Построение VoR-дерева с использованием технологии MapReduce

Амосов Федор, СПбГУ

Руководитель: Волохов Антон, Яндекс

Постановка задачи

Дано

- 10^9 точек из $\mathbb{R}^5$
- Метрика
- Кластер из 10^3 машин

Задача

- Распараллеленно построить структуру для эффективных 1-NN и k-NN запросов в данной метрике

VoR-дерево

Определение

- VoR-дерево = R-дерево + диаграмма Вороного

В нашей задаче

- Евклидова метрика — ?
- 1–NN запрос за $O(\log n)$
- k–NN запрос за $O(\log n + k)$

R-дерево

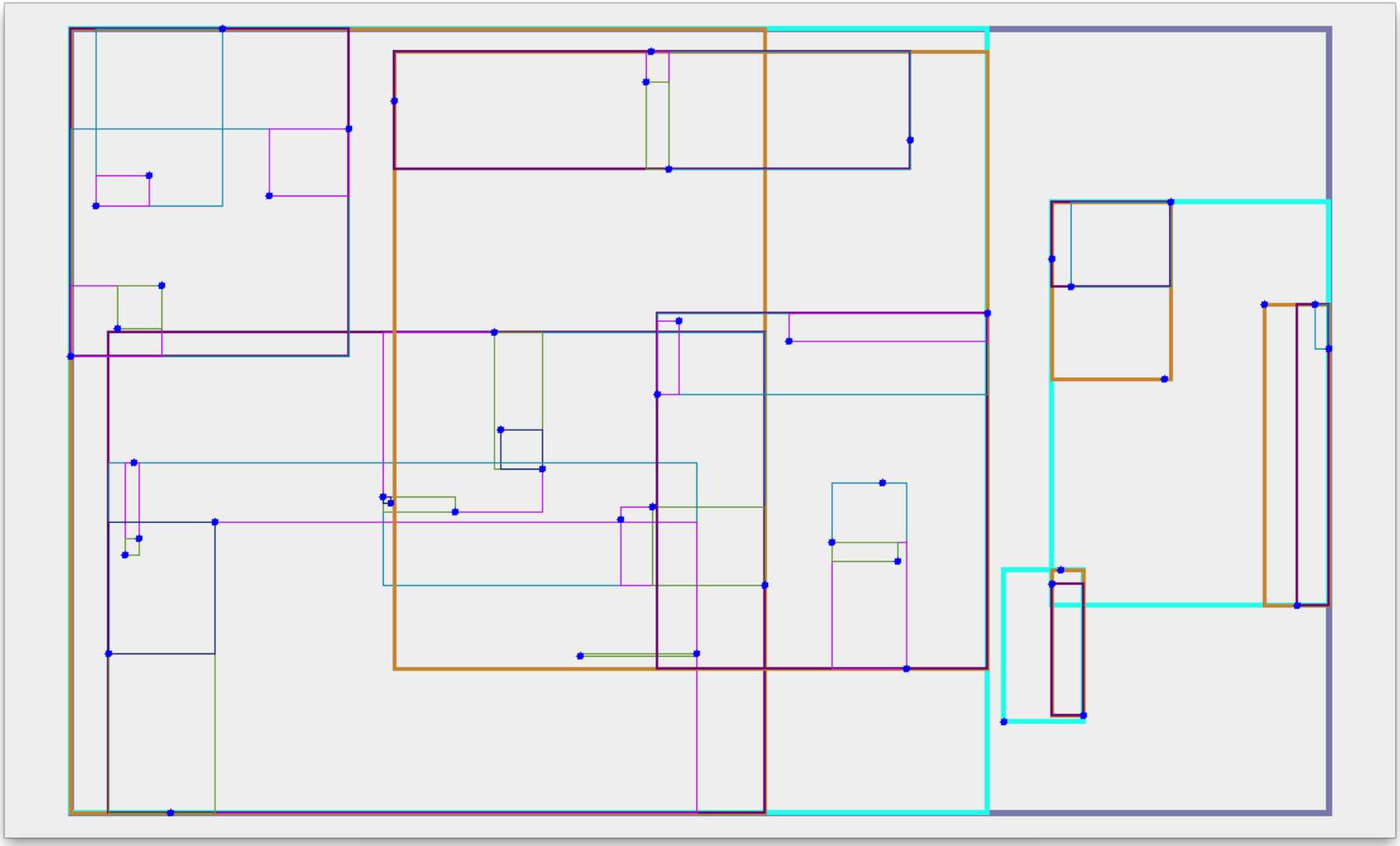


Диаграмма Вороного

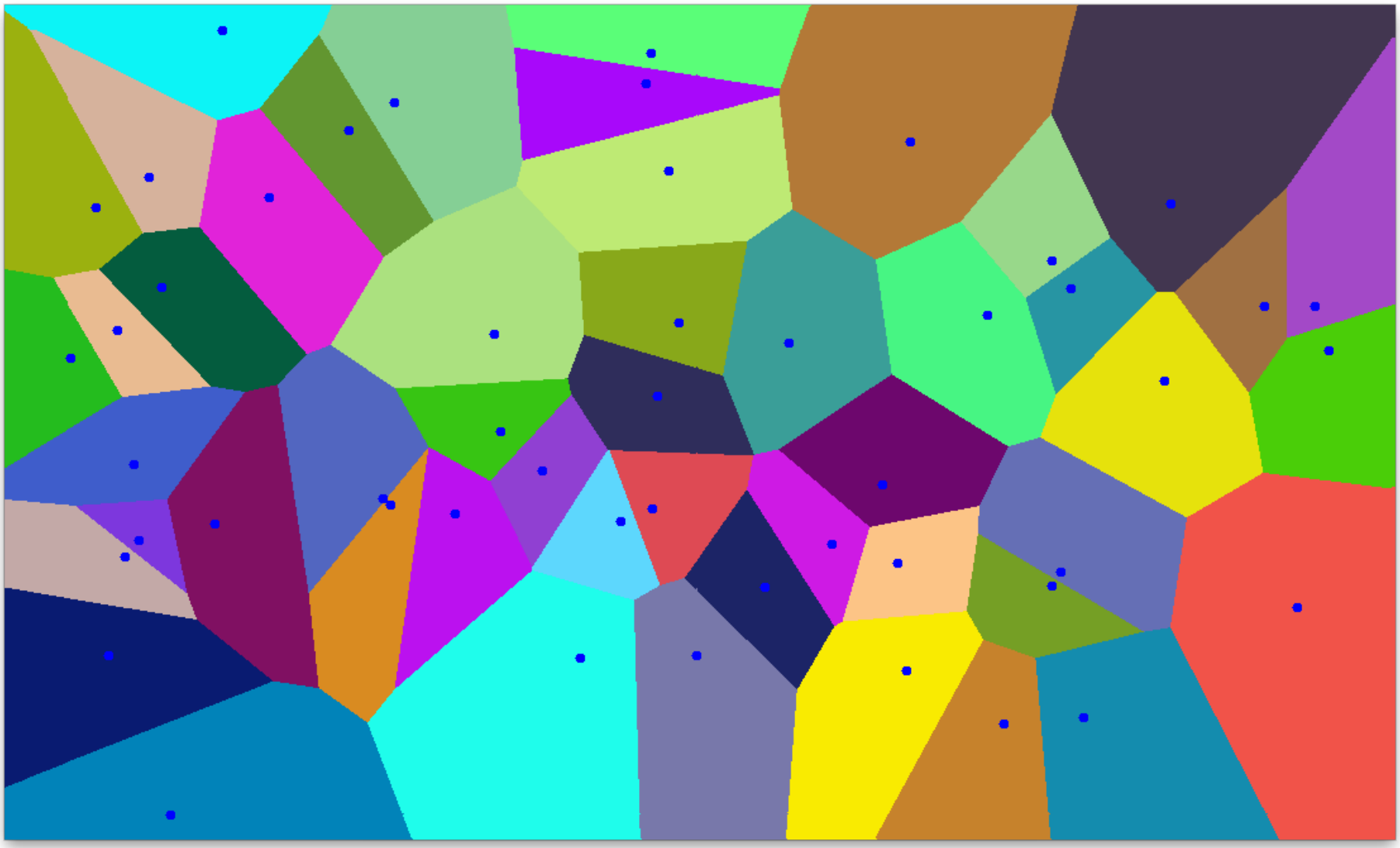
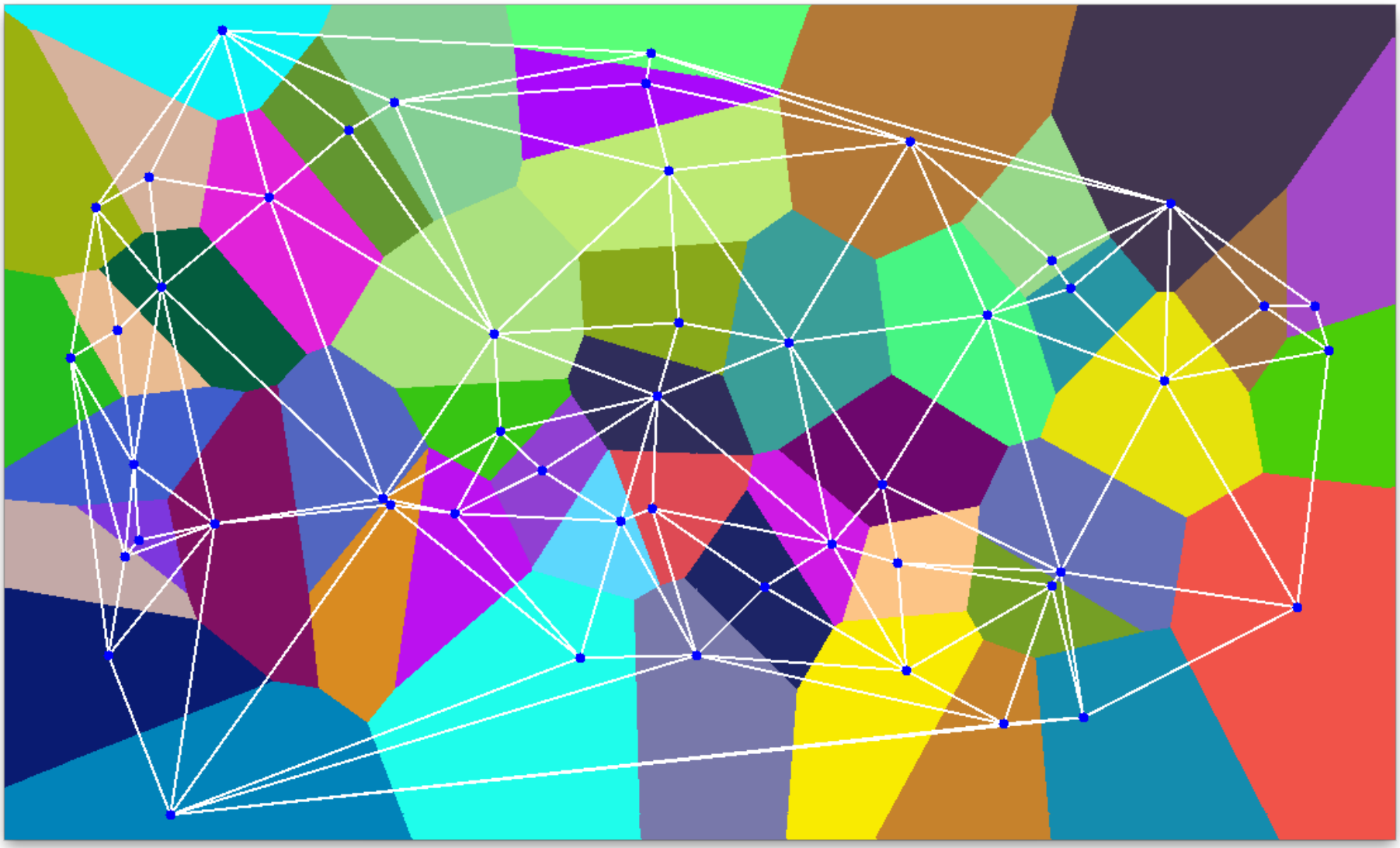
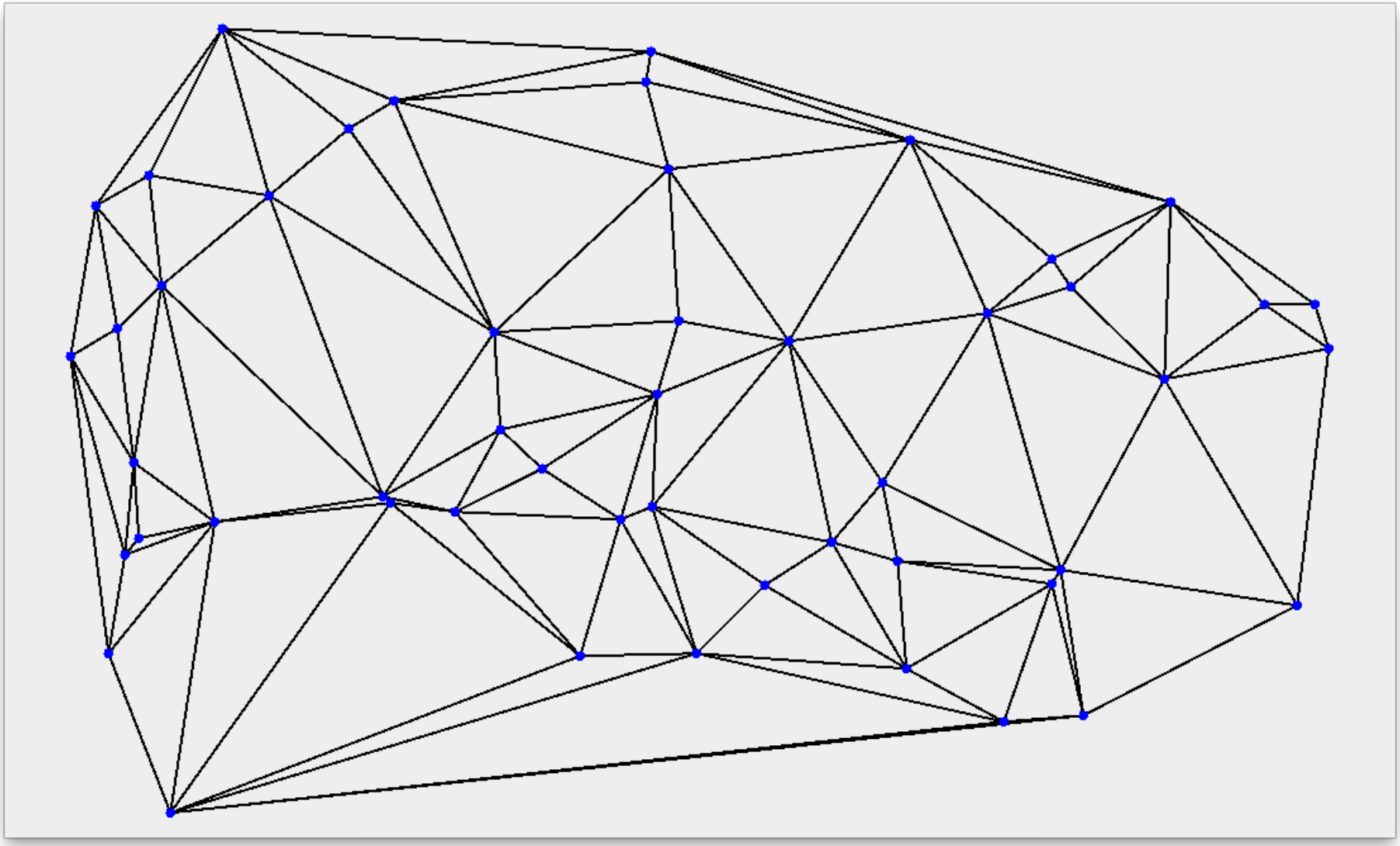


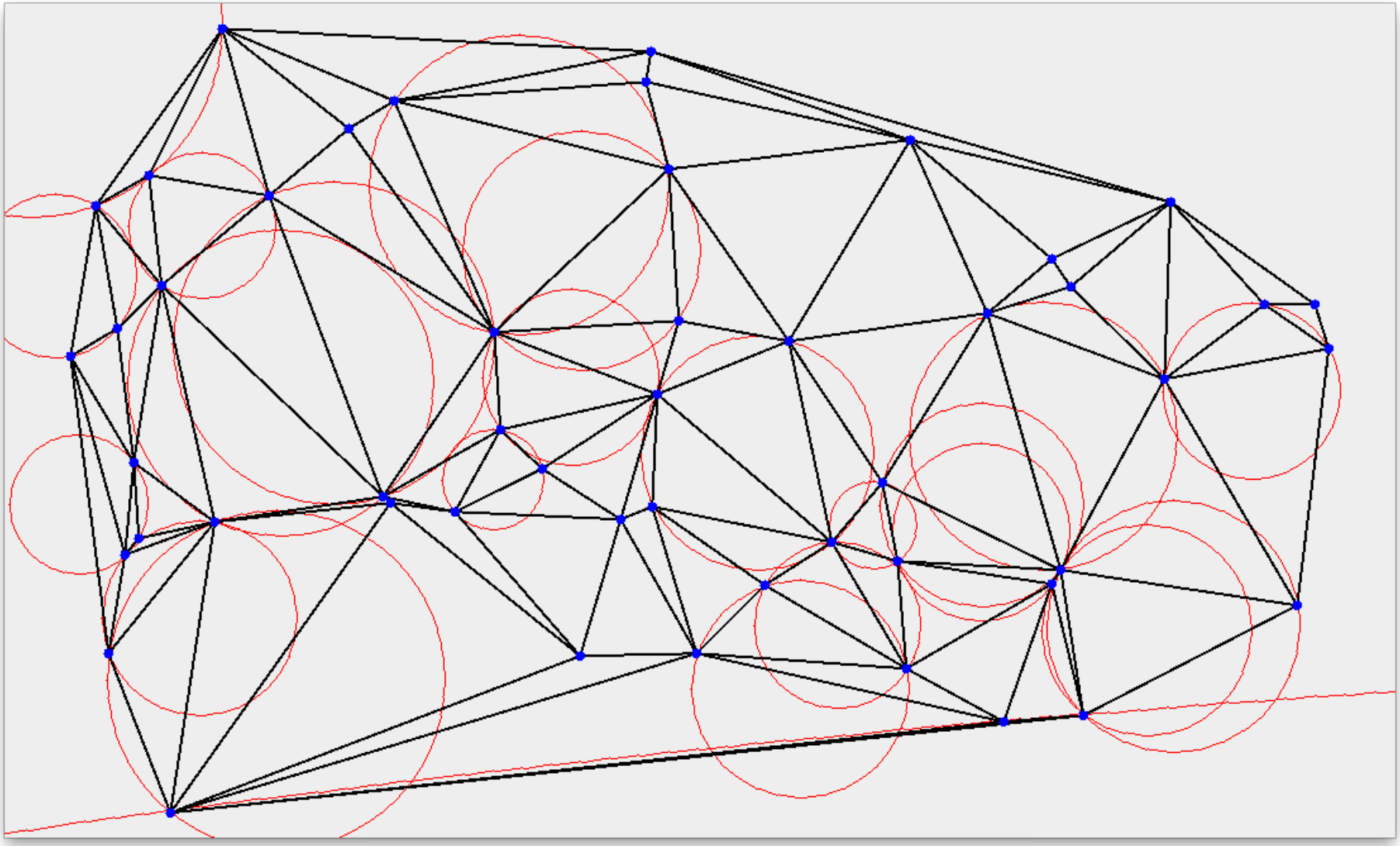
Диаграмма Вороного



Граф Делоне (Триангуляция Делоне)

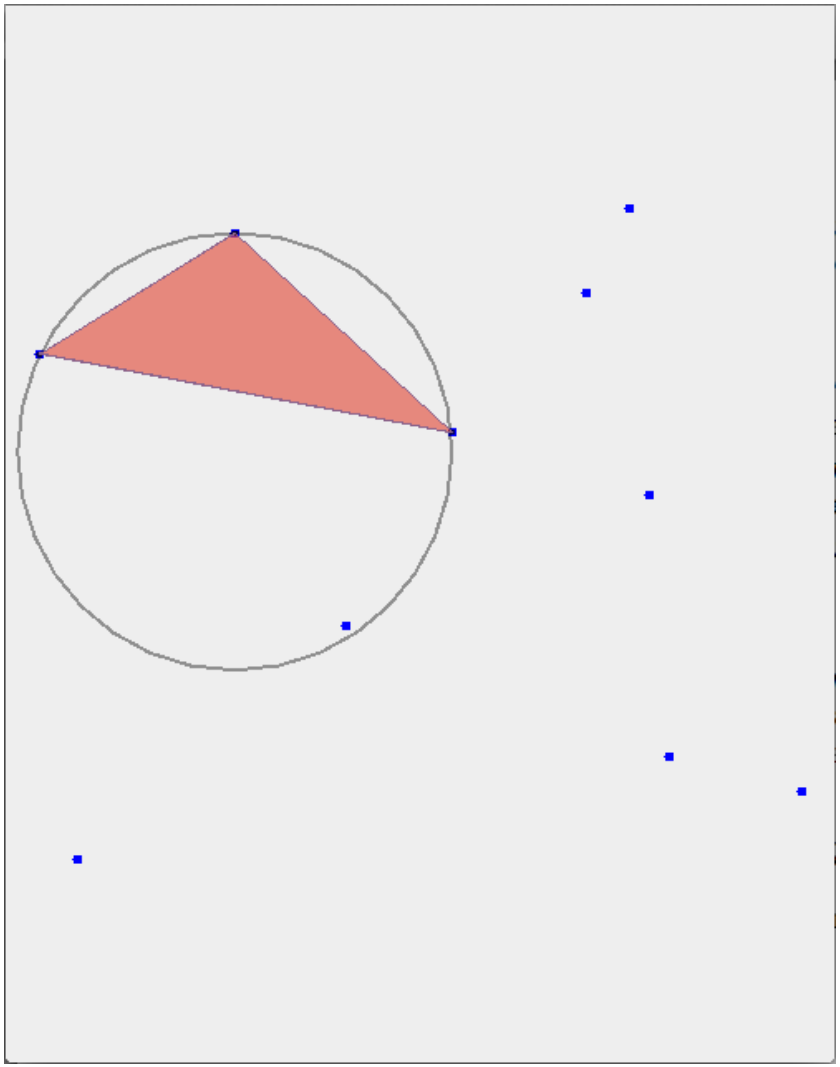


Основное свойство графа Делоне

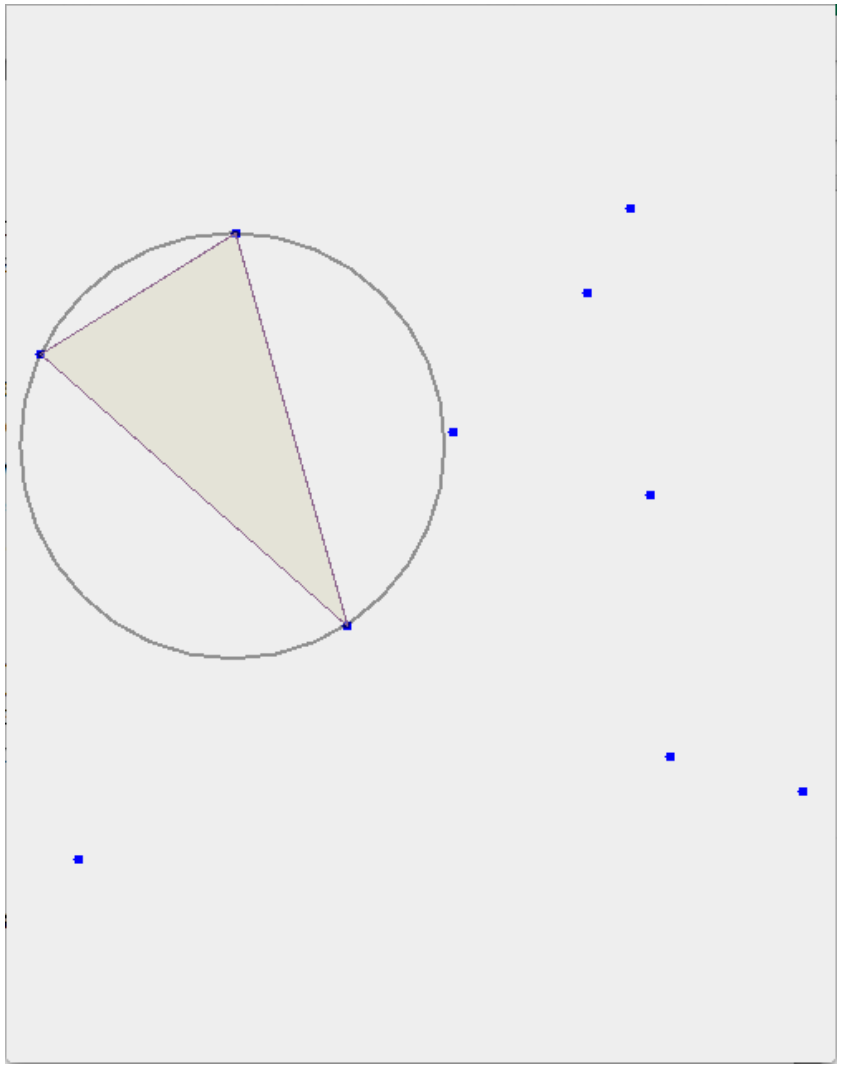


Плохие и хорошие треугольники

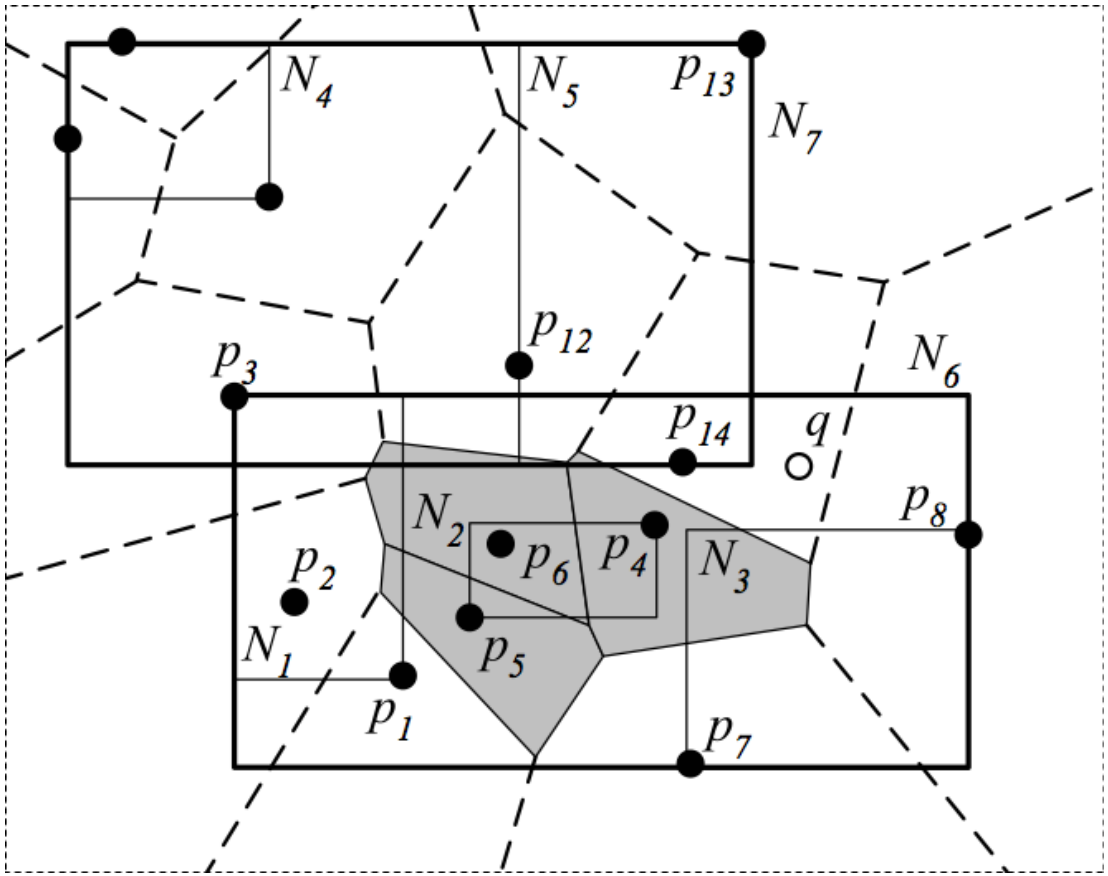
Плохой треугольник



Хороший треугольник



VoR-дерево

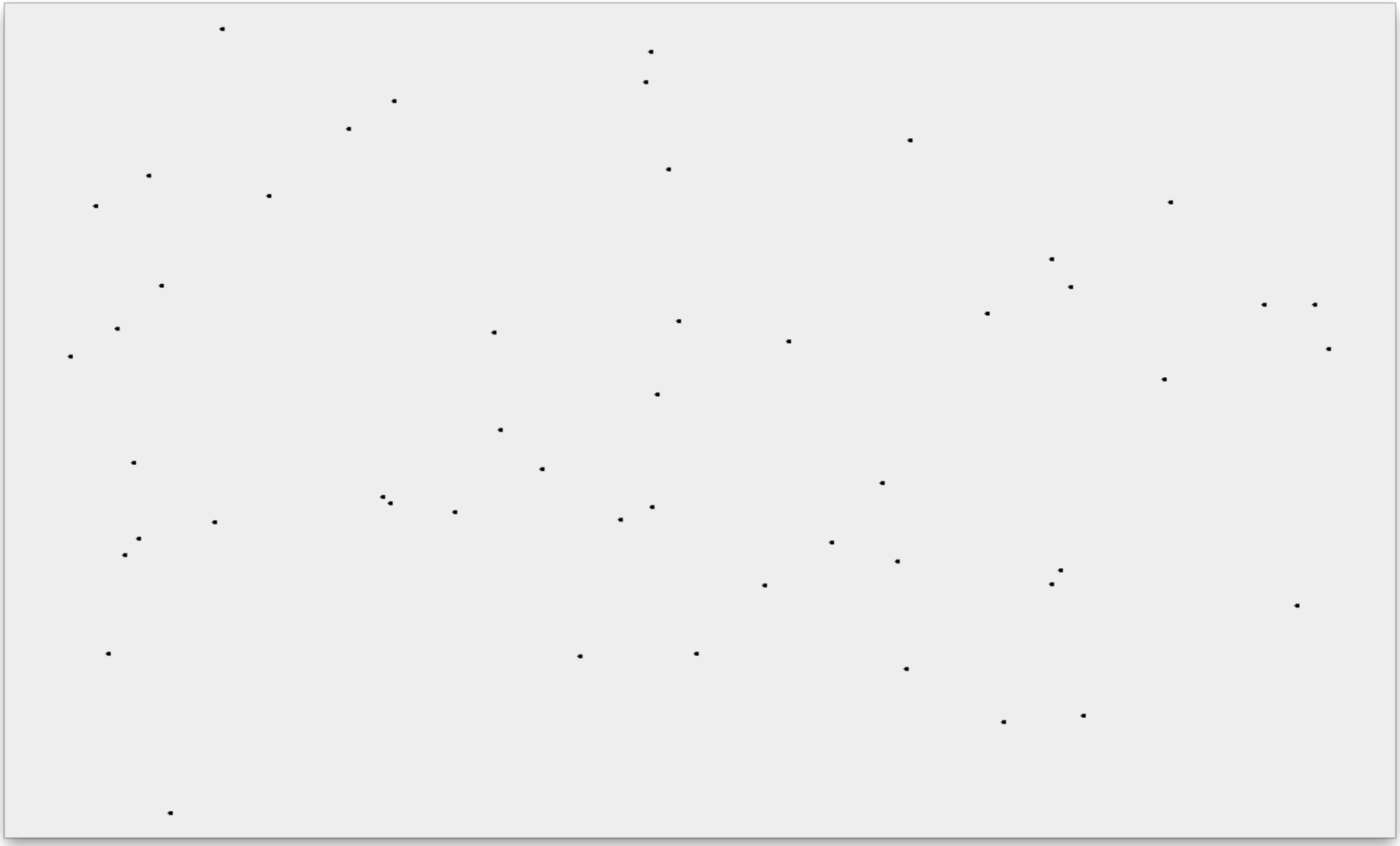


Sharifzadeh , M., Shahabi C., «VoR-Tree: R-trees with Voronoi Diagrams for Efficient Processing of Spatial Nearest Neighbor Queries» Proceedings of the VLDB Endowment, 2010

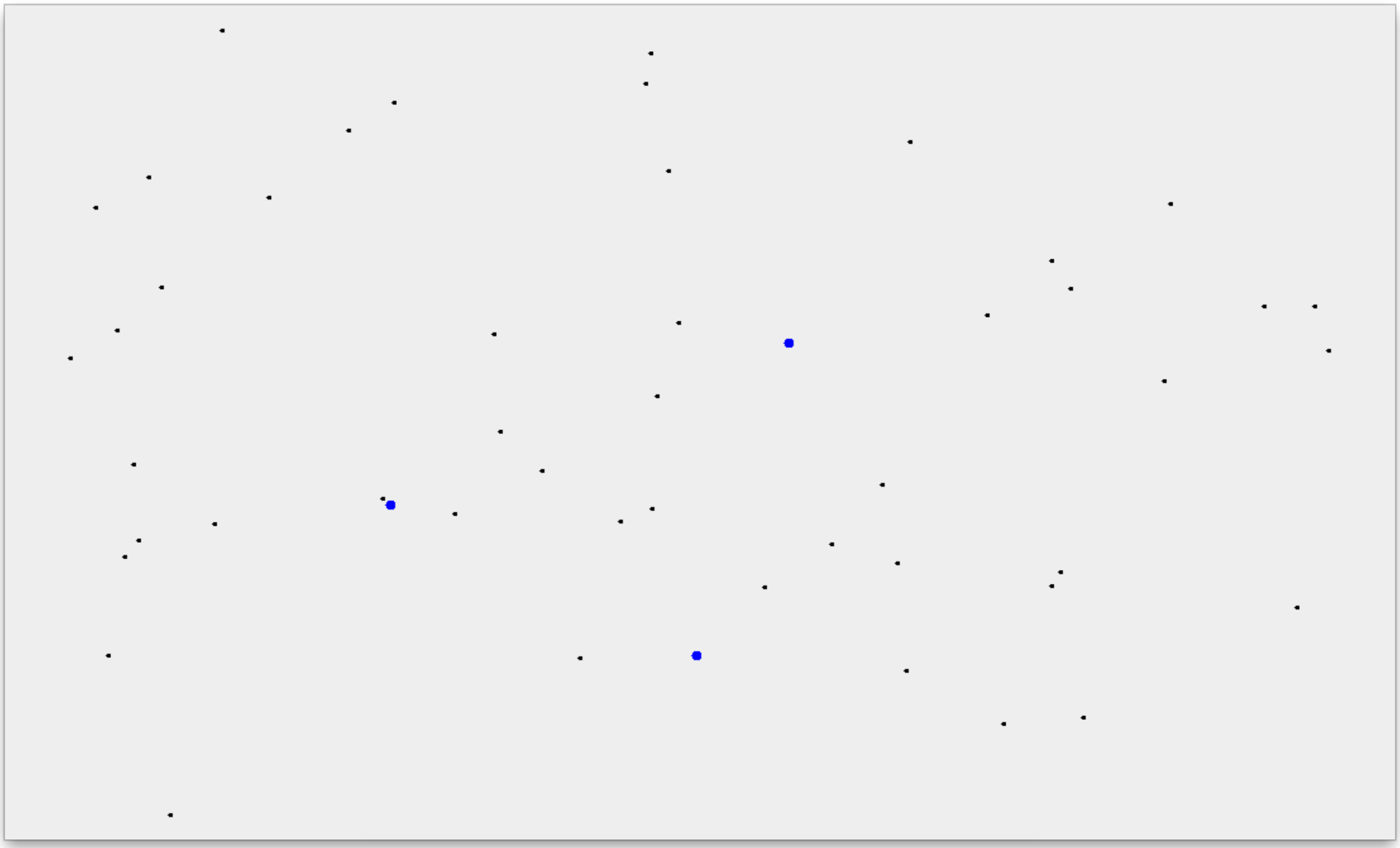
Построение VoR-дерева

- offline
- Требование распараллеленности $\Longrightarrow$ «разделяй и властвуй»
- Построить R-дерево за $O(n \log n)$ — очевидно
- Построить граф Делоне за $O(n \log n)$, следуя парадигме «разделяй и властвуй» — ?

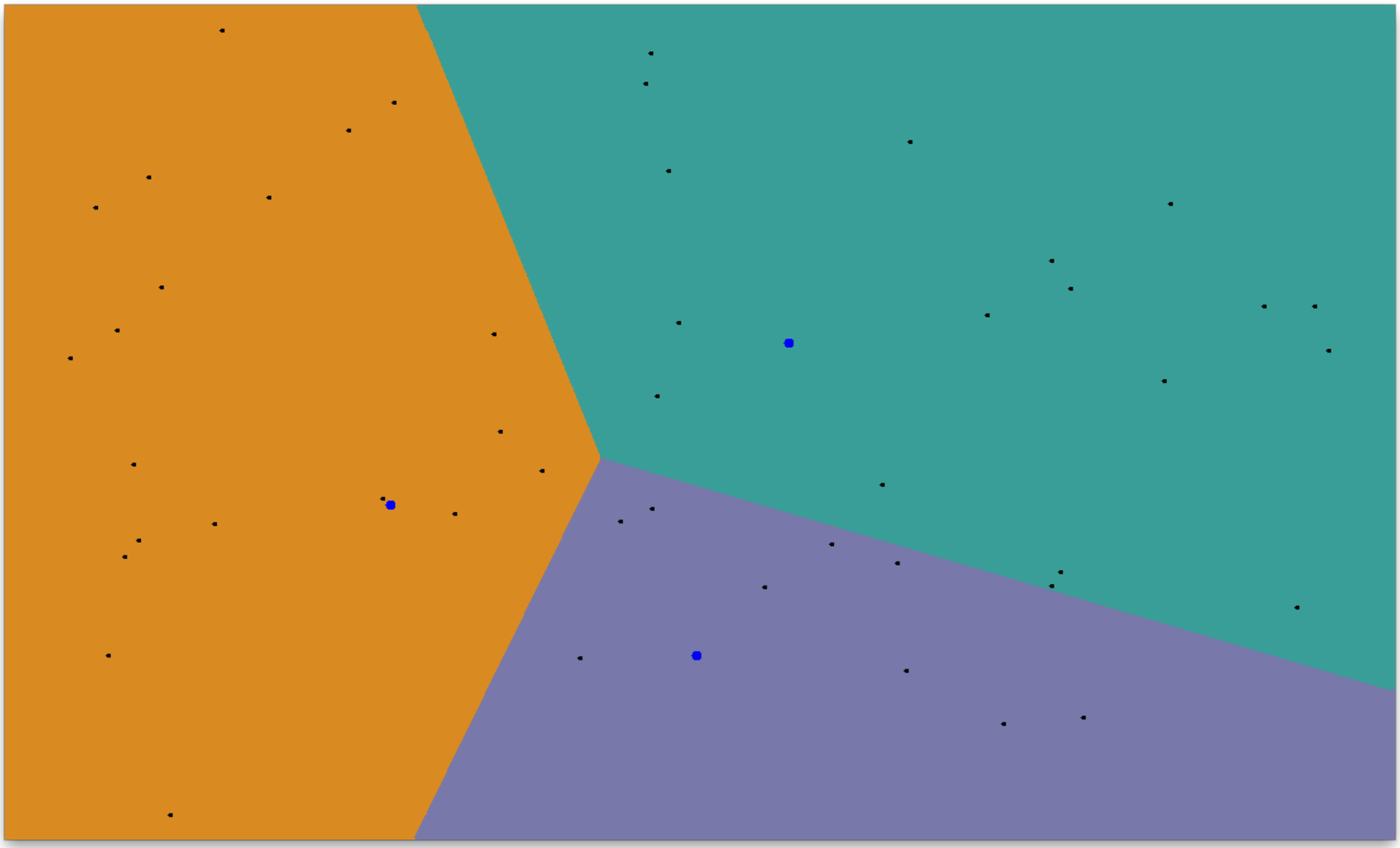
Построение графа Делоне 1



Построение графа Делоне 2

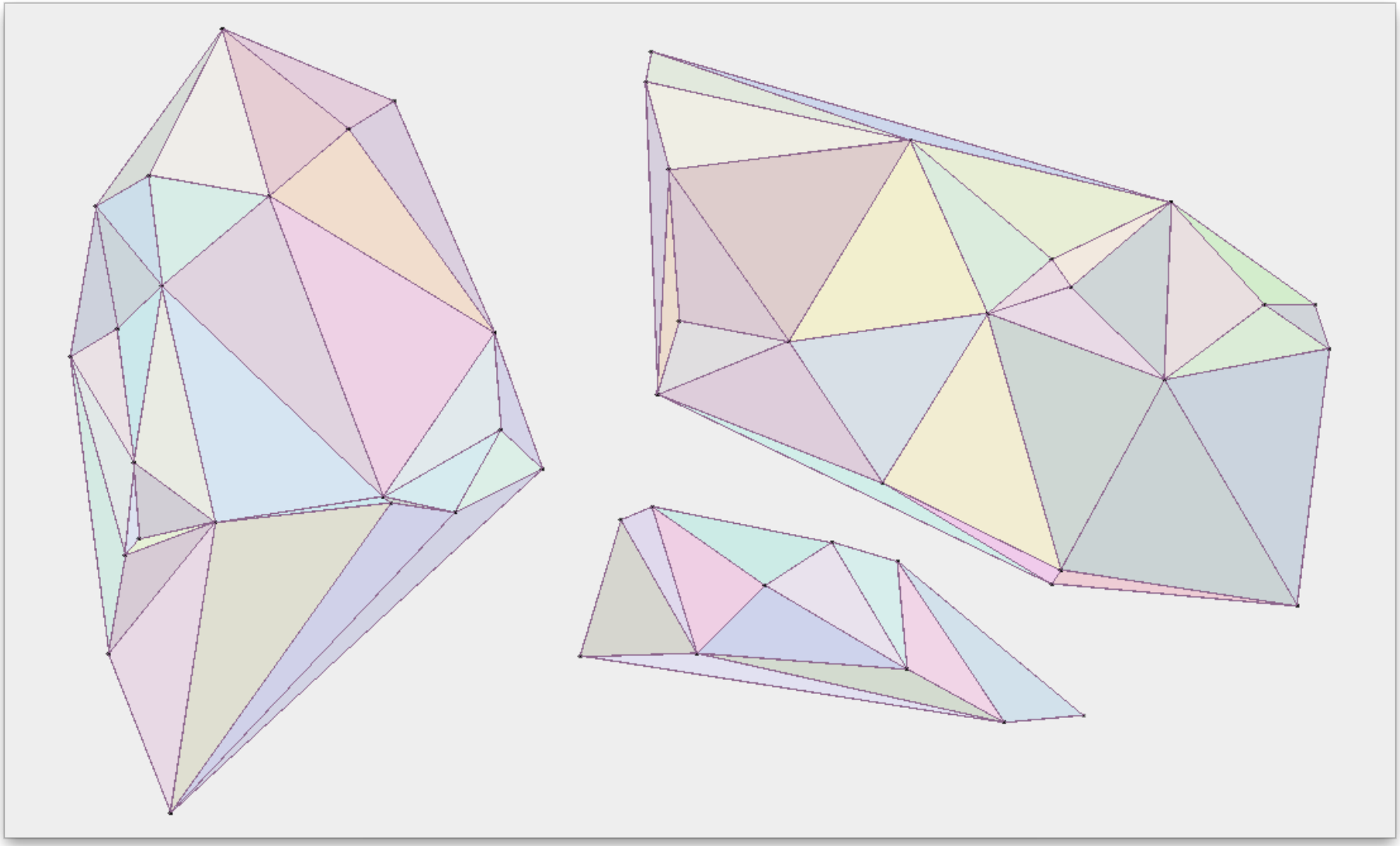


Построение графа Делоне 3

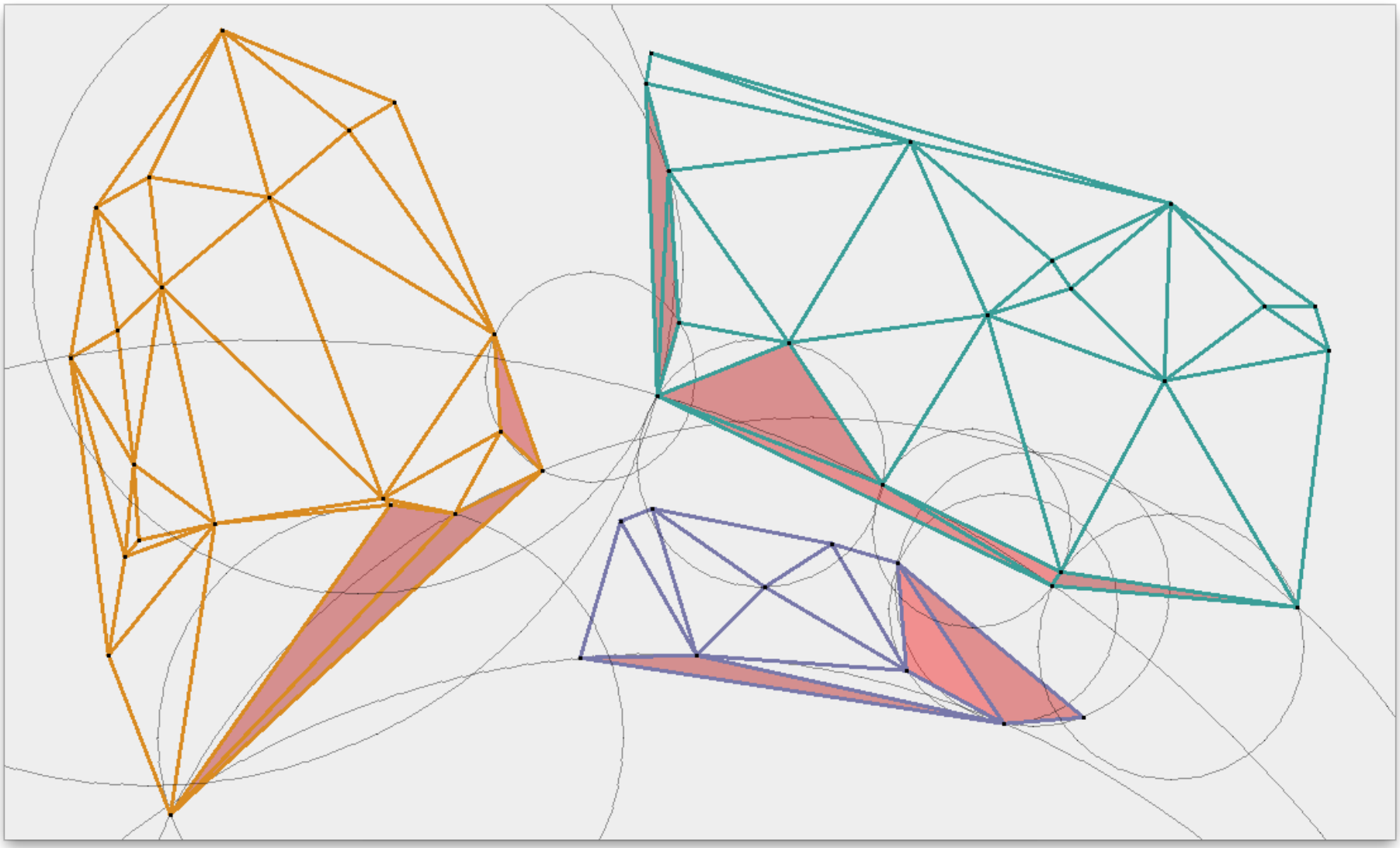


Построение графа Делоне 4

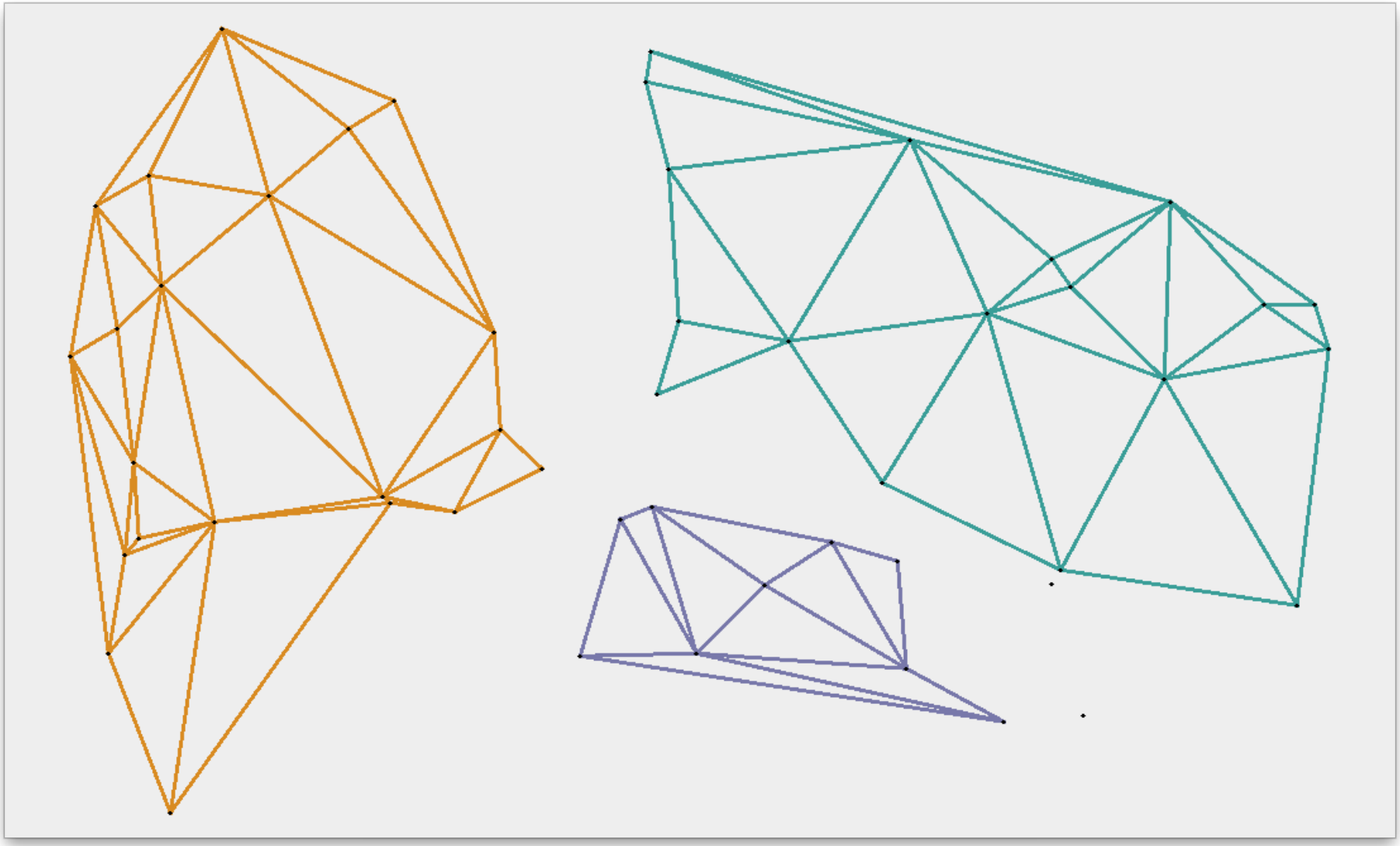
Построение графа Делоне 5



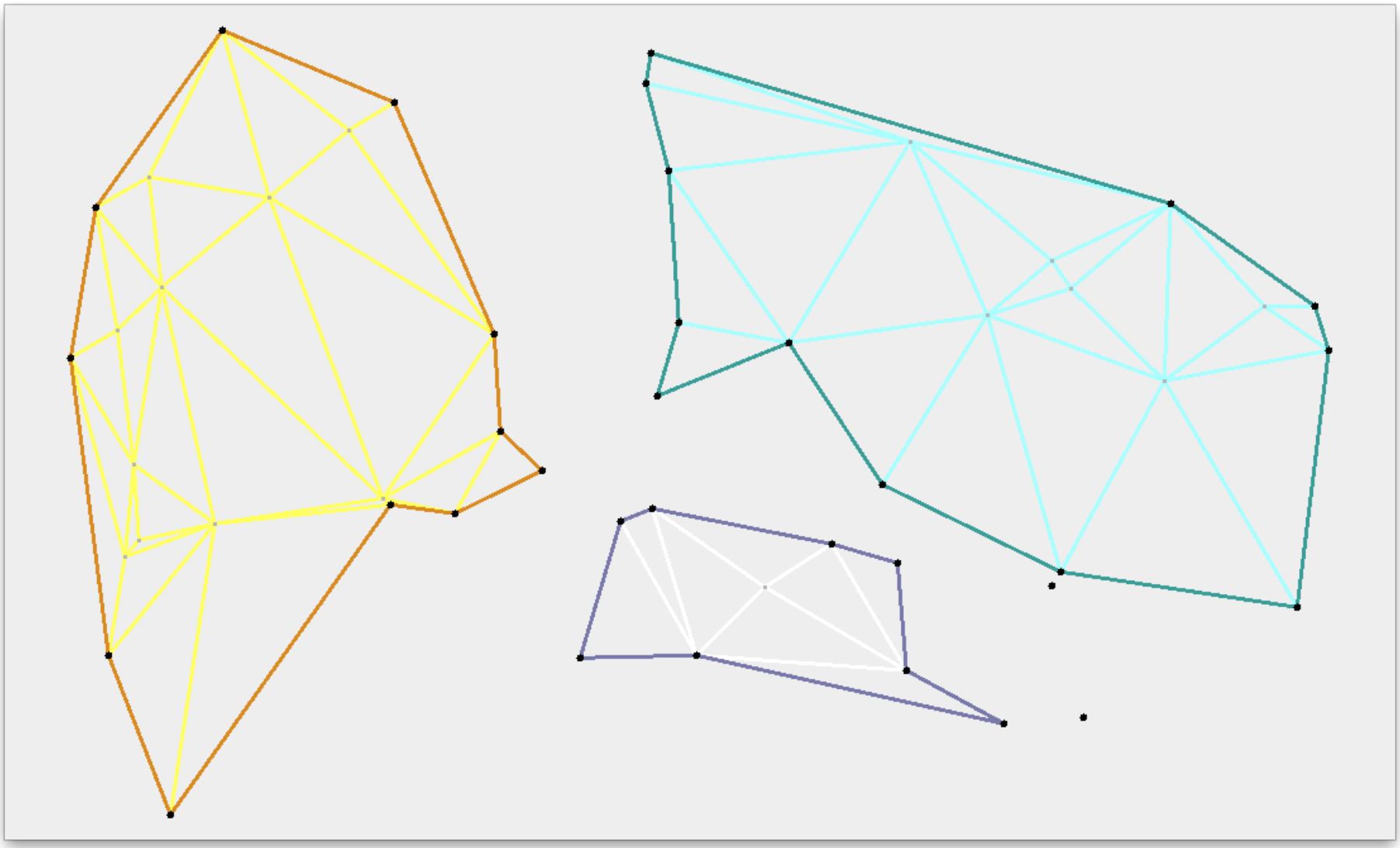
Построение графа Делоне 6



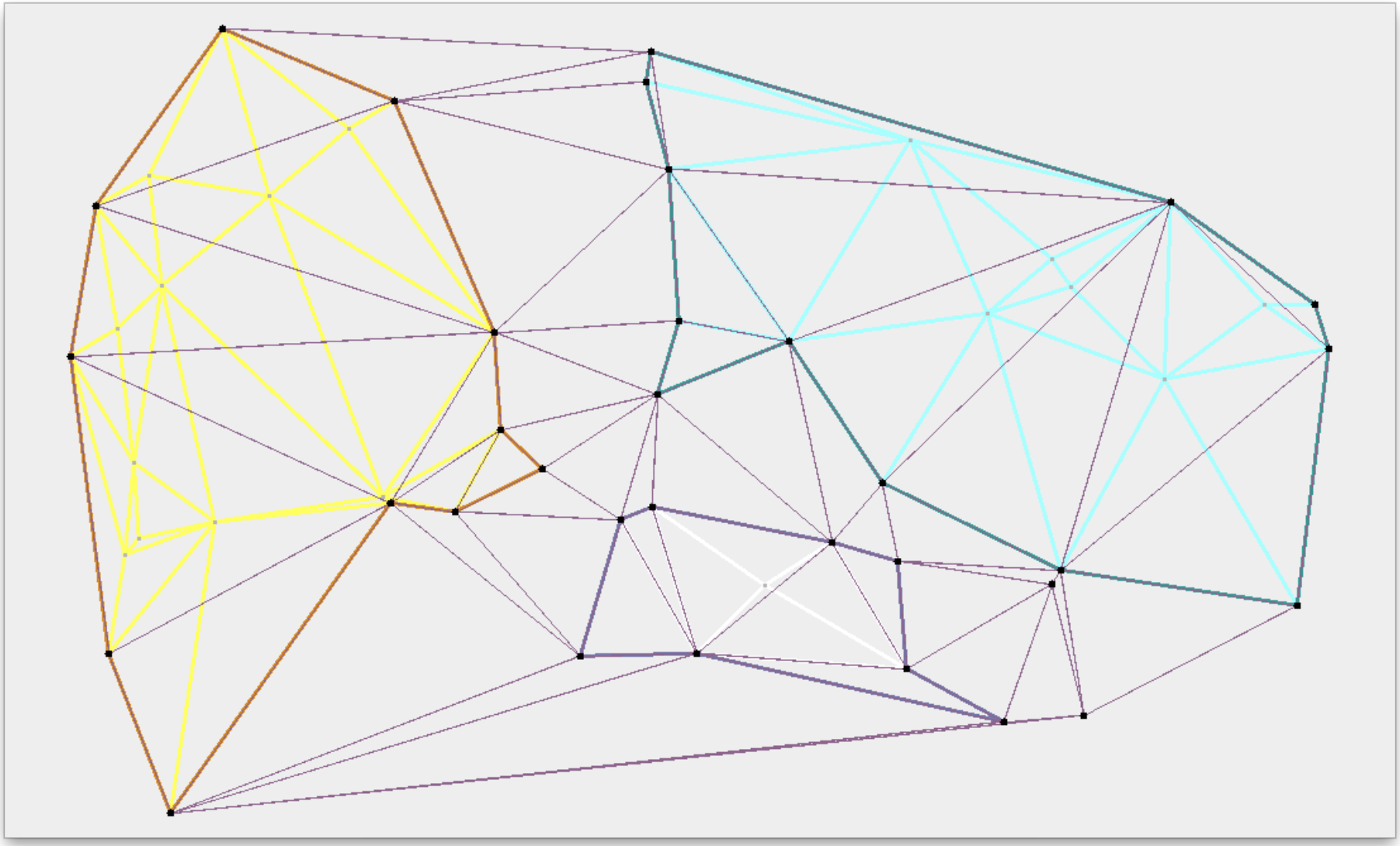
Построение графа Делоне 7



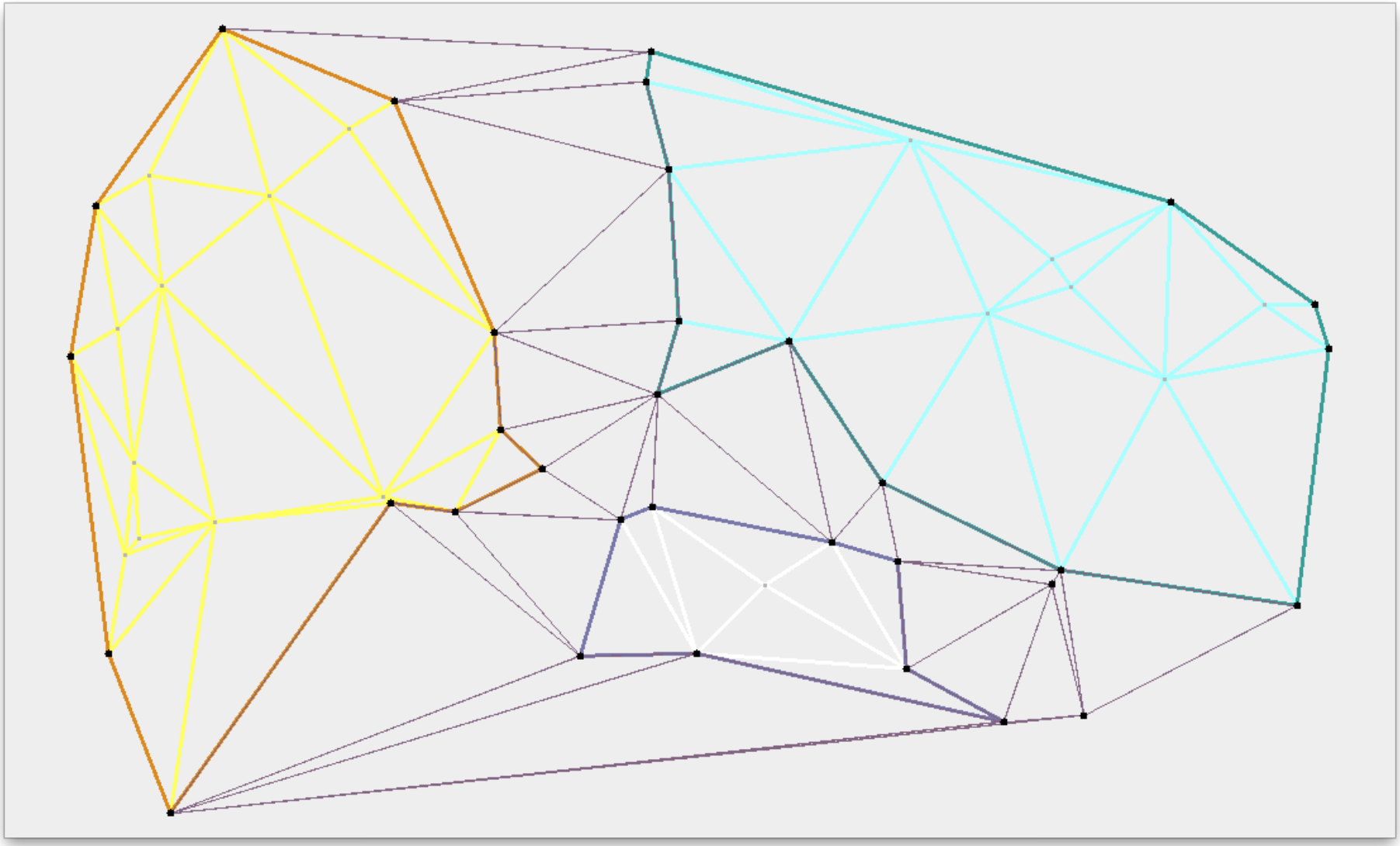
Построение графа Делоне 8



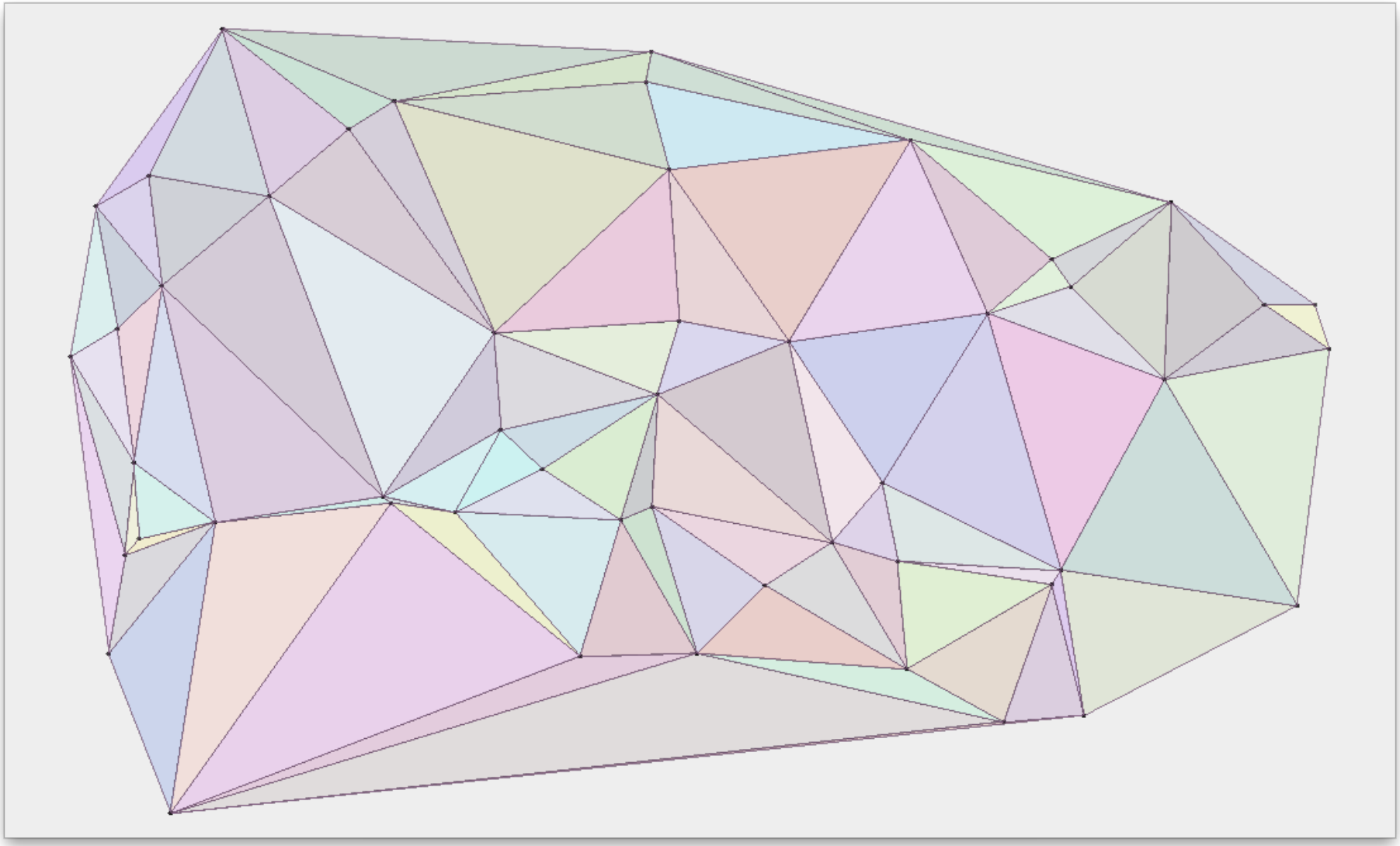
Построение графа Делоне 9



Построение графа Делоне 10



Построение графа Делоне 11



Наш алгоритм

1. Разбить множество точек на несколько частей*
2. Построить на каждой части граф Делоне
3. Удалить в каждом построенном графе «плохие» треугольники
4. Добавить все оставшееся в итоговый граф
5. Построить на границах оставшегося «склеивающий» граф Делоне
6. Добавить в итоговый граф ребра «склеивающего» графа
   - соединяющие разные части
   - удаленные на 3 этапе

Анализ алгоритма

- 2 утверждения $\Longrightarrow$ корректность
- Ускорение внешнего «склеивающего» алгоритма
  - $O(n^2) \longrightarrow O(n \log n)$
  - $O(n^k) \longrightarrow O(n^{\frac{k}{2}}), k > 2$

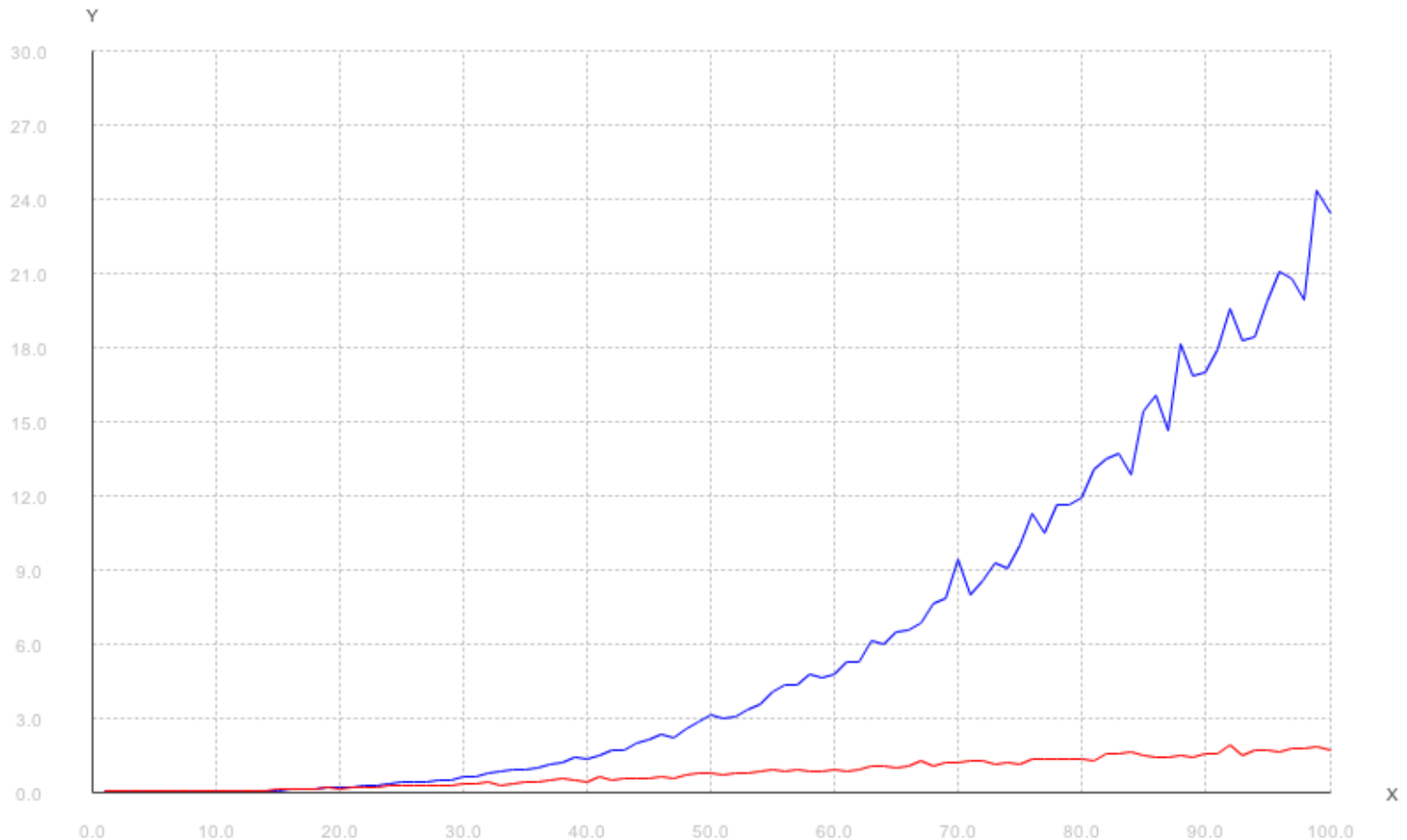
Улучшение алгоритма

1. Плохие треугольники находятся вблизи границы $\Longrightarrow$ не надо проверять все треугольники
2. Будем строить нашим алгоритмом сразу VoR-деревья $\Longrightarrow$ определение «хорошести» треугольника с помощью построенных VoR-деревьев
3. Склеивание можно проводить с помощью рекурсивного вызова нашего алгоритма. В чем подвох?
4. Внешний алгоритм только для маленьких задач

Итого, сложность всегда $O(n \log n)$

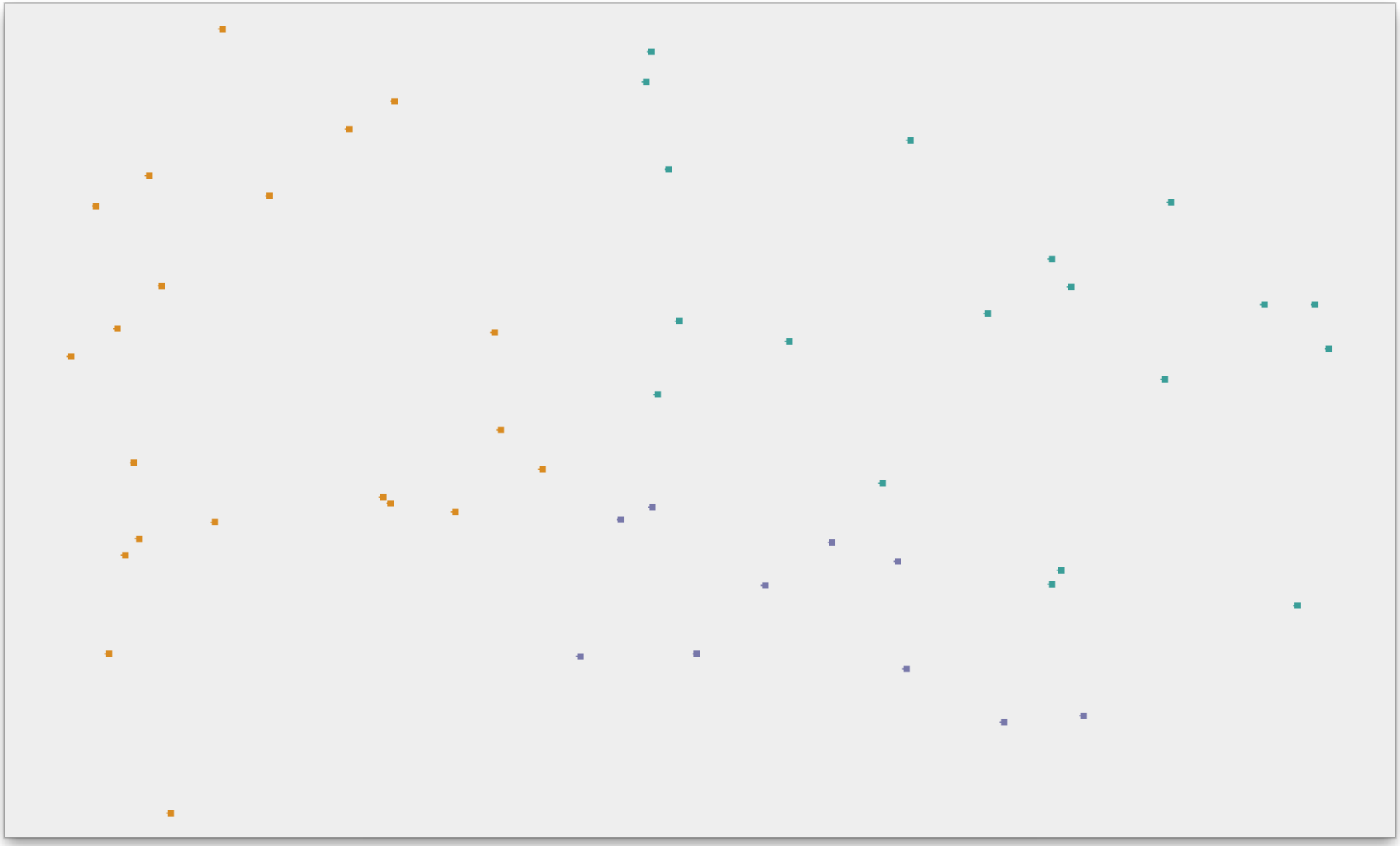
На практике

Ускорение наивного алгоритма (n^4 vs $n \log n$)

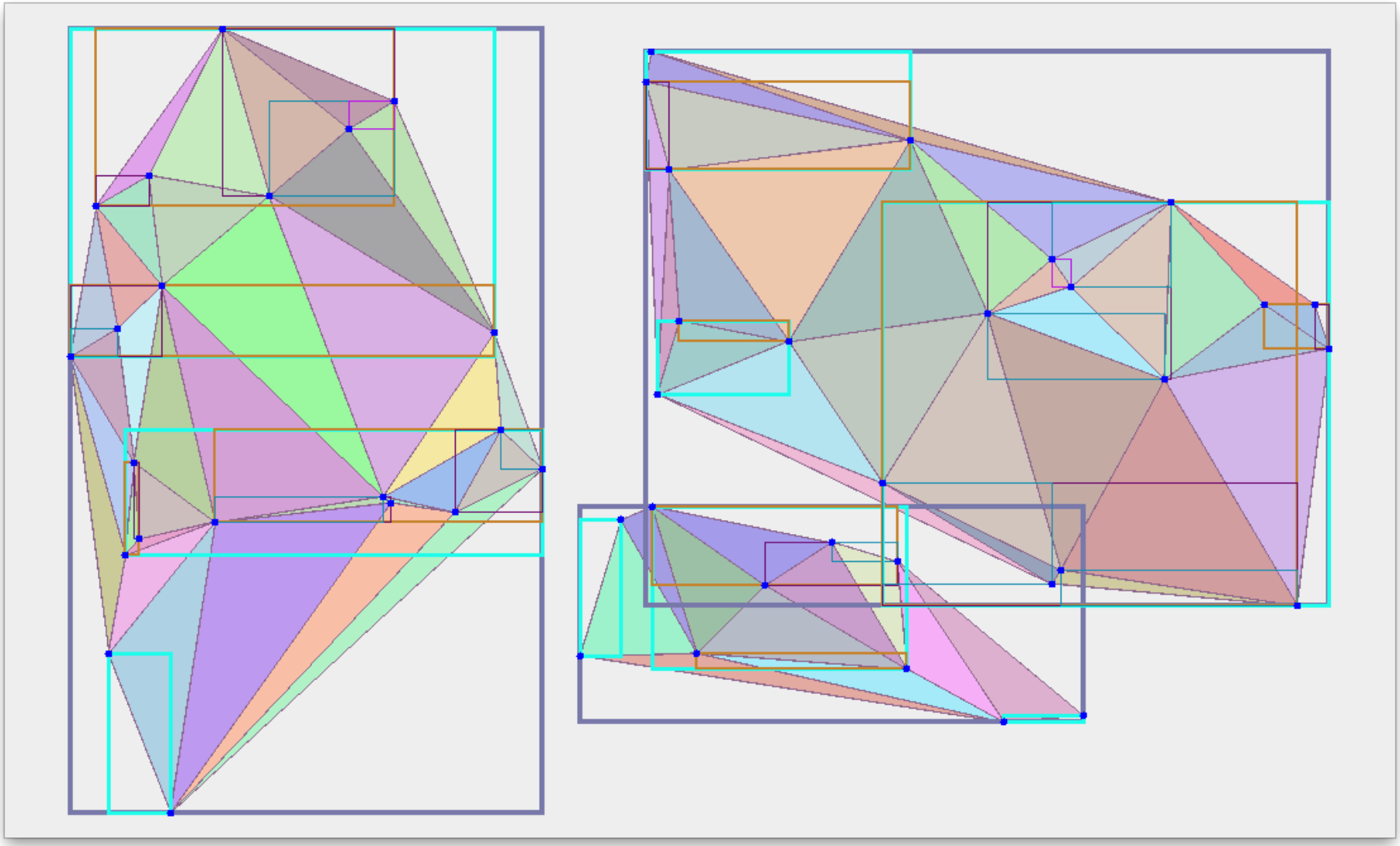


Время работы от размера задачи

Распараллеливание 1



Распараллеливание 2



Технология MapReduce

Технология

- Map — обработка данных — построение VoR-деревьев из частей
- Reduce — свертка данных — склеивание полученных VoR-деревьев

Реализация

- Apache Hadoop
- Сериализация VoR-дерева — Google Protocol Buffers

Результаты

Разработан алгоритм построения многомерного VoR-дерева

- Асимптотически эффективный
- Простой в реализации
- Хорошо параллелится

Реализован

- На Java
- С использованием MapReduce

До 10^9 точек еще далеко, но прототип работает

Q&A

Спасибо за внимание!

github.com/amosov-f/VorTree