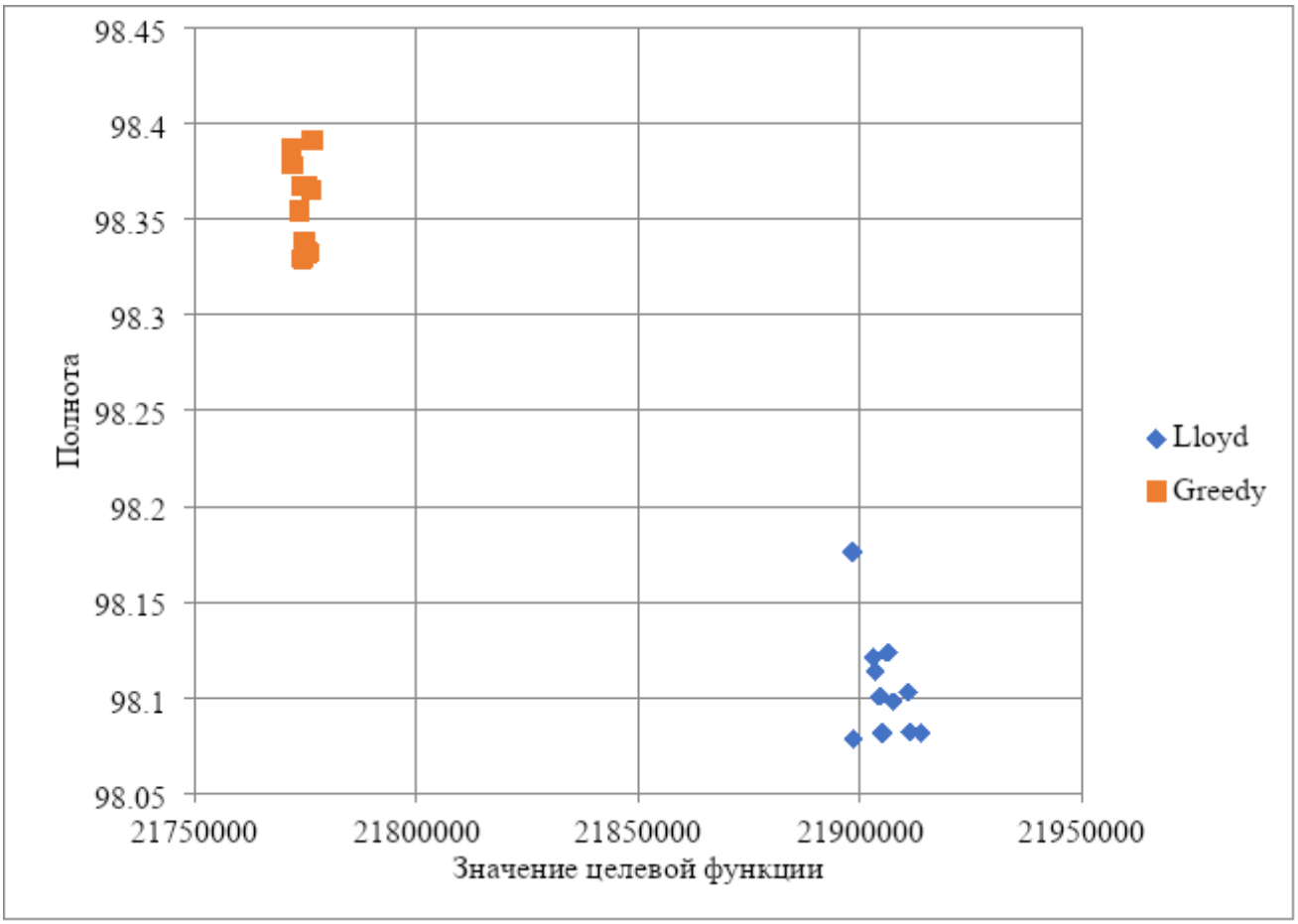

Рисунок 3.3 – Зависимость полноты поиска ближайшего соседа на основе индекса, построенного с помощью алгоритма Ллойда и жадного агломеративного алгоритма для задачи *k*-средних, от достигнутого значения целевой функции *k*-средних для набора данных SIFT1M с разделением на 1024 центра

Новый алгоритм кластеризации Greedy, основанный на жадной агломеративной процедуре для построения индекса для векторной базы данных в вычислительных экспериментах на рисунках 3.1 – 3.3 показывает, что индекс (набор центров), сгенерированный этим алгоритмом, достигает более высокой оценки полноты по сравнению с алгоритмом *k*-средних.

Как видно из таблицы, даже при очень незначительных различиях в достигнутых значениях целевой функции кластеризации значения полноты могут существенно различаться. Тем не менее, определенная корреляция между значением целевой функции и полнотой все же наблюдается (рисунок 3.1). Алгоритм инициализации k-means++ не всегда улучшает производительность агломеративного алгоритма и не является достаточно требовательным к

вычислительным ресурсам. С ростом объема данных и числа выделяемых кластеров вычислительные затраты алгоритма инициализации k-means++ начинают многократно превышать затраты основного итеративного алгоритма Ллойда.

Результаты Раздела 3

Предложенный алгоритм кластеризации для построения приближенного индекса поиска ближайших соседей, как компромисс между точностью и временем вычислений, значительно улучшает метрику полноты в задачах приближенного поиска ближайших соседей. Наши эксперименты показывают, что значения полноты, достигнутые с помощью алгоритма кластеризации на основе модели *p*-медианой, не превосходят значений, достигнутых с помощью кластеризации на основе модели *k*-средних. Учитывая меньшие вычислительные затраты, связанные с алгоритмом *k*-средних, его следует предпочесть в аналогичных условиях.

Из таблиц 3.1-3.2 видно, что алгоритм инициализации *k*-means++ не всегда улучшает результаты жадного агломеративного алгоритма. Хорошее начальное решение не всегда гарантирует лучший конечный результат при использовании более сложных алгоритмов по сравнению с классической процедурой Ллойда. Следовательно, необходимость использования *k*-means++ при построении индексов векторной базы данных требует дальнейшего изучения.

Предложенный алгоритм, обеспечивающий компромисс, демонстрирует высокую эффективность кластеризации данных, но его текущая реализация требует значительных вычислительных ресурсов по сравнению с алгоритмом Ллойда. Дальнейшие исследования будут направлены на повышения эффективности самой жадной агломеративной процедуры, т.е. уменьшения разницы во времени вычислений при работе с большими наборами данных и увеличении количества центров/центроидов в результирующем индексе.

4. АЛГОРИТМ АВТОМАТИЧЕСКОЙ ГРУППИРОВКИ ПОВТОРЯЮЩИХСЯ ФРАГМЕНТОВ БЛОКОВ ДАННЫХ

В разделе 4 рассмотрены вопросы разработки нового алгоритма автоматической группировки повторяющихся фрагментов блоков данных на основе алгоритма *k*-средних совместно с локально-чувствительным хэшированием (LSH) для использования в системах хранения данных. Результаты данного раздела были опубликованы в [118].

Предложен алгоритм автоматической группировки повторяющихся фрагментов блоков данных на основе упрощенной жадной агломеративной процедуры Greedy (с применением алгоритма *k*-средних) совместно с локально чувствительным хэшированием (LSH) для использования в системах хранения данных. Показано, что новый алгоритм повышает производительность сжатия.

4.1 Обзор литературы об алгоритмах автоматической группировки с применением хеширования с учетом местоположения LSH обзор источников

Растущее количество данных на серверах создает проблему их эффективного хранения, частично решаемую сжатием данных. Ключевым фактором эффективности сжатия является способность обнаруживать повторяющиеся или похожие фрагменты в данных. Однако данные на сервере часто представлены разнообразными файлами, которые зачастую не объединены по сходству. Большинство существующих методов сжатия ограничены словарем или длиной сжатия. Обеспечение эффективного сжатия для такого рода данных чрезвычайно сложно.

Увеличение количества и объема файлов на серверах требует повышения эффективности способа их хранения, и эта задача частично решаема сжатием блоков данных. При сжатии конечных последовательностей цифровой информации (данных) без потерь создается меньшая последовательность битов (кодов), по которой возможно точно восстановить каждый бит исходного блока,

количество которых определяет длину кода. Эффективность алгоритмов сжатия определяется отношением длины несжатых данных к сжатым данным (степенью сжатия), скоростью сжатия и восстановления. В обзоре Ватолина и др. [119] 2003 года изложены основные стандарты сжатия данных. Базовая стратегия сжатия (преобразование блоков) состоит в делении данных на блоки для трансформации и последующего сжатия, к которым относят преобразование Барроуза–Уилера.

Программы сжатия используют несколько основных концепций. Одна из них – алгоритм Хаффмана [120], предложенный еще в 1952 году. Это статистический метод, который присваивает каждому символу алфавита последовательность битов (кодовых слов) разной длины, редкие символы получают более длинные кодовые слова. Сжатие (кодирование) производится заменой символов соответствующими кодовыми словами. Алгоритм генерирует оптимальные префиксные коды за время $O(N \log_2 N)$ и выполняется за линейное время, если частоты появления блоков данных отсортированы заранее.

В 1977-78 годах Зив и Лемпель [121, 122] представили словарные методы сжатия. Эти методы обрабатывают данные слева направо и кодируют длинные повторяющиеся последовательности символов как ссылки на сжатую часть данных, чем достигается более высокий уровень сжатия по сравнению с алгоритмом Хаффмана.

В 1987 году Виттен и др. [123] предложили более эффективный метод сжатия, обеспечивающий лучшее распределение битов в соответствии с вероятностью каждого символа. В исследовании Белла и Виттена [124] 1989 года отмечалось, что статистические методы, такие как алгоритм Хаффмана, обычно медленнее, но обеспечивают лучшее сжатие, чем словарные методы Зива и Лемпеля.

В алгоритме сжатия данных Берроуза [125] 1994 года, основная идея заключается в преобразовании входного потока данных с группировкой одинаковых символов. Такое преобразование, являясь также основой для некоторых методов индексации последовательностей, само по себе не является методом сжатия, но используется для создания высокоэффективных

компрессоров, таких как bzip2, которые сочетают в себе данное преобразование с кодированием Хаффмана и другими методами.

Существует несколько основных концепций, которые используются в программах сжатия. Кодирование Хаффмана [120], предложенное в 1952 году, представляет собой статистический метод, который присваивает последовательность битов (кодовое слово) каждому символу алфавита. Кодовые слова имеют разную длину, и в соответствии с золотым правилом сжатия данных более редкие символы представлены более длинными кодовыми словами. Затем данная последовательность кодируется путем замены каждого символа соответствующим кодовым словом, что приводит к более короткой кодированной последовательности.

В 1977-78 годах Зив и Лемпель [121] предложили дискретные методы сжатия данных без потерь, алгоритм LZ77 и LZ78. Методы обрабатывают последовательности символов слева направо и кодируют, возможно, длительные повторения последовательных символов в качестве ссылок на уже сжатую часть данных. Такой подход обеспечивает более высокие коэффициенты сжатия, чем при кодировании только по Хаффману, поскольку используется тип избыточности распространенный не только в естественном языке, но и в повторяющихся словосочетаниях.

Берроуз и Уилер [125] в 1994 году предложили преобразование BWT. Преобразование BWT не является методом сжатия, он только изменяет последовательность ввода, но его можно использовать для создания высокоэффективных компрессоров. Хорошо известным представителем является bzip2 [126] Цяо 2019 года, сочетающий BWT с кодированием Хаффмана и другими методами. Еще лучшие результаты возможны при комбинировании словарных методов и кодирование Хаффмана. Очень популярная программа gzip [127] Кербириу 2019 года.

Брин [128] в 1995 году предложил алгоритм и метод обнаружения схожих документов и соответствующие метрики. По мере повсеместного распространения хранилищ становится все труднее организовывать растущие

системы репозиториев и управлять ими. Чем больше хранилище, тем больше документов будет задействовано, что приводит к увеличению вероятности появления похожих документов. Идентичные копии или более старые версии документов часто оказываются разделенными и разбросанными по структуре хранилища. Объединение или удаление нескольких версий документов становится желательным. Несколько исследований показывают, что примерно 30% документов в репозиториях схожи по своему содержанию. Наличие похожих документов может привести к неразрешенным ошибкам и проблемам, связанным с обслуживанием, увеличивая риск аномалий обновления/поиска при индексации.

Бродер [129] в 1997 году предложил метод MinHash для обнаружения дубликатов веб-страниц и исключения их из результатов поиска.

Индик и Мотвани [130] в 1998 году предложили хеширование с учетом локализации LSH для расстояния Хэмминга. Метод LSH применяется для решения задачи поиска ближайшего соседа, использует оптимальное количество нескольких хэш-функций и хэш-таблиц, чтобы иметь постоянные вероятности хэширования, и гарантировать хорошее качество поиска. В [131] Бюлер в 2001 году реализовали метод LSH в LSH-ALL-PAIRS для эффективного сравнения последовательности геномной ДНК с целью выявления консервативных особенностей генома у разных видов. LSH-ALL-PAIRS преобразует последовательности в пояс. Затем LSH применяется ко всем поясам, и пояса с одинаковым хэш-значением группируются вместе в класс. Чтобы найти похожие пояса попарное сравнение выполняется в одном классе. В [132] Азимпуркиви в 2017 голу реализовали новый подход к обнаружению аномалий видео с использованием LSH-фильтров. С использованием LSH точки данных хэшируются в набор сегментов. Используется измерение расстояния Хэмминга для определения расстояний между тренировочными и тестовым сегментом. В [132] LSH используется для сопоставления изображения с отпечатком их двоичного изображения. В [133] Цзян в 2011 году предложили новый метод S3H для поиска похожих документов в L-мерном пространстве Хэмминга с помощью LSH для генерирации отпечатков пальцев. В [134] Берлин в 2015 году применяет

LSH, вводит процесс выравнивания MinHash (MHAP) для обнаружения совпадений между зашумленными и длинными считываниями микробных геномов. MHAP использует MinHash для создания небольших отпечатков последовательных считываний с целью уменьшения размерности. Чтобы сделать это MHAP разбивает последовательности ДНК на несколько фрагментов, а затем эти фрагменты преобразуются в целочисленные отпечатки пальцев с использованием нескольких случайных хэш-функций. Для определения перекрытия между двумя поясами используется расстояние Хэмминга отпечатков пальцев для определения приближеного расстояния Жаккарда между ними. В [135] Моура в 2017 году применил LSH, чем ускорил процесс поиска паттернов в библиотеках изображений, с использованием расстояний Евклида и Хэмминга. В [136] Чен в 2016 году применил LSH для решения проблемы масштабируемости применения больших обученных моделей к огромным коллекциям мультимедиа без аннотаций. В [137] Ким в 2016 году использует два метода LSH для прогнозирования критических событий по физиологическим временным рядам. Метод L1LSH [138] Гиониса 1999 года, основанный на расстоянии Хэмминга, и E2LSH [101] Датара 2004 года, основанный на расстоянии Евклида, используются для поиска ближайших соседей по заданному запросу. Позже метка класса запроса выбирается большинством голосов его ближайших соседей.

Чарикар [140] в 2002 году предложил новый метод SimHash для поиска похожих документов, который обеспечивает почти повторение отпечатков пальцев с разницей лишь в небольшое количество битовых позиций. SimHash применим для хранилища в 8 миллиардов страниц с использованием 64-битных отпечатков SimHash.

Датар [139] 2004 года предложили метод E2LSH для расстояния Евклида. Основная идея E2LSH заключается в увеличении вероятности столкновения двух близлежащих точек при использовании нескольких хэш-таблиц и составных хэш-функций. В [141] Рююнанен в 2008 году применили для поиска музыки LSH с расстояниями Евклида. В [142] Жувикин в 2018 году предложил блокчейн-схему для защиты авторских прав на изображения и предоставляет авторские права по

сети в условиях ограничений распространения с использованием E2LSH. Ли [143] в 2014 году на основе LSH и онтологии предметной области метод крупномасштабного сжатия документов. В [144] Шривастава в 2014году предложил ALSH асимметричный метод на основе E2LSH, который для создания и зондирования сегментов использует различные хэш-функции. В [137] Ким в 2016 году использует два метода L1LSH [138], Гиониса 1999 года и E2LSH [139] Датара 2004 года, для прогнозирования критических событий по физиологическим временным рядам с использованием поиска ближайших соседей по заданному запросу, которому присваивается метка класса выбором большинством голосов его ближайших соседей. В [145] Кандзи в 2008 году реализовал алгоритм iLSH для создания масштабируемой платформы глобальной локализации и отслеживания местоположения в роботизированных системах, используя E2LSH и метод Монте-Карло. В [146] Саеки в 2009 году реализовал алгоритм LSHRANSAC при решении задачи локализации роботов на основе признаков на картах большого размера, используя инкрементные карты, основанные на iLSH [145]. Вычисленные местоположения новых объектов в реальном мире хэшируется с помощью E2LSH.

Бава [147] в 2005 году предложил новый алгоритм LSH Forest с LSH основанный на расстоянии Жаккара для создания дерева префиксов для каждой хэш-таблицы и сохранения составных хэш-ключей в деревьях префиксов, с иерархической стратегией поиска. В [148] Пробст в 2018 году предложил алгоритм MHFP6 с шестью отпечатками MinHash для повышения производительности поиска ближайших соседей и использовал индексацию методом LSH Forest [147] для эффективного извлечения ближайших соседей из больших наборов молекулярных данных. В [149] Юй в 2013 году улучшает масштабируемость приложений для обнаружения местоположения за счет обработки сотен полигонов и точек в режиме реального времени. В [150] Кочез в 2017 году использует LSH Forest [147], чтобы найти правильный контекст среды для размещения новых токенов знаний.

Кэйтон [151] в 2007 предложил алгоритмический фреймворк, который использует плотностные характеристик набора данных, чтобы выбирать хэш-функции LSH.

Лю [152] в 2007 году предложил алгоритм Multyprob, мульти-зондовый LSH для просматривания нескольких соседних сегментов, так как ближайшие соседи с большей вероятностью будут хэшированы в близлежащие сегменты. Чжан [153] в 2015 году предложил алгоритм для поиска подобия в неориентированных простых графах с метками вершин, которые не имеют собственных циклов и множественных ребер, с использованием мульти-зондового LSH [152] для запроса сгенерированых векторов. Лю [154] в 2006 году предлагает LSH с хэш-возмущением для уменьшения большого количества требуемых хэш-таблиц относительно базового LSH. Жоли [155] в 2008 году предлагает усовершенствованный мульти-зондовый LSH [152] с использованием вероятностных подходов вместо правдоподобия. Жегоу [156] в 2008 году предлагает метод адаптивного хэширования запросов, который на этапе обработки запроса использует ожидаемое измерение точности для выбора наилучших хэш-функций, более подходящих для данного запроса.

Чжан [157] в 2010 году предлагает метод, зависящий от данных, который использует анализ основных компонентов для уменьшения размеров набора данных таким образом, чтобы генерировался единообразный набор данных.

Дасгупта [158] в 2011 году использует преобразования Адамара для лучшей оценки расстояний в евклидовом и угловом пространствах и сокращения времени работы методов LSH с одним ACHash и двумя DHHash преобразованиями Адамара.

Сатулури [159] в 2011 году предложил метод BayesLSH для удаления ложных срабатываний в E2LSH [139] с применением евклидовой, угловой и жаккардовой метрики.

Кулис [160] в 2011 году предлагает метод основанный на LSH без требования знания о распределении и встраивании входных данных .

Ган [161] в 2012 году предложил алгоритм C2LSH с одной хэш-таблицейа и m случайные хэш-функции.

Пан [162] в 2012 году предлагает двухуровневый LSH для повышения точности и времени выполнения поиска.

Ванг [163] в 2012 году предложил алгоритм хеширования чувствительного к расширению границ локальности для задачи хеширования ближайших точек в границах разных сегментов, увеличивая вероятность столкновения соседей.

Джи [164] в 2012 году предложил алгоритм SBLSH на основе угловых расстояний для уменьшения дисперсии оценки в результате разделения данных на проекции (супербиты), когда угол для оценки находится в пределах $(0, \pi/2]$.

Деорович [165] в 2013 году описывал преобразование BWT, которое также является основой некоторых методов индексации последовательностей. Ключевая идея BWT заключается в перестановке входной последовательности таким образом, чтобы символы группировались по их близости (сходства), то есть символы, за которыми следуют одни и те же символы, после перестановки оказываются близкими, даже если они были далеко в исходной последовательности.

Памулапарти [166] в 2013 году показал, что ручная проверка результатов кластеризации документов возможна, если масштаб репозитория невелик, например, сотни или тысячи экземпляров. Когда количество экземпляров увеличивается до миллионов и более, очевидно, что для людей становится невозможным проверять их один за другим, что является утомительным, дорогостоящим и подверженным ошибкам. Требуется использование компьютеров для такого рода повторяемой работы, ядром которой является алгоритм для измерения различий между любой парой документов, включая дублированные и почти дублированные.

Ли [167] в 2013 году заключает, что иногда точки проецируются на границы сегментов, что делает их ложноотрицательными или ложноположительными в зависимости от сегмента запроса. Чтобы решить эту проблему в [163] вводят концепцию проецирования точки в более чем один сегмент, используя три хэш-

функции, которые сопоставляют точку с 1) текущим сегментом, 2) сегментом слева и 3) сегментом справа.

Гу [168] в 2013 году описывает процесс удаления ложноположительных результатов, которые требуют вычисления расстояния Евклида.

Инь [169] в 2013 году предложил динамический многозондовый LSH, для оптимизации эффективности ввода-вывода путем динамического изменения количества хэш-функций каждого сегмента таким образом, чтобы сегмент соответствовал одной странице диска. Для этого процесса используется B+tree, и в результате этого метода генерируются сегменты с различной степенью детализации.

Чжан [170] 2013 году предложил усовершенствованную стратегию с несколькими зондами LSH.

Ли [171] в 2013году предложил модифицированный алгоритм LSH для работы с категориальными данными для поиска матрицы сходства между всеми категориальными значениями кластеризации категориальных значений с использованием агломеративной иерархической кластеризации. Идентификаторы кластеров проецируются в разные сегменты чтобы каждое категориальное значение могло быть сопоставлено с идентификатором кластера.

Бай [172] в 2014 году для LSH предложил подход, зависящий от данных. Этот подход проецирует одну точку набора данных на несколько случайных проекций, а затем использует распределение данных для получения меньшего числа проекций, способных аппроксимировать проекции, созданные на предыдущем шаге. Этот подход предложен для расстояний Евклида и экспериментально доказал свою эффективность лучше, чем E2LSH.

Сье [173] в 2014 году используется метод *k*-средних для кластеризации набора данных в несколько групп, а затем выполняется E2LSH для каждого кластера для построения таблиц LSH.

Андони [174] в 2014 году предложен метод хэширования, зависящий от данных. Предлагаемый метод использует два уровня хэш-функций для дальнейшего сокращения прогнозов.

Ван [175] в 2014 году предложил двухуровневый метод LSH.

Сан [176] в 2014 году предложил алгоритм SRS, который использует структуры индексов R-tree для оценки исходных расстояний точек по их прогнозируемым расстояниям.

Лю [177] в 2014 году предложил алгоритм SKLSH с уменьшением случайного ввода-вывода за счет эффективного размещения близлежащих проецируемых точек на одних и тех же или близких страницах диска. Алгоритм SKLSH использует кривую заполнения пробелов и новую меру расстояния между составными хэш-ключами для оценки расстояния между точками в исходном пространстве.

Джи [178] в 2014 году предложил пакетно-ортогональное BOLSH.

Чакрабарти [179] в 2015 году улучшает BayesLSH [159], добавляя поддержку произвольных мер подобия ядра.

Хуан [180] в 2015 году предложил алгоритм QALSH для создания хэш-функции с учетом запросов.

Чжэн [181] в 2016 году предлагает алгоритм LazyLSH, для уменьшения вероятности того, что две близлежащие точки окажутся в двух разных lp пространствах.

Ю [182] в 2016 году предложил алгоритм, который выбирает небольшой репрезентативный набор из набора данных запроса, чтобы уменьшить количество запросов LSH; таким образом, увеличить время обработки.

Хуан [183] в 2017 году предложил улучшенный алгоритм QALSH [178] для работы с lp нормами, где $p \in (0, 2]$.

Лю [184] в 2019 году предложил алгоритм ILSH, в котором в процессе расширения радиуса LSH, поиска в проекциях увеличивается, чтобы искать кандидатов в соседних сегментах запроса.

Донг [185] в 2019 году предложил алгоритм, в котором для изучения характеристик набора данных используются модели машинного обучения.

Ким [186] в 2020 году предложил новый алгоритм [126], в котором для уменьшения избыточности в проекциях метод LSH последовательно обучает линейные классификаторы, используя парадигму адаптивного повышения.

Чжэн [187] в 2020 предложил алгоритм PM-LSH, в котором используют PM-tree для индексации данных и увеличения времени обработки запросов, настраивает доверительный интервал для более точной оценки расстояния и обеспечения более высокой точности результатов.

Лу [188] в 2020 году предложил новый двумерный метод R2LSH, в котором вместо одномерных проекций R2LSH используются двумерные проекции и сопоставляются точки набора данных с этими проекциями, а на этапе индексации строится B+tree для каждой из двумерных проекций/

Ронг [189] в 2018 году предлагают усовершенствованный алгоритм [188] с параллельной обработкой.

Основными преимуществами LSH являются его сублинейное время выполнения запроса и теоретические гарантии точности запроса. Различные методы LSH основаны на расстоянии Жаккарда, Минковского, Хэмминга или угловом расстоянии.

Идея переупорядочить входную последовательность таким образом, чтобы похожие (схожие) символы группировались вместе, благодаря чему соседние символы в исходной последовательности стали бы ближе, даже если изначально они находились далеко друг от друга, описана Деоровичем [165] в 2013 году.

Другая программная реализация – архиватор gzip – сочетает методы на основе словаря с кодированием Хаффмана. Схема архивирования данных, сжатых gzip, используется в методе Кербириу и Чихи [127] 2019 года для параллельного сжатия текстовых данных. В методе сжатия Шаха и Сетхи [190] 2019 года используется алгоритм gzip для сжатия данных в реальном времени с использованием адаптивного алгоритма Хаффмана вместо статического.

В составе преобразования сжатия используют методы автоматической группировки (кластеризации), такие как *k*-средних в области кластеризации изображений в работе Котелина и Матвийчука [191] 2019 года. В данной работе

указывается на влияние случайного выбора центров кластеров и улучшение кластеризации после неоднократного запуска *k*-средних. В алгоритме Мунши [192] 2021 года кластеризация используется для сжатия (уменьшения) изображений, а в подходе Дворского [193] года – для улучшения сжатия документов.

Обнаружение сходства фрагментов данных, рассматриваемое в схеме Вана [194] 2024 года, использует метод удаления избыточности (дедупликации) данных, в котором задействуется алгоритм фрагментирования Ни и Янга [195] 2019 года для разделения файла на несколько фрагментов фиксированной или переменной длины. Такой способ фрагментирования описан Ся и Чжаном [196, 197] (2015-2016 годов.). После фрагментации вычисляется значение хэш-функции SHA-1 для каждого фрагмента. Для обнаружения сходства также используются методы статистического сжатия. Так, например, SLH (вариант алгоритма LZSS, описанного Брентом [198] в 1987 году) использует хеш-таблицы для определения сходства.

В методе E2LSH Датара [139] (2004 г.) предлагается использовать хеширование с учетом местоположения для расстояний Евклида. Основная идея E2LSH заключается в использовании нескольких хэш-таблиц и составных хэш-функций для увеличения вероятности совпадения двух близлежащих точек в евклидовом пространстве. Используя несколько хэш-таблиц и несколько хэш-функций, E2LSH уменьшает количество ложноположительных и ложноотрицательных совпадений.

Алгоритм Чарикара SimHash [140] (2002 г.) задействован для выявления близких по содержанию последовательностей данных («почти дубликатов») в методе Манку и др. [199] 2007 года. Метод генерирует отпечатки для последовательностей данных, которые отличаются лишь несколькими битовыми позициями для практически идентичных исходных последовательностей. Для каждого объекта генерируется отпечаток SimHash. Подобные (схожие) отпечатки указывают на наличие «почти дубликатов». Эксперименты в [199] показывают,

что 64-битных отпечатков SimHash достаточно для организации хранения 8 миллиардов страниц (блоков данных).

Памулапати и Рао в [166] описывали применение алгоритма кластеризации совместно с SimHash. Центр тяжести кластера формируется таким образом, чтобы он был тесно связан со всеми объектами в этом кластере с точки зрения функции сходства: косинусного сходства, расстояния Евклида и Жаккара.

Цель нашего исследования – увеличить степень сжатия путем расположения одинаковых (схожих) достаточно крупных отсортированных заранее блоков данных (в сравнении с группировкой символов) последовательно для архивирования совместно с известными методами сжатия Лемпеля и Зива.

4.2 Новый гибридный алгоритм автоматической группировки повторяющихся фрагментов блоков данных

В настоящей работе предлагается новый алгоритм, идея которого состоит в разделении данных на ряды блоков, группировки рядов блоков по схожести с применением алгоритма *k*-средних и обработке всех блоков данных. Пусть N рядов блоков по M блоков в каждом составляют все данные. Заменим каждый блок значением его хеш-функции. Каждый ряд блоков заменим вектором значений хеш-функции соответствующих блоков. Данные заменим N векторами по M значений хеш-функции соответствующих блоков.

Схожесть определяем между рядами блоков, векторами значений хеш-функции соответствующих блоков, для последующей группировки алгоритмом *k*-средних.

Для определения сходства между рядами блоков используем локально чувствительную хеш-функцию (LSH) SimHash для каждого ряда. Полученные векторы значений хеш-функции соответствующих рядов блоков перегруппировываем с применением алгоритма *k*-средних, после чего архивируем данные (применяем известные алгоритмы компрессии данных).

Мы используем избыточную информацию (повторяющиеся и схожие фрагменты, которые алгоритм *k*-средних относит в одну группу), содержащуюся в рядах блоков, для сокращения общего объема хранимых данных.

Разделим исходные данные на N рядов блоков по M блоков в каждом ряду и вычислим для каждого локально чувствительную хэш-функцию SimHash (Алгоритм 4.1). Из предварительно обработанных блоков данных создаем набор векторов значений хеш-функции соответствующих рядов блоков с применением алгоритма SimHash для генерации f-разрядного (в нашей реализации алгоритма – 64-битного) значения хеш-функции каждого блока. Затем используем расстояния Хэмминга для поиска схожих рядов блоков. Расстояния Хэмминга выбраны для уменьшения вычислительных затрат на поиск похожих рядов блоков.

Основная схема использования алгоритма SimHash для измерения сходства двух рядов блоков описывается следующим образом.

Алгоритм 4.1 Вычисление расстояния Хэмминга с SimHash

Дано: индексы двух рядов блоков $\boldsymbol{i,j} \in \{\boldsymbol{1 \ldots N}\}$, для которых требуется определить сходство.

Шаг 1а. Для каждого блока данных *i*-го ряда вычислить признаки $\boldsymbol{f_1 \ldots f_M}$, применяя к каждому блоку хеш-функцию SimHash.

Шаг 1б. Для каждого блока данных *j*-го ряда вычислить признаки $\boldsymbol{f_1 \ldots f_M}$, применяя к каждому блоку хеш-функцию SimHash.

Шаг 2. Создать векторы $\boldsymbol{x_i = f_1 \ldots f_M}$ и $\boldsymbol{x_j = f_1 \ldots f_M}$ размерности $\boldsymbol{M}$ из f-разрядных значений хеш-функций для каждого рядов $\boldsymbol{i}$ и $\boldsymbol{j}$.

Шаг 3. Вычислить расстояния Хэмминга между векторами *i*-го и j-го ряда блоков.

Шаги 1 и 2 Алгоритма 4.1 составляют шаги предварительной обработки данных для последующего применения метода *k*-средних. Шаг 3 выполняется в новом алгоритме LSH *K*-MEANS для расчета расстояний между парами группируемых рядов блоков.

После завершения предварительной обработки в Алгоритме 4.1 получаем f-разрядный вектор значений хеш-функций каждого ряда блоков, используя алгоритм SimHash.

Для сравнения эффективности применения локально-чувствительной хеш-функцию (LSH) SimHash используем хеш-функцию md5 [200].

Измерение значения сходства двух рядов блоков данных в Алгоритме 4.1 используем в Алгоритме 4.2 LSH *K*-MEANS для группировки рядов блоков по схожести с применением алгоритмом *k*-средних и обработке всех блоков данных.

Алгоритм 4.2 LSH *K*-MEANS

Шаг 1. Подготовить данные.

Шаг 1.1. Разделить данные на блоки из *N* строк, каждая из которых разбита на *M* блоков фиксированного размера.

Шаг 1.2. Для каждого блока вычислить 64-битный отпечаток с использованием алгоритма SimHash.

Шаг 1.3. Объединить хэш-значения блоков каждой строки в векторы длины *M*, для получения *N* векторов, каждый из которых состоит из *M* 64-битных значений.

Шаг 2. Инициализировать центры кластеров.

Шаг 2.1. Задать количество кластеров *k*.

Шаг 2.2. Случайным образом выбрать *k* векторов из подготовленных данных в качестве начальных центров кластеров.

Шаг 3. Кластеризовать данные.

Шаг 3.1. Для каждого вектора вычислить расстояние Хэмминга до каждого из k центров кластеров. Каждому вектору кластера назначить ближайший центр.

Шаг 3.2. Пересчитать центры кластеров алгоритмом Greedy, минимизируя сумму расстояний Хэмминга до векторов, принадлежащих кластеру.

Шаг 4. Сгруппировать повторяющиеся фрагменты блоков одного кластера.

Предлагается решение с поиском блоков данных по сходству с локально-чувствительным хэшированием (LSH), основанным на создании функций

хеширования, которое с большей вероятностью присвоит одинаковый хеш повторяющимся фрагментам блоков данных. Из предварительно обработанных блоков данных на Шаге 1 алгоритма LSH *K*-MEANS создадим массив признаков (хешей), с помощью алгоритма LSH SimHash для генерации 64-битного отпечатка (признака вектора данных) каждого блока данных, из которых сформируем фрагменты данных (вектор признаков данных). Затем с помощью алгоритма кластеризации *k*-средних с использованием расстояния Хэмминга (также известного как манхэттенское расстояние) сгруппируем повторяющиеся фрагменты данных. Алгоритм LSH *K*-MEANS отличается от алгоритма *k*-средних дополнительной операцией Шаг 1. Назовем фрагментом данных несколько отпечатков блоков данных.

4.3 Результаты вычислительного эксперимента по решению задач автоматической группировки повторяющихся фрагментов блоков данных

Для сравнения эффективности LSH SimHash для группировки схожих рядов блоков используем функцию хеширования md5, не зависящую от локальности.

В экспериментах использовалась известная программа сжатия данных gzip. Алгоритм 4.2 (LSH *K*-MEANS) был реализован в одном потоке на центральном процессоре. Тестовая система состояла из процессора Intel Core i3-3220 с 8 ГБ оперативной памяти.

Данные для сжатия состоят из рядов блоков, схожесть между которыми определяем с помощью расстояний Хемминга в Алгоритме 4.1. Используем Алгоритм 4.2 (LSH *K*-MEANS) для группировки схожих блоков.

В случае использования в gzip в данной работе удалось улучшить степень сжатия по сравнению с максимальным сжатием стандартного алгоритма gzip с параметрами -9 и -extreme.

Для тестирования производилось архивирование и сжатие начальных рядов блоков жесткого диска хранилища данных. Наивное сжатие с помощью XZ_OPT=9 tar Jcvf sdd.iso.tar.xz sdd.iso. Команда tar формирует архив с именем

sdd.iso.tar.xz, включающий файл sdd.iso. Опция «J» указывает, что архив создан в формате xz, а опция «c» указывает на создание архива.

Предварительная сортировка файлов выполнялась с помощью хеш-функции md5. Архив данных sdd.iso разбит на блоки данных по 1Мб - командой “split ../sdd.iso b 1M d”, каждый файл пронумерован по порядку, содержит фактические данные архива.

Во-первых, добавим все файлы в архив tar с помощью команды “tar cvf - sdd.tar *”, где каждый блок данных будет добавлен в архив последовательно. Повторяющиеся блоки данных алгоритм gzip ищет на расстоянии менее 32 КБ друг от друга. При уличении параметра тщательности сжатия до 9 xz ищет повторяющиеся блоки данных на расстоянии менее 64 МБ друг от друга.

Во-вторых, отсортируем все похожие блок данных по хэшу md5 и архивируем в одном файле. Результаты показаны в таблице 4.1. Для сравнения проведем сжатие 3960Мб блоков данных, отсортированных с помощью хеш-функции SimHash.

Результаты вычислительных экспериментов, проводимые для данных общим объемом 3960Мб, представлены в таблице 4.1. Для сравнения эффективности сжатия были применены следующие способы:

1. В первом способе проводилось архивирование без сжатия на блоки данных, каждый объемом 1Мб, и сжатие компрессором gzip.

2. Во втором методе проводилось архивирование без сжатия на блоки, каждый объемом 1Мб, с сортировкой блоков данных с помощью хеш-функции md5 для группировки повторяющихся фрагментов блоков данных и последующим сжатием компрессором gzip.

3. В третьем методе проводилось архивирование без сжатия на блоки, каждый объемом 1Мб, с группировкой Алгоритмом 4 LSH *K*-MEANS и последующим сжатием компрессором gzip.

Результаты приведены в Таблицах 4.1 – 4.7.

Таблица 4.1 – Сравнительные результаты сжатия VDI_3960 данных по 3960 блоков данных, объемом 1Мб

Способ сжатия	Время сжатия, секунд	Время сжатия, секунд/Мбайт	Размер архива, байт
1	418,920	0,106	363897476
2	437,144	0,110	353975608
3	461,062	0,116	321329584

Применение случайной сортировки с использованием хеш-функции md5 приводит к уменьшению размера архива относительно результатов стандартного компрессора gzip, а с SimHash к более значительному сжатию. Таким образом, применение *k*-средних совместно с LSH для сортировки блоков данных для лучшей сжимаемости архивов файлов 3960Мб является сравнительно более эффективным.

Для сравнения результатов сжатия новым алгоритмом LSH *K*-MEANS используем наборы данных из теста сжатия большого текста [201] состоящие из файла enwik8 размером 10^8 байт и файла enwik9 размером 10^9 байт XML текстового архива английской версии Википедии от 3 марта 2006г, распространяемые в виде архивов 36445475 и 322592222 байт соответственно.

Таблица 4.2 – Сравнительные результаты сжатия 10^8 байт enwik8 по 96 блоков данных, объемом 1Мб

Способ сжатия	Время сжатия, секунд	Время сжатия, секунд/Мбайт	Размер архива, байт
1	54,617	0,566	24865036
2	55,254	0,577	24892848
3	56,027	0,587	24892728

Таблица 4.3 – Сравнительные результаты сжатия 10^9 байт enwik9 с сортировкой 954 блоков данных, объемом 1Мб

Способ сжатия	Время сжатия, секунд	Время сжатия, секунд/Мбайт	Размер архива, байт
1	539,866	0,565	216228064
2	780,006	0,818	216263632
3	1440,045	1,510	215428988

Для сравнения результатов сжатия новым алгоритмом LSH *K*-MEANS используем наборы данных для приблизительного поиска ближайшего соседа [202] ANN_SIFT1B размером $132x10^9$ байт, распространяемые в виде архивов gzip размером 97941899519 байт (91.21Гб). Для сжатия использовали часть архива объемом $132x10^8$ байт.

Таблица 4.4 – Сравнительные результаты сжатия $132x10^8$ байт ANN_SIFT1B по 10^4 блоков данных, объемом $132x10^4$ байт

Способ сжатия	Время сжатия, минут, секунд	Время сжатия, секунд/Мбайт	Размер архива, байт
1	10541	0,837	9302739340
2	10595	0,842	9303610788
3	10700	0,850	9303687068

Таблица 4.5 – Сравнительные результаты сжатия VDI_1904, 10739515392 байт

Способ сжатия	Время сжатия, минут, секунд	Время сжатия, секунд/Мбайт	Размер архива, байт
1	83	0,008	1615084
2	109	0,011	1611536
3	261	0,025	1611588

Таблица 4.6 – Сравнительные результаты сжатия VDI_2004, 7702839296 байт

Способ сжатия	Время сжатия, минут, секунд	Время сжатия, секунд/Мбайт	Размер архива, байт
1	2071	0,282	2215866744
2	1701	0,232	1946039624
3	1653	0,225	1650728728

Таблица 4.7 – Сравнительные результаты сжатия VDI_2204, 5309988864 байт

Способ сжатия	Время сжатия, минут, секунд	Время сжатия, секунд/Мбайт	Размер архива, байт
1	1269	0,251	1498186792
2	1361	0,269	1671935704
3	1254	0,248	1400761320

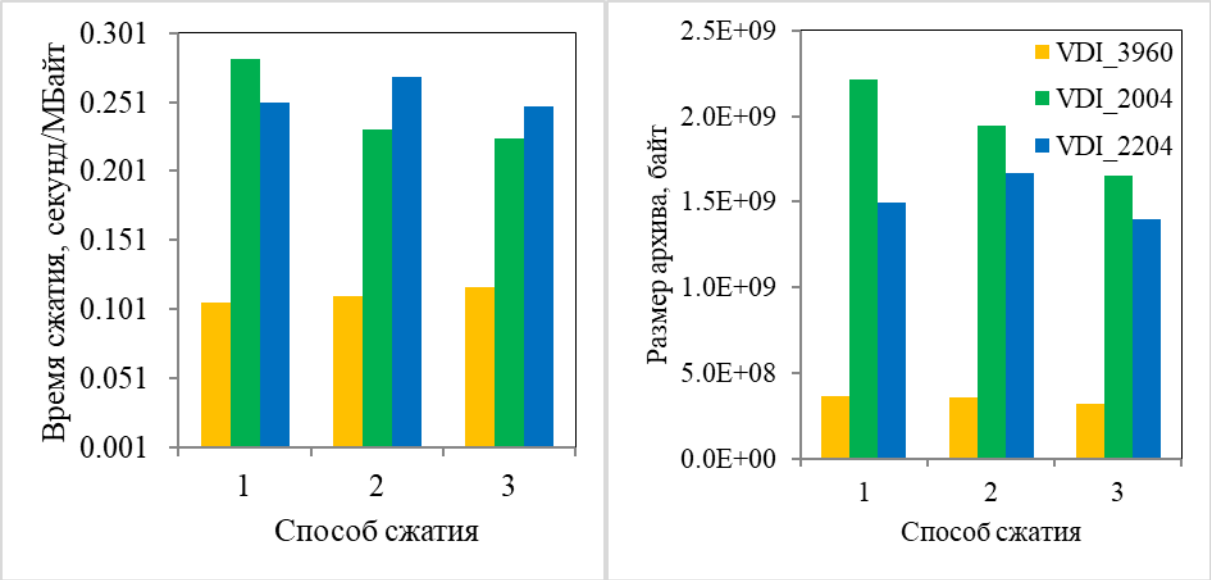

Рисунок 4.1 – Сравнительные результаты сжатия данных новым алгоритмом LSH *K*-MEANS

Для сравнения различных способов сжатия данных компрессором gzip используем файлы виртуальных операционных систем linux, файл VDI_1904 размером 10739515392 байт, файл VDI_2004 размером 7702839296 байт, файл VDI_2204 размером 5309988864 байт.

Применение *k*-средних совместно с LSH для сортировки блоков данных для лучшей сжимаемости файлов архивов виртуальных операционных систем linux на рисунке 4.1 является сравнительно более эффективным относительно результатов стандартного компрессора gzip.

Результаты Раздела 4

Применение сжатия данных для сохранения, передачи информации, резервного копирования, работы с электронными таблицами осуществляется регулярно [203] для предотвращения перегрузки средств хранения и каналов передачи данных [204] с помощью наиболее подходящих алгоритмов сжатия данных [205]. Комплексный подход к сжатию данных также, по мнению [206], позволяет добиться высокого коэффициента сжатия, в том числе с использованием алгоритмов поблочного сжатия [207].

Применение Алгоритма 4.2 LSH *K*-MEANS для сортировки блоков данных для лучшей сжимаемости является эффективным по сравнению с известным компрессором gzip, использующим методы на основе словаря совместно с кодированием Хаффмана. Результаты оценки эффективности показали, что использование LSH может повысить эффективность сжимаемости данных и снизить затраты на обработку данных. Новый алгоритм автоматической группировки блоков данных, основанный на алгоритме *k*-средних совместно с алгоритмом хеширования с учетом местоположения был успешно применен в составе системы управления дисковыми хранилищами в ООО «Центр вычислительных технологий», Приложение А.

Преимущество применения нового алгоритма сжатия основанного на архивации с помощью упрощенной жадной агломеративной процедура Greedy, основанной на алгоритме *k*-средних, совместно с LSH состоит в адаптации к базовому распределению данных, что повышает производительность сжатия.

5. АЛГОРИТМ ИНИЦИАЛИЗАЦИИ ЦЕНТРОВ КЛАСТЕРОВ ДЛЯ АЛГОРИТМОВ КЛАСТЕРИЗАЦИИ

В разделе 5 рассмотрены вопросы разработки новой процедуры инициализации центров кластеров для алгоритмов кластеризации, представляющей собой модификацию алгоритма инициализации *k*-means++. Полученная процедура инициализации применяется к большим данным. Учитывая меньшие вычислительные затраты, связанные с алгоритмом *k*-средних по сравнению с агломеративным алгоритмом, их применение в сочетании с модифицированным алгоритмом инициализации *k*-means++ к большим данным позволит уменьшить вычислительную сложность агломеративного алгоритма и улучшить его производительность. Результаты данного раздела были опубликованы в [208, 101].

Быстрые алгоритмы кластеризации больших и сверхбольших объемов данных востребованы в различных областях, например, векторных базах данных (см. Раздел 3), в системах дискового хранения данных (см. Раздел 4). Алгоритм *k*-средних остается самым популярным, однако он во многом зависит от метода инициализации. *K*-means++ – это хорошо установленный порядок выбора начальных центров кластеров (центроидов) для алгоритма *k*-средних. В этой работе мы представляем новый алгоритм, способный решить задачу инициализации *k*-means++ для получения хорошего начального решения. Известный алгоритм *k*-means++ хорошо работает с наборами данных из N векторов данных в M-мерном пространстве при выполнении одного прохода через входные данные за $O(k)$ итераций, и каждая итерация характеризуется сложностью $O(NMk)$, где k – количество центров (кластеры). Итак, общее время выполнения равно $O(NMk^2)$. Поскольку алгоритм *k*-means++ требует, чтобы k проходил через данные для инициализации, он плохо масштабируется до больших наборов данных. Мы предлагаем алгоритм Mini-Batch *K*-means++ с одним проходом по данным и общим ожидаемым временем выполнения $O(Mk(b + \log_2 B))$, где B – количество пакетов данных, а b – количество

векторов данных в одном пакете, $N \leq Bb$. Высокую сравнительную эффективность нового алгоритма Mini-Batch *K*-means++ для больших данных показал эксперимент. Минимальный объем ресурсов памяти для запуска алгоритма ожидается $B = N^{1/2}$ с неоптимальным временем выполнения $O(Mk(b + \log_2 B))$. Для оптимального сценария запуска нового алгоритма для векторов данных 2^w ожидаемые затраты времени равны $O(Mkw)$. В эксперименте показано, что качество решений не уступает классическим *k*-means++. Учитывая меньшие вычислительные затраты, связанные с алгоритмом *k*-средних по сравнению с агломеративным алгоритмом, их применение в сочетании с модифицированным алгоритмом инициализации *k*-means++ к большим данным позволит уменьшить вычислительную сложность агломеративного алгоритма и улучшить его производительность.

5.1 Обзор литературы об алгоритмах инициализации центров кластеров для алгоритмов кластеризации

В анализе данных кластеризация с использованием алгоритмического подхода используется для извлечения информации, обработки, классификации и сжатия данных [209]. Набор данных разделяют на кластеры, оптимизируя целевую функцию, основанную на расстояниях между объектами.

Эвристические методы расчета целевой функции используют приближенные решения задачи [210]. Среди эвристических алгоритмов наиболее часто используемым является алгоритм Ллойда [31], который часто называют алгоритмом *k*-средних. Алгоритм случайным образом выбирает *k* произвольных центров из *N* векторов данных. Центры *k* присваиваются ближайшим векторам данных. Среднее (среднее значение) векторов данных, присвоенных ближайшему центру, выбираются в качестве нового центра кластера. Заранее определенный критерий окончания алгоритма завершает эти два итерационных шага.

Алгоритм оптимизации целевой функции *k*-средних (суммы квадратов расстояний от векторов данных до ближайшего центра) часто сходится к

локальному минимуму. Одной из причин этого является случайный выбор произвольных центров при инициализации алгоритма. Адаптивные методы инициализации алгоритма *k*-средних исключают более медленную оптимизацию целевой функции *k*-средних и более высокую вероятность застрять в плохих локальных минимумах [211].

Недетерминированные методы инициализации с линейной временной сложностью относительно количества векторов данных используются для многократного запуска алгоритма инициализации и получения выходных данных с наименьшей суммой квадратов ошибок в кластере.

Методы инициализации центров на основе случайного выбора были впервые предложены одновременно с внедрением алгоритма *k*-средних. Инициализация вектора данных одним из центров равномерно и случайным образом реализована в методе Форги [212], который расширяется в методе Спата [213] путем циклического присвоения вектора данных центрам. Использование первых векторов из набора данных в качестве начальных центров и использование случайного выбора векторов данных в качестве начальных центров было предложено Маккуином [87]. Метод Болла и Холла [214] использует в качестве первого центра центр всех векторов данных или первый вектор данных, а остальные центры выбираются на заданном расстоянии от него. Метод Тоу [215] аналогичен. Метод максимина использует вектор данных как центр, имеющий наибольшее минимальное расстояние до ранее выбранных центров, с произвольно выбранным первым центром [216, 217]. Метод, предложенный Аль–Даудом, использует равномерно разделенное пространство данных на несколько непересекающихся гиперкубов, центры которых выбираются равномерно случайным образом [218]. Используя аналогичный метод, который итеративно разбивает пространство данных на два гиперкуба за итерацию, выбор центров из наиболее плотных гиперкубов представлен в методе Пиццути [219]. В методе Маккуина используется случайное разбиение набора данных на подмножества для отдельной кластеризации подмножеств с получением промежуточных центров. Полученные промежуточные центры объединяются в надмножества для выбора

кластеризации, дающей наименьшую среднеквадратическую ошибку, центры кластеров которой принимаются в качестве конечных центров в методе Брэдли и Файяда [220]. В методе Су и Ди используется разделяющий иерархический подход, основанный на анализе главных компонент [221] с гиперплоским разбиением набора векторов данных через центр кластера в направлении, ортогональном основному вектору ковариационной матрицы кластера [222], а центрами этих кластеров назначаются центры. При использовании двухфазного пирамидального подхода центры выбираются из набора векторов данных и уточняются с использованием процедуры *k*-средних в методе Лу [223]. Вектор данных с наименьшим косинусным расстоянием от вектора случайных данных до независимых компонентов в качестве центра используется в методе Оноды [224].

Детерминированные методы инициализации начальных центров кластеров с высокой временной сложностью требуют меньше итераций для удовлетворения критерия сходимости алгоритма *k*-средних и компенсации потерь времени на этапе инициализации [222, 49, 225].

Улучшенные методы инициализации рассматриваются в первую очередь для работы с большими данными. В методе Хартигана используется усовершенствованный метод инициализации Маккуина с начальной сортировкой векторов данных по их расстоянию до случайного вектора данных в качестве центра [226]. Сортировка векторов данных по признакам с наибольшей дисперсией, группировка и выделение векторов данных в качестве центров, соответствующих медианам этих групп, используется в методе Аль–Дауда [227, 228]. Оценка плотности с помощью kd-дерева и модифицированного метода максимина для выбора центров из густонаселенных сегментов листьев используется в методе Редмонда и Хенегана [49]. В работе Редмонда и Хенегана используется локальный фактор выброса [229], определяющий векторы выбросов как центры [228, 229]. В методе Астрахана используются двухпороговые значения расстояний для сортировки векторов данных по плотности и иерархическая кластеризация для группировки центров векторов данных [231]. Выходные данные алгоритма иерархической кластеризации для инициализации центров

векторов данных используются в методе Ланса–Вильямса [232]. В качестве центра используются векторы данных, которые в наименьшей степени уменьшают среднеквадратическую ошибку в методе Кауфмана и Русси [227]. В качестве первых центров в методе Цао используются плотности в рамках модели пересеченной местности, основанной на соседстве с сортировкой векторов данных в порядке убывания связности [233]. В методе Линде [234] используется двоичное разбиение вектора случайных данных на два центра с учетом фиксированного вектора возмущения и уточненного использования *k*-средних. Бинарное расщепление с направленным поиском с учетом конкретного вектора возмущений и с использованием анализа главных компонент используется в методе Хуанга и Харриса [235]. В методе Ликаса используется метод глобальных *k*-средних с поочередным рассмотрением следующего вектора данных в качестве центра [236]. В методе Бабу и Мурти [237] используются метаэвристические алгоритмы моделирования отжига и генетические алгоритмы, использующие *k*-средние для оценки центров на каждой итерации.

Метод инициализации *k*-средних Артура и Васильвицкого [100], названный *k*-means++, с использованием методов Маккуина и максимина случайным образом равномерно выбирает первый центр из набора данных, а следующий центр из набора векторов равновероятно сумме квадратов расстояний от центров до векторов данных. *K*-means++ позволяет избежать зацикливания на локальных минимумах за счет выбора начальных центров на основе распределения данных и дает более точные результаты кластеризации для алгоритма *k*-средних [100].

В работе предлагается новая процедура, которая может решить проблему *k*-means++ с линейной временной сложностью относительно количества векторов данных.

Учитывая N векторов данных, алгоритм *k*-means++ выполняется за $O(k)$ итераций. Итерации занимают время $O(NMk)$. Таким образом, общее время выполнения равно $O(NMk^2)$. В 2022 году Лян и др. [238] предложили новый алгоритм FastKmeans++, в котором общее время выполнения составляет всего $O(NM + Nk^2)$.

Поскольку *k*-means++ требует *k* проходов по данным для инициализации, он плохо масштабируется для больших наборов данных. Бахмани и др. [239] предложили масштабируемый вариант *k*-means++, названный *k*-means||, который обеспечивает теже теоретические гарантии, но обладает высокой масштабируемостью. Время работы *k*-means|| составляет два раунда за $O(k^{\frac{2}{3}} N^{\frac{1}{2}})$.

Экспериментальные исследования *k*-means|| алгоритма были выполнены для $k \in \{20, 50, 100\}$ по сравнению с алгоритмом *k*-means++ и показали, что результаты разбиения будут идентичны результатам *k*-means++ с точки зрения стоимости целевой функции. Разделение наборов данных выполняется гораздо быстрее при большом количестве центров $k \in \{500, 1000\}$ с помощью *k*-means|| алгоритм в сравнительных экспериментах со случайным разделением. Набор данных KDDCup1999 [240] состоит из 4,8 миллионов векторов данных в 42 измерениях. Это исследование показывает, что для больших наборов векторов данных (около миллиарда) необходимы другие алгоритмы, в которых вычислительная сложность снижается относительно количества векторов данных.

Алгоритм *k*-means++, в сравнении с равномерным выбором векторов данных в качестве начальных центров кластеров, позволяет более равномерно распределить начальные центры, существенно улучшая начальное и, как следствие, конечное решение алгоритмов кластеризации. С ростом объема данных и числа выделяемых кластеров $O(NMk^2)$ вычислительные затраты алгоритма инициализации *k*-means++ (предложенного Артуром и Васильвицким) начинают многократно превышать затраты основных итеративных алгоритмов, таких как алгоритм Ллойда или PAM. Для больших наборов векторов данных (порядка миллиарда и более) нужны алгоритмы, в которых вычислительная сложность уменьшается относительно числа векторов данных.

5.2 Новый алгоритм инициализации центров кластеров для алгоритмов кластеризации использующий вспомогательную структуру данных

Рассмотрим постановку задачи *k*-means++ и теоретическое решение нового алгоритма инициализации центров кластеров. Пусть имеется набор векторов данных a_i, $i \in \{1, \dots, N\}$ в *M*-мерном пространстве измерений. Целью алгоритма *k*-means++ является нахождение центров x_j, $j \in \{1, \dots, k\}$ кластеров. Первый центр выбирается случайно из набора векторов данных. Алгоритм *k*-means++ для нахождения центра x_{j+1} вычисляет сумму квадратов расстояний от центра x_j до вектора данных a_i по формуле

$$\sum_{i=1}^{N} \left(d(x_j, a_i)\right)^2 = \sum_{i=1}^{N} (x_j - a_i)^2. \quad (5.1)$$

Раскроем скобки:

$$N x_j^2 - 2 \sum_{i=1}^{N} (x_j a_i) + \sum_{i=1}^{N} a_i^2. \quad (5.2)$$

Алгоритм *k*-means++ для наборов данных из *N* векторов данных в *M*-мерном пространстве при однократном проходе по входным данным выполняет $O(k)$ итераций; каждая итерация характеризуется сложностью $O(NMk)$, где *k* – число центров. Таким образом, общее время выполнения равно $O(NMk^2)$. Поскольку для инициализации *k*-means++ требуется *k* проходов через данные, он плохо масштабируется для больших наборов данных.

Предлагаем в новой процедуре инициализации центров кластеров определить сумму квадратов расстояний от *k* центров x_j до N векторов данных a_i по формуле (5.12).

Определяем термины для вычисления суммы квадратов расстояний в Алгоритме 5.2, с целью поиска в упорядоченном массиве заранее вычисленных слагаемых для определения ближайшего вектора данных на расстоянии, не превышающем заданное значение R.

Определяем сумму A_i от первого до i-го вектора данных $a_{i`}$, где $i` \in \{1,\dots,i\}$ для координаты $M` \in M$ отдельного пространства:

$$A_{M`i} = \sum_{i`=1}^{i} a_{M`i`}. \tag{5.3}$$

Определим сумму N векторов данных для координаты $M`$:

$$A_{M`} = \sum_{i=1}^{N} a_{M`i}. \tag{5.4}$$

Сумма от первого центра до j, где $j' \in \{1,\dots,j\}$, для координаты $M`$ есть

$$X_{M`j} = \sum_{j`=1}^{j} x_{M`j`}\ . \tag{5.5}$$

Определим сумму k центров координаты M' как

$$X_{M`} = \sum_{j=1}^{k} x_{M`j}. \tag{5.6}$$

Сумма квадратов векторных координат в M-мерном пространстве вектора данных a_i равна

$$A_i^2 = \sum_{M`=1}^{M} A_{M`i}^2. \tag{5.7}$$

Определим сумму квадратов координат вектора в M-мерном пространстве центра x_j:

$$X_j^2 = \sum_{M`=1}^{M} X_{M`j}^2. \tag{5.8}$$

Определим сумму сумм квадратов векторных координат в M-мерном пространстве для N векторов данных a_i , $i \in \{1,\dots,N\}$:

$$A^2 = \sum_{i=1}^{N} A_i^2. \tag{5.9}$$

Аналогично определим сумму сумм квадратов координат вектора в M-мерном пространстве для k центров

$$X^2 = \sum_{j=1}^{k} X_j^2. \qquad (5.10)$$

Таким образом, определяем, что сумма квадратов расстояний от центра x_j до N векторов данных a_i равна

$$\sum_{i=1}^{N} \left(d(x_j, a_i)\right)^2 = N\, X_j^2 - 2 \sum_{M`=1}^{M} \left(A_{M`} x_{M`j}\right) + A^2, \qquad (5.11)$$

где X^2 – сумма сумм квадратов координат вектора в M-мерном пространстве для k центров x_j; A^2 – сумма сумм квадратов векторных координат в M-мерном пространстве для N векторов данных a_i. За счет сокращения количества операций суммирования по i и j для вычисления суммы квадратов расстояний снижается сложность вычислений. Процедура Mini-Batch K-means++ последовательно вычисляет промежуточные значения суммы квадратов расстояний от центров x_j до векторных координат a_i в пакетах B.

Аналогично определяем сумму квадратов расстояний от k центров x_j до N векторов данных a_i как

$$\sum_{j=1}^{k} \sum_{i=1}^{N} \left(d(x_j, a_i)\right)^2 = N\, X^2 - 2 \sum_{M`=1}^{M} \left(A_{M`} X_{M`}\right) + k\, A^2. \qquad (5.12)$$

Сначала вычислим слагаемые (5.3) и (5.7) в Алгоритме 5.1.

Вспомогательная структура данных представлена упорядоченным массивом предварительно вычисленных слагаемых $A_{M`B`}$, $A_{B`}^2$ и A^2 вычисляется в Алгоритме 5.1.

Алгоритм 5.1 Вспомогательная структура данных

Шаг 1. Разделить $\boldsymbol{N}$ векторов данных на количество пакетов $\boldsymbol{B}$, $\boldsymbol{B`} \in \{\boldsymbol{1} \ldots \boldsymbol{B}\}$ с $\boldsymbol{b}$ векторами данных в каждом пакете, $\boldsymbol{N} \leq \boldsymbol{Bb}$. Обозначаем количество векторов данных от первого во всем наборе данных до последнего в пакете $\boldsymbol{B`}$ как $\boldsymbol{N_{B`}}$, где $\boldsymbol{i`} \in \{\boldsymbol{1} \ldots \boldsymbol{N_{B`}}\}$.

Шаг 2. Для каждой пакета $\boldsymbol{B`} \in \{\boldsymbol{1} \ldots \boldsymbol{B}\}$ для каждой координаты

$\boldsymbol{M`} = \{\boldsymbol{1} \ldots \boldsymbol{M}\}$: вычислить (5.3):

$$A_{M`B`} = \sum_{i`=1}^{N_{B`}} a_{Mi`}. \tag{5.13}$$

Шаг 3. Для каждого пакета $\boldsymbol{B`} \in \{\boldsymbol{1} \ldots \boldsymbol{B}\}$: вычислить (7):

$$A_{B`}^2 = \sum_{M`=1}^{M} A_{M`B`}^2. \tag{5.14}$$

Шаг 4. Рассчитать сумму сумм квадратов расстояний пакета $\boldsymbol{B}$:

$$A^2 = \sum_{B`=1}^{B} A_{B`}^2 . \tag{5.15}$$

Таким образом, согласно (5.11) алгоритм Mini-Batch *K*-means++ для центра x_j вычисляет сумму квадратов расстояний до $N_{B`}$ векторов данных a_i:

$$\sum_{i=1}^{N_{B`}} \left(d(x_j, a_i)\right)^2 = N_{B`}\, X_j^2 - 2 \sum_{M`=1}^{M} \left(A_{M`B`} x_{M`j}\right) + A_{B`}^2. \tag{5.16}$$

Аналогично для k центров x_j вычисляем сумму квадратов расстояний до $N_{B`}$ векторов данных a_i:

$$\sum_{j=1}^{k} \sum_{i=1}^{N_{B`}} \left(d(x_j, a_i)\right)^2 = N_{B`}\, X_j^2 - 2 \sum_{M`=1}^{M} \left(A_{M`B`}\, X_{M`}\right) + k\, A^2. \tag{5.17}$$

За счет сокращения количества операций суммирования по i и j для вычисления суммы квадратов расстояний снижается сложность вычислений. Для алгоритма Mini-Batch *K*-means++ последовательно вычисляем промежуточные значения суммы квадратов расстояний от центров x_j до векторов данных a_i для пакетов $B`$.

Процедура Mini-Batch *K*-means++ выполняет k итераций, $j \in \{1, \ldots, k`, \ldots, k\}$.

На первом этапе находим сумму расстояний от N векторов данных до первого центра x_1 для выбора следующего центра x_j .

Поиском в упорядоченном массиве предварительно вычисленных слагаемых A_{MB} и A_B^2 мы определяем пакет $B`$, в котором находится искомый центр

x_j. Далее мы итеративно уточняем желаемый центр x_j, используя векторы данных a_i выбранного пакета $B`$.

На второй и последующих итерациях суммируются суммы расстояний от N векторов данных a_i до найденных центров x_j.

Алгоритм 5.2 Mini-Batch *K*-means++:

Шаг 1. Выбрать равномерно случайным образом один центр $\boldsymbol{x_1}$ среди векторов данных $\boldsymbol{a_i}$. Пусть $\boldsymbol{B}$ количество пакетов, на которые разделен набор данных размера $\boldsymbol{N}$, $\boldsymbol{b}$ количество векторов данных в одном пакете, $\boldsymbol{N \leq Bb}$ и $\boldsymbol{N_{B`}}$ количество векторов данных из первого из всего набора данных до последнего в пакете $\boldsymbol{B`}$ включительно, где $\boldsymbol{i` \in \{1 \dots N_{B`}\}}$. Пусть слагаемые $A_{MB`}$, $A^2_{B`}$ предварительно рассчитаны для всех пакетов $\boldsymbol{B}$.

Шаг 2. Вычислить слагаемые X_M и X^2 для центров $\boldsymbol{x_j}$. Вычисляем сумму квадратов расстояний от $\boldsymbol{k}$ центров до векторов данных $\boldsymbol{a_i}$, используя (5.3), чтобы найти второй центр, и (5.4) следующий (случайно выбранный) *j*-й центр $\boldsymbol{x_j}$.

Шаг 3. Выбрать новый центр с вероятностью, пропорциональной сумме квадратов расстояний от $\boldsymbol{k`}$ выбранных центров $\boldsymbol{x_j}$ до N векторов данных $\boldsymbol{a_i}$. Для этого выбираем случайное расстояние

$$\boldsymbol{R} \in \left\{ \boldsymbol{0}, \dots, \sum_{\boldsymbol{j=1}}^{\boldsymbol{k`}} \sum_{\boldsymbol{i=1}}^{\boldsymbol{N}} \left(\boldsymbol{d}(\boldsymbol{x_j}, \boldsymbol{a_i}) \right)^2 \right\},$$

где $\boldsymbol{k`}$ – количество найденных центров. В алгоритме 5.3 находим пакет $\boldsymbol{B`}$, который содержит самый дальний вектор данных на расстоянии не больше $\boldsymbol{R}$. Шаги 2 и 3 повторяются пока не выбраны все k центров.

После выполнения Алгоритма 5.3 мы получаем центры, которые используются для получения исходного результата кластеризации и сравниваем с результатом начальной кластеризации, полученным с использованием классического *k*-means++. Качество решения не уступает классическому *k*-means++, поскольку последовательность шагов в новой процедуре Mini-Batch *K*-means++ аналогична классическому *k*-means++ за исключением того, что новая

процедура увеличивает скорость вычисления следующего центра (из всех векторов данных) так, чтобы вероятность выбора вектора была пропорциональна вычисленному квадрату расстояния до всех выбранных центров.

Алгоритм 5.3 Поиск в упорядоченном массиве предварительно вычисленных слагаемых A_{MB}, A_B^2 для определения ближайшего вектора данных на расстоянии не более $\boldsymbol{R}$.

Шаг 1. Установить максимальный и минимальный номер пакета $\boldsymbol{B`}$, $\boldsymbol{i_{min} = 0}$ и $\boldsymbol{i_{max} = B}$.

Шаг 2. Повторять до тех пор, пока $\boldsymbol{|i_{min} - i_{max}| > 1}$:

Шаг 3. Определить текущий номер пакета как

$$t = \frac{(\boldsymbol{i_{min} + i_{max}})}{\boldsymbol{2}}.$$

Шаг 4. Вычислить сумму квадратов расстояний до $\boldsymbol{k`}$ найденных центров от $\boldsymbol{n_t}$ векторов данных:

$$\sum_{j=1}^{k`}\sum_{i=1}^{n_t}\left(\boldsymbol{d(x_j, a_i)}\right)^2 = \boldsymbol{n_t X^2 - 2}\sum_{M`=1}^{M}(\boldsymbol{A_{M`t}\, X_{M`}}) + \boldsymbol{k`\, A^2}.$$

Шаг 5. Если

$$\boldsymbol{R} > \sum_{j=1}^{k`}\sum_{i=1}^{n_t}\left(\boldsymbol{d(x_j, a_i)}\right)^2.$$

Затем установить минимальный номер пакета $\boldsymbol{i_{min} = t}$, а сумма квадратов расстояний от $\boldsymbol{k`}$ центров до их векторов данных больше, чем

$$\boldsymbol{S_{min}} = \sum_{j=1}^{k`}\sum_{i=1}^{n_t}\left(\boldsymbol{d(x_j, a_i)}\right)^2.$$

В противном случае установить максимальный номер пакета $\boldsymbol{i_{max} = t}$.

Шаг 6. Конец цикла (Шаг 2).

Шаг 7. Повторять до тех пор, пока $\boldsymbol{S_{min} < R}$, где $\boldsymbol{S}$ — сумма квадратов расстояний до вектора данных $\boldsymbol{b`} \in \{\boldsymbol{1 \ldots b}\}$ из пакета $\boldsymbol{i_{min}}$:

Шаг 8. Рассчитать

$$S_{min} = S_{min} + \sum_{j=1}^{k} \left(d\left(x_j, a_{(t+b`)}\right)\right)^2.$$

Шаг 9. Конец цикла (шаг 7). Выбираем следующий центр набора данных x_j из векторов данных a_i с индексом $i = t + b`$.

Шаг 10. Вычислить слагаемые $X_{M`j}$, X_j^2, X^2 для набора данных x_j.

Алгоритм 5.3 может привести к увеличению вычислительной сложности для многомерных наборов данных, как показано на рисунках 5.1 и 5.2 (см. графики 4 и 6). Вычисление слагаемых для центра x_j происходит во время выполнения алгоритма 5.2 (Mini-Batch *K*-means++).

5.3 Вычислительные эксперименты с новым алгоритмом инициализации

Для наших экспериментов мы использовали известные реализации алгоритмов *k*-means++ [100] в сравнении с новым алгоритмом Mini-Batch *K*-means++. Все алгоритмы были реализованы в одном потоке на центральном процессоре. Тестовая система состояла из процессора Intel Corei9-12900F, 32 ГБ оперативной памяти. Для всех наборов данных было предпринято 30 запусков алгоритма, с расстояниями Евклида и $k = 100$ центров инициализации. Результаты представлены в таблицах 5.1–5.4. Для целевой функции (суммы квадратов расстояний) указаны минимальное, максимальное, среднее значения и стандартное отклонение. Значение целевой функции указывается в числителе, а также в знаменателе указывается время выполнения алгоритма в секундах.

Для наших экспериментов мы использовали самые большие известные наборы данных, чтобы оценить скорость выполнения нового алгоритма Mini-Batch *K*-means++ и экспериментально обосновать снижение вычислительной сложности, а также оценить статистическую значимость результатов:

1. ANN SIFT1B, ANN GIFT1M и ANN SIFT1M [241]: наборы данных для поиска приблизительных ближайших соседей, в наборе данных ANN SIFT1B

имеется 10^9 базовых векторов размерности 128. В наборе данных ANN GIFT1M имеется 10^6 базовых векторов размерности. 960, в наборе данных ANN SIFT1M имеется 10^6 базовых векторов размерности 128. Набор данных SIFT1M использовался в наших исследованиях для разработки нового алгоритма кластеризации, основанного на жадной агломеративной процедуре для построения индекса для векторной базы данных.

2. BIRCH3 [242]: набор данных из 10^6 векторов размерности 2.

3. IHEPC [243]: более чем 2 миллионов векторов размерности 7.

4. SUSY [243]: набор данных 5×10^6 векторов размерности 18.

Сравнительный анализ эффективности алгоритма *k*-means++ с новой процедурой Mini-Batch *K*-means++ с использованием *t*-критерия и *U*–критерия Манна-Уитни для большинства наборов данных демонстрирует (в таблицах 5.1 – 5.4) такую же эффективность (разница в результатах статистически незначима). Результаты сравнения времени запуска алгоритма *k*-means++ (KM++), Mini-Batch *K*-means++ (MBKM++) и алгоритма FastKmeans++ с разными наборами данных представлены на рисунке 2 для 100 центров инициализации. Исследуемые алгоритмы используют один проход по набору данных и используют одинаковое количество итераций k для расчета центров. При этом затрачиваемое время значительно меньше с новым алгоритмом Mini-Batch *K*-means++. В таблицах 5.1 – 5.4 приведены *p*-значения и статистическая значимость разницы при p=0,05 (U-тест Вилкоксона).

Таблица 5.1 – Экспериментальные результаты для набора данных ANN SIFT1M. N=1000000 векторов данных, k=2 кластера, 30 запусков

Алгоритм	Значение целевой функции / Время, секунд.					
	минимальное	максимальное	медианное	среднее	ст. отклонение	*p*-значение, *U*- тест
k-means++	1,616E+11	2,886E+11	2,235E+11	2,193E+11	0,355E+11	p=0,82588
	1,247951 с.	1,528814 с.	1,426723 с.	1,40777 с.	0,0875 с.	незначимо
Mini-Batch *K*-means++	1,705E+11	2,850E+11	2,291E+11	2,213E+11	0,372E+11	
	0,000020 с.	0,000026 с.	0,000023 с.	0,000023 с.	0,000002 с.	

Таблица 5.2 – Результаты экспериментов для набора данных BIRCH3. *N*=100000 векторов данных, *k*=100 кластеров, 30 запусков.

Алгоритм	Значение целевой функции / Время, секунд.					
	минимальное	максимальное	медианное	среднее	ст. отклонение	*p*-значение, *U*- тест
k-means++	1,072E+14	2,3814E+14	1,4909E+14	1,4643E+14	3,6546E+13	p=0,41794
	1,1088 с.	1,22598 с.	1,15584 с.	1,15609 с.	0,02734 с.	незначимо
Mini-Batch	1,125E+14	2,0838E+14	1,6485E+14	1,5817E+14	2,9829E+13	
K-means++	0,00055 с.	0,00345 с.	0,00056 с.	0,00058 с.	0,000526 с.	

Таблица 5.3 – Экспериментальные результаты для набора данных IHEPC. *N*=2075259 векторов данных, k=100 кластеров, 30 запусков.

Алгоритм	Значение целевой функции / Время, секунд.					
	минимальное	максимальное	медианное	среднее	ст. отклонение	*p*-значение, *U*- тест
k-means++	8187	23435	11629	11526	31504	p<0.00001.
	92 с.	101 с.	96 с.	96 с.	2,1 с.	значимо
Mini-Batch	10232	41628	17518	19090	10830	
K-means++	0,00062 с.	0,00237 с.	0,00063 с.	0,0006 с.	0,00032 с.	

Таблица 5.4 – Экспериментальные результаты для набора данных SUSY. *N*=5000000 векторов данных, k=25 кластеров, 30 запусков.

Алгоритм	Значение целевой функции / Время, секунд.					
	минимальное	максимальное	медианное	среднее	ст. отклонение	*p*-значение, *U*- тест
k-means++	1345876	1537414	1417747	1419697	43747	p=0.19706,
	51,5 с.	59,8 с.	56,4 с.	56,4 с.	2,3 с.	незначимо
Mini-Batch	1301506	1503959	1399338	1402135	54719	
K-means++	0,000182 с.	0,000199 с.	0,000185 с.	0,000185 с.	0,0032 с.	

Преимущество нового алгоритма сохраняется для всех наборов данных; для набора данных IHEPC различия в целевой функции статистически значимы.

Для наборов данных ANN SIFT1M, BIRCH3, SUSY новый алгоритм Mini-Batch *K*-means++ демонстрирует одинаковую эффективность (разница в

результатах статистически незначима). Кроме того, новый алгоритм демонстрирует лучшую стабильность результатов (меньшее стандартное отклонение целевой функции).

Расчет вспомогательной структуры данных в Алгоритме 5.1 для набора данных ANN SIFT1B (10^9 векторов данных) проводился с количеством пакетов $B = 10^7$ и векторов данных в пакете $b = 10^2$, составляющим 258 секунд. Для набора данных ANN GIFT1M (10^6 векторов данных) обработка проводилась с количеством пакетов $B = 10^5$ и $b = 10$ векторов данных на пакет. Для набора данных BIRCH3 (10^5 векторов данных) обработка данных проводилась с количеством пакетов $B = 10^4$ и $b = 10$ векторов данных на пакет. Как видно из таблиц 5.1-5.4, время работы сокращается на несколько десятичных порядков, а качество результата, оцениваемое по достигнутому значению целевой функции, в большинстве случаев статистически неотличимо.

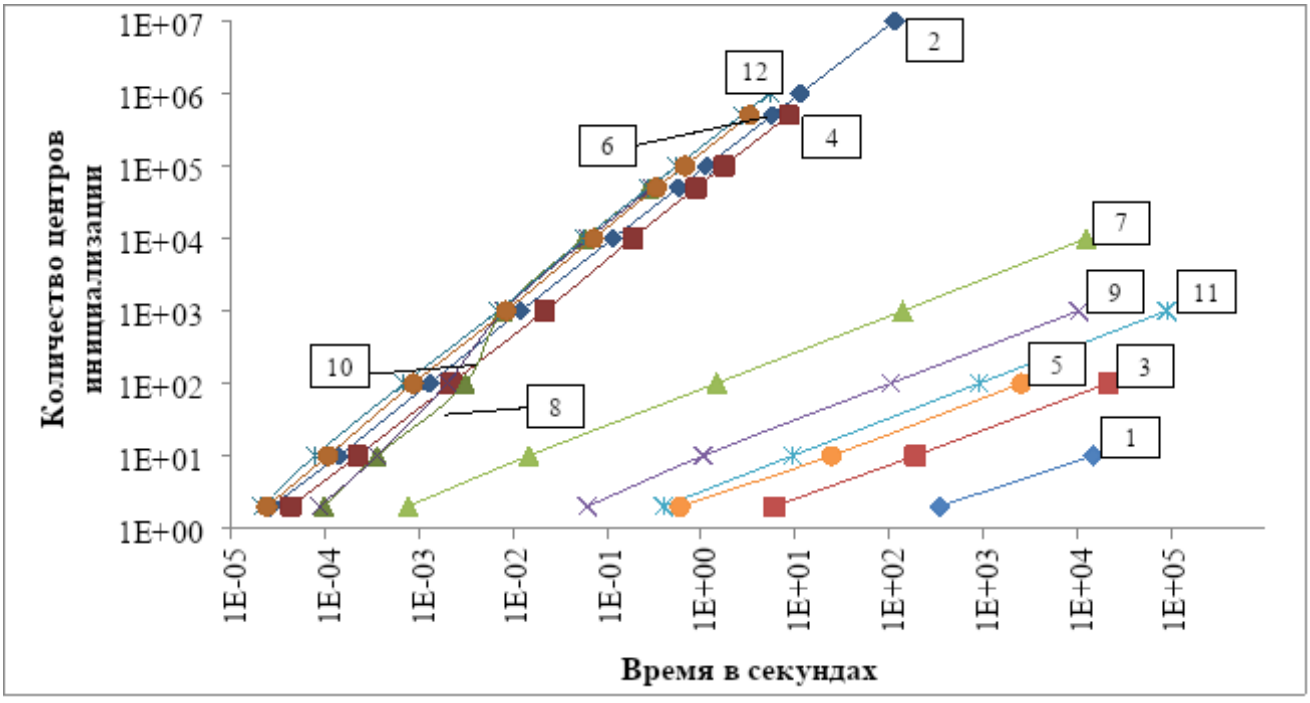

Рисунок 5.1 – Сравнение времени работы алгоритма *k*-means++(KM++) и Mini-Batch *K*-means++(MBKM++), наборы данных: 1 – ANN_SIFT1B(KM++), 2 – ANN_SIFT1B(MBKM++), 3 – ANN_ GIFT1M (KM++), 4 – ANN_GIFT1M(MBKM++), 5 – ANN_SIFT1M(KM++), 6 – ANN_SIFT1M(MBKM++), 7 – BIRCH3(KM++), 8 – BIRCH3(MBKM++), 9 – IHEPC (KM++), 10 – IHEPC (MBKM++), 11 – SUSY(KM++), 12 – SUSY(MBKM++)

Время выполнения алгоритма *k*-means++ линейно зависит от количества векторов данных *N*, необходимых для инициализации фиксированного числа $(k = 100)$ центров; вычислительная сложность алгоритма равна $O(N)$. Учитывая набор из *N* векторов данных в *M* измерениях, алгоритм *k*-means++ выполняется за $O(k)$ итераций, и каждая итерация занимает время $O(NMk)$. Таким образом, общее время выполнения равно $O(NMk^2)$.

Результаты экспериментальных расчетов в таблице 5.5 показывают, что алгоритм Mini-Batch *K*-means++ в отличие от алгоритма *k*-means++ с увеличением числа векторов *N* набора данных при инициализации фиксированного числа $(k = 100)$ центров тратит значительно меньше времени, зависимость времени выполнения от объема данных можно выразить как $O(log\ N)$. На рисунках 5.1 и 5.2 количество центров выбрано с логарифмическим шагом, чтобы показать небольшое увеличение вычислительной сложности с увеличением количества центров инициализации. Влияние количества векторов данных N для фиксированного числа кластеров сводится к $O(log\ N)$. Это означает, что в оптимальном сценарии запуска нового алгоритма для векторов данных 2^w минимальные затраты времени должны быть равны $O(Mkw)$, поскольку общее время работы алгоритма Mini-Batch *K*-means++ составляет $O(Mk(\ b\ +\ logB))$.

Алгоритм FastKmeans++, предложенный Ляном в [238], занимает всего $O(NM + Nk^2)$ единиц времени по общей сложности и линейно зависит от количества векторов данных N. Сравнение вычислительной сложности алгоритма Mini-Batch *K*-means++, алгоритма *k*-means++ и алгоритма FastKmeans++ приведено в таблице 5.5. Расчеты проводились для набора данных ANN SIFT1B с шагом одного десятичного порядка от количества векторов данных от 10^3 до 10^9. Для наборов данных ANN GIFT1M, ANN SIFT1M, BIRCH3, IHEPC, SUSY расчеты проводились с шагом одного десятичного порядка от числа векторов данных от 10^3 до 10^6, ячейки таблицы от 10^7 до 10^9 для них оставались пустыми.

Таблицы 5.1-5.4 приведены для сравнения результатов (значений целевой функции) нового алгоритма с современным алгоритмом *k*-means++ и показывают,

что разница в полученных результатах статистически незначима. Другие алгоритмы, такие как Fast *K*-means++, будучи более быстрыми, не дают результатов лучше, чем *k*-means++. Таким образом, наш новый Алгоритм 5.2, будучи быстрее *k*-Means++ и FastKMeans++, дает почти те же результаты, что и *k*-Means++ (разница незначительна).

Таблица 5.5 – Результаты эксперимента (затраты времени): Mini-Batch *K*-means++, стандартные *k*-means++ и FastKmeans++ (затраты времени, сек/потребление памяти, Кб)

Набор данных	Количество векторов данных (*N* первых векторов данных в наборе данных)						
	1.E+03	1.E+04	1.E+05	1.E+06	1.E+07	1.E+08	1.E+09
Mini-Batch *K*-means++							
SIFT1B	0,00095	0,00101	0,00108	0,00113	0,0012	0,0013	0,0015
	6212	6204	6712	11248	56600	51017	504528
GIFT1M	0,00261	0,00176	0,00377	0,00401			
	6956	10324	44108	381844			
SIFT1M	0,00057	0,00058	0,00071	0,00077			
	6204	6712	11248	56600			
BIRCH3	0,00262	0,00276	0,00317	0,00241			
	6072	6072	6228	7248			
HOUSE2	0,00270	0,00275	0,00285	0,00188			
	6072	6072	6480	9200			
SUSY	0,003273	0,001623	0,000506	0,00059			
	6072	6220	6904	13496			
K-means++							
SIFT1B	2,29595	22,2425	233,789				
	6264	7460	18800				
GIFT1M	19,8423	201,062					
	10200	43984					
BIRCH3	0,01564	0,12443	1,14740	10,8869			
	6068	6256	7240	17788			
HOUSE2	0,04807	0,45637	4,40178	43,7735			
	6068	6384	9196	37320			
SUSY	0,17073	1,67682	16,3763	171,5998			
	6212	6812	13492	80288			
SIFT1M	2,354202	22,81933	231,3778				
	6572	11108	56460				
Fast*K*-means++							
BIRCH3	0,01043	0,04321	0,34610	3,34359			
	6068	6264	7240	17788			

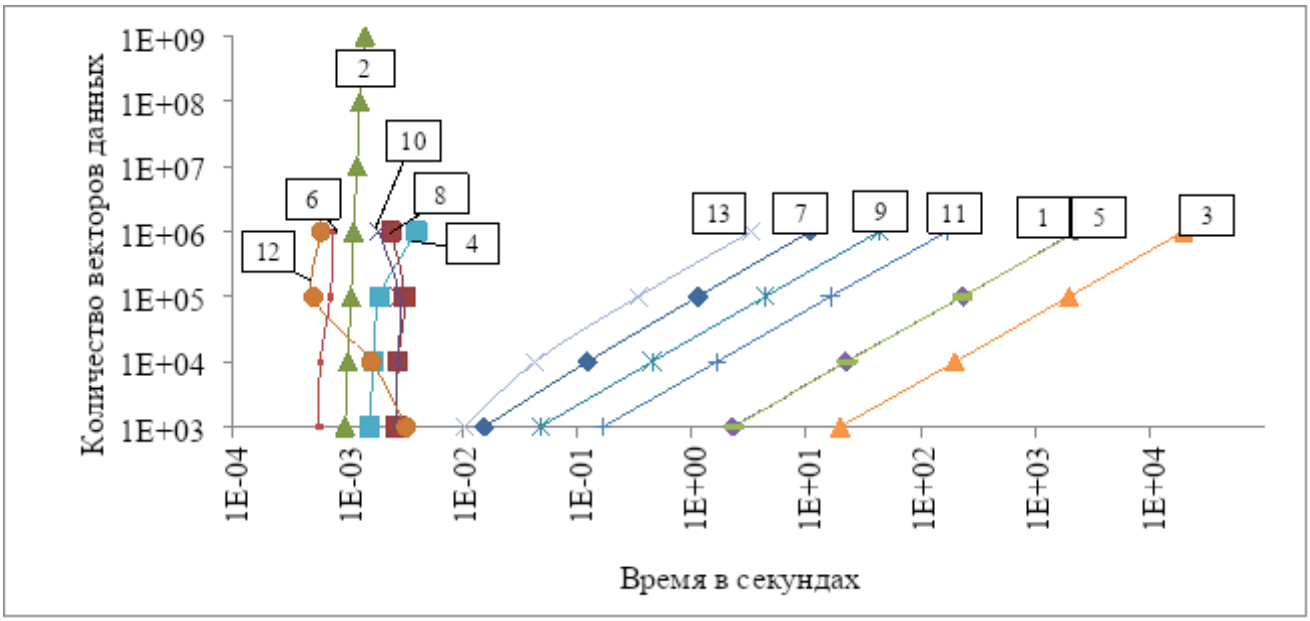

Рисунок 5.2 – Сравнение вычислительной сложности алгоритма Mini-Batch *K*-means++, алгоритма *k*-means++ и алгоритма FastKmeans++, наборы данных: 1 – ANN_SIFT1B(KM++), 2 – ANN_SIFT1B(MBKM++), 3 – ANN_ GIFT1M (KM++), 4 – ANN_GIFT1M(MBKM++), 5 – ANN_SIFT1M(KM++), 6 – ANN_SIFT1M(MBKM++), 7 – BIRCH3(KM++), 8 – BIRCH3(MBKM++), 9 – IHEPC (KM++), 10 – IHEPC (MBKM++), 11 – SUSY(KM++), 12 – SUSY(MBKM++), 13 – BIRCH3(FAST *K*-means++)

Результаты экспериментов по сравнению вычислительной сложности алгоритмов Mini-Batch *K*-means++, *k*-means++ и FastKMeans++ на большом наборе реальных и синтетических данных, приведенные в таблице 5.5 и на рисунке 5.2, экспериментально обосновывают логарифмическую вычислительную сложность нового алгоритма, который может решить задачу *k*-means++ с линейной временной сложностью относительно количества векторов данных. К дополнительным исследованиям нового алгоритма можно добавить сравнительные эксперименты с точностью кластеризации.

Результаты Раздела 5

В настоящей работе обзор методов инициализации *k*-средних показал, что совместная вычислительная сложность зависит от количества векторов данных. Отсутствуют методы инициализации центров кластеров алгоритма *k*-средних при

решении задач с большими наборами, порядки одного миллиона (миллиарда) векторов данных. В этом исследовании предложен быстрый и эффективный (в большинстве случаев статистически неотличимый с точки зрения достигнутых результатов целевой функции) алгоритм инициализации центра для алгоритма кластеризации *k*-средних. Мы показали, что качество решений нового алгоритма Mini-Batch *K*-means++ не уступает классическому *k*-means++. Основное преимущество в скорости инициализации центров, после вычисления вспомогательной структуры данных представленой упорядоченным массивом предварительно вычисленных слагаемых (с вычислительной сложностью $O(MkN)$), связано с уменьшением сложности вычислений (до $O(Mklog_2N)$, где N – количество векторов данных, M – количество измерений координат векторов данных, k – количество кластеров) с помощью формулы пакетного вычисления суммы квадратов расстояний путем поиска в упорядоченном массиве предварительно вычисленных слагаемых.

Предложенная процедура инициализации центров кластеров для быстрых алгоритмов кластеризации больших и сверхбольших объемов данных, по результатам проведенных нами экспериментов, в том числе на тех же задачах, востребованы в векторных базах данных и в системах дискового хранения данных.

В дальнейших исследованиях мы считаем перспективным совместить новый алгоритм Mini-Batch *K*-means++ с параллельной реализацией инициализации центров в алгоритме *k*-means||, который демонстрирует масштабируемость по мере увеличения количества кластеров в экспериментах на больших наборах данных. Также нас интересует инициализация центров кластеров по новому алгоритму при решении задачи p-медианы [244] с евклидовым и манхэттенским расстояниями, а также задачи p-medoid [245].

ЗАКЛЮЧЕНИЕ

В диссертации предложены новые решения, повышающие эффективность (точность и стабильность результатов, т.е. улучшение достигаемых значений целевой функции за заданное время) алгоритмов автоматической группировки объектов при большом объеме входных данных.

Цель диссертации достигается путем решения поставленных задач, а именно:

Разработанный новый подход к нормализации данных для предобработки входных данных, может использоваться в системах анализа данных результатов неразрушающих испытаний образцов промышленной продукции. Он комбинирует нормализацию по допустимым значениям параметров оцениваемых значений продукции и оценке Джеймса-Штейна. Кроме того, он позволяет выявлять наиболее значимые контролируемые параметры продукции, что в перспективе позволит сократить затраты на испытания за счет уменьшения числа контролируемых параметров.

Разработанный новый алгоритм кластеризации для системы анализа данных электрорадиоизделий на основе данных тестовых испытаний характеризуется вполне приемлемой точностью и скоростью работы программной реализации. Он расширяет набор (ансамбль) алгоритмов для применения в системах анализа данных электрорадиоизделий (разделения смешанной партии ЭРИ на однородные партии продукции) с применением задачи *k*-средних для нормализованных данных.

Разработанный новый алгоритм кластеризации для построения индекса векторной базы данных для построения приближенного индекса поиска ближайших соседей, реализован в рамках хоз договора в составе векторной. Дальнейшие исследования будут направлены на уменьшения разницы во времени вычислений при работе с большими наборами данных и увеличении количества центров/центроидов в результирующем индексе.

Разработанный новый алгоритм автоматической группировки повторяющихся фрагментов блоков данных на основе алгоритма *k*-средних, совместно с LSH для использования в системах хранения данных, реализован в рамках работ по договору СФУ для разработки модели оптимального использования дискового пространства с учетом компрессии данных.

Разработанная новая процедура инициализации центров кластеров для быстрых алгоритмов кластеризации больших и сверхбольших объемов данных, при поддержке Министерства науки и высшего образования Российской Федерации, по результатам проведенных нами экспериментов, востребована в системах дискового хранения данных и в векторных базах данных Российской технической компанией.

Результаты, изложенные в диссертации, имеют непосредственное практическое применение, в настоящее время новые разработанные алгоритмы был успешно применены в составе системы управления дисковыми хранилищами в ООО «Центр вычислительных технологий», а также в составе векторной СУБД Российской технологической компанией.

СПИСОК ЛИТЕРАТУРЫ

1. О нормализации данных в задаче автоматической группировки промышленной продукции по однородным производственным партиям / Ф. Г. Ахматшин, И. Р. Насыров, В. Л. Казаковцев, Л. А. Казаковцев // Системы управления и информационные технологии. – 2020. – № 2(80). – С. 86-89.

2. Reducing the James–Stein Shrinkage Estimator for Automatically Grouping Heterogeneous Production Batches / F. G. Akhmatshin, I. A. Petrova, L. A. Kazakovtsev, I. N. Kravchenko // Journal of Machinery Manufacture and Reliability. – 2024. – Vol. 53, No. 3. – P. 254-262.

3. Орлов, В. И. Качество электронной компонентной базы - залог длительной работоспособности космических аппаратов / В. И. Орлов, В. В. Федосов // Решетневские чтения. – 2013. – Т. 1. – С. 238-241.

4. Федосов, В. В. Повышение надежности радиоэлектронной аппаратуры космических аппаратов при применении электрорадиоизделий, прошедших дополнительные отбраковочные испытания в специализированных испытательных технических центрах / В. В. Федосов, В. Е. Патраев // Авиакосмическое приборостроение. – 2006. – № 10. – С. 50-56.

5. Патраев, В. Е. Анализ показателей качества и надежности при эксплуатации современных космических аппаратов / В. Е. Патраев, И. В. Трифанов // Вестник Сибирского государственного аэрокосмического университета им. академика М.Ф. Решетнева. – 2010. – № 2(28). – С. 110-113.

6. Данилин, Н. С. Диагностика и контроль качества изделий цифровой микроэлектроники [Текст] / Н.С.Данилин, Ю.Л.Нуров. - М. : Изд-во стандартов, 1991. - 175 с.

7. Данилин Н. Проблемы применения современной индустриальной электронной компонентной базы иностранного производства в ракетно-космической технике / Н. Данилин, С. Белослудцев // Современная электроника. - 2007. - №. 7. - С. 8-12.

8. Ooi M. P. L. et al. Getting more from the semiconductor test: Data mining with defect-cluster extraction //IEEE Transactions on Instrumentation and Measurement. – 2011. – Vol. 60. – No. 10. – P. 3300-3317.

9. ОТ 510-5608-05. Анализ технического состояния и оценка уровня фактической надежности и готовности к целевому использованию космических аппаратов народнохозяйственного назначения по результатам изготовления и эксплуатации в 2005 году. Анализ динамики изменения показателей надежности за период с 1994 по 2005 гг. Железногорск: НПОПМ, 2005. 176 с.

10. Направления развития системы управления качеством радиационных испытаний электронной компонентной базы / Ю. А. Ожегин, А. Ю. Никифоров, В. А. Телец [и др.] // Спецтехника и связь. – 2011. – № 4-5. – С. 59-62.

11. Казаковцев, Л. А. Метод жадных эвристик для систем автоматической группировки объектов : специальность 05.13.01 "Системный анализ, управление и обработка информации (по отраслям)" : диссертация на соискание ученой степени доктора технических наук / Казаковцев Л. А. – Красноярск, 2016. – 429 с.

12. Rozhnov, I. Ensembles of clustering algorithms for problem of detection of homogeneous production batches of semiconductor devices / I. Rozhnov, V. Orlov, L. Kazakovtsev // CEUR Workshop Proceedings, Omsk, 08–14 июля 2018 года. – Omsk, 2018. – P. 338-348.

13. Kazakovtsev, L. A. Fast deterministic algorithm for EEE components classification / L. A. Kazakovtsev, A. N. Antamoshkin, I. S. Masich // IOP Conference Series: Materials Science and Engineering : International Scientific and Research Conference on Topical Issues in Aeronautics and Astronautics (Dedicated to the 55th Anniversary from the Foundation of SibSAU), Krasnoyarsk, 06–10 апреля 2015 года. Vol. 94. – Krasnoyarsk: Institute of Physics Publishing, 2015. – P. 012015.

14. Additional screening tests at the testing technical center for ground power equipment / Y. V. Aliseenko, M. V. Nesterishin, E. O. Vorontsova [et al.] // Siberian Journal of Science and Technology. – 2019. – Vol. 20, No. 4. – P. 458-464.

15. Ахматшин, Ф. Г. Подбор свободного параметра алгоритма FOREL-2 в задаче автоматической группировки промышленной продукции по однородным

производственным партиям / Ф. Г. Ахматшин // Системы управления и информационные технологии. – 2021. – № 4(86). – С. 28-31.

16. Mathai A., Provost S., Haubold H. Chapter 11: Factor Analysis //Multivariate Statistical Analysis in the Real and Complex Domains. – Cham : Springer International Publishing, 2022. – P. 679-710.

17. Reddy C. K. Data clustering: algorithms and applications. – Chapman and Hall/CRC, 2018. 652 p.

18. Kumar S., Tripathi Y. M., Misra N. James–Stein type estimators for ordered normal means //Journal of Statistical Computation and Simulation. – 2005. – Vol. 75. – No. 7. – P. 501-511.

19. Масич, И. С. Отбор закономерностей для построения решающего правила в логических алгоритмах распознавания / И. С. Масич, Е. М. Краева // Системы управления и информационные технологии. – 2013. – № 1-1(51). – С. 170-173.

20. Kazakovtsev, L. A. Genetic algorithm with fast greedy heuristic for clustering and location problems / L. A. Kazakovtsev, A. N. Antamoshkin // Informatica (Ljubljana). – 2014. – Vol. 38, No. 3. – P. 229-240.

21. Farahani R. Z., Hekmatfar M. (ed.). Facility location: concepts, models, algorithms and case studies. – Springer Science & Business Media, 2009. 560 p.

22. Казаковцев, Л. А. Выбор метрики для системы автоматической классификации электрорадиоизделий по производственным партиям / Л. А. Казаковцев, А. А. Ступина, В. И. Орлов // Программные продукты и системы. – 2015. – № 2. – С. 124-129.

23. Алгоритм поиска в чередующихся окрестностях для задачи выделения однородных производственных партий электрорадиоизделий / В. И. Орлов, И. П. Рожнов, Л. А. Казаковцев, М. Н. Гудыма // Решетневские чтения. – 2018. – Т. 1. – С. 315-316.

24. Li Y., Wu H. A clustering method based on *K*-means algorithm //Physics Procedia. – 2012. – Vol. 25. – P. 1104-1109.

25. Efficiency of distance measures in the automatic grouping of electronic radio devices by k-means algorithm / G. Sh. Shkaberina, E. M. Tovbis, L. A. Kazakovtsev [et

al.] // IOP Conference Series: Materials Science and Engineering, Krasnoyarsk, 18–21 ноября 2019 года / Krasnoyarsk Science and Technology City Hall of the Russian Union of Scientific and Engineering Associations. Vol. 734. – Krasnoyarsk: Institute of Physics and IOP Publishing Limited, 2020. – P. 12136.

26. Hossain M. Z. et al. A dynamic *K*-means clustering for data mining //Indonesian Journal of Electrical engineering and computer science. – 2019. – Vol. 13. – No. 2. – P. 521-526.

27. Pérez-Ortega J., Almanza-Ortega N. N., Romero D. Balancing effort and benefit of *K*-means clustering algorithms in Big Data realms //PLoS One. – 2018. – Vol. 13. – No. 9. – P. e0201874.

28. Patel V. R., Mehta R. G. Modified *k*-means clustering algorithm //International Conference on Computational Intelligence and Information Technology. – Berlin, Heidelberg : Springer Berlin Heidelberg, 2011. – P. 307-312.

29. Na S., Xumin L., Yong G. Research on *k*-means clustering algorithm: An improved *k*-means clustering algorithm //2010 Third International Symposium on intelligent information technology and security informatics. – Ieee, 2010. – P. 63-67.

30. Gao J., Hitchcock D. B. James–Stein shrinkage to improve *k*-means cluster analysis //Computational Statistics & Data Analysis. – 2010. – Vol. 54. – No. 9. – P. 2113-2127.

31. Lloyd S. Least squares quantization in PCM //IEEE transactions on information theory. – 1982. – Vol. 28. – No. 2. – P. 129-137.

32. Загоруйко Н. Г. Прикладные методы анализа данных и знаний/Н. Г. Загоруйко -Новосибирск Изд-во Ин-та математики СО РАН, 1999 -270 с.

33. Rand W. M. Objective criteria for the evaluation of clustering methods //Journal of the American Statistical association. – 1971. – Vol. 66. – No. 336. – P. 846-850.

34. Ахматшин, Ф. Г. Подбор свободного параметра алгоритма FOREL-2 в задаче автоматической группировки промышленной продукции по однородным производственным партиям / Ф. Г. Ахматшин // Системы управления и информационные технологии. – 2021. – № 4(86). – С. 28-31.

35 Ахматшин, Ф. Г. Алгоритм FOREL-2 с жадной эвристикой выбора радиуса поиска локальных сгущений / Ф. Г. Ахматшин, Л. А. Казаковцев // Системы управления и информационные технологии. – 2022. – № 3(89). – С. 39-42.

36. Елкина В. Н., Елкин Е. А., Загоруйко Н. Г. О возможности применения методов распознавания образов в палеонтологии //Геология и геофизика. – 1967. – №. 9. – С. 8-15.

37. Функции конкурентного сходства в алгоритмах распознавания комбинированного типа / Н. Г. Загоруйко, И. А. Борисова, В. В. Дюбанов, О. А. Кутненко // Вестник Сибирского государственного аэрокосмического университета им. академика М.Ф. Решетнева. – 2010. – № 5(31). – С. 19-21.

38. Attribute selection through decision rules construction (algorithm FRiS-GRAD) / N. G. Zagoruiko [et al.] // Pattern Recognition and Image Analysis: New Information Technologies: Proc. of 9th Intern Conf. Nizhni Novgorod. 2008. Vol. 2. P. 335-338.

39. Загоруйко, Н. Г. Интеллектуальный анализ данных, основанный на функции конкурентного сходства / Н. Г. Загоруйко // Автометрия. – 2008. – Т. 44, № 3. – С. 30-40.

40. Identification of the Optimal Set of Informative Features for the Problem of Separating of Mixed Production Batch of Semiconductor Devices for the Space Industry / G. S. Shkaberina, V. I. Orlov, E. M. Tovbis, L. A. Kazakovtsev // Communications in Computer and Information Science. – 2019. – Vol. 1090. – P. 408-421.

41. Казаковцев, Л. А. Решение задачи Вебера для специальных случаев размещения на плоскости / Л. А. Казаковцев, М. Н. Гудыма // Решетневские чтения. – 2014. – Т. 2. – С. 52-53.

42. Ахматшин Ф.Г. Об алгоритме кластеризации для решения задачи поиска ближайшего соседа//Системы управления и информационные технологии, 3(97), 2024. С. 4-8.

43. McLachlan G. J. Mahalanobis distance //Resonance. – 1999. – Vol. 4. – No. 6. – P. 20-26.

44. Abbasifard M. R., Ghahremani B., Naderi H. A survey on nearest neighbor search methods //International Journal of Computer Applications. – 2014. – Vol. 95. – No. 25.

45. Bhatia N. et al. Survey of nearest neighbor techniques //arXiv preprint arXiv:1007.0085. – 2010.

46. Ponomarenko A. et al. Approximate nearest neighbor search small world approach //International Conference on Information and Communication Technologies & Applications. – 2011. – Vol. 17.

47. Hwang Y., Han B., Ahn H. K. A fast nearest neighbor search algorithm by nonlinear embedding //2012 IEEE conference on computer vision and pattern recognition. – IEEE, 2012. – P. 3053-3060.

48. Weber R., Schek H. J., Blott S. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces //VLDB. – 1998. – Vol. 98. – P. 194-205.

49. Redmond S. J., Heneghan C. A method for initialising the *K*-means clustering algorithm using kd-trees //Pattern recognition letters. – 2007. – Vol. 28. – No. 8. – P. 965-973.

50. Guttman A. R-trees: A dynamic index structure for spatial searching //Proceedings of the 1984 ACM SIGMOD international conference on Management of data. – 1984. – P. 47-57.

51. Jagadish H. V. et al. iDistance: An adaptive B+-tree based indexing method for nearest neighbor search //ACM Transactions on Database Systems (TODS). – 2005. – Vol. 30. – No. 2. – P. 364-397.

52. Song Z. et al. The B+-tree-based Method for Nearest Neighbor Queries in Traffic Simulation Systems //TELKOMNIKA Indonesian Journal of Electrical Engineering. – 2014. – Vol. 12. – No. 12. – P. 8175-8192.

53. Jafari O. et al. Optimizing fair approximate nearest neighbor searches using threaded b+-trees //Similarity Search and Applications: 14th International Conference, SISAP 2021, Dortmund, Germany, September 29–October 1, 2021, Proceedings 14. – Springer International Publishing, 2021. – P. 133-147.

54. Kraus P., Dzwinel W. Nearest neighbor search by using Partial KD-tree method //Theor. Appl. Genet. – 2008. – Vol. 20. – P. 149-165.

55. Yen S. H. et al. Nearest neighbor searching in high dimensions using multiple

KD-trees //Proceedings of the 10th WSEAS international conference on Signal processing, computational geometry and artificial vision. – 2010. – P. 40-45.

56. Papadopoulos A., Manolopoulos Y. Performance of nearest neighbor queries in R-trees //Database Theory—ICDT'97: 6th International Conference Delphi, Greece, January 8–10, 1997 Proceedings 6. – Springer Berlin Heidelberg, 1997. – P. 394-408.

57. Cheung K. L., Fu A. W. C. Enhanced nearest neighbour search on the R-tree //ACM SIGMOD Record. – 1998. – Vol. 27. – No. 3. – P. 16-21.

58. Beygelzimer A., Kakade S., Langford J. Cover trees for nearest neighbor //Proceedings of the 23rd international conference on Machine learning. – 2006. – P. 97-104.

59. Elkin Y. A new compressed cover tree for *k*-nearest neighbour search and the stable-under-noise mergegram of a point cloud. – The University of Liverpool (United Kingdom), 2022. 144 p.

60. Karger D. R., Ruhl M. Finding nearest neighbors in growth-restricted metrics //Proceedings of the thiry-fourth annual ACM symposium on Theory of computing. – 2002. – P. 741-750.

61. Beeri C., Buneman P. (ed.). Database Theory-ICDT'99: 7th International Conference, Jerusalem, Israel, January 10-12, 1999, Proceedings. – Springer, 2003. 489 p.

62. Ponomarenko A. et al. Approximate nearest neighbor search small world approach //International Conference on Information and Communication Technologies & Applications. – 2011. – Vol. 17.

63. Scalable distributed algorithm for approximate nearest neighbor search problem in high dimensional general metric spaces / Y. Malkov, A. Ponomarenko, A. Logvinov, V. Krylov // Lecture Notes in Computer Science. – 2012. – Vol. 7404 LNCS. – P. 132-147.

64. Approximate nearest neighbor algorithm based on navigable small world graphs / Y. Malkov, A. Ponomarenko, A. Logvinov, V. Krylov // Information Systems. – 2014. – Vol. 45. – P. 61-68.

65. Malkov, Y. A. Efficient and Robust Approximate Nearest Neighbor Search Using

Hierarchical Navigable Small World Graphs / Y. A. Malkov, D. A. Yashunin // IEEE Transactions on Pattern Analysis and Machine Intelligence. – 2020. – Vol. 42, No. 4. – P. 824-836.

66. Cover T., Hart P. Nearest neighbor pattern classification //IEEE transactions on information theory. – 1967. – Vol. 13. – No. 1. – P. 21-27.

67. Ge T. et al. Optimized product quantization for approximate nearest neighbor search //Proceedings of the IEEE conference on computer vision and pattern recognition. – 2013. – P. 2946-2953.

68. Casey M. A., Slaney M. Locality Sensitive Hashing for Large Music Databases //Signal Processing Magazine, IEEE. – 2008. – Vol. 25. – No. 2. – P. 128-131.

69. Zhao K., Lu H., Mei J. Locality preserving hashing //Proceedings of the AAAI Conference on Artificial Intelligence. – 2014. – Vol. 28. – No. 1.

70. Tsai Y. H., Yang M. H. Locality preserving hashing //2014 IEEE International Conference on Image Processing (ICIP). – IEEE, 2014. – P. 2988-2992.

71. Blott S., Weber R. A simple vector-approximation file for similarity search in high-dimensional vector spaces //ESPRIT Technical Report TR19, ca. – 1997.

72. Sahu M., Yerpude P. Vector Approximation File: Cluster Bounding in High-Dimension Data Set., 2000.

73. Kriegel H. P. et al. Efficient query processing in arbitrary subspaces using vector approximations //18th International Conference on Scientific and Statistical Database Management (SSDBM'06). – IEEE, 2006. – P. 184-190.

74. Jegou H., Douze M., Schmid C. Product quantization for nearest neighbor search //IEEE transactions on pattern analysis and machine intelligence. – 2010. – Vol. 33. – No. 1. – P. 117-128.

75. Geist M., Pietquin O., Fricout G. Kernelizing vector quantization algorithms //ESANN'2009. – 2009. – P. 541-546.

76. Kalantidis Y., Avrithis Y. Locally optimized product quantization for approximate nearest neighbor search //Proceedings of the IEEE conference on computer vision and pattern recognition. – 2014. – P. 2321-2328.

77. Ge T. et al. Optimized product quantization for approximate nearest neighbor

search //Proceedings of the IEEE conference on computer vision and pattern recognition. – 2013. – P. 2946-2953.

78. Yu T. et al. Product quantization network for fast visual search //International Journal of Computer Vision. – 2020. – Vol. 128. – No. 8. – P. 2325-2343.

79. Zhang M., Zhe X., Yan H. Orthonormal product quantization network for scalable face image retrieval //Pattern Recognition. – 2023. – Vol. 141. – P. 109671.

80. Gu L. et al. Entropy-Optimized Deep Weighted Product Quantization for Image Retrieval //IEEE Transactions on Image Processing. – 2024. – P. 1162-1174.

81. Wang J. et al. Milvus: A purpose-built vector data management system //Proceedings of the 2021 International Conference on Management of Data. – 2021. – P. 2614-2627.

82. Jin Y. et al. Curator: Efficient Indexing for Multi-Tenant Vector Databases //arXiv preprint arXiv:2401.07119. – 2024. [Электронный ресурс]. URL: https://arxiv.org/abs/2401.07119. (дата обращения: 24.12.2024).

83. Johnson J., Douze M., Jégou H. Billion-scale similarity search with GPUs //IEEE Transactions on Big Data. – 2019. – Vol. 7. – No. 3. – P. 535-547.

84. Alp O., Erkut E., Drezner Z. An efficient genetic algorithm for the p-median problem //Annals of Operations research. – 2003. – Vol. 122. – P. 21-42.

85. Kochetov Y. Large neighborhood local search for the p-median problem //Yugoslav Journal of Operations Research. – 2016. – Vol. 15. – No. 1.

86. Fränti P., Sieranoja S. *K*-means properties on six clustering benchmark datasets //Applied intelligence. – 2018. – Vol. 48. – C. 4743-4759.

87. MacQueen J. et al. Some methods for classification and analysis of multivariate observations //Proceedings of the fifth Berkeley symposium on mathematical statistics and probability. – 1967. – Vol. 1. – No. 14. – P. 281-297.

88. Ahmed M., Seraj R., Islam S. M. S. The *k*-means algorithm: A comprehensive survey and performance evaluation //Electronics. – 2020. – Vol. 9. – No. 8. – P. 1295.

89. Golasowski M., Martinovič J., Slaninová K. Comparison of K-means clustering initialization approaches with brute-force initialization //Advanced Computing and Systems for Security: Volume Three. – 2017. – P. 103-114.

90. Steinhaus H. et al. Sur la division des corps matériels en parties //Bull. Acad. Polon. Sci. – 1956. – Vol. 1. – No. 804. – P. 801.

91. Weiszfeld E. Sur le point pour lequel la somme des distances de n points donnés est minimum //Tohoku Mathematical Journal, First Series. – 1937. – Vol. 43. – P. 355-386.

92. Cooper L., Katz I. N. The Weber problem revisited //Computers & Mathematics with Applications. – 1981. – Vol. 7. – No. 3. – P. 225-234.

93. Kuhn H. W. A note on Fermat's problem //Mathematical programming. – 1973. – Vol. 4. – P. 98-107.

94. Ostresh Jr L. M. On the convergence of a class of iterative methods for solving the Weber location problem //Operations Research. – 1978. – Vol. 26. – No. 4. – P. 597-609.

95. Plastria F., Elosmani M. On the convergence of the Weiszfeld algorithm for continuous single facility location–allocation problems //Top. – 2008. – Vol. 16. – No. 2. – P. 388-406.

96. Vardi Y., Zhang C. H. The multivariate L 1-median and associated data depth //Proceedings of the National Academy of Sciences. – 2000. – Vol. 97. – No. 4. – P. 1423-1426.

97. Bādoiu M., Har-Peled S., Indyk P. Approximate clustering via core-sets //Proceedings of the thiry-fourth annual ACM symposium on Theory of computing. – 2002. – P. 250-257.

98. Kuenne R. E., Kuhn H. W. An efficient algorithm for the numerical solution of the generalized weber problem in spatial economics //General Equilibrium Economics: Space, Time and Money. – 1992. – P. 223-240.

99. Kazakovtsev L. A., Rozhnov I. P. Comparative study of local search in SWAP and agglomerative neighbourhoods for the continuous p-median problem //IOP Conference Series: Materials Science and Engineering. – IOP Publishing, 2021. – Vol. 1047. – No. 1. – P. 012079.

100. Arthur D., Vassilvitskii S. *k*-means++: The advantages of careful seeding. – Stanford, 2006. – P. 1027-1035.

101. Ahmatshin F. G., Kazakovtsev L. A. Mini-batch *K*-means++ clustering initialization. //XXIII International Conference Mathematical Optimization Theory and Operations Research MOTOR-2024 Omsk Russia, June 30 - July 06, 2024. – P. 293-307.

102. Self-adjusting variable neighborhood search algorithm for near-optimal k-means clustering / L. Kazakovtsev, I. Rozhnov, A. Popov, E. Tovbis // Computation. – 2020. – Vol. 8, No. 4. – P. 1-32.

103. Hansen P., Mladenović N. J-means: a new local search heuristic for minimum sum of squares clustering //Pattern recognition. – 2001. – Vol. 34. – No. 2. – P. 405-413.

104. Hansen P., Mladenović N. Variable neighborhood search: Principles and applications //European journal of operational research. – 2001. – Vol. 130. – No. 3. – P. 449-467.

105. Ribeiro C. C. et al. Developments of variable neighborhood search. – Springer US, 2002. – P. 415-439.

106. Mladenović N., Hansen P. Variable neighborhood search //Computers & operations research. – 1997. – Vol. 24. – No. 11. – P. 1097-1100.

107. Doerr B. et al. The (1+ λ) evolutionary algorithm with self-adjusting mutation rate //Proceedings of the Genetic and Evolutionary Computation Conference. – 2017. – P. 1351-1358.

108. Kazakovtsev L., Rozhnov I., Kazakovtsev V. A (1+ λ) evolutionary algorithm with the greedy agglomerative mutation for p-median problems //AIP Conference Proceedings. – AIP Publishing, 2023. – Vol. 2700. – No. 1.

109. Droste S., Jansen T., Wegener I. On the analysis of the (1+ 1) evolutionary algorithm //Theoretical Computer Science. – 2002. – Vol. 276. – No. 1-2. – P. 51-81.

110. Borisovsky, P.A.; Eremeev, A.V. A study on performance of the (1+1)-Evolutionary Algorithm. In Foundations of Genetic Algorithms, De Jong, K., Poli, R., Rowe, J. Eds.; Morgan Kaufmann, San Francisco, 2003, P. 271–287.

111. Borisovsky, P. A. Comparing evolutionary algorithms to the (1+1) -EA / P. A. Borisovsky, A. V. Eremeev // Theoretical Computer Science. – 2008. – Vol. 403, No. 1.

– P. 33-41.

112. Kazakovtsev, L. Self-configuring (1 + 1)-evolutionary algorithm for the continuous p-median problem with agglomerative mutation / L. Kazakovtsev, I. Rozhnov, G. Shkaberina // Algorithms. – 2021. – Vol. 14, No. 5.

113. Anda S., Kikuchi M., Ozono T. Developing a component comment extractor from product reviews on e-commerce sites //2022 12th International Congress on Advanced Applied Informatics (IIAI-AAI). – IEEE, 2022. – P. 83-88.

114. Powers D. M. W. Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation //arXiv preprint arXiv:2010.16061. – 2020.

115. Kazakovtsev, L. A. Greedy heuristic method for location problems / L. A. Kazakovtsev, A. N. Antamoshkin // Vestnik SibSAU. Aerospace tehnologies and control systems. – 2015. – Vol. 16, No. 2. – P. 317-325.

116. Bandyopadhyay S., Maulik U. An evolutionary technique based on *K*-means algorithm for optimal clustering in RN //Information Sciences. – 2002. – Vol. 146. – No. 1-4. – P. 221-237.

117. Maulik U., Bandyopadhyay S. Genetic algorithm-based clustering technique //Pattern recognition. – 2000. – Vol. 33. – No. 9. – P. 1455-1465.

118. Ахматшин Ф.Г. О сжатии данных с использованием алгоритма кластеризации *K*-mean// Системы управления и информационные технологии, №3(97), 2024. С. 68-72.

119. Методы сжатия данных. Устройство архиваторов, сжатие изображений и видео / В. Юкин, М. Смирнов, А. Ратушняк, Д. Ватолин : Диалог-МИФИ, 2003. – 381 с.

120. Huffman D. A. A method for the construction of minimum-redundancy codes //Proceedings of the IRE. – 1952. – Vol. 40. – No. 9. – P. 1098-1101.

121. Ziv J., Lempel A. A universal algorithm for sequential data compression //IEEE Transactions on information theory. – 1977. – Vol. 23. – No. 3. – P. 337-343.

122. Ziv J., Lempel A. Compression of individual sequences via variable-rate coding //IEEE transactions on Information Theory. – 1978. – Vol. 24. – No. 5. – P. 530-536.

123. Witten I. H., Neal R. M., Cleary J. G. Arithmetic coding for data compression

//Communications of the ACM. – 1987. – Vol. 30. – No. 6. – P. 520-540.

124. Bell T., Witten I. H., Cleary J. G. Modeling for text compression //ACM Computing Surveys (CSUR). – 1989. – Vol. 21. – No. 4. – P. 557-591.

125. Burrows M. A bloc*k*-sorting lossless data compression algorithm //SRS Research Report. – 1994. – Vol. 124.

126. Qiao W. et al. An FPGA-based BWT accelerator for Bzip2 data compression //2019 IEEE 27th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM). – IEEE, 2019. – P. 96-99.

127. Kerbiriou M., Chikhi R. Parallel decompression of gzip-compressed files and random access to DNA sequences //2019 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW). – IEEE, 2019. – P. 209-217.

128. Brin S., Davis J., Garcia-Molina H. Copy detection mechanisms for digital documents //Proceedings of the 1995 ACM SIGMOD international conference on Management of data. – 1995. – P. 398-409.

129. Broder A. Z. On the resemblance and containment of documents //Proceedings. Compression and Complexity of SEQUENCES 1997 (Cat. No. 97TB100171). – IEEE, 1997. – P. 21-29.

130. Indyk P., Motwani R. Approximate nearest neighbors: towards removing the curse of dimensionality //Proceedings of the thirtieth annual ACM symposium on Theory of computing. – 1998. – P. 604-613.

131. Buhler J. Efficient large-scale sequence comparison by locality-sensitive hashing //Bioinformatics. – 2001. – Vol. 17. – No. 5. – P. 419-428.

132. Azimpourkivi M., Topkara U., Carbunar B. A secure mobile authentication alternative to biometrics //Proceedings of the 33rd Annual Computer Security Applications Conference. – 2017. – P. 28-41.

133. Jiang Q., Sun M. Semi-supervised simhash for efficient document similarity search //Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies. – 2011. – P. 93-101.

134. Berlin K. et al. Assembling large genomes with single-molecule sequencing and locality-sensitive hashing //Nature biotechnology. – 2015. – Vol. 33. – No. 6. – P. 623-630.

135. Moura P. et al. LSHSIM: a locality sensitive hashing based method for multiple-point geostatistics //Computers & Geosciences. – 2017. – Vol. 107. – P. 49-60.

136. Chen D. Structural Nonparallel Support Vector Machine Based on LSH for Large-Scale Prediction //2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW). – IEEE, 2016. – P. 839-846.

137. Kim Y. B., O'Reilly U. M. Analysis of locality-sensitive hashing for fast critical event prediction on physiological time series //2016 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC). – IEEE, 2016. – P. 783-787.

138. Gionis A. et al. Similarity search in high dimensions via hashing //Vldb. – 1999. – Vol. 99. – No. 6. – P. 518-529.

139. Datar M. et al. Locality-sensitive hashing scheme based on p-stable distributions //Proceedings of the twentieth annual symposium on Computational geometry. – 2004. – P. 253-262.

140. Charikar M. S. Similarity estimation techniques from rounding algorithms //Proceedings of the thiry-fourth annual ACM symposium on Theory of computing. – 2002. – P. 380-388.

141. Ryynanen M., Klapuri A. Query by humming of midi and audio using locality sensitive hashing //2008 IEEE International Conference on Acoustics, Speech and Signal Processing. – IEEE, 2008. – P. 2249-2252.

142. Zhuvikin A. A BLOCKCHAIN OF IMAGE COPYRIGHTS USING ROBUST IMAGE FEATURES AND LOCALITY-SENSITIVE HASHING //International Journal of Computer Science & Applications. – 2018. – Vol. 15. – No. 1.

143. Li H. et al. Large-scale documents reduction based on domain ontology and E2LSH //Proceedings of the 11th IEEE International Conference on Networking, Sensing and Control. – IEEE, 2014. – P. 24-29.

144. Shrivastava A., Li P. Asymmetric LSH (ALSH) for sublinear time maximum inner product search (MIPS) //Advances in neural information processing systems. – 2014. – Vol. 27.

145. Kanji Tanaka and Eiji Kondo. 2008. A scalable localization algorithm for high dimensional features and multi robot systems. In 2008 IEEE International Conference on Networking, Sensing and Control. IEEE, 920–925.

146. Saeki K., Tanaka K., Ueda T. Lshransac: An incremental scheme for scalable localization //2009 IEEE International Conference on Robotics and Automation. – IEEE, 2009. – С. 35233530.

147. Bawa M., Condie T., Ganesan P. LSH forest: self-tuning indexes for similarity search //Proceedings of the 14th international conference on World Wide Web. – 2005. – P. 651-660.

148. Probst D., Reymond J. L. A probabilistic molecular fingerprint for big data settings //Journal of cheminformatics. – 2018. – Vol. 10. – P. 1-12.

149. Yu Y., Tang S., Zimmermann R. Edge-based locality sensitive hashing for efficient geo-fencing application //Proceedings of the 21st ACM SIGSPATIAL international conference on advances in geographic information systems. – 2013. – P. 576-579.

150. Cochez M., Terziyan V., Ermolayev V. Large scale knowledge matching with balanced efficiency-effectiveness using lsh forest //Transactions on Computational Collective Intelligence XXVI. – Springer International Publishing, 2017. – P. 46-66.

151. Cayton L., Dasgupta S. A learning framework for nearest neighbor search //Advances in Neural Information Processing Systems. – 2007. – Vol. 20.

152. Lv Q. et al. Multi-probe LSH: efficient indexing for high-dimensional similarity search //Proceedings of the 33rd international conference on Very large data bases. – 2007. – P. 950-961.

153. Zhang B., Liu X., Lang B. Fast graph similarity search via locality sensitive hashing //Advances in Multimedia Information Processing--PCM 2015: 16th Pacific-Rim Conference on Multimedia, Gwangju, South Korea, September 16-18, 2015, Proceedings, Part I 16. – Springer International Publishing, 2015. – P. 623-633.

154. Lv Q. et al. A time-space efficient locality sensitive hashing method for similarity search in high dimensions //Technical report, Tech. Rep. – 2006.

155. Joly A., Buisson O. A posteriori multi-probe locality sensitive hashing //Proceedings of the 16th ACM international conference on Multimedia. – 2008. – P. 209-218.

156. Jégou H. et al. Query adaptative locality sensitive hashing //2008 IEEE International Conference on Acoustics, Speech and Signal Processing. – IEEE, 2008. – P. 825-828.

157. Zhang W. et al. Data-oriented locality sensitive hashing //Proceedings of the 18th ACM international conference on Multimedia. – 2010. – P. 1131-1134.

158. Dasgupta A., Kumar R., Sarlós T. Fast locality-sensitive hashing //Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining. – 2011. – P. 1073-1081.

159. Satuluri V., Parthasarathy S. Bayesian locality sensitive hashing for fast similarity search //arXiv preprint arXiv:1110.1328. – 2011.

160. Kulis B., Grauman K. Kernelized locality-sensitive hashing //IEEE Transactions on Pattern Analysis and Machine Intelligence. – 2011. – Vol. 34. – No. 6. – P. 1092-1104.

161. Gan J. et al. Locality-sensitive hashing scheme based on dynamic collision counting //Proceedings of the 2012 ACM SIGMOD international conference on management of data. – 2012. – P. 541-552.

162. Pan J., Manocha D. Bi-level locality sensitive hashing for *k*-nearest neighbor computation //2012 IEEE 28th International Conference on Data Engineering. – IEEE, 2012. – P. 378-389.

163. Wang Q. et al. Boundary-expanding locality sensitive hashing //2012 8th International Symposium on Chinese Spoken Language Processing. – IEEE, 2012. – P. 358-362.

164. Ji J. et al. Super-bit locality-sensitive hashing //Advances in neural information processing systems. – 2012. – Vol. 25.

165. Deorowicz S. et al. Whisper: Read sorting allows robust mapping of sequencing data //bioRxiv. – 2017. – P. 240358.

166. Pamulaparty L., Rao C. V. G. A novel approach to perform document clustering using effectiveness and efficiency of simhash //International Journal of Engineering and Advanced Technology. – 2013. – Vol. 2. – No. 3. – P. 312-315.

167. Lee K. M. A projection-based locality-sensitive hashing technique for reducing false negatives //Applied Mechanics and Materials. – 2013. – Vol. 263. – P. 1341-1346.

168. Gu X. et al. An improved method of locality sensitive hashing for indexing large-scale and high-dimensional features //Signal Processing. – 2013. – Vol. 93. – No. 8. – P. 2244-2255.

169. Yin S., Badr M., Vodislav D. Dynamic multi-probe lsh: An i/o efficient index structure for approximate nearest neighbor search //Database and Expert Systems Applications: 24th International Conference, DEXA 2013, Prague, Czech Republic, August 26-29, 2013. Proceedings, Part I 24. – Springer Berlin Heidelberg, 2013. – P. 48-62.

170. Zhang L. et al. Distribution-aware locality sensitive hashing //Advances in Multimedia Modeling: 19th International Conference, MMM 2013, Huangshan, China, January 7-9, 2013, Proceedings, Part II. – Springer Berlin Heidelberg, 2013. – P. 395-406.

171. Lee K. M., Lee K. M. A locality sensitive hashing technique for categorical data //Applied Mechanics and Materials. – 2013. – Vol. 241. – P. 3159-3164.

172. Bai X. et al. Data-dependent hashing based on p-stable distribution //IEEE Transactions on Image Processing. – 2014. – Vol. 23. – No. 12. – P. 5033-5046.

173. Xie H. et al. Data-dependent locality sensitive hashing //Advances in Multimedia Information Processing–PCM 2014: 15th Pacific-Rim Conference on Multimedia, Kuching, Malaysia, December 1-4, 2014, Proceedings 15. – Springer International Publishing, 2014. – P. 284-293.

174. Andoni A. et al. Beyond locality-sensitive hashing //Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms. – Society for Industrial and Applied Mathematics, 2014. – P. 1018-1028.

175. Wang P., Yin D., Sun T. Bi-Level Locality Sensitive Hashing Index Based on Clustering //Applied Mechanics and Materials. – 2014. – Vol. 556. – P. 3804-3808.

176. Sun Y. et al. SRS: solving c-approximate nearest neighbor queries in high dimensional euclidean space with a tiny index //Proceedings of the VLDB Endowment. – 2014. – P. 1-12.

177. Liu Y. et al. SKLSH: an efficient index structure for approximate nearest neighbor search //Proceedings of the VLDB Endowment. – 2014. – Vol. 7. – No. 9. – P. 745-756.

178. Ji J. et al. Batch-orthogonal locality-sensitive hashing for angular similarity //IEEE transactions on pattern analysis and machine intelligence. – 2014. – Vol. 36. – No. 10. – P. 1963-1974.

179. Chakrabarti A. et al. A bayesian perspective on locality sensitive hashing with extensions for kernel methods //ACM Transactions on Knowledge Discovery from Data (TKDD). – 2015. – Vol. 10. – No. 2. – P. 1-32.

180. Huang Q. et al. Query-aware locality-sensitive hashing for approximate nearest neighbor search //Proceedings of the VLDB Endowment. – 2015. – Vol. 9. – No. 1. – P. 1-12.

181. Zheng Y. et al. Lazylsh: Approximate nearest neighbor search for multiple distance functions with a single index //Proceedings of the 2016 International Conference on Management of Data. – 2016. – P. 2023-2037.

182. Yu C. et al. A generic method for accelerating LSH-based similarity join processing //IEEE Transactions on Knowledge and Data Engineering. – 2016. – Vol. 29. – No. 4. – P. 712-726.

183. Huang Q. et al. Query-aware locality-sensitive hashing scheme for lp norm //The VLDB Journal. – 2017. – Vol. 26. – No. 5. – P. 683-708.

184. Liu W. et al. I-LSH: I/O efficient c-approximate nearest neighbor search in high-dimensional space //2019 IEEE 35th International Conference on Data Engineering (ICDE). – IEEE, 2019. – P. 1670-1673.

185. Dong Y. et al. Learning space partitions for nearest neighbor search //arXiv preprint arXiv:1901.08544. – 2019.

186. Kim S., Yang H., Kim M. Boosted locality sensitive hashing: Discriminative binary codes for source separation //ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). – IEEE, 2020. – P. 106-110.

187. Zheng B. et al. PM-LSH: A fast and accurate LSH framework for high-dimensional approximate NN search //Proceedings of the VLDB Endowment. – 2020. – Vol. 13. – No. 5. – P. 643-655.

188. Lu K., Kudo M. R2LSH: A nearest neighbor search scheme based on two-dimensional projected spaces //2020 IEEE 36th International Conference on Data Engineering (ICDE). – IEEE, 2020. – P. 1045-1056.

189. Rong K. et al. Locality-sensitive hashing for earthquake detection: A case study of scaling data-driven science //arXiv preprint arXiv:1803.09835. – 2018.

190. Shah A. S., Sethi M. A. J. The improvised GZIP, a technique for real time lossless data compression //EAI Endorsed Transactions on Context-aware Systems and Applications. – 2019. – Vol. 6. – No. 17. – P. e5-e5.

191. Котелина Н. О., Матвийчук Б. Р. Кластеризация изображения методом *k*-средних //Вестник Сыктывкарского университета. Серия 1. Математика. Механика. Информатика. – 2019. – No. 32. – P. 101-112.

192. Munshi A. et al. Image compression using *K*-mean clustering algorithm //International Journal of Computer Science & Network Security. – 2021. – Vol. 21 – No. 9. – P. 275-280.

193. Dvorský J. et al. Document Compression Improvements Based on Data Clustering //Web Intelligence and Intelligent Agents. – 2010. – P. 133.

194. Wang C. et al. Chunk2vec: A novel resemblance detection scheme based on Sentence-BERT for post-deduplication delta compression in network transmission //IET Communications. – 2024. – Vol. 18. – No. 2. – P. 145-159.

195. Ni F., Jiang S. RapidCDC: Leveraging duplicate locality to accelerate chunking in CDC-based deduplication systems //Proceedings of the ACM symposium on cloud computing. – 2019. – P. 220-232.

196. Xia W. et al. {FastCDC}: A fast and efficient {Content-Defined} chunking approach for data deduplication //2016 USENIX Annual Technical Conference

(USENIX ATC 16). – 2016. – P. 101-114.

197. Zhang Y. et al. AE: An asymmetric extremum content defined chunking algorithm for fast and bandwidth-efficient data deduplication //2015 IEEE Conference on Computer Communications (INFOCOM). – IEEE, 2015. – P. 1337-1345.

198. Brent R. P. A linear algorithm for data compression //Australian Computer Journal. – 1987. – Vol. 19. – No. 2. – P. 64-68.

199. Manku G. S., Jain A., Das Sarma A. Detecting near-duplicates for web crawling //Proceedings of the 16th international conference on World Wide Web. – 2007. – P. 141-150.

200. Rivest R. The MD5 message-digest algorithm. – 1992. – No. rfc1321.

201. Mahoney M. Large text compression benchmark [Электронный ресурс]. URL: https://mattmahoney.net/dc/text.html. (дата обращения: 24.12.2024).

202. Amsaleg L., Jegou H. Datasets for approximate nearest neighbor search. – 2010.

203. Минь Д. Б. и др. Сжатие данных //Проблемы современной науки и образования. – 2017. – No. 1 (83). – P. 55-56.

204. Бурцев В. Л. и др. Области применения и классификация методов сжатия данных //Открытое образование. – 2011. – No. 4. – P. 57-64.

205. Катиева Л. М. Методы сжатия данных //Молодой ученый. – 2020. – №. 36. – С. 12-15.

206. Жиляков Е. Г. и др. Сжатие речевых данных как средство обеспечения скрытности речевых сообщений //Вестник Национального технического университета Харьковский политехнический институт. Серия: Информатика и моделирование. – 2009. – No. 43. – P. 75-83.

207. Исмагилов И. И., Васильева М. Ю. Сжатие цифровых изображений с использованием преобразований Уолша: алгоритмы и сравнительный анализ их эффективности //Известия высших учебных заведений. Проблемы энергетики. – 2008. – No. 9-10. – P. 91-99.

208. Ахматшин Ф.Г. О методе инициализации для алгоритмов кластеризации// Системы управления и информационные технологии, №1(95), 2024. С. 4-10.

209. Jain A. K., Murty M. N., Flynn P. J. Data clustering: a review //ACM computing surveys (CSUR). – 1999. – Vol. 31. – No. 3. – P. 264-323.

210. Tarsitano A. A computational study of several relocation methods for *k*-means algorithms //Pattern recognition. – 2003. – Vol. 36. – No. 12. – P. 2955-2966.

211. Celebi M. E. Improving the performance of *k*-means for color quantization //Image and Vision Computing. – 2011. – Vol. 29. – No. 4. – P. 260-271.

212. Forgy E. W. Cluster analysis of multivariate data: efficiency versus interpretability of classifications //biometrics. – 1965. – Vol. 21. – P. 768-769.

213. Späth H. Computational experiences with the exchange method: Applied to four commonly used partitioning cluster analysis criteria //European Journal of Operational Research. – 1977. – Vol. 1. – No. 1. – P. 23-31.

214. Ball G. H., Hall D. J. A clustering technique for summarizing multivariate data //Behavioral science. – 1967. – Vol. 12. – No. 2. – P. 153-155.

215. Tou J. T., Gonzalez R. C. Pattern recognition principles. – 1974. – 377 p.

216. Gonzalez T. F. Clustering to minimize the maximum intercluster distance //Theoretical computer science. – 1985. – Vol. 38. – P. 293-306.

217. Katsavounidis I., Kuo C. C. J., Zhang Z. A new initialization technique for generalized Lloyd iteration //IEEE Signal processing letters. – 1994. – Vol. 1. – No. 10. – P. 144-146.

218. Moh'd B A. D., Roberts S. A. New methods for the initialisation of clusters //Pattern Recognition Letters. – 1996. – Vol. 17. – No. 5. – P. 451-455.

219. Pizzuti C., Talia D., Vonella G. A divisive initialisation method for clustering algorithms //Principles of Data Mining and Knowledge Discovery: Third European Conference, PKDD'99, Prague, Czech Republic, September 15-18, 1999. Proceedings 3. – Springer Berlin Heidelberg, 1999. – P. 484-491.

220. Bradley P. S., Fayyad U. M. Refining initial points for *k*-means clustering //ICML. – 1998. – Vol. 98. – P. 91-99.

221. Hotelling H. Simplified calculation of principal components //Psychometrika. – 1936. – Vol. 1. – No. 1. – P. 27-35.

222. Su T., Dy J. G. In search of deterministic methods for initializing *K*-means and Gaussian mixture clustering //Intelligent Data Analysis. – 2007. – Vol. 11. – No. 4. – P. 319-338.

223. Lu J. F. et al. Hierarchical initialization approach for *K*-Means clustering //Pattern Recognition Letters. – 2008. – Vol. 29. – No. 6. – P. 787-795.

224. Onoda T., Sakai M., Yamada S. Careful seeding method based on independent components analysis for *k*-means clustering //Journal of Emerging Technologies in Web Intelligence. – 2012. – Vol. 4. – No. 1. – P. 51-59.

225. Al Hasan M. et al. Robust partitional clustering by outlier and density insensitive seeding //Pattern Recognition Letters. – 2009. – Vol. 30. – No. 11. – P. 994-1002.

226. Hartigan J. A., Wong M. A. Algorithm AS 136: A *k*-means clustering algorithm //Journal of the royal statistical society. series c (applied statistics). – 1979. – Vol. 28. – No. 1. – P. 100-108.

227. Kaufman L., Rousseeuw P. J. Finding groups in data: an introduction to cluster analysis. – John Wiley & Sons, 2009. – 344 p.

228. Aloise D. et al. NP-hardness of Euclidean sum-of-squares clustering //Machine learning. – 2009. – Vol. 75. – P. 245-248.

229. Breunig M. M. et al. LOF: identifying density-based local outliers //Proceedings of the 2000 ACM SIGMOD international conference on Management of data. – 2000. – P. 93-104.

230. Mahajan M., Nimbhorkar P., Varadarajan K. The planar *k*-means problem is NP-hard //Theoretical Computer Science. – 2012. – Vol. 442. – P. 13-21.

231. Astrahan M. M. Speech analysis by clustering, or the hyperphoneme method. – Standford University, 1970. – 24 p.

232. Lance G. N., Williams W. T. A general theory of classificatory sorting strategies: II. Clustering systems //The computer journal. – 1967. – Vol. 10. – No. 3. – P. 271-277.

233. Cao F., Liang J., Jiang G. An initialization method for the *K*-Means algorithm using neighborhood model //Computers & Mathematics with Applications. – 2009. – Vol. 58. – No. 3. – P. 474-483.

234. Linde Y., Buzo A., Gray R. An algorithm for vector quantizer design //IEEE Transactions on communications. – 1980. – Vol. 28. – No. 1. – P. 84-95.

235. Huang C. M., Harris R. W. A comparison of several vector quantization codebook generation approaches //IEEE Transactions on Image Processing. – 1993. – Vol. 2. – No. 1. – P. 108-112.

236. Likas A., Vlassis N., Verbeek J. J. The global *k*-means clustering algorithm //Pattern recognition. – 2003. – Vol. 36. – No. 2. – P. 451-461.

237. Babu G. P., Murty M. N. Simulated annealing for selecting optimal initial seeds in the *k*-means algorithm //Indian Journal of Pure and Applied Mathematics. – 1994. – Vol. 25. – No. 1-2. – P. 85-94.

238. Liang J. et al. A Faster $ k $-means++ Algorithm //arXiv preprint arXiv:2211.15118. – 2022.

239. Bahmani B. et al. Scalable *k*-means++ //arXiv preprint arXiv:1203.6402. – 2012.

240. Cup K. D. D. [Электронный ресурс]. - 1999. - URL: http://kdd. ics. uci. edu/databases/kddcup99/kddcup99 (дата обращения 30.07.2024).

241. Amsaleg, Laurent, and Hervé Jégou. "Datasets for approximate nearest neighbor search, 2010." [Электронный ресурс]. - 2010. - URL: http://corpus-texmex. irisa. Fr (дата обращения 30.07.2024).

242. Zhang T., Ramakrishnan R., Livny M. BIRCH: A new data clustering algorithm and its applications //Data mining and knowledge discovery. – 1997. – Vol. 1. – P. 141-182.

243. UCI Machine Learning Repository. Available online [Электронный ресурс]. - 1987. - URL: https://archive. ics. uci. edu/ (дата обращения 21.02.2024).

244. Rozhnov I. P., Orlov V. I., Kazakovtsev L. A. VNS-based algorithms for the centroid-based clustering problem //Facta Universitatis, Series: Mathematics and Informatics. – 2019. – P. 957-972.

245. Казаковцев Л. А. Детерминированный алгоритм для задачи *k*-средних и *k*-медоид //Системы управления и информационные технологии. – 2015. – №. 1. – С. 95-99.

ПРИЛОЖЕНИЕ А. АКТ О ВНЕДРЕНИИ РЕЗУЛЬТАТОВ ДИССЕРТАЦИИ

АКТ

о внедрении результатов диссертационного исследования

Ахматшина Фарида Галиулловича

Настоящим актом подтверждается, что ООО «Центр вычислительных технологий» в составе системы управления дисковыми хранилищами был успешно применен разработанный Ахматшиным Ф.Г. алгоритм автоматической группировки блоков данных, основанный на алгоритме k-средних совместно с алгоритмом хеширования с учетом местоположения LSH, позволяющий повысить компрессию архивируемых данных на на благодаря упорядочению блоков данных по схожести.

Применение нового алгоритма, разработанного в рамках диссертационного исследования соискателя ученой степени кандидата технических наук Ахматшина Фарида Галиуловича, обеспечил увеличение эффективности сжатия данных в системах хранения данных в среднем на 1.8 %.

Директор
ООО «Центр
вычислительных технологий» Д.С.Переверзев