А.В. Волохов
Е.Г. Михайлова
Б.А. Новиков

Санкт-Петербургский государственный университет

Приближенное индексирование многомерных объектов

Постановка задачи

Данные – многомерные вектора

- (изображения, тексты, музыка и пр.)

- Ищем похожие данные – найти вектора, близкие к искомому в многомерном пространстве

Проклятие размерности

Есть много вариантов индексирования много мерного пространства, с ростом размерности любой поиск по индексу начинает вести себя хуже, чем последовательный просмотр.

Цели

- Поиск ближайших соседей в n-мерном пространстве
- Построение субоптимальной индексной структуры векторов (приближенная точность + нелинейный рост)
- Сравнение с вероятностным подходом

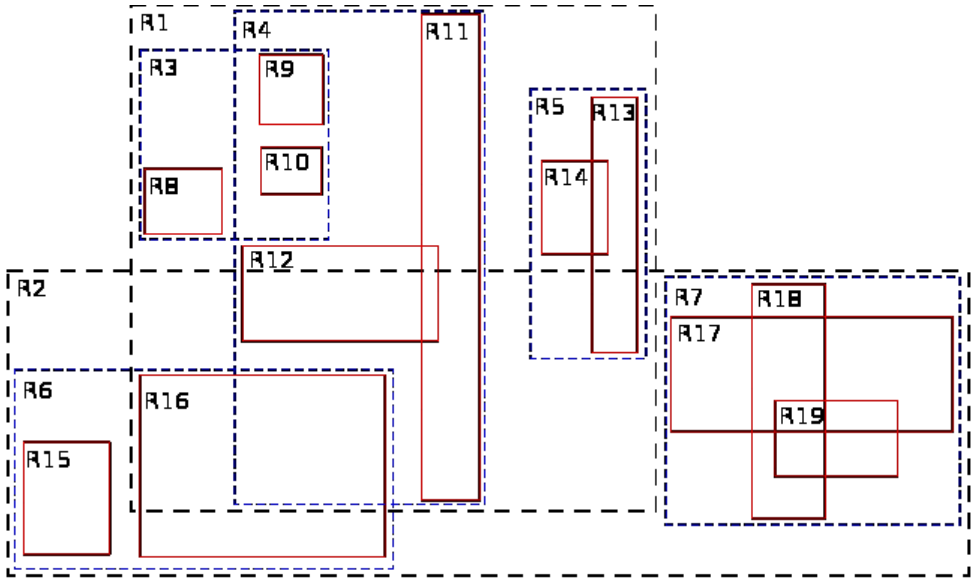
Родственные работы

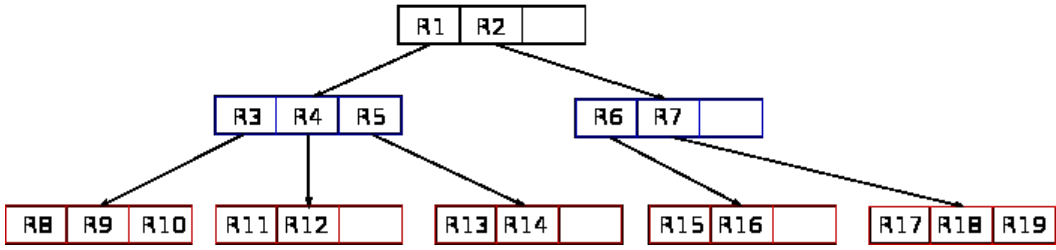
Классифицируем на подходы:

- *строить деревья*
- *ускорять последовательный просмотр*
- *уменьшать размерность*

R-деревья

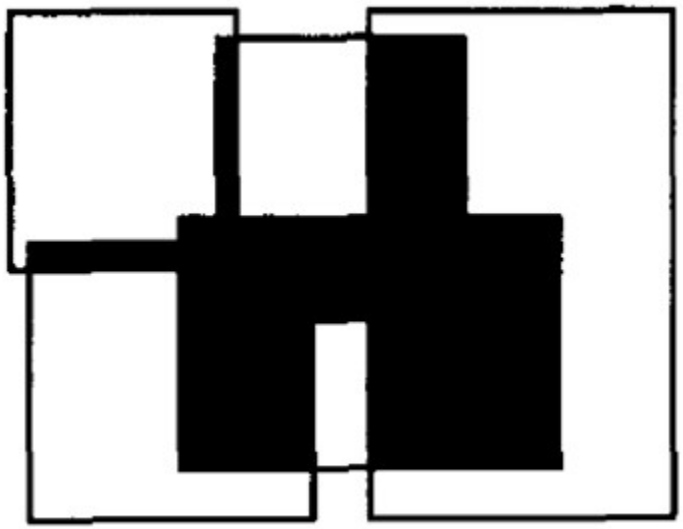
Структура для поиска протяженных объектов – окружаем объекты ограничивающим прямоугольником

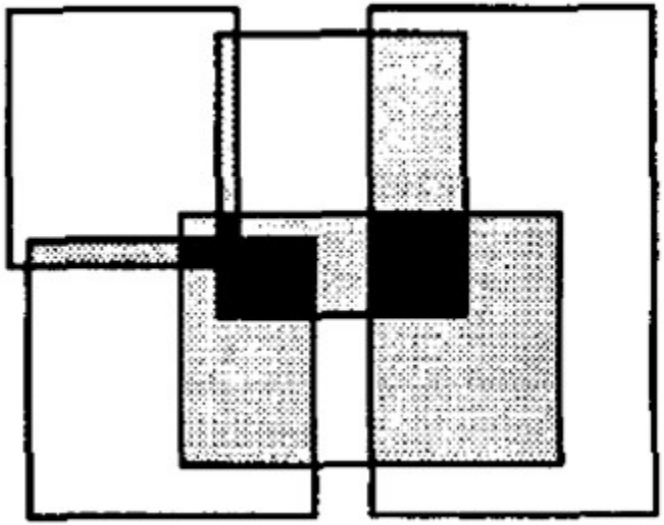




X-деревья

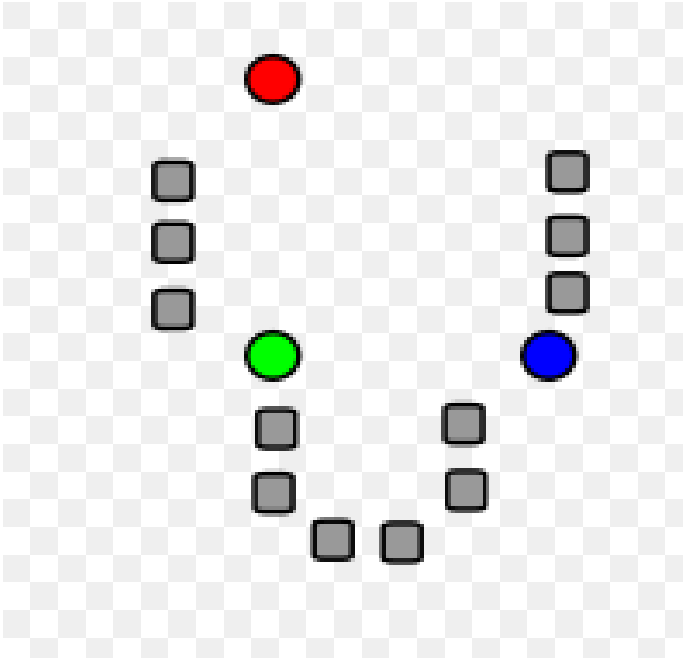
гибрид линейного просмотра и иерархической структуры:
для данных, которые при разбиении на кластеры будут давать много перекрытий, организуется линейный просмотр





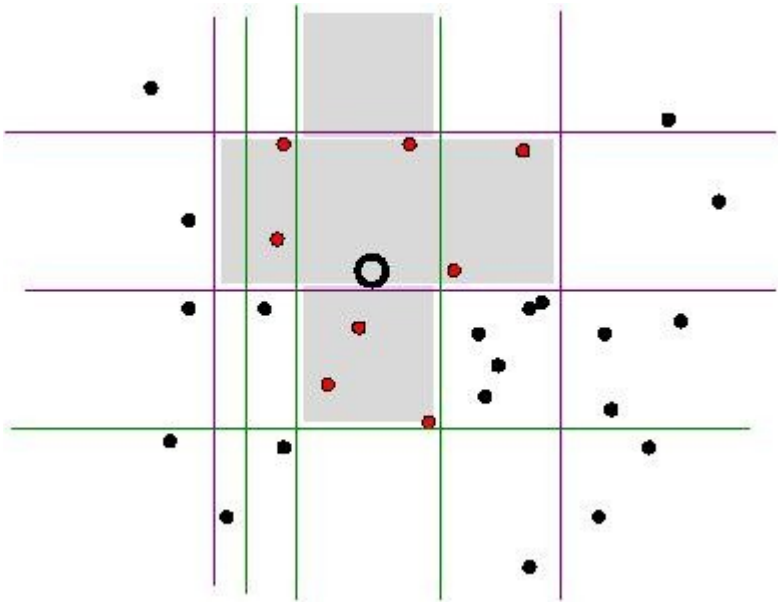
K-Means

Выбираем «центры масс», притягиваем объекты к ближайшему центру, пересчитываем центры масс



LSH - locality sensitive hashing

Для некоторых наборов координат подбираем функции хеширования, чтобы подобные элементы с большой вероятностью оказывались в одной корзине



Цели:

- Построить структуру для быстрого поиска n-мерных векторов, «похожих» на искомый
- Поиск должен давать приближенное решение, т.к. точное решение упрется в проклятие размерности.

Исходные данные

- Вектора произвольной размерности
- Функция расстояния (L1)

$$r(x, \ \ y) = \Sigma \left| x_i - y_i \right|$$

- Можно использовать также Евклидову метрику.
- Наш метод от метрики не зависит.
- Работает с обобщённым метрическим пространством.

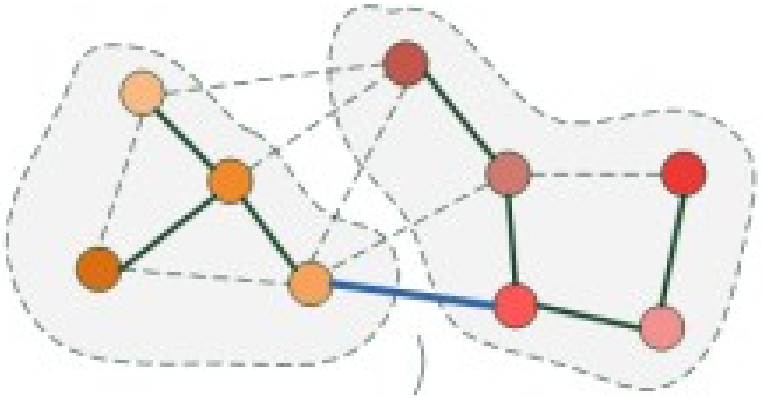
Алгоритмы

- Построение индексной структуры
- Поиск
- Добавление

Индексная структура – дерево

Построение минимального остовного дерева.

- Ограничение – количество дуг, входящих в каждую вершину.



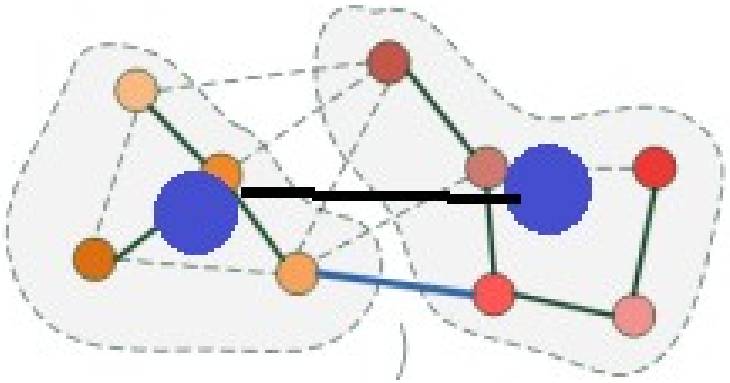
Индексная структура – компоненты

Деление дерева на компоненты связности путем удаления из дерева максимальных дуг.

- Для каждой компоненты связности вводилось ограничение на минимальное и максимальное количество элементов.

Индексная структура – уровни

- Нахождение центра компоненты связности.
- Построение следующего уровня индексной структуры.



Индексная структура – добавление элемента

- 1. Поиск ближайшего к добавляемому элемента (1-NN запрос)

- 2. Добавление элемента в кластер:
  Если размер кластера меньше максимально допустимого, то добавление в кластер
  Иначе — разбиение кластера и перекластеризация

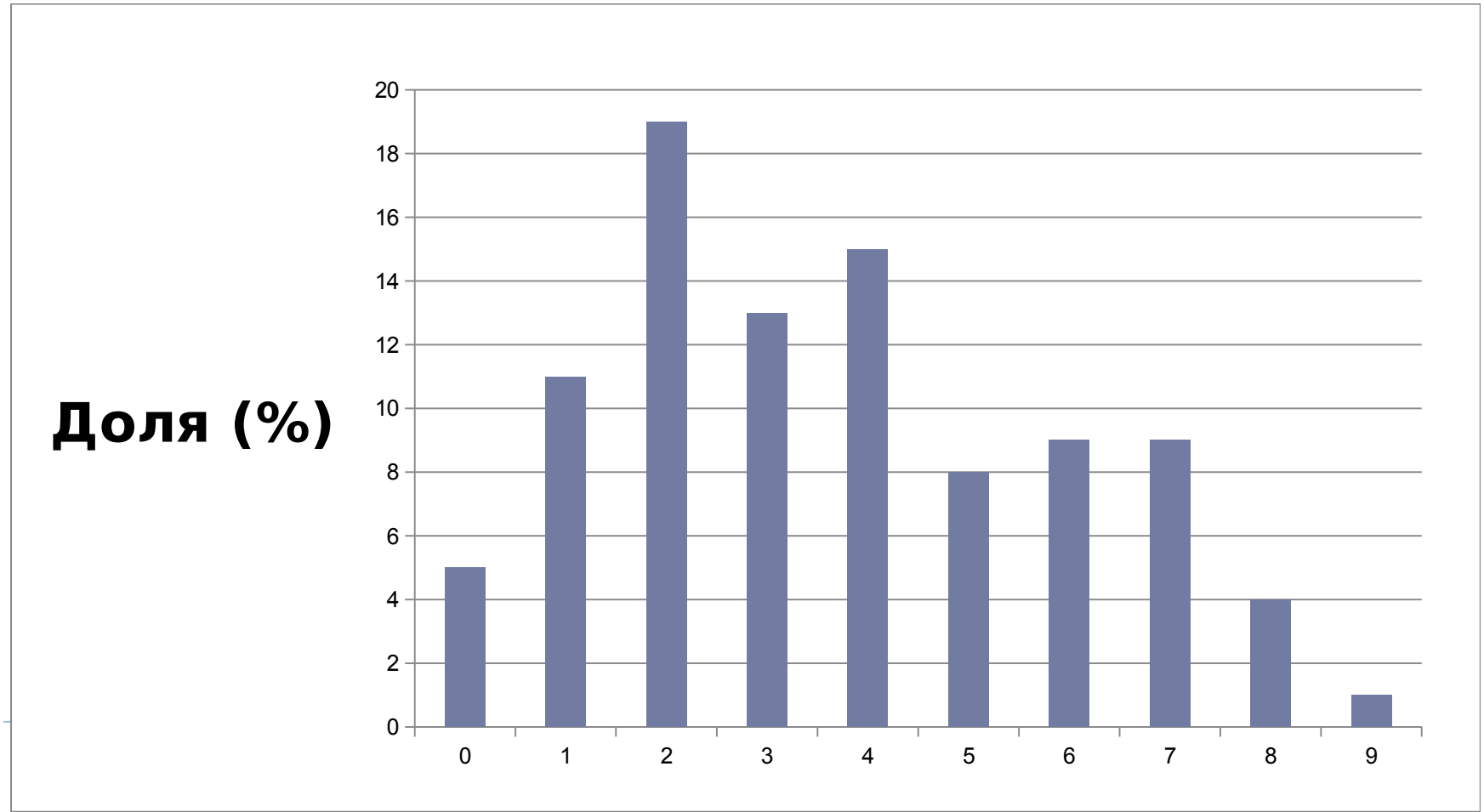
Поиск

- Задан вектор, ищем «похожие»
- Вычисляем расстояние от заданного вектора до центров компонент верхнего уровня
- Находим компоненту верхнего уровня, центр которой является ближайшим к искомому вектору.
- Если количество векторов, входящих в данную компоненту, меньше количества искомых, то ищем следующую ближайшую компоненту

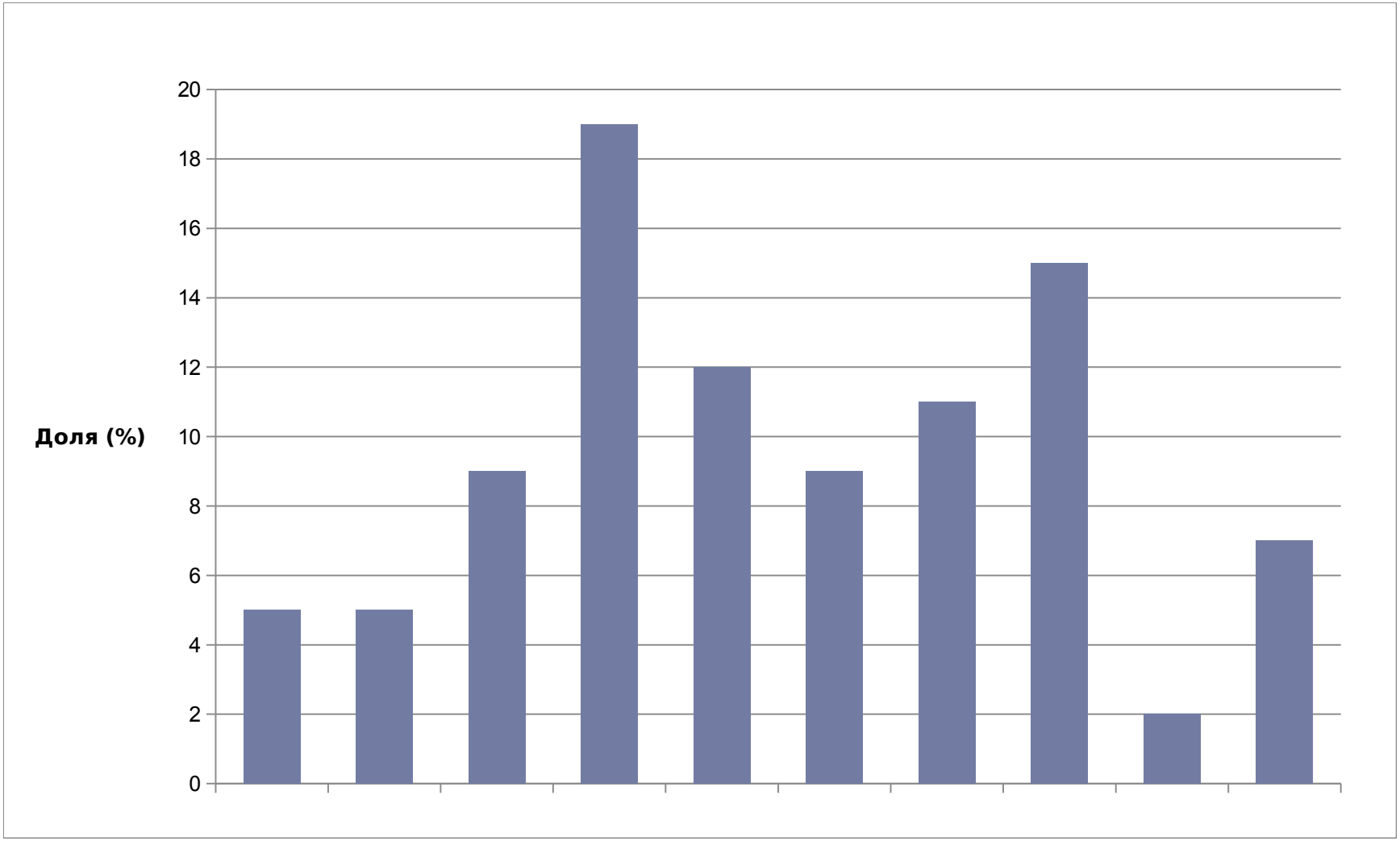
Эксперимент

- 30000 векторов, в каждом из которых до 40 координат.
- Вектора — представления характеристик изображений.
- Результаты показали, что если просматривать на нижнем уровне не одну, а несколько компонент, то точность поиска существенно возрастает

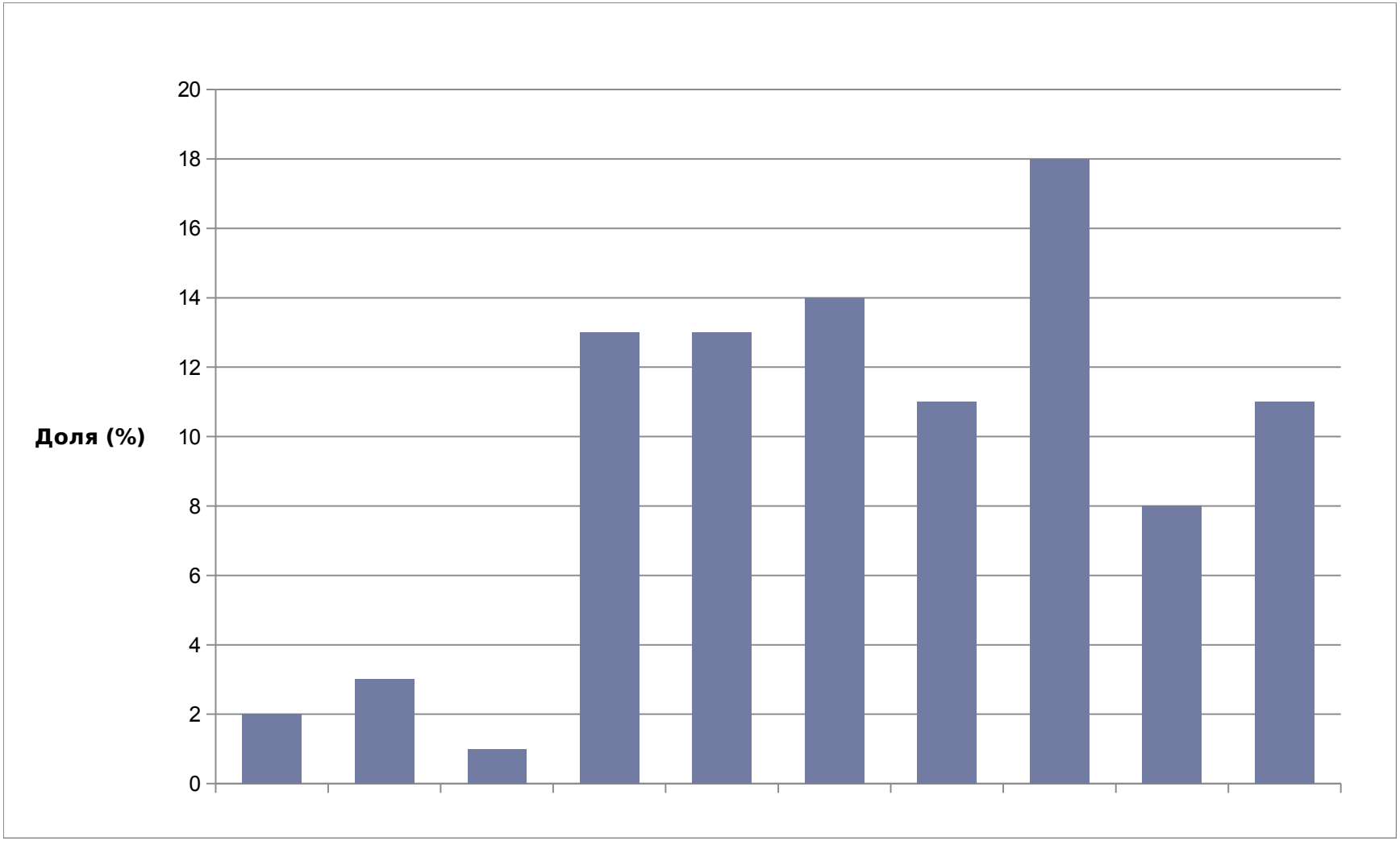
Точность приближенного поиска при выборе 1-й компоненты связности



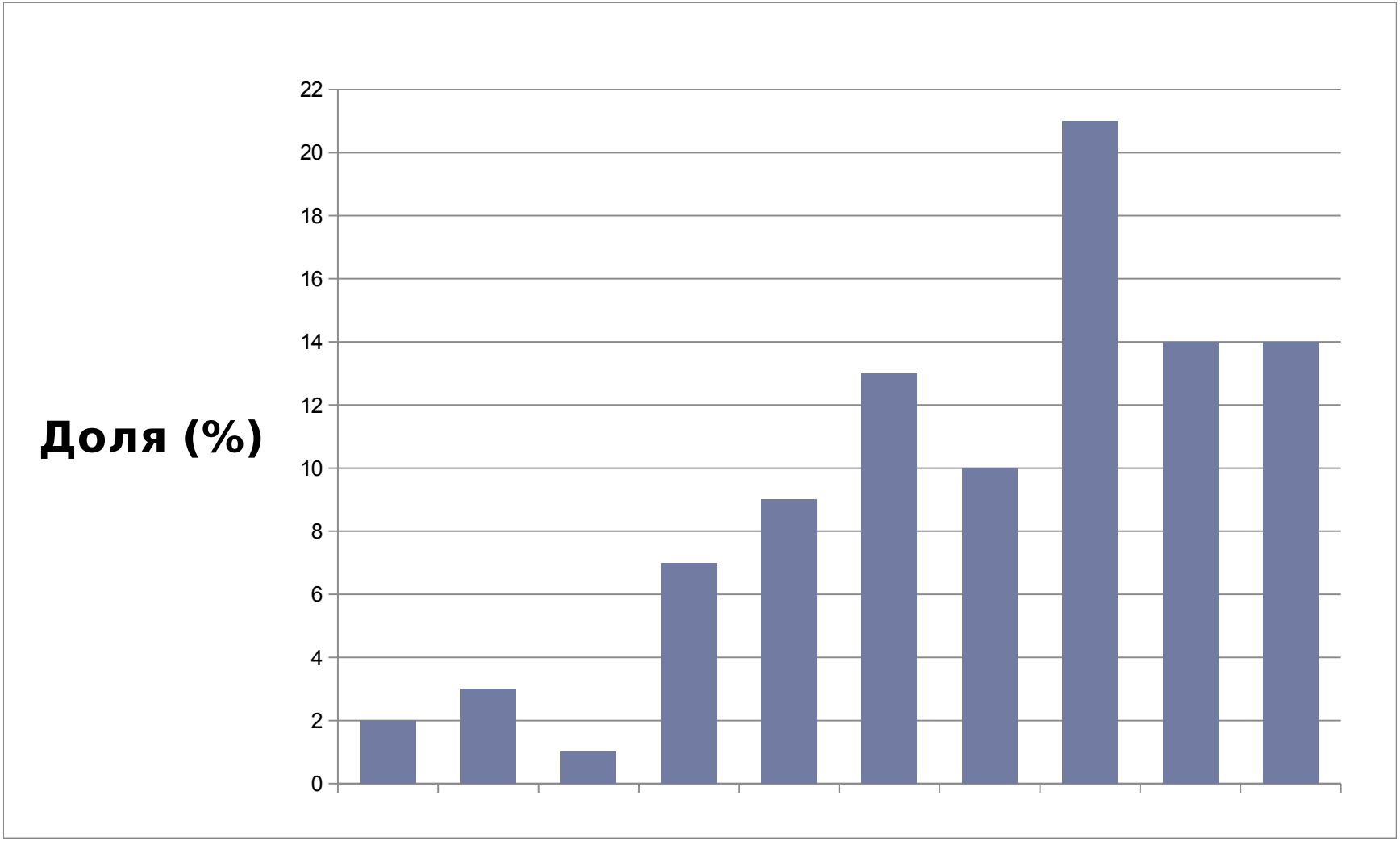
Точность приближенного поиска при выборе 2-х компонент связности



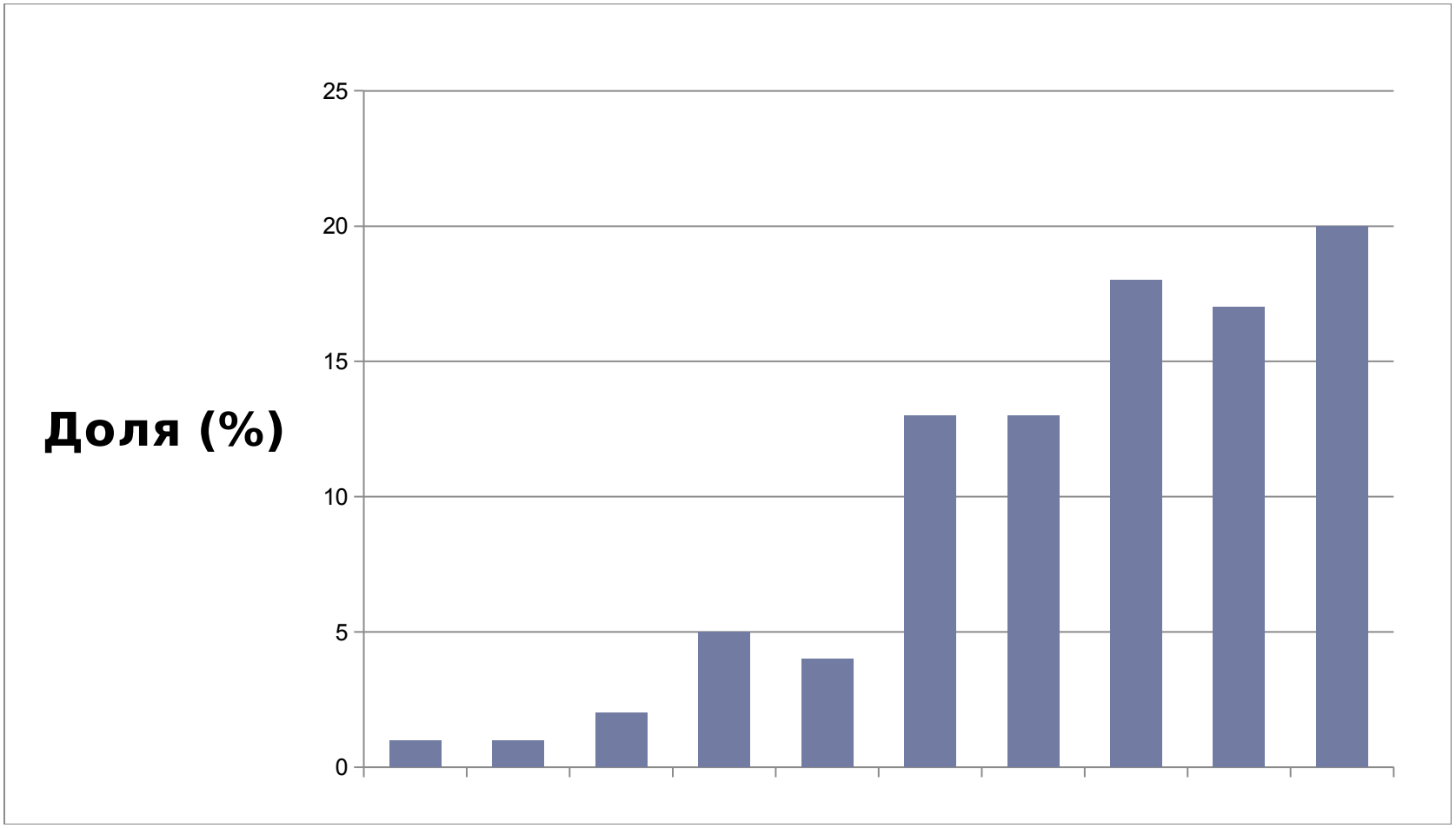
Точность приближенного поиска при выборе 3-х компонент связности



Точность приближенного поиска при выборе 4-х компонент связности



Точность приближенного поиска при выборе 5-и компонент связности



Сравненительный анализ

- Алгоритм пространственно чувствительного хеширования (LSH)
  - Время ответа на запрос
  - Точность возвращаемых ответов.

- Зависимость от типа запроса

LSH. Основные понятия

- Вероятность коллизии близких точек выше, чем вероятность коллизии далёких

- Семейство LSH функций, заданное параметрами

LSH. Построение

- Семейство пространственно-чувствительных функций:

$$\begin{cases} \|v_1 - v_2\| < r \rightarrow Pr_h\left(h(v_1) = h(v_2)\right) > P_1 \\ \|v_1 - v_2\| > c * r \rightarrow Pr_h\left(h(v_1) = h(v_2)\right) < P_2 \end{cases}$$

- L хеш функций, полученных из конактенации k функций семейства
- Минимизация времени ответа на запрос

$$L \geq \frac{\log 1/\delta}{-\log\left(1 - P_1^k\right)}$$

LSH. Поиск

- 1. Прохешировать точку запроса построенной функцией.
- 2. Получить все элементы из корзины, в которую попала точка.
- 3. Если точек не достаточно для удовлетворения запроса поиска ближайших соседей, выбрать следующую функцию и перейти к п. 1.
- 4. Отсортировать точки по расстоянию до точки запроса и вернуть необходимое количество в качестве ответа.

Заключение

- Построен алгоритм, который позволяет с удовлетворительную точность достаточно быстро находить многомерные вектора, близкие к искомому
- Проведен эксперимент для оценки точности поиска при помощи предложенной структуры
- Проведено сравнение с одним из существующих вероятностных подходов

Результаты: (1)

- Проанализированы следующие параметры алгоритма:
- Максимальное количество дуг, входящих в каждую вершину при построении остовного дерева – размерность пространства n
- Минимальное количество векторов, входящих в каждую компоненту связности – n
- Максимальное количество векторов, входящих в каждую компоненту связности - 2n

Результаты: (2)

- Сравнение с существующим подходом индексирования хешем:
- На порядок большая точность возвращаемого ответа:
  - 1-NN query
    Иерархическая кластеризация: 87%
    LSH : 62%
  - 10-NN query
    Иерархическая кластеризация: 79%
    LSH : 50%%
  - 50-NN query
    Иерархическая кластеризация: 78%
    LSH : 34%
- Значительный проигрыш во времени ответа на запрос