НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

НАУЧНЫЙ ДОКЛАД

по результатам подготовленной научно-квалификационной работы (диссертации)

Пономаренко Александр Александрович

Направление подготовки 09.06.01 Информатика и вычислительная техника
Профиль (направленность) программы
05.13.18 – Математическое моделирование, численные методы и комплексы программ
Аспирантская школа по компьютерным наукам

Аспирант_____/Пономаренко А.А./

подпись

Научный руководитель_____/Крылов В.В. /

подпись

Директор Аспирантской школы_____/Объедков С.А. /

подпись

Нижний Новгород – 2018

ОБЩАЯ ХАРАКТЕРИСТИКА РАБОТЫ

Актуальность исследования

В информационно-вычислительных системах время доступа к информации в хранилище данных во многом определяет скорость работы всей системы. Поэтому алгоритмы поиска данных в хранилищах должны поддерживать высокий уровень параллелизма и быть вычислительно эффективными. Технологии больших данных требуют от систем оставаться эффективными как с увеличением объёма хранимых данных, так и с возрастающей скоростью их поступления. Для многих приложений, особенно связанных с алгоритмами анализа данных и машинного обучения, практически весь объем хранилища должен поддерживаться в активном состоянии и обеспечивать быстрое нахождение и чтение данных, как правило по принципу похожести с предъявляемыми образцами – запросами. Такой вид процессов в хранилищах называют поиском данных на основе похожести (близости). Для этого на множестве всевозможных хранимых данных D (domain) вводят функцию расстояния $\sigma: D \times D \longrightarrow R_{[0,+\infty)}$, отражающую семантику предметной области. Тогда задача построения эффективного хранилища данных может быть сведена к задаче построения структуры данных S над конечным множеством $X \subset D$, позволяющей вычислительно эффективного производить поиск данных «по похожести», другими словами структура данных S должна позволять эффективно решать задачу поиска ближайшего соседа, то есть позволять эффективно вычислять функцию $\underset{x \in X}{\operatorname{argmin}}(\sigma(q,x))$, где $q \in D$ некоторый информационный объект –запрос.

Важный подкласс таких информационных хранилищ – это хранилища «ключ-значение». Такие системы главным образом поддерживают две операции: поиск и хранение данных по заданному ключу используя точное совпадение. На хранилищах «ключ-значение» основаны многие популярные Web-сервисы. Например, система обмена сообщениями Facebook и хранилище сообщений системы SoundCloud используют Apache Cassandra; многие продукты Google – такие как Gmail, YouTube, Google Maps – основаны на системе BigTable, которая также является хранилищем «ключ-значение».

Одним из возможных подходов к построению масштабируемых хранилищ и в частности хранилищ «ключ-значение», является использование и построение структурированных Peer-to-Peer (P2P-сетей). Посредством транспортного уровня, каждый узел P2P сети имеет возможность передавать и принимать сообщение напрямую от любого другого участника сети (от этого и происходит название Peer-to-Peer). Как правило, в качестве транспортного уровня выступает TCP/IP сеть.

Каждый узел P2P сети может быть вовлечён в процесс хранения и поиска, а также любой узел может выступать в качестве инициатора поиска. В этом заключается главная особенность структурированных P2P сетей, то есть в распределённой архитектуре, в которой отсутствует какой-либо центральный элемент, ответственный за координацию работы всей системы.

Каждый узел хранит информацию только о небольшом числе узлов в сравнении с общим числом узлов в сети, тем не менее, сеть строится таким образом, что операция поиска имеет логарифмическую ($\sim log(n)$) вычислительную сложность относительно общего числа узлов n. Данная особенность позволяет создавать распределенные системы хранения, имеющие возможность масштабироваться, как по нагрузке, так и по вместимости.

Теоретическим аспектам построения масштабируемых хранилищ данных на основе P2P сетей посвящены работы J. Klainberg, A-M. Kermarrec, O. Beaumont, I. Stoica, M. Ripeanu, P. Maymounkov, A. Rowstron и других специалистов.

Известные в настоящее время алгоритмы поиска данных, применяемые для распределенных масштабируемых хранилищ, ограничены поиском на точное совпадение или же поиском ближайшего соседа в Евклидовых пространствах. В свою очередь, алгоритмам поиска в более общем случае – метрических пространствах посвящены многочисленные работы D. Knut, P. Zezula, S. Arya, E. Chavez, G. Navarro, G. Amato, Ю. Лифшиц, А. Савченко, А. Бабенко и других специалистов. Однако, существующие методы поиска в метрических пространствах изначально разрабатывались без учёта требования эффективной работы в условиях распределенного оборудования и как правило основываются на структурах имеющих вид дерева или ассоциативного массива, которые в свою очередь, имеют присущие им проблемы масштабирования.

Таким образом разработка алгоритмов поиска данных в метрических пространствах для распределённых масштабируемых хранилищ является актуальной задачей.

Целью диссертационной работы является исследование и разработка структур для хранения данных в масштабируемых хранилищах и алгоритмов эффективного поиска данных методом ближайшего соседа, обладающих следующими свойствами:

1. Масштабируемость по количеству информационных объектов
2. Масштабируемость производительности за счёт добавления вычислительных узлов
3. Способность работать в условиях распределенного оборудования
4. Независимость от формы представления информационных объектов

Для достижения поставленной цели были решены следующие ***задачи диссертационной работы:***

1. Разработать распределённый алгоритм построения графа с навигационными свойствами тесного мира, не использующий векторное представление данных
2. Провести исследование формируемых графов (диаметр, коэффициент кластеризации, распределение степеней вершин).
3. Разработать алгоритм улучшения навигационных свойств графа

4. Разработать алгоритм поиска k-ближайших соседей в графе со свойством навигационного тесного мира
5. Провести эмпирический сравнительный анализ разработанных алгоритмов с наиболее известными методами поиска ближайшего соседа.
6. Разработать математическую модель оптимальной конфигурации рёбер графа для поиска ближайшего соседа жадным алгоритмом
7. Найти точные и эвристические решения предложенной математической модели для некоторых частных случаев целочисленной решётки.
8. Разработать программную реализацию предложенных алгоритмов на языке программирования Java.

Объектом исследования являются масштабируемые хранилища данных.

Предметом исследования являются структуры данных для построения распределенных хранилищ и эффективные алгоритмы поиска информации в таких хранилищах.

Область исследования

В части исследования и разработки структур данных и алгоритмов, для масштабируемых хранилищ, работа соответствует пункту 2 паспорта специальности 05.13.17 «Теоретические основы информатики» - Исследование информационных структур, разработка и анализ моделей информационных процессов и структур.

Разработка алгоритмов поиска информации соответствует также пункту 9 специальности 05.13.17. – «Разработка новых интернет-технологий, включая средства поиска, анализа и фильтрации информации, средства приобретения знаний и создания онтологии, средства интеллектуализации бизнес-процессов».

Методы исследования

Для решения поставленных задач использовались математические модели теории графов, теории случайных графов, методы дискретной оптимизации, численные эксперименты

Научная новизна работы:

1. Разработаны и исследованы алгоритмы построения структур данных для распределённых масштабируемых хранилищ, отличающиеся использованием графов со свойствами навигационного тесного мира, позволяющие эффективно осуществлять поиск ближайшего соседа в метрическом пространстве.
2. Предложен алгоритм модификации графа в реальном времени, позволяющий улучшать поисковые свойства путём обнаружение и устранения локальных минимумов, отличающийся использованием информации о поступающих запросах на этапе исполнения.

3. Разработан алгоритм, позволяющий производить поиск k-ближайших соседей в метрическом пространстве, использующий только вычисление меры близости между информационными объектами и не использующий векторное представление данных, в отличие от существующих алгоритмов.
4. Разработана математическая модель, позволяющая находить оптимальную конфигурацию рёбер графа, отличающаяся использованием в качестве целевой функции оценки вычислительной сложности алгоритма поиска ближайшего соседа GreedyWalk.

Достоверность и обоснованность результатов определяется использованием корректного математического аппарата и обеспечивается экспериментальной проверкой. Все исходные коды программ, используемые для получения результатов, приводимые в работе, находятся в свободном доступе.

Реализация и внедрение работы

Предложенные в диссертации алгоритмы были использованы в следующих разработанных при участии автора программных продуктах:

«Skoal – Система поиска химических соединений по структурному сходству». Подтверждено свидетельством о государственной регистрации программы для ЭВМ №2012619905.

«Cloud MSW – Система облачного хранения данных с возможностью поиска в метрическом пространстве». Подтверждено свидетельством о государственной регистрации программы для ЭВМ №2012612167.

«Non-Metric Space Library» - открытая свободно-распространяемая библиотека методов поиска ближайшего соседа доступная по адресу https://github.com/searchivarius/nmslib.

Результаты исследований внедрены в отчеты по выполнению научно-исследовательских работ при поддержки следующих грантов:

Грант Правительства Российской Федерации для государственной поддержки научных исследований, проводимых под руководством ведущих ученых в российских образовательных организациях высшего образования (договор № 11.G34.31.0057 от 21 октября 2011 г.). Подтверждено «Актом внедрения результатов исследований».

Грант РНФ 14-41-00039. Подтверждено «Актом внедрения результатов исследований».

Апробация результатов исследования

Основные положения диссертации были представлены и доложены на следующих научных конференциях и семинарах:

1. Workshop on Critical and collective effects in graphs and networks (Москва, Россия, 2016)
2. 8th International Conference on Similarity Search and Applications (Глазго, Великобритания, 2015).

3. XVI-я Байкальская международная школа-семинар "Методы оптимизации и их приложения" (о. Ольхон, Иркутская область, Россия, 2015).
4. The 4th International Conference on Network Analysis (Нижний Новгород, 2015).
5. 5th International Conference on Similarity Search and Applications (Торонто, Канада, 2012)
6. International Conference on Information and Communication Technologies and Applications ICTA 2011 (Орландо, Флорида, США, 2011).

Основные положения, выносимые на защиту

1. Алгоритмы построения графа MSWConstruction и ConstuctionByReparing
2. Алгоритм K-NNSearch для поиска k-ближайших соседей в метрическом пространстве
3. Алгоритм улучшения поисковых свойств графа на этапе исполнения запросов RepairByQuery
4. Математическая модель оптимальной конфигурации рёбер графа для поиска ближайшего соседа алгоритмом GreedyWalk.
5. Реализация алгоритмов на языке программирования Java.

Публикации результатов исследования

По теме диссертации опубликовано 16 работ, в том числе 6 работ в изданиях, рекомендованных ВАК [1-6], получено два свидетельства о государственной регистрации программ для ЭВМ [15-16].

Личный вклад автора

Автору принадлежит лично алгоритм построения графов со свойствами навигационного тесного мира MSWConstruction, алгоритм поиска k-ближайших соседей K-NNSearch, а также алгоритм улучшения поисковых свойств структуры на этапе исполнения запросов RepairByQuery. При его участии была разработана модель оптимальной структуры для вычислительно эффективного поиска. Им лично выполнены программные разработки алгоритмов и проведены основные экспериментальные исследования. Им лично и при его участии выполнена подготовка публикаций по представленной работе и докладов на научных конференциях и семинарах.

Структура и объем работы

Диссертация состоит из введения, пяти глав, заключения, списка использованной литературы из 69 наименований. Общий объем работы 135 страницы текста, содержащего 58 рисунков (включая описания алгоритмов)

КРАТКОЕ СОДЕРЖАНИЕ РАБОТЫ

Во введении обосновывается актуальность диссертационной работы, формулируются основные цели и задачи исследования

В первой главе в первом разделе анализируются известные алгоритмы построения эффективных структур для поиска данных на точное совпадение т.е. имеющие функциональность поиска «ключ-значение». Во втором разделе разбираются алгоритмы, имеющие более сильные поисковые возможности, по сравнению с поиском на точное совпадение. Рассматриваются алгоритмы позволяющие строить структуры с возможностью поиска ближайшего соседа в Евклидовых пространствах.

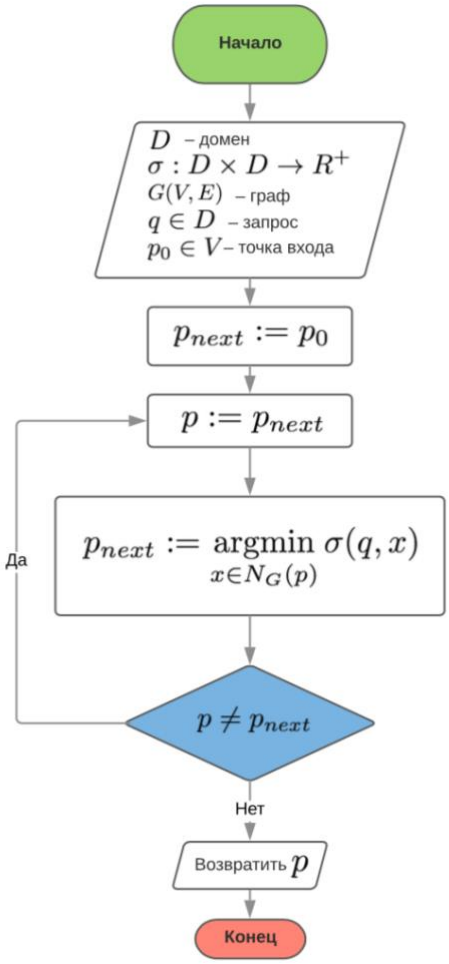


Рис 1. Блок-схема алгоритма жадного поиска GreedyWalk

Во второй главе предлагаются новые алгоритмы поиска и конструирования структуры данных имеющей вид графа $G(V, E)$, с возможностью эффективного поиска k-ближайших соседей в метрическом пространстве $M(D, \sigma)$ без использования векторного представления данных. Алгоритм поиска k-ближайших строится на основе алгоритма GreedyWalk, идея которого во многом схожа с широко известным методом градиентного спуска. (Блок-схема алгоритма GreedyWalk приведена на рис. 1).

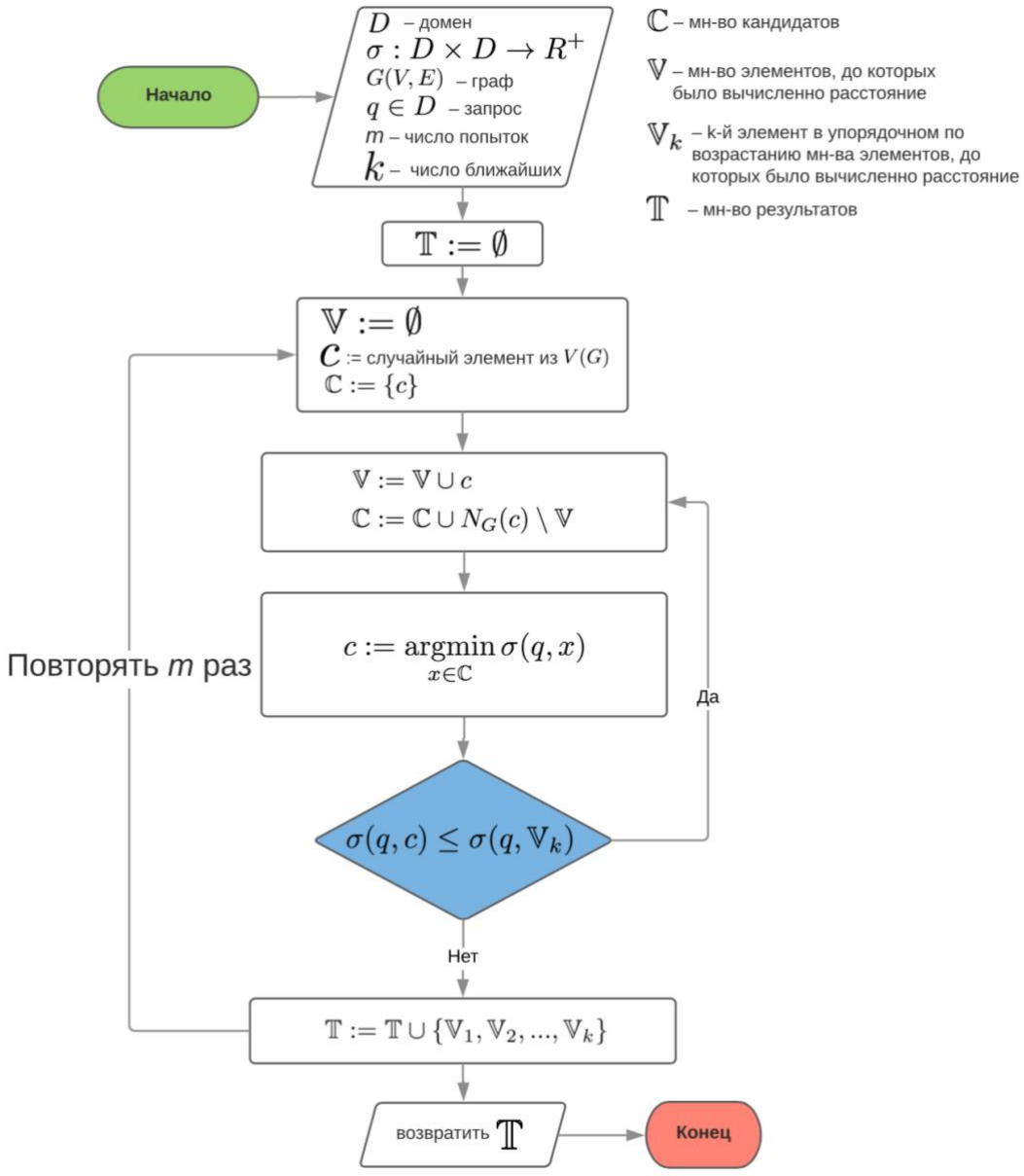


Рис. 2. Блок-схема алгоритма поиска k-ближайших соседей в графе G(V,E). (Алгоритм K-NNSearch)

Целью алгоритма GreedyWalk является обнаружение вершины в графе $G(V,E)$ ближайшей к запросу q. Алгоритм принимает на вход два параметра: запрос $q \in D$ и вершину $p_0 \in V$, с которой начинается поиск, также называемой точкой входа. Вершину в которой останавливается жадный алгоритм будем называть локальным минимумом. Алгоритм работает итеративно. На каждой итерации, называемой шагом, вычисляется расстояние между запросом q и всеми вершинами из окрестности $N_G(p)$ вершины p ($N_G(p) = \{x \in V_G : (p,x) \in E_G\}$). Из окрестности $N_G(p)$ выбирается вершина p_{next} расстояние от запроса q до которой минимально. В случае если $N_G(p)$ содержит вершину, которая ближе к q, чем p, итерация

повторяется присваивая p равной ближайшей вершине к q из множества $N_G(p)$. В другом случае, когда в окрестности p $N_G(p)$ нет вершины, которая была бы ближе к запросу, чем p, поиск останавливается и p возвращается в качестве искомой вершины.

Алгоритм поиска k-ближайших соседей приведён на рисунке 2. В нём поддерживается три множества $\mathbb{T}$, $\mathbb{V}$ и $\mathbb{C}$. В упорядоченном множество результатов $\mathbb{T}$ аккумулируются известные ближайшие вершины к запросу; в множестве «кандидатов» $\mathbb{C}$ хранятся вершины до которых было вычислено расстояние s. Множество посещённых вершин $\mathbb{V}$ (visitedSet) – хранит вершины, чьи окрестности использовалось для расширения множества $\mathbb{C}$. На очередной итерации вычисляется расстояние до всех элементов из окрестности вершины c. Далее из множества кандидатов $\mathbb{C}$ алгоритм выбирает элемент ближайшую к запросу q и помещает его в переменную c, после чего итерация повторяется. Критерием остановки служит условие того, что элемент c находится на расстояние от запроса, не дальше, чем k-й ближайший элемент к q из множества $\mathbb{V}$. Такой более «слабый» критерий остановки позволяет алгоритму выходить из локальных минимумов и помогает сформировать множество результатов $\mathbb{T}$ состоящее минимум из k элементов. Так же в алгоритме используется повторные поиски от случайных вершин для увеличения точности (параметр m).

Алгоритм построения структуры MSWConstruction

Алгоритм принимает на вход четыре аргумента: функцию расстояния σ, конечное множество объектов X, над которым необходимо построить структуру и два параметра - целые положительные числа w и u. На каждой итерации алгоритм выбирает из множества X произвольный элемент c. Далее для элемента c алгоритм формирует окрестность $N(c)$ из u ближайших элементов к элементу c, среди тех, которые есть на данный момент времени в структуре. Для этого используется алгоритм K-NNSearch, в который передаются параметры $k=u$ и $m=w$. После чего в граф G добавляются рёбра соединяющие элемент c и элементы из множества $N(c)$. Алгоритм работает до тех пор, пока все элементы из множества X не будут добавленных в структуру. Блок-схема алгоритма приведена на рисунке 3.

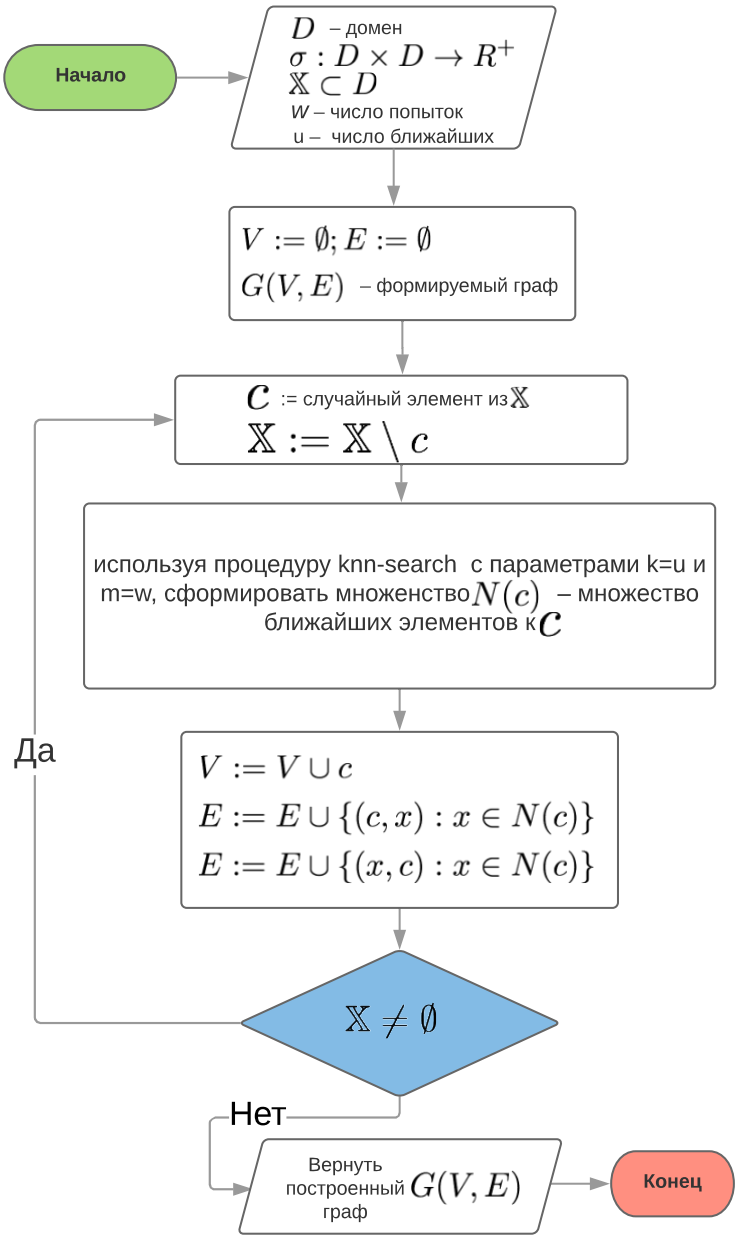


Рис 3. Алгоритм построения структуры MSWConstruction

Алгоритм добавления основанный на устранении локальных минимумов InsertByRepairing

Существование локальных минимумов в сети, является основной проблемой для корректной работы алгоритмов поиска основанных на идее градиентного спуска. С этой проблемой можно бороться различными способами. Один возможных вариантов – пытаться устранять локальные минимумы явным образом, в момент их обнаружения. Критерий того, что некоторый элемент $a \in V_G$ является локальным минимум в графе G относительно некоторого произвольного элемента $b \in D$ - это факт того, что в $N_G(a)$ нет элемента, который был ближе к b, чем a. Обнаружить локальный минимум можно на любом этапе существования структуры, то есть как на этапе добавления элементов в структуру, так и на этапе исполнения запросов.

```
GetLocalMinimums(q ∈ D, G(V,E), t ∈ N)
01 L ← {}; τ' ← 0
02 while τ' < τ do
03   v_start ← Random(V) //put to v_start random vertex from V
04   v, P ← Greedy_Walk(q, G, v_start)
05   if v ∉ L then τ' ← 0; L ← L ∪ v
08   else τ' ← τ'+1
09 end while
11 return L, P
```

Рис. 4 Алгоритм нахождения локальных минимумов. Алгоритм возвращает множество локальных минимумов L, и множество элементов P до которых было вычислено расстояние в ходе работы алгоритма

На рисунке 4 представлен псевдокод алгоритма для нахождения локальных минимумов. Этот алгоритм так же опирается на идею серии запусков жадного алгоритма от произвольной вершины графа $G(V,E)$.

Для того чтобы устранить локальный минимум a относительно элемента b достаточно добавить ребро, соединяющее локальный минимум с элементов расстояние до которого от b меньше, чем до элемента a.

Добавляемый элемент в структуру можно так же рассматривать в качестве локального минимума, поскольку изолированная вершина полностью удовлетворяет вышеназванному критерию. Это обстоятельство делает возможным определить процедуру добавления (вставки) в структуру нового элемента, как процедуру устранения локального минимума (см. рисунок 5).

```
InsertByRepairing(x ∈ X, G(V,E), t ∈ N)
01  V ← V ∪ x; P ← {}
02  L, P ← GetLocalMinimums(x, G, t)
03  P ← P ∪ x
04  for each z ∈ L do
05    P' ← {y ∈ P : d(y,x) < d(z,x)}
06    x' ← arg min_{y ∈ P'} (d(z,y))
07    E ← E ∪ (x',z) ∪ (z,x')
08  end for
```

Рис. 5 Алгоритм добавления нового элемента использующий идею устранения локальных минимумов

Алгоритм **InsertByRepairing** принимает на вход три аргумента: добавляемый элемент $x \in X$, текущий граф $G(V,E)$, и натуральное число – параметр $t \in \mathbb{N}$. Сначала алгоритм получает список локальных минимумов в структуре относительно элемента x. Затем каждый локальный минимум $z \in L$ устраняется следующим образом. Среди множества элементов просмотренных в процессе поиска локальных минимумов, отбираются

элементы, которые могут быть использованы для устранения локального минимума z т.е. те элементы расстояние от которых до добавляемого элемента x меньше, чем от локального минимума z (строка 05). Локальный минимум z устраняется, путём добавления ребра от локального минимума, к элементу x', расстояние до которого минимально среди всех известных алгоритму элементов из множества P', позволяющих устранить локальный минимум z.

RepairByQuery($q \in D, G(V,E), t \in \mathbb{N}$)

01 $L,P \leftarrow$ Get_Local_Minimums(q, G, t)

02 $x \leftarrow \arg\min_{y \in L}(d(y,q))$

03 **for each** $z \in L \setminus x$ **do**

04 $P' \leftarrow \{y \in P : d(y,q) < d(z,q)\}$

05 $x' \leftarrow \arg\min_{y \in P'}(d(z,y))$

06 $E \leftarrow E \cup (x',z) \cup (z,x')$

07 **end for**

Рис. 6 Процедура улучшения поисковых свойств структуры путём устранения локальных минимумов на этапе исполнения поисковых запросов.

Процедура улучшения поисковых свойств структуры на этапе исполнения поисковых запросов

Обнаружить локальный минимум можно на любом этапе существования структуры, как на этапе добавления элементов в структуру, так и на этапе исполнения запросов. Пользуясь данным обстоятельством предлагается следующая процедура улучшения поисковых свойств структуры на этапе исполнения запросов RepairByQuery (см. рисунок 6). Процедура RepairByQuery аналогична процедуре InsertByRepairing с тем исключением, что локальные минимумы ищутся относительно запроса – некоторого элемента q из множества всевозможных объектов D.

В третьей главе эмпирические исследуются свойства структуры формируемой алгоритмами MSWConstrution и InsertByReparing. Производится анализ таких характеристик как, средняя длина кратчайшего пути между всеми парами вершин, диаметр графа, коэффициент кластеризации, средняя длинна пути пройденная жадным алгоритмом, устойчивость поисковых свойств к выпадению случайных узлов, точность поиска k-ближайших, а так же количество вычислений функции расстояния алгоритмом поиска. Устанавливается взаимосвязь этих характеристик и параметров алгоритмов.

Одной из важнейших характеристик формируемого графа с точки зрения поиска, является возможность базовым алгоритмом GreedyWalk находить вершину, ближайшую к заданному объекту q относительно функции расстояния σ. Эффективность алгоритма GreedyWalk зависит от количества итераций совершаемых до момента остановки; другими словами от длинны совершённого пути т.е. количества совершенных шагов, иначе –

«хопов». В особенности эта характеристика существенно влияет на скорость работы поиска, для случая, когда вершины располагаются на различных физических узлах и получение списка соседей у соседней вершины (совершение «шага») сопряжено с передачей данных по сети и требует относительно большого времени. На рисунке 7. приведен график распределения длинны пути совершаемым алгоритмом GreedyWalk до момента остановки в точке локального минимума (ряд «GW»). Так же на графике приведена информация о процентном соотношении количества успешных поисков имеющих определенную длину (ряд «SUCCEED»).

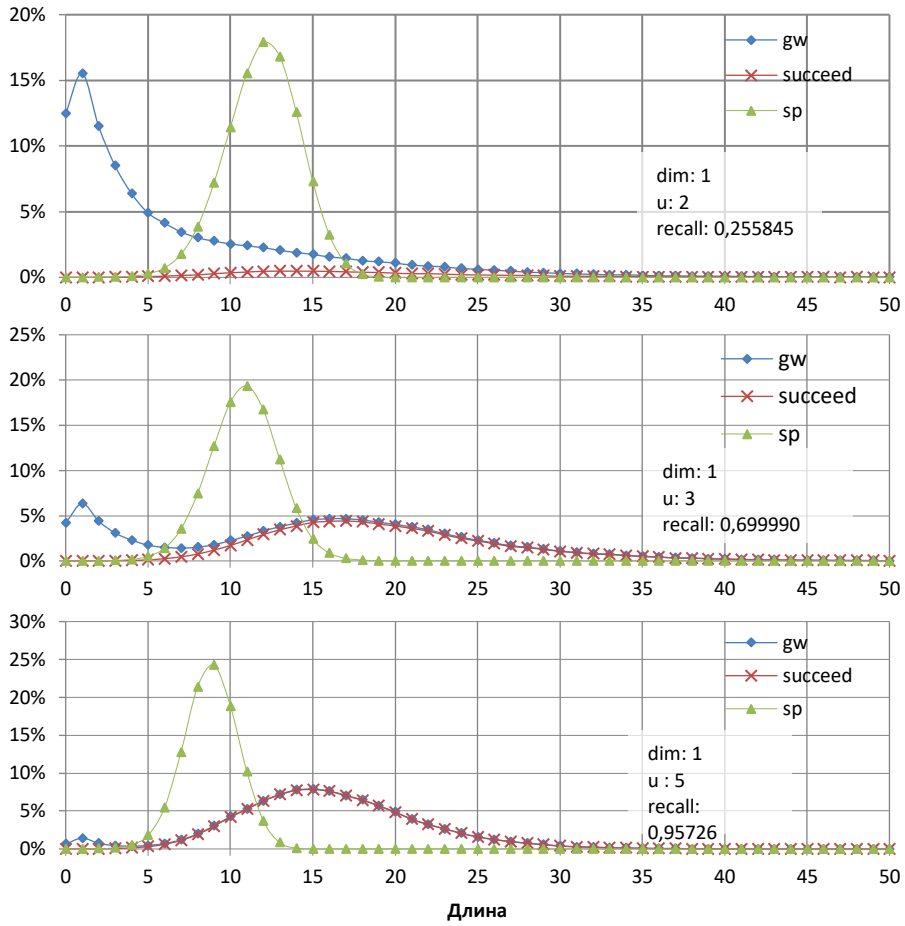


Рис. 7. Распределение длины пути жадного алгоритма до остановки в локальном минимуме – ряд «gw». Процент успешных поисков данной длинны – ряд «succeed» Распределение длин кратчайшего пути - ряд «sp». Входные данные – 100,000 точек равномерно распределенных в единичном отрезке. Параметры алгоритма добавления: w = 20; u=2;3;4;5 для графиков с верху вниз соответственно.

В совокупности алгоритмы MSWConstruction и K-NNSearch рассматриваются в качестве структуры данных для поиска ближайшего соседа, именуемой MSW (Metrized Small World). Структура данных MSW сравнивается с другими алгоритмами для поиска ближайшего соседа.

Зависимость вычислительной сложности алгоритма поиска от количества элементов в структуре MSW оценивалось в эксперименте с точками распределенными равномерно в единичном гиперкубе Евклидова пространства размерностью от 1 до 15 (параметр dim). Для каждого набора

параметров (dim, nn) эксперименты производились на 20, независимо собранных структурах. Значения усреднялись. Ось ординат соответствует квадратному корню от среднего числа просмотренных элементов. Как можно видеть из графиков на рисунке 8, точки ложатся в близи примой линии, из чего можно сделать предположение о зависимости $\sim\log(n)^2$.

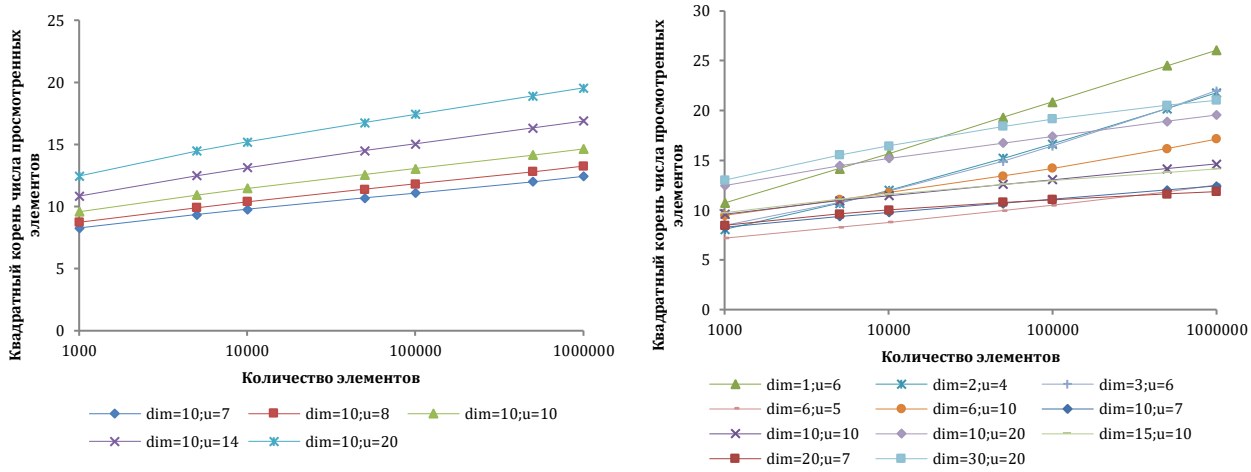


Рис. 8. Зависимость среднего числа элементов, до которых было вычислено расстояние жадным алгоритмом.

Для сравнения MSW с другими алгоритмами для поиска ближайшего соседа использовались следующие наборы входных данных.

1) *CoPhIR (L2):* коллекция 208-мерных векторов извлечённых из изображений в формате MPEG7. Вектора составлены из пяти различных свойств MPEG7.
2) *SIFT (L2):* часть коллекции данных TexMex. Содержит 1 миллион 128-мерных векторов. Каждый вектор соответствует дескрипторам извлеченных из изображений методом «Scale Invariant Feature Transformation» (SIFT)
3) *Википедия (cosine similarity):* коллекция состоящая из 3.2 миллионов векторов в разряженном формате. Каждый вектор хранит частоту встречаемости каждого слова на определённой странице Wikipedia. Вектора были построены при помощи библиотеки *gensim*. Вектора в этом наборе данных имеют сверхвысокую размерность – более 100 тысяч компонентов. Однако, вектора разряженные. В среднем, каждый вектор имеет только около 150 ненулевых компонент.
4) *Unif64 (L2):* синтетический набор 64-мерных векторов сгенерированных случайно с равномерным распределением в единичном гиперкубе.
5) *Final16, Final64 и Final256 (KL-дивергенция):* набор из 500 тысяч гистограмм заголовков полученных с помощью латентного размещения Дирихле (LDA). Числовой суффикс названия коллекции соответствует размерности, которая равна количеству LDA-заголовков (топиков).

Результаты экспериментов сравнения эффективности с 6-ю различными методами (Vantage Point Tree, 3 варианта Permutation Index,

Locality Sensitive Hashing, List of Clusters) для поиска 10-ближайших соседей с функцией расстояния L_2 на наборах данных CoPhIR, SIFT, Unif64 приведены на рисунке 9.

Результаты сравнения методов на данных Wikipedia представлены на рис. 10. График изображает во сколько раз методы работают быстрее полного перебора для разных значений точности.

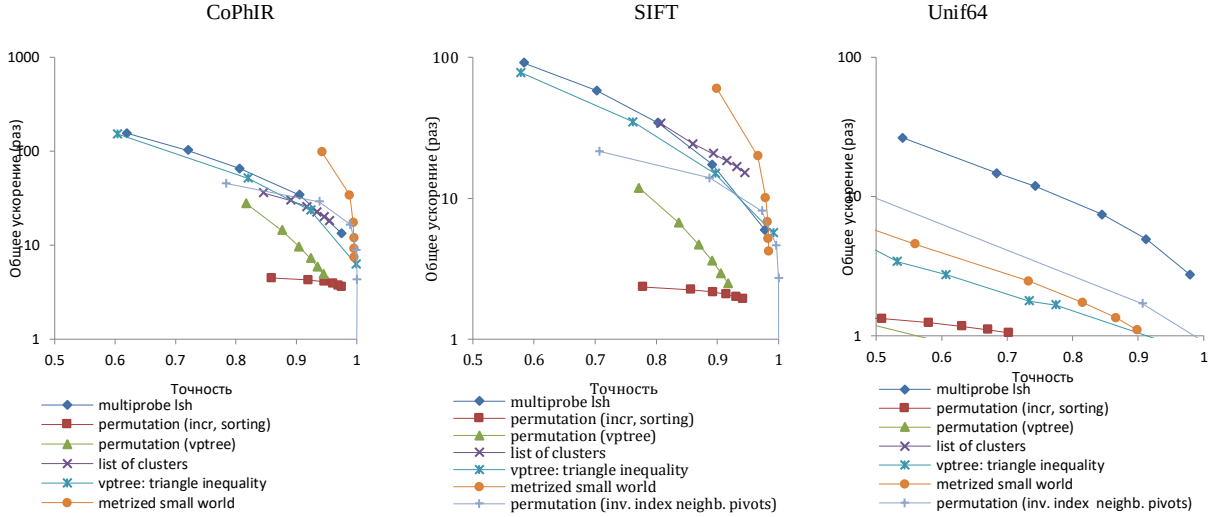
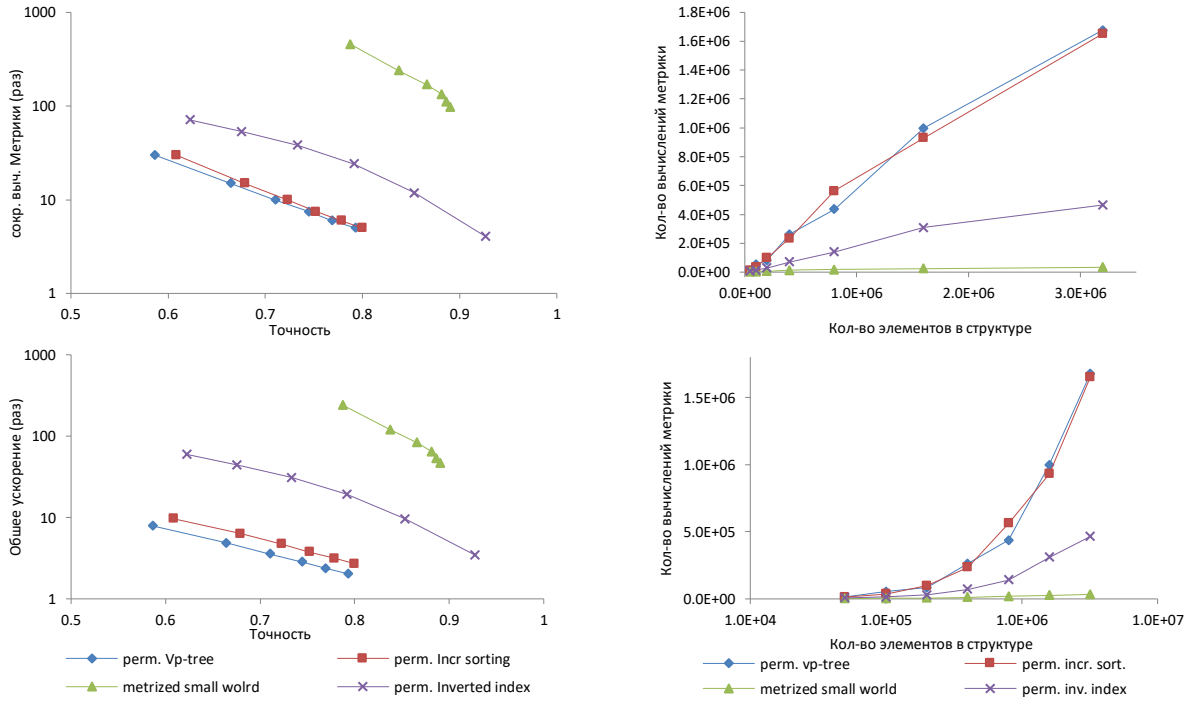


Рис. 9. Результаты экспериментов поиска 10-ближайших с функцией расстояния L_2. Графики из одного столбца соответствуют одному набору данных. CoPhIR - первый столбец, SIFT - второй, Unif64 - третий.



(а) Фиксированный набор состоящий из 3 200 000 элементов.

(б) Зависимость скорости поиска от числа проиндексированных элементов. Нижний график построен в логарифмическом масштабе по оси абсцисс.

Рис. 10. Результаты с набором данных Wikipedia. В качестве расстояния использовалась функция косинуса угла между векторами.

Как видно из рис. 10(б, верхний), все сравниваемые методы, основанные на перестановках, имеют зависимость от количества объектов

близкую к линейной. В то время как MSW демонстрирует зависимость близкую к логарифмической – зелёная прямая линия на графике 10(б, нижний), и при этом имеет большую точность (0.9 против 0.8). Таким образом, MSW имеет принципиально другую вычислительную сложность и превосходит в производительности сравниваемые методы тем больше, чем больше набор входных данных.

В четвёртой главе поднимается вопрос о существовании оптимальной структуры с точки зрения эффективности фиксированного алгоритма поиска. При этом учитывается требование на то, что алгоритм поиска должен всегда заканчиваться успехом, начиная работу от произвольного узла сети, то есть в результате поиска должен быть обнаружен узел, являющийся ближайшим к заданному запросу. Под эффективностью алгоритма поиска понимается его вычислительная сложность. В качестве элементарной операции, вносящей основной вклад во время работы алгоритма и в создаваемую нагрузку на систему в целом, можно рассматривать две операции: операцию получения списка смежных вершин и операцию вычисления расстояния между любыми двумя объектами (вершинами) известными алгоритму на текущий момент.

Таким образом в главе демонстрируется идея существования для фиксированного алгоритма поиска A и для фиксированного множества объектов X с заданной функцией расстояния $\sigma: D \times D \to R^+$ оптимальной конфигурации рёбер графа с точки зрения вычислительной сложности алгоритма поиска A. В данном случае в качестве алгоритма поиска рассматривается алгоритм GreeeyWalk, а в качестве элементарной операции рассматривается операция вычисления функции расстояние.

Математическая модель задачи построения оптимальной структуры для поиска алгоритмом GreedyWalk описывается набором неравенств (1)-(9).

Переменные

$$x_{ij} = \begin{cases} 1, \text{ if edge } (i, j) \text{ belongs to the solution} \\ 0, \text{ otherwise} \end{cases} \quad (1)$$

$$y_{ij}^{k} = \begin{cases} 1, \text{ if vertex } k \text{ belongs to the greedy walk from } i \text{ to } j \\ 0, \text{ otherwise} \end{cases} \quad (2)$$

Целевая функция

$$\min \frac{1}{n} \sum_{i=1}^{n} \sum_{q \in D} O(i, j_q) f_q, \quad (3а)$$

$$\min \frac{1}{n} \sum_{i=1}^{n} \int_{D} O(i, j_q) f(q) dq, \quad (3б)$$

где $j_q = \arg \min_{j=1,n} d(j, q)$ (4)

$$O(i,j) = \left| \left\{ l \hat{\mathrm{I}}\ V : \$k\, x_{lk} = 1 \text{and } y_{ij}^{k} = 1 \right\} \right| \quad (5)$$

Ограничения

$$x_{ii} = 0 \quad \forall i \in V \quad (6)$$

$$y_{ij}^{i} = y_{ij}^{j} = 1 \quad \text{"}\ i, j \hat{\mathrm{I}}\ V \quad (7)$$

$$\sum_{k=1}^{n} x_{lk} y_{ij}^{k} \geq y_{ij}^{l} \quad \forall i, j, l \in V \quad (8)$$

$$l^{*} = \underset{l \hat{\mathrm{I}}\ V : x_{kl}=1}{\arg\min}(s(l,j)) \ \text{Þ} \ \ y_{ij}^{l^{*}} \ ^{3}\ y_{ij}^{k} \quad \text{"}\ i, j, k \hat{\mathrm{I}}\ V, j \ ^{1}\ i, k \ ^{1}\ j \quad (9)$$

Переменные x_{ij} (1) определяют матрицу смежности оптимального графа, который требуется найти. Переменные индикаторы y_{ij}^{k} (2) используются для подсчёта числа операций $O(i, j_q)$ выполняемых во время процедуры поиска от вершины i, вершины j_q, которая является ближайшей к запросу q (4). В данном случае это количество различных вершин, от запроса до которых было вычислено расстояние. Это число равно мощности объединения окрестности вершины k, для которых $y_{ij}^{k} = 1$ (5).

Так как требуется найти оптимальную структуру для всех возможных вариантов запросов и стартовых вершин, целевая функция выглядит как среднее число операций необходимое для того, чтобы достигнуть вершины (3а, 3б, 4, 5). Ограничение (6) гарантирует отсутствие циклов. В свою очередь ограничение (7) требует, что процедура GreedyWalk(i, j) начиналась в вершин i и заканчивалась в вершине j. Неравенство (8) связывает переменные x_{ij} и y_{ij}^{k}, при этом оно требует, чтобы алгоритм поиска проходил, через какую-либо вершину из окрестности вершины l, если он проходит через вершину l. Ограничение (9) описывает жадную стратегию алгоритма поиска: если вершина k принадлежит пути жадного алгоритма от вершины i до вершины j_q $(y_{ij}^{k} = 1)$, тогда вершина l^{*} ближе к q, чем любая другая вершина из окрестности l, и она тоже принадлежит жадному пути $(y_{ij_{\text{й}}}^{k} = 1)$.

Данная модель применима для произвольного метрического пространства. Далее в разделе 5.3 приведены результаты для частного случая, когда вершины совпадают с узлами регулярной целочисленной двумерной решётки с функцией расстояния L_1, L_2 и L_∞.

В пятой главе обсуждаются различные аспекты реализации предложенных алгоритмов и кода, с помощью которого проводились большинство численных экспериментов. Главной целью, которая учитывалась, при проектирование архитектуры классов данной реализации, было желание смоделировать условия работы распределенного

оборудования. Таких условий при которых алгоритмы не могли бы использовать информацию о произвольных частях графа, а опирались бы только на локальную информацию о структуре известной им части сети, т.е. только такого множества узлов, чьи идентификаторы стали известны в течении одной сессии работы того или иного испытуемого алгоритма, например, алгоритма поиска или добавления. Предполагается, что коммуникация осуществляется на основе какого-либо транспортного, протокола, например, tcp/ip. Зная идентификатор узла, любой другой узел используя транспортный протокол может направлять сообщения ему.

В реализации активно применялись шаблоны проектирования такие, как “Strategy”, “Factory”, для того, чтобы избежать дублирования кода, при написания окружения позволяющее производить численные эксперименты с различными версиями алгоритма, для различных пространств. Поэтому, в некотором смысле, данная реализация может рассматриваться в качестве фреймворка для исследования алгоритмов работающих в условиях распределённого оборудования.

Заключение

Настоящая работа посвящена алгоритмам построения распределенных хранилищ данных основанных на использовании сетевых структур и алгоритмам поиска в них.

Основными результатами работы являются:

1. Предложены алгоритмы организации данных в виде графа MSWConstruction, ConstructionByReparing, отличающиеся главным образом от ранее известных алгоритмов тем, что предложенные алгоритмы не используют векторное представление данных
2. Предложен алгоритм приближённого поиска k-ближайших соседей K-NNSearch, основанного на идее жадного направленного поиска и поиске с запретами
3. Исследование свойств графов формируемых предложенными алгоритмами над некоторыми конечными подмножествами метрического пространства показало, что при определённых параметрах, предложенные алгоритмы позволяют строить графы со свойствами навигационного тесного мира
4. Было установлено, что данные графы могут эффективно использоваться для поиска k ближайших соседей. В выдвинуто предположение, о том что вычислительная сложность алгоритма поиска k-ближайших k-NNSearch оценивается как $O(\log(n)^2)$
5. Произведено исследование предложенных алгоритмов, включающее большой сравнительный анализ эффективности алгоритмов с существующими алгоритмами поиска ближайшего соседа, продемонстрировавшее значительное превосходство предложенных алгоритмов.
6. Предложена математическая модель оптимальной структуры графа для поиска ближайшего соседа.

7. Найдены решения точные и эвристические решения предложенной модели оптимальной структуры графа для некоторых частный случаев целочисленной решетки.

Таким образом, учитывая полилогарифмическую сложность всех предложенных алгоритмов, и факт использования алгоритмами только локальной информации о структуре сети, это даёт основания полагать, что результаты данной диссертации, открывают возможность для создания распределённых систем со сложной поисковой функциональностью способных к расширению до масштабов всемирной паутины.

ПУБЛИКАЦИИ ПО ТЕМЕ ИССЛЕДОВАНИЯ

Статьи в рецензируемых изданиях, рекомендованных ВАК:

1. Ponomarenko, A. A. An overlay network for distributed exact and range search in one-dimensional space / A. A. Ponomarenko, Y. A. Malkov, A. A. Logvinov, V. V. Krylov //Бизнес-информатика. – 2016. – №. 1 (35). – С. 26-36.
2. Пономаренко, А. А. Структура со свойствами тесного мира для решения задачи поиска ближайшего соседа в метрическом пространстве / А. А. Пономаренко, Ю. А. Мальков, А. А. Логвинов, В. В. Крылов // Вестник Нижегородского университета им. Н.И. Лобачевского. 2012. № 5. С. 409-415.
3. Malkov, Y. A. Growing Homophilic Networks Are Natural Navigable Small Worlds / Y. A. Malkov, A. A. Ponomarenko // Plos One. 2016. Vol. 11. No. 6 doi (Scopus)
4. Ponomarenko, A. A. Query-Based Improvement Procedure and Self-Adaptive Graph Construction Algorithm for Approximate Nearest Neighbor Search / A. A. Ponomarenko // International Conference on Similarity Search and Applications. – Springer International Publishing, 2015. – С. 314-319. (Scopus)
5. Malkov, Y. A. Approximate nearest neighbor algorithm based on navigable small world graphs / Y. A. Malkov, A. A. Ponomarenko, V. V. Krylov, A. A. Logvinov // Information Systems. 2014. Vol. 45. No. DOI 10.1016/j.is.2013.10.006. P. 61-68. (Scopus)
6. Malkov, Y. A. Scalable Distributed Algorithm for Approximate Nearest Neighbor Search Problem in High Dimensional General Metric Spaces / Y. A. Malkov, A. A. Ponomarenko, V. V. Krylov, A. A. Logvinov // Lecture Notes in Computer Science. 2012. No. 7404. P. 132-147. (Scopus)

Статьи в индексируемых РИНЦ:

7. Пономаренко, А. А. Сравнительный анализ структур данных для приближенного поиска ближайшего соседа / А. А. Пономаренко, Н. С. Аврелин, Б. С. Найдан, Л. М. Бойцов // Алгоритмы, методы и системы обработки данных. 2015. Т. 4. № 33. С. 91-106.

Статьи в сборниках трудов конференций:

8. Ponomarenko, A. A. Comparative analysis of data structures for approximate nearest neighbor search / A. A. Ponomarenko, N. S. Avrelin, B. S. Naidan, L. M. Boytsov // Proceedings of Data Analytics. – 2014. – C. 125-130. ISBN: 978-1-61208-358-2 P. 125-13

9. Пономаренко, А. А. Организация быстрого поиска без индекса / А. А. Пономаренко // Труды 38-й конференции "Информационные технологии и системы - 2014". Н. Новгород . 2014.
10. Ponomarenko, A. A. Approximate Nearest Neighbor Search Small World Approach / A. A. Ponomarenko, Y. A. Malkov, A. A. Logvinov, V. V. Krylov // Proceedings of International Conference on Information and Communication Technologies and Applications (ICTA) 2011, Orlando, Florida, USA.
11. Krylov, V. V. Active database architecture for XML documents / V. V. Krylov, A. A. Logvinov, A. A. Ponomarenko, D. M. Ponomarev // Proceedings of the ISCA 23rd International Conference on Computer Applications in Industry and Engineering (CAINE), 2008.
12. Ponomarenko, A. A. Metrized Small World Approach for Nearest Neighbor Search / A. A. Ponomarenko, Y. A. Malkov., V. V. Krylov, A. A. Logvinov // Труды 4-ого Весеннего/летнего коллоквиума молодых исследователей в области программной инженерии (SYRCoSE 2010), 1-2 июня 2010 г. – Нижний Новгород, Россия / Ed. by A. Kamkin, A. Petrenko, A. Terekhov. Nizhny Novgorod , 2010. P. 151-156.
13. Ponomarenko, A. A Metrized small world properties data structure / A. A. Ponomarenko, V. V. Krylov, A. A. Logvinov, D. M. Ponomarev // Proceedings of the 17th International Conference on Software Engineering and Data Engineering (SEDE), Los Angeles, California, June 30 to July 2, 2008, P. 203-208.

Препринты:

14. Ponomarenko, A. A. A Model of Optimal Network Structure for Decentralized Nearest Neighbor Search / A. A. Ponomarenko, I. E. Utkina, M. V. Batsyn //arXiv preprint arXiv:1712.08437. – 2017.

Свидетельства:

15. «Skoal – Система поиска химических соединений по структурному сходству» – свидетельство о государственной регистрации программы для ЭВМ №2012619905. Авторы: Логвинов Андрей Александрович, Пономаренко Александр Александрович, Новикова Елена Сергеевна
16. «Cloud MSW – Система облачного хранения данных с возможностью поиска в метрическом пространстве» – свидетельство о государственной регистрации программы для ЭВМ №2012612167. Авторы: Логвинов Андрей Александрович, Пономаренко Александр Александрович