Московский государственный университет имени М. В. Ломоносова
Факультет Вычислительной Математики и Кибернетики
Кафедра Математических Методов Прогнозирования

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Слияние перекрывающихся триангуляций Делоне на основе минимальных остовных деревьев

Выполнила:
студентка 417 группы
Готман Мария Леонидовна

Научный руководитель:
д.т.н., профессор
Местецкий Леонид Моисеевич

Москва, 2015

Содержание

1 Введение **3**

2 Терминология и постановка задачи **4**

2.1 Основные определения 4

2.2 Постановка задачи 5

3 Метод слияния триангуляций **6**

4 Описание алгоритма **8**

4.1 Выбор структуры данных 8

4.2 Построение разрезов и швов триангуляций Делоне 9

4.3 Поиск стартеров 12

4.3.1 Поиск первого стартера 14

4.3.2 Минимальные остовные деревья 15

4.3.3 Поиск последующих стартеров 18

4.4 Общая структура алгоритма 20

5 Оценка вычислительной сложности алгоритма **20**

5.1 Сложность поиска стартеров 21

5.2 Сложность построения разрезов и швов 21

6 Эксперименты **22**

6.1 Программная реализация 22

6.2 Вычислительный эксперимент 23

7 Заключение **23**

Использованная литература **25**

Аннотация

В данной работе рассмотрен алгоритм слияния двух триангуляций Делоне, причем множества, на которых заданы исходные триангуляции, допускают полное перемешивание. Метод, позволяющий решить данную задачу, имеет линейную сложность в худшем случае. Он основан на построении минимальных остовных деревьев исходных триангуляций при помощи алгоритма Черитона-Тарьяна. Это является ключевой частью алгоритма, за счет чего достигается его линейная сложность.

1 Введение

Рассматривается задача слияния двух триангуляций Делоне, линейная разделимость исходных облаков точек которых не предполагается. Впервые задача построения триангуляции Делоне была рассмотрена в 1934 году российским математиком Борисом Делоне. На практике триангуляции Делоне применяются во многих прикладных задачах: сравнение 3D поверхностей, интерполяция разреженных геонаучных данных, аппроксимация географических поверхностей на плоскости и т.д.

Несмотря на то, что построение триангуляции Делоне является распространенной задачей в вычислительной геометрии [1], [2], слияние неразделенных триангуляций исследовано мало. Такую задачу решают обычно одним из следующих способов. Первый способ основан на том, что информация об исходных триангуляциях никак не учитывается и искомая триангуляция строится по объединенному множеству точек с нуля. Сложность такого алгоритма $O(n \log n)$ в худшем случае. Такая асимптотика достигается при помощи парадигмы «разделяй и властвуй» [1, стр. 44-45]. Множество точек рекурсивно делится на два линейно разделимых подмножества. Рекурсия прекращается, когда в подмножестве останется две или три точки на которых строится тривиальная триангуляция в виде линии или треугольника соответственно. Затем происходит процесс слияния разделенных триангуляций за линейное время. Второй способ разрушает только одну триангуляцию, затем последовательно добавляет по одной точке разрушенной триангуляции с помощью инкрементного алгоритма. Такой подход имеет сложность $O(n^2)$ в худшем случае.

Как видно, оба подхода не используют всю информацию о исходных триангуляциях, поэтому естественно кажется, что существует алгоритм, помогающий осуществить слияние за линейное время. В данной работе рассматривается линейный алгоритм слияния триангуляций Делоне. Алгоритм основан на построении минимальных остовных деревьев (МОД) исходных триангуляций при помощи алгоритма Черитона-Тарьяна. Известно, что МОД является подграфом триангуляции Делоне. Эта информация используется при поиске новых ребер, соединяющих две исходные триангуляции. Если найдено ребро, соединяющее точки из разных исходных триангуляций, называемое стартером, то можно строить «швы», соединяющие исходные триангуляции, и «разрезы», содержащие ребра исходных триангуляций, которые перестали удовлетворять условию Делоне. Подзадача построения швов и разрезов является наиболее сложно осуществимой с технической точки зрения. Поиск стартеров технически проще, но понимание его математического обоснования требует должных усилий.

2 Терминология и постановка задачи

2.1 Основные определения

Пусть на евклидовой плоскости задано **S** — множество из не менее трех точек, не все из которых лежат на одной прямой. Используя [1, стр. 7-8], [3], введем ряд ключевых определений которые понадобятся нам для дальнейшего понимания задачи.

Точки, входящие во множество **S**, будем называть *сайтами*. *Триангуляцией* конечного множества точек **S** называется планарный граф с вершинами из **S**, все внутренние области которого являются треугольниками. Триангуляция называется *выпуклой*, если минимальный многоугольник, охватывающий все ее треугольники, будет выпуклым. Далее под термином *грань* будем понимать только конечную треугольную грань триангуляции. Ребро и грань называются *инцидентными*, если они имеют две общие вершины. Ребра, инцидентные одной грани, называются *смежными*. Ребро, имеющее менее двух инцидентных граней, называется *открытым*.

Окружность называется *пустой*, если она не содержит внутри себя сайтов. Прямая линия, по одну сторону от которой нет сайтов, называется *несобственной* пустой окружностью. Окружность, проходящая через сайт, называется *инцидентной* этому сайту. *Ребром Делоне* называется ребро, инцидентные сайты которого имеют общую пустую инцидентную окружность. *Гранью Делоне* называют грань, вершины которой имеют общую пустую инцидентную окружность.

Дадим два эквивалентных определения:

Определение. *Триангуляцией Делоне (ТД) Del*($\boldsymbol{S}$) множества точек **S** называется выпуклая триангуляция, у которой описанная окружность каждой треугольной грани является пустой, т.е. все грани которой являются гранями Делоне.

Определение. *Триангуляцией Делоне* множества точек **S** называется выпуклая триангуляция, у которой для каждого ребра существует пустая инцидентная сайтам-вершинам окружность, т.е. все ребра которой являются ребрами Делоне.

Пучком сайта $p \in \mathbf{S}$ называют множество ребер триангуляции, инцидентных сайту p. Пучок сайтов будем представлять в виде двунаправленного циклического списка сайтов $p_1, \ldots, p_k$, смежных с p в триангуляции, так чтобы они шли в направлении обхода против часовой стрелки относительно p.

2.2 Постановка задачи

Задача слияния двух перекрывающихся триангуляций Делоне ставится следующим образом. Даны два конечных линейно неразделимых множества сайтов $\mathbf{B}$ и $\mathbf{W}$ (допускается их полное перемешивание), а также их триангуляции Делоне $Del(\mathbf{B})$ и $Del(\mathbf{W})$. Нужно построить триангуляцию Делоне на объединенном множестве сайтов $Del(\mathbf{B} \cup \mathbf{W})$.

Будем считать, что сайты раскрашены в два цвета: множество черных сайтов $\mathbf{B}$ и множество белых сайтов $\mathbf{W}$. Триангуляции множеств $\mathbf{B}$ и $\mathbf{W}$ будем называть исходными, искомую триангуляцию $Del(\mathbf{B} \cup \mathbf{W})$ будем называть объединенной. Пример исходных данных и искомой триангуляции приведен на рис. 1.

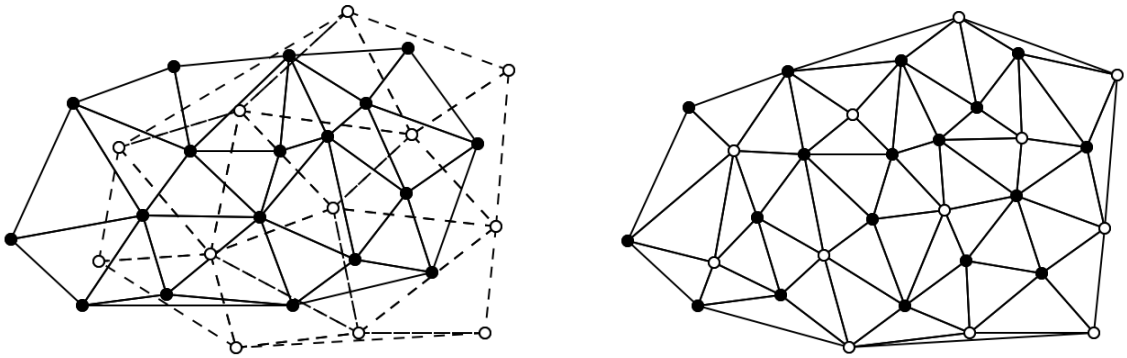

Рис. 1: Исходные триангуляции и объединенная триангуляция

Нижняя оценка для задачи построения триангуляции Делоне множества точек $\mathbf{S}$ известна и имеет сложность $O(n \log n)$, где $n = |\mathbf{S}|$ [4, лекции 10-11]. Такую сложность имеет, например, алгоритм, основанный на парадигме «разделяй и властвуй» [1, стр.44-45]. Этот алгоритм имеет сложность $O(n \log n)$ в среднем и худшем случае. Такой алгоритм позволяет производить слияние разделенных триангуляций Делоне за линейное по числу точек время. Однако вопрос о слиянии перекрывающихся триангуляций Делоне остается открытым. Возможным подходом является построение объединенной триангуляции на множестве $\mathbf{B} \cup \mathbf{W}$ без учета информации о исходных триангуляциях. Такой подход не является эффективным. В [3] доказана теоретическая оценка времени слияния неразделенных триангуляций Делоне, равная в худшем случае $O(n)$. Ранее были попытки реализовать метод, описанный в [3], но полностью линейный алгоритм так и не был реализован.

В настоящей работе будет описан линейный алгоритм слияния триангуляций Делоне, показывающий правильность описанного в [3] метода, а также проведены эксперименты, доказывающие его теоретическую оценку.

3 Метод слияния триангуляций

Используя терминологию, введенную в [3] определим несколько понятий.

В объединенной триангуляции Делоне $Del(\mathbf{B} \cup \mathbf{W})$ существуют ребра двух типов: одноцветные, которые были взяты из исходных триангуляций, и разноцветные, сайты-вершины которых взяты из разных исходных триангуляций.

Множество вершин и ребер (ребер и граней) называется *связным*, если для любой пары элементов этого множества существует цепь из попарно инцидентных вершин и ребер (ребер и граней), принадлежащих этому множеству.

В объединенной триангуляции существуют максимальные связные одноцветные подмножества вершин и ребер, переходящие без изменений из исходных триангуляций, которые будем называть *лоскутами*. Ребра и грани, не вошедшие в лоскуты, должны быть разрушены. Такие связные подмножества будем называть *разрезами*. При построении разрезов будут удалены некоторые грани, что приведет к образованию открытых ребер. Связную цепочку вершин и открытых ребер, состоящих из ребер одного цвета, будем называть *краем*. Максимальные связные подмножества разноцветных ребер и граней объединенной триангуляции будем называть *швами*. Разноцветные ребра будем называть *стежками*.

При данных определениях построение объединенной триангуляции Делоне состоит из нескольких частей: построения разрезов, выделения лоскутов исходных триангуляций и построения швов. Каждый шов соединяет два разноцветных края, образованных после разрушения ребер, вошедших в разрезы. В начале работы алгоритма краями исходных триангуляций являются их выпуклые оболочки. По мере построения разрезов и швов количество краев меняется (увеличивается при построении разрезов и уменьшается при построении швов). В объединенной триангуляции остается только один край — граница выпуклой оболочки.

Стартером будем называть пару разноцветных сайтов, образующих ребро Делоне, еще не включенное в объединенную триангуляцию. Стартер нужен для запуска процесса построения разрезов и швов. Построение смежных стежков для стартера можно вести по разные стороны от него. Для определенности будем полагать, что стартер имеет левый и правый сайты, построение стежка происходит в направлении перед стартером. На рис. 2 черная жирная линия обозначает стартер, жирная пунктирная новый стежок, построенный в направлении перед стартером.

Шов может быть двух типов. *Разомкнутым* швом называется шов, имеющий два стежка, принадлежащих выпуклой оболочке объединенной триангуляции, то есть принадлежащих граничным ребрам $Del(\mathbf{B} \cup \mathbf{W})$. *Циклическим* называется шов, име-

Рис. 2: Построение новых стежков

ющий только внутренние ребра объединенной триангуляции.

В процессе построения разомкнутого шва сначала находим одну его часть, затем продолжаем движение по другую сторону стартера и находим оставшуюся часть разреза. В циклическом шве в некоторый момент времени стартер совпадет с новым стежком, на этом построение шва будет закончено.

Введем понятие минимального остовного дерева. *Минимальным остовным деревом (МОД)* триангуляции Делоне называется ее связный подграф, имеющий наименьшую суммарную длину ребер. Пусть на плоскости задано N точек. *Евклидовым МОД (ЕМОД)* называется связный подграф, вершинами которого являются все N точек, суммарная длина всех ребер которого минимальна. Известно, что МОД ТД является евклидовым минимальным остовным деревом для множества сайтов ТД [5, стр. 229, 277]. МОД исходных триангуляций Делоне понадобятся в процессе построения стартеров.

Согласно определениям, данным выше, можно построить общую схему алгоритма слияния перекрывающихся триангуляций Делоне.

1. Построить минимальные остовные деревья для обеих исходных триангуляций.
2. Построить начальный стартер
3. Построить разрез и шов
   (a) Объявить стартер текущим стежком
   (b) Провести коррекцию одноцветных ребер (построение разреза)
   (c) Построить новый стежок
   (d) Если построить новый стежок не удалось, развернуть стартер и перейти к пункту 3
   (e) Если построить новый стежок не удалось во второй раз, закончить построение шва

(f) Удалить ребра, пересекающие новую грань

(g) Проверить совпадение со стартером, в случае совпадения закончить построение шва, иначе перейти к шагу 3c

4. Поиск очередного стартера

5. Если стартер найден, перейти к пункту 3, иначе закончить работу алгоритма

4 Описание алгоритма

В данном разделе будет приведено подробное описание алгоритма слияния перекрывающихся триангуляций Делоне на основе теоретического метода, описанного в [3].

Пусть множество $\mathbf{B}$ содержит n_1 точек с координатами $(x_{11}, y_{11}), (x_{12}, y_{12}), \ldots, (x_{1n_1}, y_{1n_1})$, множество $\mathbf{W}$ — n_2 точек с координатами $(x_{21}, y_{21}), (x_{22}, y_{22}), \ldots, (x_{2n_2}, y_{2n_2})$. На вход алгоритму подаются координаты первого и второго множества точек исходных триангуляций:

$$\begin{aligned} points[0] &= \{(x_{11}, y_{11}), (x_{12}, y_{12}), \ldots, (x_{1n_1}, y_{1n_1})\} \\ points[1] &= \{(x_{21}, y_{21}), (x_{22}, y_{22}), \ldots, (x_{2n_2}, y_{2n_2})\} \end{aligned} \tag{1}$$

Объединенное множество точек $\mathbf{B} \cup \mathbf{W}$, содержащее $n = n_1 + n_2$ точек, будем обозначать:

$$points[2] = points[0] + points[1]\} \tag{2}$$

На вход также подается некая структура данных, содержащая информацию о ребрах и/или гранях исходных триангуляций. Выбор структуры данных сильно влияет на дальнейшую сложность вычислений. Поэтому, рассмотрим более подробно этот пункт.

4.1 Выбор структуры данных

В книге [1, стр. 11-17] описаны несколько структур данных, которые могут быть использованы для решения поставленной задачи.

В нашей задаче будут часто производиться коррекции пучков сайтов, поэтому операцию добавления и удаления ребер из пучка нужно производить очень эффективно. В связи с этим требованием была выбрана структура данных *«Узлы с соседями»* (рис.3). Такая структура для каждого сайта хранит его координаты на плоскости и список номеров смежных сайтов в обходе против часовой стрелки. Рассмат-

риваемая структура данных неявно содержит ребра триангуляции, грани в данной структуре не хранятся вообще, что в нашей задаче, вообще говоря, и не нужно.

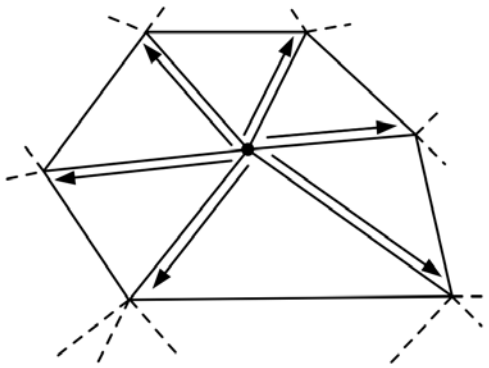

Рис. 3: Структура данных *«Узлы с соседями»*

Для хранения описанной выше информации для каждого сайта создается двунаправленный список номеров смежных сайтов в порядке обхода против часовой стрелки в соответствии с их номером в объединенном множестве точек (2):

$$neighbors = \{list_1, list_2, \ldots, list_n\}, \tag{3}$$

где $list_i$ — список смежных сайтов в порядке обхода против часовой стрелки для i-ого сайта.

Пусть сайт A предшествует сайту B в пучке сайта C, тогда верно:

$$\begin{aligned} A_{num} &= neighbors[C_{num}].prev(B_{num}) \\ B_{num} &= neighbors[C_{num}].next(A_{num}) \end{aligned} \tag{4}$$

4.2 Построение разрезов и швов триангуляций Делоне

Процесс построения разрезов и швов основан на проверке условия Делоне для ребер триангуляции, которую можно осуществлять при помощи углового критерия [3] :

Лемма 1. *Пусть* $\boldsymbol{S}$ *— множество сайтов, для пары* $A, B \in \boldsymbol{S}$ *выполнено условие Делоне* $\Leftrightarrow$ *для любых других двух сайтов* $C, D \in \boldsymbol{S}$*, лежащих по разные стороны от* AB*, справедливо:* $\angle ACB + \angle ADB \leq 180^\circ$ *и* $\angle ABD \geq \angle ACD$

Пусть найдена разноцветная пара сайтов A и B, такая что ребро AB образует ребро Делоне в объединенной триангуляции, ещё не включенное в нее. Например,

AB является стартером. Присоединение ребра AB к пучкам сайтов-вершин A и B может привести к нарушению условия Делоне для некоторых ребер из этих пучков. Все ребра, не удовлетворяющие условию Делоне, должны быть разрушены на этом этапе. Рассмотрим, как вставляется ребро AB в пучок сайта A.

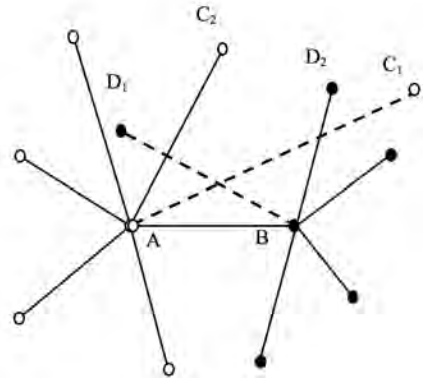

Рис. 4: Коррекция одноцветных ребер

Пусть AC_1 и AC_2 одноцветные ребра пучка A, что сайты B, C_1 и C_2 идут в порядке обхода против часовой стрелки в пучке сайта A (рис. 4). Для ребра AC_1 нарушено условие Делоне, если $\angle AC_2C_1 + \angle ABC_1 > 180°$. В этом случае ребро AC_1 удаляется из объединенной триангуляции, а ребро AC_2 подвергается аналогичной проверке. Если же для ребра AC_1 условие Делоне выполнено, то ребро AC_1 сохраняется в объединенной триангуляции, проверка условия Делоне для остальных ребер не производится.

Аналогичным образом выполняется коррекция пучка сайта A в направлении обхода по часовой стрелке. После коррекции пучка сайта A в обе стороны ребро AB разворачивается в противоположную сторону BA и осуществляется аналогичная проверка пучка сайта B при добавлении в него сайта A.

Приведем формальное описание процедуры коррекции пучков сайтов (алгоритм 1), параметрами являются номера сайтов A и B, для которых проводится коррекция.

Пучки, для которых выполнена описанная коррекция, будем называть *правильными*. Рассмотрим процесс образования новых разноцветных ребер объединенной триангуляции Делоне (лемма 2, [3]):

Лемма 2. *Пусть пучки сайтов A и B разноцветного ребра AB являются правильными. Точка C следует за точкой B в пучке сайта A, точка D предшествует точке A в пучке сайта B, тогда если $\angle ACB > \angle ADB$, то CB является новым ребром Делоне, иначе AD является новым ребром Делоне.*

Лемма 2 описывает процесс образования новых разноцветных стежков при по-

Алгоритм 1 Коррекция пучков сайтов

```
1: procedure deleteWrongeEdges(a, b, reverse = 0)
2:     while True do
3:         c_1 := neighbors[a].prev(b) // c_1 := neighbors[a].next(b)
4:         c_2 := neighbors[a].prev(c_1) // c_2 := neighbors[a].next(c_1)
5:         if ∠abc_1 + ∠ac_2c_1 ≤ 180° then
6:             break
7:         else
8:             Удалить ребро ac_1, продолжить коррекцию
9:         end if
10:    end while
11:    if reverse == 0 then
12:        deleteWrongeEdges(b, a, 1)
13:    end if
14: end procedure
```

строении шва. Из начального стежка AB получаем следующий стежок AD или CB. Основная сложность слияния неразделенных триангуляций состоит в том, что новая грань может быть пересечена ребрами исходных триангуляций, которые должны быть удалены во время построения очередного стежка.

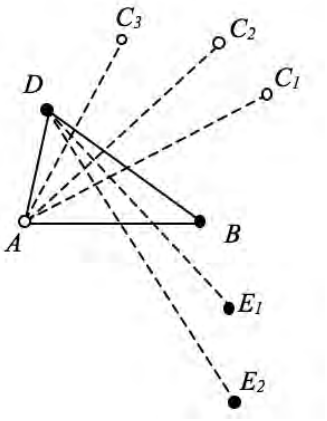

Рис. 5: Пресечение ребер исходных триангуляций с новой гранью $\triangle ABD$

Пусть AB — текущее разноцветное ребро Делоне, не ограничивая общности будем считать, что AD — новое разноцветное ребро (рис. 5).

По лемме 2 AD не может иметь пересечений со старыми одноцветными ребрами из пучков сайтов A, B и D. Про два оставшихся ребра AB и BD так утверждать нельзя. Действительно, все ребра в пучке A, лежащие между AB и AD, обязательно

пересекают одноцветное ребро BD, по алгоритму построения ребра AD все эти ребра не войдут в объединенную триангуляцию и должны быть разрушены.

Рассмотрим теперь пучок сайта D, в который включено вновь образованное ребро Делоне AD. В этом пучке также могут найтись одноцветные ребра, лежащие между DA и DB, которые пересекают ребро Делоне AB и должны быть разрушены. Поиск и разрушение указанных ребер осуществляется на основе последовательного просмотра пучка A от AB до AD против часовой стрелки и пучка D от DA до DB по часовой стрелке.

Таким образом, если найдено разноцветное ребро, удовлетворяющее условию Делоне, то процесс построения разреза и шва можно проделывать рассмотренным способом. Процедура, описывающая этот процесс, представлена в алгоритме 2. На вход подаются номера точек a и b, являющихся стартером.

4.3 Поиск стартеров

В задаче поиска первого и последующих стартеров используется различная информация о триангуляциях. Первый стартер ищется на основе начальной информации о триангуляциях, в процессе поиска последующих используется информация о уже построенных швах и разрушенных при этом одноцветных ребер.

Рассмотрим условия существования стартеров (разноцветных ребер Делоне). Лемма 3 [3] заключает в себе основную идею построения стартеров, поэтому приведем ее с доказательством.

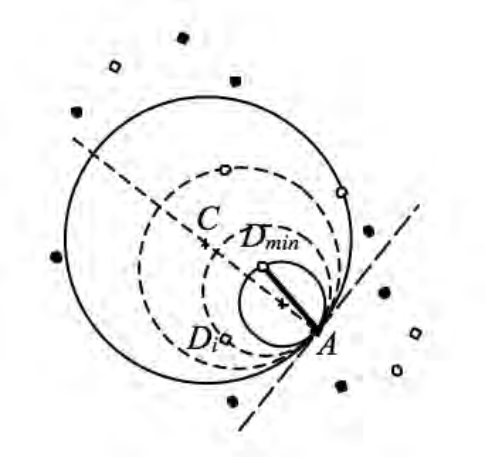

Рис. 6: Условие существования стартера

Лемма 3. *Сайт имеет инцидентное разноцветное ребро Делоне ⇔ существует инцидентная ему окружность, внутри которой нет сайтов одного с ним цвета, но есть сайты противоположного цвета на ней или внутри нее.*

Алгоритм 2 Построение разреза и шва триангуляции

```
1: procedure sewTriangle(starter, reverse = 0)
2:     [a, b] := starter
3:     while True do
4:         deleteWrongEdges(a, b)
5:         c := neighbors[a].prev(b)
6:         d := neighbors[b].next(a)
7:         if (∠acb > ∠adb & ∠acb < π) ∨ (∠acb < π & ∠adb ≥ π) then
8:             Добавить ребро cb в триангуляцию          ▷ cb является новым ребром
9:             Удалить все ребра между ca и cb в обходе против часовой стрелки
10:            Удалить все ребра между bc и ba в обходе против часовой стрелки
11:            if [c, b] ≠ starter then
12:                a := c
13:            else
14:                reverse := 1                          ▷ Шов является циклическим
15:                break
16:            end if
17:        else if (∠adb > ∠acb & ∠adb < π) ∨ (∠adb < π & ∠acb ≥ π) then
18:            ad является новым ребром, аналогично ребру cb
19:        else
20:            break
21:        end if
22:    end while
23:    if reverse == 0 then
24:        sewTriangle(b, a, 1)
25:    end if
26: end procedure
```

Доказательство. Достаточность. Пусть для некоторого сайта A существует инцидентная ему окружность с центром C, внутри которой нет ни одного сайта того же цвета, что и A, но есть сайты противоположного цвета $D_1, D_2, \ldots, D_k, k \geq 1$ другого цвета внутри или на окружности (рис. 6). Рассмотрим множество окружностей, инцидентных парам сайтов A и $D_i, i = 1, \ldots, k$, имеющих в точке A общую касательную с окружностью с центром в точке C. Окружность, имеющая минимальный радиус, является пустой, и она инцидентна разноцветной паре сайтов. Значит, эта пара сайтов образует ребро Делоне и может являться стартером.

Необходимость. Есть пара сайтов, образующая ребро Делоне с инцидентной пустой окружностью. Эта окружность и будет удовлетворять условию леммы. □

Согласно лемме 3, если существует сайт A и инцидентная ему окружность O, удовлетворяющая условию леммы, то поиск стартера сводится к перебору сайтов $D_1, D_2, \ldots, D_k, k \geq 1$, отличного от A цвета, внутри этой окружности и выбору сайта D_{min}, такого что инцидентная A и D_{min} окружность имеет с окружностью O общую касательную в точке A и минимальный радиус из всех таких окружностей, инцидентных A и D_i. Тогда ребро AD_{min} можно использовать в качестве стартера.

4.3.1 Поиск первого стартера

Будем говорить, что сайт A *лежит левее* сайта B, если A предшествует B при лексикографическом упорядочивании сайтов по возрастанию координат (x, y).

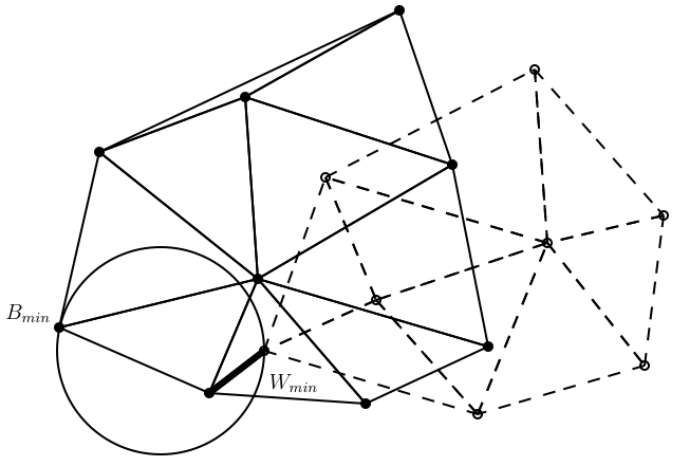

Рис. 7: Поиск первого стартера

Пусть **B** и **W** множества сайтов исходных триангуляций. Найдем сайты B_{min} и W_{min}, являющиеся самыми левыми в соответствующем множестве точек **B** и **W**. Не ограничивая общности, будем считать, что B_{min} лежит левее, чем W_{min} (рис. 7). Рассмотрим окружность O, инцидентную сайтам B_{min} и W_{min} с центром на луче, параллельном оси Ox, выходящим из точки W_{min} влево. Окружность O не содержит ни одного сайта из **W**, потому что лежит левее самого левого сайта этого множества W_{min}, и содержит сайт B_{min} на границе окружности. Возможно, ещё часть сайтов из **B** попадет внутрь этой окружности. Тогда точка W_{min} и окружность O удовлетворяют условиям леммы 3. То есть первый стартер найден. Алгоритм 3 описывает поиск первого стартера.

Алгоритм 3 Поиск первого стартера

```
1: procedure FINDFIRSTSTARTER()
2:     b — самая левая точка множества points[0]
3:     w — самая левая точка множества points[1]
4:     starter_begin := (b левее w) ? w : b
5:     num := (b левее w) ? 0 : 1
6:     R_min := Inf
7:     for p ∈ points[num] do
8:         r — радиус окружности с центром на луче из точки w влево, инцидентной
       сайтам p и w
9:         if r < R_min then
10:            R_min := r
11:            starter_end := p                    ▷ Объявить p концом стартера
12:        end if
13:    end for
14:    return [starter_begin, starter_end]
15: end procedure
```

Время поиска первого стартера складывается из вычисления самых левых точек множеств **B** и **W** и перебора точек из множества **B** по алгоритму из леммы 3, то есть линейно по общему числу сайтов.

4.3.2 Минимальные остовные деревья

Задача поиска последующих стартеров основана на использовании минимальных остовных деревьев исходных триангуляций Делоне. Теоретическая оценка трудоемкости задачи построения минимального остова составляет $(n \log n)$. В то же время известно, что на основе триангуляции Делоне ЕМОД может быть построен за линейное время (рис. 8). Такой вычислительной сложностью обладает алгоритм Черитона-Тарьяна [5, стр. 278-280],

Построение минимальных остовных деревьев осуществляется в процессе предобработки, предшествующем непосредственному слиянию исходных триангуляций. Рассмотрим подробнее алгоритм Черитона-Тарьяна. *Лесом* называется упорядоченное множество деревьев. На каждом шаге построения МОД алгоритм обрабатывает лес, содержащий несколько деревьев, которые в ходе данного алгоритма станут поддеревьями искомого остовного дерева. Сначала каждое дерево леса состоит из одной вершины триангуляции без ребер, размер леса равен числу точек в триангуляции.

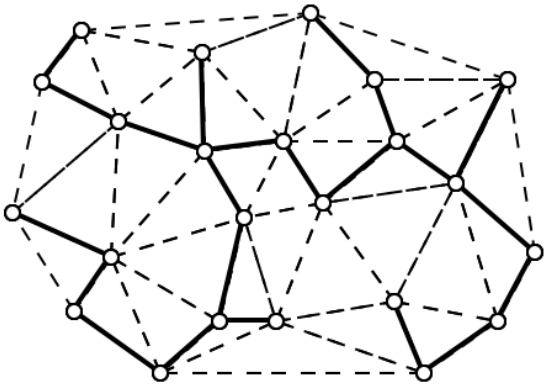

Рис. 8: Евклидово минимальное остовное дерево триангуляции Делоне

Для выбора дерева T (которое будет объединено с другим деревом леса) они указали однородное правило выбора, перед использованием которого в очередь помещаются все деревья, содержащие по одной вершине.

1. Выбрать дерево из начала очереди.

2. Если дерево T'' получено в результате объединения дерева T с некоторым другим деревом T', то удалить T и T' из очереди и добавить T'' в конец очереди.

Каждому дереву T из очереди поставим в соответствие целое число, называемое *этапом* и вычисляемое следующим образом: $stage(T) = 0$, если $|T| = 1$, и $stage(T) = \min(stage(T'), stage(T'')) + 1$, если T является объединением деревьев T' и T''. В любой момент работы алгоритма последовательность чисел $stage(T)$ является неубывающей от начала к концу очереди. Будем говорить, что этап j завершен, если из очереди удалено дерево T, для которого $stage(T) = j$, и ни для одного другого дерева T в очереди значение $stage(T)$ не равно j.

Далее выбрав из очереди первый элемент T ($stage(T) = j$), среди ребер, соединяющих вершины дерева T с вершинами вне его, ищем ребро наименьшей длины. При такой организации обработки на каждом этапе каждое ребро просматривается не более двух раз (кроме ребер, уже вошедших в некоторое дерево T).

Метод, позволяющий получить Евклидово МОД из триангуляции Делоне за линейное время, использует операцию «очистки», предложенную Черитоном и Тарьяном. Цель операции очистки состоит в том, чтобы сжать исходный граф G (граф триангуляции Делоне), преобразовав его в некоторый граф G^*, который в любой момент работы алгоритма содержит лишь необходимую информацию. Это означает, что каждое дерево T, входящее в лес F, сжимается в единственную вершину графа G (т. е. удаляются все неотобранные ребра, соединяющие вершины дерева T)

и удаляются все ребра, за исключением самого короткого из неотобранных, соединяющие деревья T' и T''. Очистка графа производится сразу же после завершения этапа. Предположим, что с каждым деревом T связан список неотобранных ребер, инцидентных вершинам дерева T. Формальное описание процедуры построения евклидова минимального остовного дерева описано в алгоритме 4.

Алгоритм 4 Построение ЕМОД по триангуляции Делоне

```
1:  procedure makeMST()
2:      F := ∅
3:      for i := 1 to n do
4:          stage(points[2][i]) := 0
5:          F.insert(points[2][i])
6:      end for
7:      j := 1
8:      while F ≠ ∅ do
9:          T' := F.get()
10:         if stage(T') == j then
11:             Очистить граф
12:             j := j + 1
13:         end if
14:         (u, v) — кратчайшее из неотобранных ребер, инцидентных T' (u ∈ T')
15:         T'' := дерево в F, содержащее v
16:         T := merge(T', T'')
17:         Удалить T'' из F
18:         stage(T) := min(stage(T'), stage(T'')) + 1
19:         F.insert(T)
20:     end while
21: end procedure
```

Поиск последующих стартеров основан на том, что при построении разреза в триангуляции Делоне нарушается ее связность, при этом среди разрушенных ребер обязательно окажется ребро МОД этой триангуляции Делоне.

Будем называть окружность, диаметром которой является ребро МОД, *окружностью влияния ребра*. Леммы 4 и 5 описывают свойства минимальных остовных деревьев, которые понадобятся для теоретической оценки сложности алгоритма.

Лемма 4. *Окружность влияния ребра МОД является пустой окружностью ТД.*

Лемма 5. *Расстояние между центрами двух ребер МОД не меньше, чем половина длины каждого ребра.*

4.3.3 Поиск последующих стартеров

Мостами будем называть ребра исходных триангуляций, входящие в МОД, разрушенные при построении очередного разреза. Сайт, инцидентный мосту, называется *свободным*, если он имеет только одноцветные инцидентные ребра. Сайт называется *закрепленным*, если он инцидентен мосту, но среди инцидентных ему ребер есть разноцветные. Свободные сайты могут быть присоединены к объединенной триангуляции только при помощи построения новых швов, поэтому существование свободного сайта говорит о том, что построение объединенной триангуляции ещё не завершено:

Лемма 6. *Существует стартер, инцидентный свободному сайту.*

Пусть сайты A и B — разноцветные, сайт A входит в триангуляцию Делоне T. *Максимальными пустыми окружностями* будем называть пустые окружности, которые не содержатся внутри других пустых окружностей, не совпадающих с ними. Рассмотрим множество окружностей, инцидентных A и являющихся пустыми относительно сайтов триангуляции T. В этом множестве существую максимальные пустые окружности, которые являются описанными окружностями граней триангуляции T, инцидентные сайту A. В случае, когда A принадлежит границе выпуклой оболочки триангуляции Делоне T, несобственные окружности также будут максимальными пустыми окружностями, инцидентными сайту A.

Лемма 7. *Пусть AB — стартер. Тогда сайт B попадает внутрь хотя бы одной из максимальных пустых окружностей в T, инцидентных сайту A.*

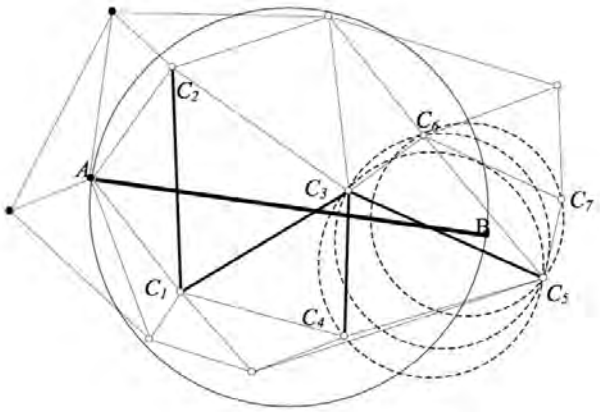

Рис. 9: Поиск последующих стартеров

Пусть AB — мост, сайт A является закрепленным, сайт B — свободным (рис. 9). Сайты A и B имеют один и тот же цвет (на рисунке – черный) и в окружности влияния моста находятся лишь сайты противоположного цвета. Согласно лемме 6 свободный сайт B может быть выбран в качестве первого сайта стартера. Сайт B попадает внутрь максимальных пустых окружностей триангуляции белых сайтов. На рисунке это описанные окружности треугольных граней $\triangle C_3C_4C_5,\ \triangle C_3C_5C_6,\ \triangle C_6C_5C_7$. Согласно лемме 7 для нахождения второго сайта стартера можно выполнять перебор не по всем вершинам, а только по вершинам граней, внутри описанных окружностей которых лежит сайт B. Таким образом, поиск второго сайта осуществляется среди вершин этих граней, причем только тех из них, которые попадают внутрь окружности влияния моста, в нашем примере среди сайтов C_3, C_4 и C_6.

Сайт A уже включен в объединенную триангуляцию, поэтому можно оценить положение свободного сайта B относительно этой триангуляции. Сайт B либо попадает внутрь какой-то грани, либо лежит вне выпуклой оболочки второй триангуляции. В последнем случае можно найти ребро выпуклой оболочки, определяющее полуплоскость, в которой лежит B. Задачу определения местоположения свободного сайта относительно триангуляции, в которую уже включен закрепленный сайт, будем называть *локализацией моста*. Рассмотрим множество ребер триангуляции, которые пересекает мост. Эти ребра естественным образом упорядочиваются в направлении от закрепленного сайта к свободному. Легко видеть, что последнее из пересекаемых мостом ребер либо инцидентно треугольной грани, содержащей сайт B, либо определяет полуплоскость, содержащую B. Найденная грань или полуплоскость есть решение задачи локализации моста. Таким образом, задача локализации моста решается путем трассировки ребер и граней триангуляции вдоль моста.

Найденная грань имеет описанную окружность, которая содержит свободный сайт B. Поиск остальных граней триангуляции, чьи описанные окружности содержат B, осуществляется путем проверки смежных граней. Объединением этих граней является многоугольник, содержащий сайт B. В рассматриваемом примере это многоугольник $C_3C_4C_5C_7C_6$. Среди вершин многоугольника обязательно найдутся сайты, лежащие внутри окружности влияния моста (в противном случае мост не был бы разрушен). Один из этих сайтов и образует стартер вместе с сайтом B.

Таким образом, поиск стартера на основе свободного сайта и инцидентного ему моста может быть осуществлен алгоритмом 5:

Алгоритм 5 Поиск последующих стартеров

procedure FINDNEXTSTARTER($bridge$)
    $open$ — открытый сайт ребра $bridge$
    Локализация $open$ относительно триангуляции, которой он не принадлежит
    $tempPoints$ — множество сайтов, инцидентных грани, содержащей $open$
    $faces$ — смежные грани триангуляции, в описанные окружности которых попадает сайт $open$
    Вершины, инцидентные граням из $faces$ добавляем в $tempPoints$
    for $p \in tempPoints$ **do**
        if p не попадает в окружность влияния $bridge$ **then**
            Удалить p из $tempPoints$
        end if
    end for
    $R_{min} := Inf$
    $starter_{end} := -1$
    for $p \in tempPoints$ **do**
        r — радиус окружности с центром на $bridge$, инцидентной сайтам p и $open$
        if $r < R_{min}$ **then**
            $R_{min} := r$
            $starter_{end} := p$
        end if
    end forreturn $[open, starter_{end}]$
end procedure

4.4 Общая структура алгоритма

В предыдущих пунктах был подробно описан каждый шаг алгоритма слияния перекрывающихся триангуляции. Обобщим полученные результаты и напишем формальную процедуру слияния (алгоритм 6).

5 Оценка вычислительной сложности алгоритма

Алгоритм слияния перекрывающихся триангуляций Делоне можно разделить на три части: построение минимальных остовных деревьев исходных триангуляций Делоне, поиск стартеров, построение разрезов и швов. Алгоритм построения МОД Черитона-Тарьяна имеет сложность $O(n)$, что доказано в [5]. Вычислительную сложность других частей алгоритма рассмотрим более подробно.

Алгоритм 6 Слияние триангуляций

```
1: procedure makeConcatenation()
2:     makeMST()
3:     starter := findFirstStarter()
4:     Добавить starter в триангуляцию
5:     bridges — множество всех мостов, пополняется при построении разрезов и швов
6:     sewTriangle(starter)
7:     while bridges ≠ ∅ do
8:         starter := findNextStarter(bridges.get())
9:         Добавить starter в триангуляцию
10:        sewTriangle(starter)
11:    end while
12: end procedure
```

5.1 Сложность поиска стартеров

Пусть $n = |\mathbf{B}| + |\mathbf{W}|$ — общее количество сайтов объединенной триангуляции. Поиск первого стартера имеет сложность $O(n)$. Поиск последующих стартеров включает в себя два перебора точек: во-первых, это выбор моста —- ребра МОД, разрушенного при разрезании триангуляций, во-вторых, это пересечение мостом ребер триангуляции при локализации моста. Можно построить примеры таких «худших случаев», когда число мостов окажется $O(n)$ и число ребер объединенной триангуляции, пересекающих отдельный мост, также $O(n)$. Ниже будет показано, что общее число всех пересечений мостов и ребер исходных триангуляций все равно имеет сложность $O(n)$.

Рассмотрим процесс выбора очередного моста для трассировки. Число ребер МОД равно $O(n)$. При построении разрезов и швов все мосты заносятся в отдельный список. Для дальнейшего поиска стартера выбирается любой мост, то есть это занимает времени $O(1)$. В [3] доказано, что поиск последующих стартеров имеет сложность $O(n)$. То есть общее время на поиск стартеров также равно $O(n)$.

5.2 Сложность построения разрезов и швов

На этом шаге алгоритма разрушаются одноцветные ребра, которые перестали удовлетворять условию Делоне в объединенной триангуляции, и строятся разноцветные ребра. Построение нового ребра осуществляется на основе углового критерия

(лемма 1) для пары конкурирующих ребер, т.е. занимает фиксированное время $O(1)$. Для каждого нового разноцветного ребра выполняется оценка корректности соседних с ним одноцветных ребер. Такая проверка дважды запускает угловой критерий. Если по результатам тестирования какое-то ребро уничтожается, то ставшее соседним другое одноцветное ребро вновь подвергается тесту. Таким образом, время на удаление одного одноцветного ребра также оценивается как постоянное $O(1)$.

К общему времени стоит ещё добавить время включения нового ребра в триангуляцию. При включении первого стежка, образуемого из стартера, может потребоваться полный перебор всех ребер, входящих в пучки двух инцидентных ему сайтов. Поэтому, общее время включения всех начальных ребер в пучки оценивается как $O(n)$. А вот включение каждого очередного ребра уже не требует перебора ребер, поскольку новое ребро устанавливается в пучок вслед за текущим анализируемым ребром. Поэтому общее время на включение всех ребер есть $O(n)$.

Общее число построенных ребер не превосходит количество ребер в выходной триангуляции, то есть $O(n)$, а общее число разрушенных ребер не превосходит общее число ребер в исходных триангуляциях — тоже $O(n)$.

Таким образом, получили, что время построения разрезов и швов равно $O(n)$. А, следовательно, и общее время работы алгоритма равно $O(n)$.

6 Эксперименты

В данном разделе будет описана программная реализация рассматриваемого алгоритма, а также проведен ряд экспериментов по замеру времени работы алгоритма на входных данных разного размера.

6.1 Программная реализация

Для реализации алгоритма слияния перекрывающихся триангуляций Делоне был разработан класс на языке программирования Python. Данный класс имеет поля, хранящие для каждого сайта исходных триангуляций информацию о его координатах и о упорядоченном списке инцидентных сайтов. А также содержит все вышеописанные методы: поиск первого стартера, реализация минимальных остовных деревьев по триангуляции Делоне, построение разреза и шва от заданного стартера и поиск последующих стартеров.

Для тестирования программы можно использовать как автоматически сгенерированные облака точек исходных триангуляций, так и введенные вручную точки.

6.2 Вычислительный эксперимент

Эксперимент по проверке вычислительной сложности предложенного алгоритма проводился на данных разной размерности. Случайным образом были сгенерированы множества точек для двух триангуляций:

$$n_1 = n_2 = 10,\ 20,\ 30,\ \ldots,\ 700$$

В сравнение описанному линейному алгоритму ставился алгоритм «разделяй и властвуй», который строит триангуляцию Делоне на объединенном множестве точек, не учитывая информацию о исходных триангуляциях, и имеет сложность $O(n \log n)$. График замера времени показан на рисунке 10. Синим обозначено усредненное время работы линейного алгоритма по нескольким запускам данных фиксированного размера, зеленым — алгоритма «разделяй и властвуй». По графику видно, что предложенный алгоритм слияния работает быстрее, чем построение триангуляции по объединенному множеству точек, и имеет выраженную линейную тенденцию.

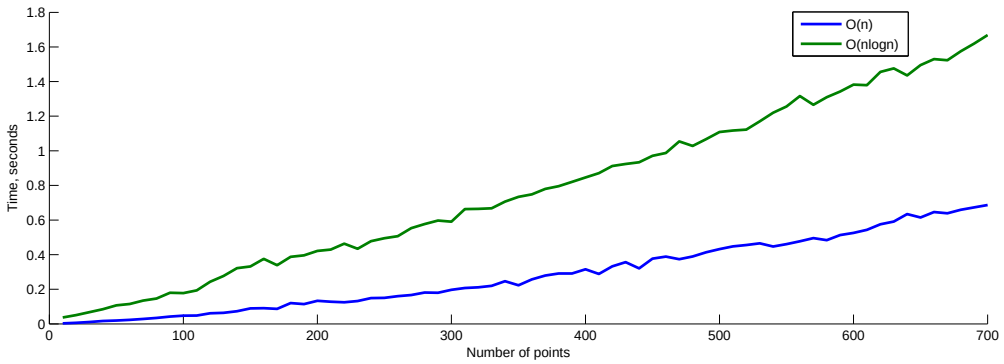

Рис. 10: Время работы алгоритма слияния триангуляций

Пример работы алгоритма представлен на рис. 11. Как и требуется в алгоритме, в качестве исходных данных подаются две триангуляции, результатом работы алгоритма является объединенная триангуляция.

7 Заключение

В данной работе рассматривалась задача слияния двух перекрывающихся триангуляций Делоне за линейное время. По теоретическому методу [3] был разработан алгоритм, позволяющий решать данную задачу. Проведенные вычислительные эксперименты подтвердили корректность и эффективность разработанного алгоритма. Было выявлено, что слияние перекрывающихся триангуляций Делоне предпо-

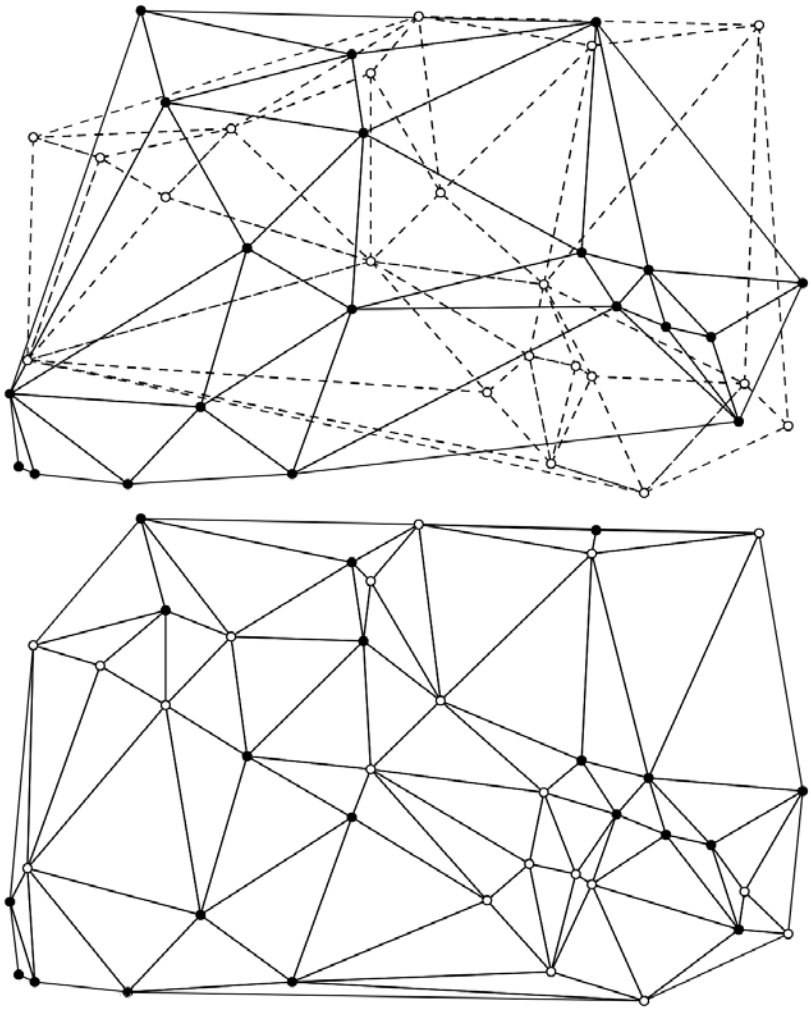

Рис. 11: Входные данные и результат слияния

чтительнее разрушения обеих триангуляций и построения искомой триангуляции по объединенному множеству точек.

Основным результатом данной работы является разработка структур данных и алгоритма слияния линейно неразделимых триангуляций Делоне.

На защиту выносятся:

1. Разработка структур данных
2. Разработка алгоритма слияния перекрывающихся триангуляций Делоне за линейное время
3. Программная реализация алгоритма

Список литературы

[1] А.В. Скворцов. Триангуляция Делоне и ее применение. Москва: Издательство Томского университета, 2002.

[2] Н.Ф. Дышкант. Эффективные алгоритмы сравнения поверхностей, заданных облаками точек. Кандидатская диссертация, МГУ имени М.В.Ломоносова, 2011.

[3] Л.М. Местецкий, Е.В. Царик. Слияние неразделенных триангуляций Делоне // Сложные системы: обработка информации, моделирование и оптимизация. Вып. 2. Тверь: Тверской гос.университет, 2004. С. 216–231.

[4] Л.М. Местецкий. Лекции по вычислительной геометрии. Москва: ВМК МГУ, 2013.

[5] Ф. Препарата, М. Шеймос. Вычислительная геометрия: введение. Москва: Издательство Мир, 1989.