$O(n)$-алгоритм слияния перекрывающихся триангуляций Делоне

Леонид Местецкий[1], Мария Готман[1], Наталья Дышкант[2] и Елена Царик[3]

[1] *ВМК МГУ*
[2] *НИВЦ*
[3] *ТвГУ*

Аннотация

???

1 Введение

Задача построения триангуляции Делоне (ТД) n сайтов-точек имеет нижнюю оценку вычислительной сложности $O(n \log n)$. В случаях, когда имеется некоторая дополнительная информация о структуре и связях сайтов, эта оценка может быть улучшена. В частности, представляет интерес задача слияния двух триангуляций, когда исходное множество сайтов состоит из двух непересекающихся подмножеств $\mathbf{S} = \mathbf{S_1} \cup \mathbf{S_2}$, и триангуляции Делоне $Del(\mathbf{S_1})$ и $Del(\mathbf{S_2})$ исходных множеств $\mathbf{S_1}$ и $\mathbf{S_2}$ уже построены. В [1] описан алгоритм Ли-Шехтера слияния ТД за время $O(n)$ в случае, когда множества точек $\mathbf{S_1}$ и $\mathbf{S_2}$ линейно разделимы. Мы рассматриваем более общую постановку задачи слияния ТД, когда множества сайтов $\mathbf{S_1}$ и $\mathbf{S_2}$ не пересекаются $\mathbf{S_1} \cap \mathbf{S_2} = \emptyset$, но перекрываются, т.е. пересекаются их выпуклые оболочки $Conv(\mathbf{S_1}) \cap Conv(\mathbf{S_2}) \neq \emptyset$. Даны ТД $Del(\mathbf{S_1})$ и $Del(\mathbf{S_2})$, нужно построить ТД $Del(\mathbf{S_1} \cup \mathbf{S_2})$ за время $O(n)$ в худшем случае.

В такой постановке задача слияния ТД до сих пор не рассматривалась, возможно, потому, что не представляла практического интереса. Наш интерес к этой постановке связан с построением алгоритма для вычисления расстояния в метрическом пространстве функций двух переменных, заданных на конечных нерегулярных множествах точек. Эта задача возникает, в частности, при анализе 3D поверхностей человеческих лиц, полученных в результате пространственного сканирования [2]. Предложенная модель определяет расстояние между такими функциями на основе построения общей ТД. При этом расстояние между моделями лиц определяется как минимальное по всем движениям множеств $\mathbf{S_1}$ и $\mathbf{S_2}$ на плоскости. Таким образом, осуществляется подгонка пары поверхностей друг к другу. И на каждой итерации подгонки выполняется слияние триангуляций.

Поскольку ТД и диаграмма Вороного (ДВ) n сайтов являются двойственными графами и могут быть получены один из другого за время $O(n)$, для решения

задачи в принципе можно использовать известные алгоритмы Киркпатрика [3] и Шазеля [4]. Алгоритм Киркпатрика [3] строит ДВ $Vor(\mathbf{S_1} \cup \mathbf{S_2})$ перекрывающихся множеств сайтов $\mathbf{S_1}$ и $\mathbf{S_2}$ на основе слияния ДВ $Vor(\mathbf{S_1})$ и $Vor(\mathbf{S_2})$. Для решения нашей задачи нужно преобразовать исходные ТД $Del(\mathbf{S_1})$, $Del(\mathbf{S_2})$ в $Vor(\mathbf{S_1})$, $Vor(\mathbf{S_2})$, построить $Vor(S_1 \cup S_2)$ с помощью алгоритма [3], а затем преобразовать $Vor(\mathbf{S_1} \cup \mathbf{S_2})$ в $Del(\mathbf{S_1} \cup \mathbf{S_2})$. Алгоритм Шазеля [4] строит пересечение двух выпуклых многогранников в 3D-пространстве. Для его использования применительно к нашей задаче требуется построить из исходных ТД $Del(\mathbf{S_1})$, $Del(\mathbf{S_2})$ сначала ДВ $Vor(\mathbf{S_1})$, $Vor(\mathbf{S_2})$, затем выпуклые многогранники. Затем нужно построить пересечение этих многогранников с помощью алгоритма [4]. После этого из многогранника получается сначала ДВ $Vor(\mathbf{S_1} \cup \mathbf{S_2})$, а после $Del(\mathbf{S_1} \cup \mathbf{S_2})$. Таким образом, используя алгоритм [3, 4], мы получаем решение за 3 и 5 шагов соответственно. И хотя вычислительная сложность этих решений остается $O(n)$, их практическая реализация весьма проблематична. В отличие от этого, предлагаемый нами алгоритм находит $Del(\mathbf{S_1} \cup \mathbf{S_2})$ непосредственно на основе слияния исходных ТД $Del(\mathbf{S_1})$ и $Del(\mathbf{S_2})$.

Ещё одной причиной для прямого использования триангуляций Делоне служит то, что структура данных, описывающая ТД, обычно намного проще, чем структура ДВ. В ТД все ребра одного типа, и имеют конечную длину. Напротив, в ДВ существует специальный массив, хранящий бесконечные ребра диаграммы Вороного. Наличие этого факта существенно осложняет задачу описания алгоритмов слияния. Это и было одной из причин для разработки алгоритма Ли-Шехтера [1], хотя аналогичный алгоритм для ДВ, описанный в [5], был уже известен. Таким образом, актуальна задача описания и разработки эффективного алгоритма слияния пересекающихся триангуляций Делоне.

Структура статьи следующая:

2 Терминология и постановка задачи

Пусть на евклидовой плоскости задано $\mathbf{S}$ — множество из $n \geq 3$ точек, не все из которых лежат на одной прямой. Эти точки будем называть *сайтами*.

Триангуляцией конечного множества точек $\mathbf{S}$ называется планарный граф с вершинами из $\mathbf{S}$, 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)* $Del(\mathbf{S})$ множества точек $\mathbf{S}$ называется выпуклая триангуляция, у которой описанная окружность каждой тре-

угольной грани является пустой, т.е. все грани которой являются гранями Делоне.

Определение. *Триангуляцией Делоне* множества точек $\mathbf{S}$ 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.

Задача слияния двух перекрывающихся триангуляций Делоне ставится следующим образом. Даны два конечных линейно неразделимых множества сайтов $\mathbf{B}$ и $\mathbf{W}$ (считаем, что сайты раскрашены в два цвета: множество черных сайтов $\mathbf{B}$ и множество белых сайтов $\mathbf{W}$), а также их триангуляции Делоне $Del(\mathbf{B})$ и $Del(\mathbf{W})$. Нужно построить триангуляцию Делоне на объединенном множестве сайтов $Del(\mathbf{B} \cup \mathbf{W})$.

Пример исходных данных и искомой триангуляции приведен на рис. 1.

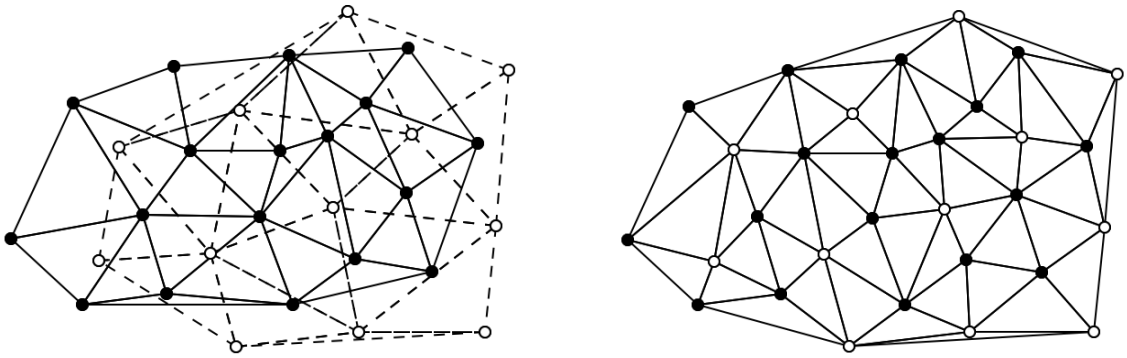

Рис. 1: Исходные триангуляции и объединенная триангуляция

3 Problem discussion and related work

4 Структура алгоритма

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.

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 при слиянии триангуляций, должны быть разрушены. Такие связные подмножества будем называть *разрезами*. При построении разрезов будут удалены некоторые грани, что приведет к образованию открытых ребер. При объединении будут образовываться новые разноцветные ребра, называемые *стежками*. Максимальные связные подмножества разноцветных ребер и граней объединенной триангуляции будем называть *швами*.

При данных определениях построение объединенной триангуляции Делоне состоит из нескольких частей: построения разрезов, выделения лоскутов исходных триангуляций и построения швов.

Стартером будем называть пару разноцветных сайтов, образующих ребро Делоне, еще не включенное в объединенную триангуляцию. Процесс нахождения стартера будет подробнее описан ниже. Стартер нужен для запуска процесса построения разрезов и швов. На рис. 2 черная жирная линия обозначает стартер, жирная пунктирная новый стежок, построенный в направлении перед стартером.

Рис. 2: Построение новых стежков

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, имеющий 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 ЕМОД для множества сайтов ТД [6, стр. 227, 277]. МОД исходных триангуляций Делоне понадобятся в процессе построения стартеров.

Общая схема слияния перекрывающихся триангуляций Делоне:

1. Поиск начального стартера
2. Построение разреза и шва от найденного стартера
3. Поиск очередного стартера
4. Если стартер найден, перейти к пункту 2, иначе закончить работу алгоритма

5 Описание алгоритма

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:

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

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

$$points = pointsB + pointsW \tag{2}$$

На вход также подается 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.

5.1 Выбор структуры данных

В нашей задаче будут часто производиться коррекции пучков сайтов, поэтому операцию добавления и удаления ребер из пучка нужно производить очень эффективно. Из предложенных в [7, стр. 11-17] для нашей задачи лучше всего подходит 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. Рассматриваемая структура данных неявно содержит ребра триангуляции, грани в данной структуре не хранятся вообще, что в нашей задаче, вообще говоря, и не нужно.

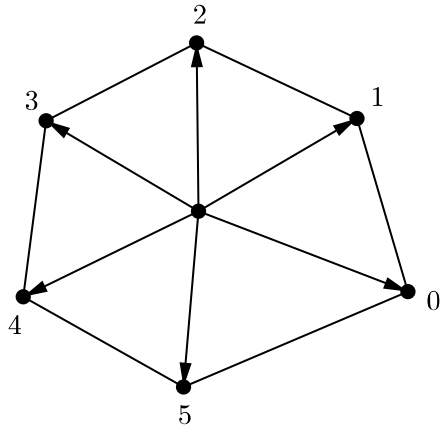

Рис. 3: Структура данных *«Узлы с соседями»*

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:

$$\begin{aligned} A &= neighbors[C].prev(B) \\ B &= neighbors[C].next(A) \end{aligned} \tag{4}$$

5.2 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:

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 $\angle ABD \geq \angle ACD$*

Доказательство. Докажем, что $\angle ACB + \angle ADB \leq 180^\circ$. Рассмотрим окружность, инцидентную сайтам A и B (рис.4 левый). Пусть E — середина AB. Отрезки EC и ED пересекают окружность в точках C' и D' соответственно. В четырехугольнике $AC'BD'$, вписанном в окружность, имеем $\angle AC'B + \angle AD'B = 180^\circ$. Если окружность пустая, то $\angle ACB \leq \angle AC'B$ и $\angle ADB \leq \angle AD'B$. Тогда получаем: $\angle ACB + \angle ADB \leq \angle AC'B + \angle AD'B = 180^\circ$. И наоборот, если $\angle ACB + \angle ADB > 180^\circ$, то пустой окружности, инцидентной A и B, не существует, что и доказывает утверждение.

Докажем второе утверждение. Предположим противное, пусть $\angle ABD < \angle ACD$. Построим описанную окружность для треугольника $\triangle ABD$ (рис.4 правый). Тогда C лежит внутри нее. Пусть C' — точка пересечения прямой CD с окружностью. В четырехугольнике $AC'BD$, вписанном в окружность, имеем $\angle AC'B + \angle ADB = 180^\circ$. Поскольку $\angle AC'B < \angle ACB$, получаем, что $\angle ACB + \angle ADB > 180^\circ$. А тогда для AB согласно первому утверждению леммы не выполняется условие Делоне. Значит, предположение $\angle ABD < \angle ACD$ неверно, и утверждение леммы выполняется. ■

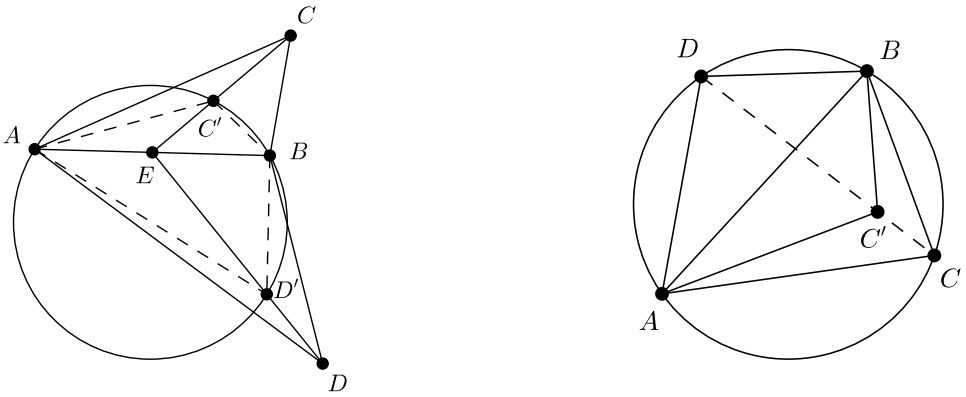

Рис. 4: Пояснение к лемме 1

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

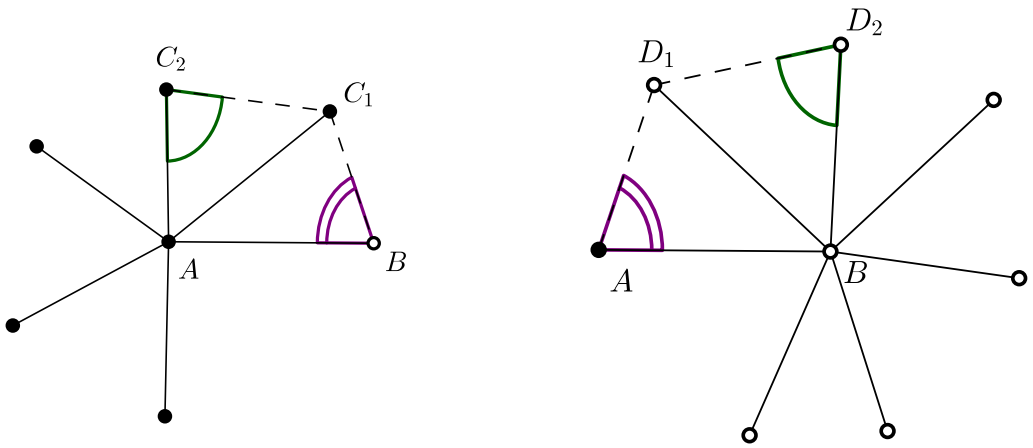

Рис. 5: Коррекция одноцветных ребер

ребер из этих пучков. Все ребра, не удовлетворяющие условию Делоне, должны быть разрушены на этом этапе. Рассмотрим, подробнее этот процесс.

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. 5 левый). 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 B в направлении обхода по часовой стрелке (рис. 5 правый).

После этого ребро AB разворачивается, и снова производятся действия по коррекции ребер. Таким образом, пучки сайтов A и B будут скорректированы как по часовой стрелке, так и против нее. На этом вставка нового ребра AB заканчивается.

Во время выполнения алгоритма часть ребер будет удалена. Все удаленные ребра занесем в отдельный список *deleteEdges*, эта информация понадобится нам при поиске последующих стартеров. Об этом подробно будет рассказано ниже.

Приведем формальное описание процедуры коррекции пучков сайтов (алгоритм 1), параметрами являются номера сайтов A и B, для которых проводится коррекция, *reverse* является необязательным параметром, нужен, чтобы развернуть ребро AB и продолжить коррекцию ребра BA.

Пучки, для которых выполнена описанная коррекция, будем называть *правильными*. Рассмотрим процесс образования новых разноцветных ребер объединенной триангуляции Делоне:

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.*

Доказательство. Поскольку AB ребро Делоне, то существует пустая окружность, инцидентная сайтам A и B (рис.6). Значит, сайты C и D лежат вне этой окружности. Рассмотрим описанные окружности треугольников $\triangle ACB$ и

Алгоритм 1 Коррекция пучков сайтов

```
1: procedure deleteWrongeEdges(a, b, reverse = 0)
2:     c_1 := neighbors[a].prev(b)                    ▷ Обрабатываем пучок сайта A
3:     c_2 := neighbors[a].prev(c_1)
4:     while ∠abc_1 + ∠ac_2c_1 > 180° do
5:         deleteEdges.add(ac_1)
6:         Удалить ребро ac_1 из ТД
7:         c_1 := c_2
8:         c_2 := neighbors[a].prev(c_2)
9:     end while
10:    d_1 := neighbors[b].next(a)                    ▷ Обрабатываем пучок сайта B
11:    d_2 := neighbors[b].next(d_1)
12:    while ∠bad_1 + ∠bd_2d_1 > 180° do
13:        deleteEdges.add(bd_1)
14:        Удалить ребро bd_1
15:        d_1 := d_2
16:        d_2 := neighbors[b].next(d_2)
17:    end while
18:    if reverse == 0 then          ▷ Развернуть ребро AB, продолжить проверку
19:        deleteWrongeEdges(b, a, 1)
20:    end if
21: end procedure
```

$\triangle ADB$. Очевидно, что дуги этих окружностей, лежащие ниже AB, целиком попадают внутрь пустой окружности ребра Делоне, следовательно, снизу от AB внутри этих окружностей не может оказаться какой-либо сайт. А выше ребра AB одна из этих окружностей также является пустой, причем та, у которой угол треугольника, лежащий против стороны AB, больше (если углы $\angle ACB$ и $\angle ADB$ равно, то можно выбрать любое ребро). Это вытекает из правильности пучков сайтов A и B. ■

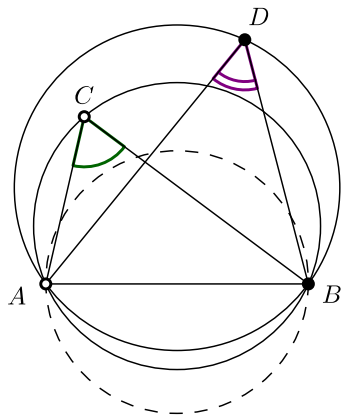

Рис. 6: Пояснение к лемме 2

Лемма 2 описывает процесс образования новых разноцветных стежков при построении шва. Из начального стежка AB получаем следующий стежок AD

(или BC). Далее возвращаемся к предыдущему шагу, производим коррекции пучков сайтов A и D (или B и C соответственно), пока очередной стежок не окажется ребром выпуклой оболочки или шов не окажется циклическим. Если очередной стежок оказался ребром выпуклой оболочки, то нужно развернуть стартер, и продолжить построение шва с другой стороны от него.

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.

Алгоритм 2 Построение разреза и шва триангуляции

```
 1: procedure SEWTRIANGLE(a_0, b_0, reverse = 0)
 2:     [a, b] := [a_0, b_0]
 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 then                ▷ cb — новое ребро
 8:             Добавить ребро cb в триангуляцию
 9:             if [c, b] ≠ [a_0, b_0] then
10:                 a := c
11:             else
12:                 reverse := 1                            ▷ Шов является циклическим
13:                 break
14:             end if
15:         else if ∠adb > ∠acb  &  ∠adb < π then           ▷ ad — новое ребро
16:             Аналогично ребру cb
17:         else
18:             break
19:         end if
20:     end while
21:     if reverse == 0 then          ▷ Построение шва с другой стороны от стартера
22:         sewTriangle(a_0, b_0, 1)
23:     end if
24: end procedure
```

5.3 Поиск стартеров

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

Рассмотрим условия существования стартеров (разноцветных ребер Делоне).

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.*

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. 7). 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. ■

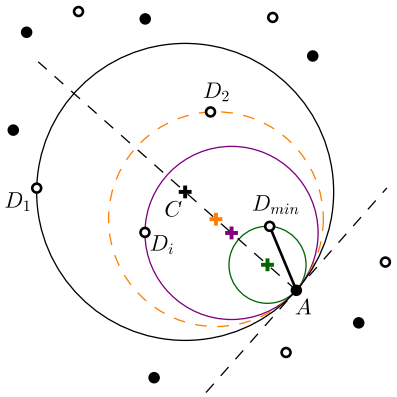

Рис. 7: Условие существования стартера

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.

5.3.1 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).

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. 8). 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. Внутри этой окружности может не оказаться ни одного сайта, тогда пара W_{min} и B_{min} образует стартер. Если же она не пустая, то часть

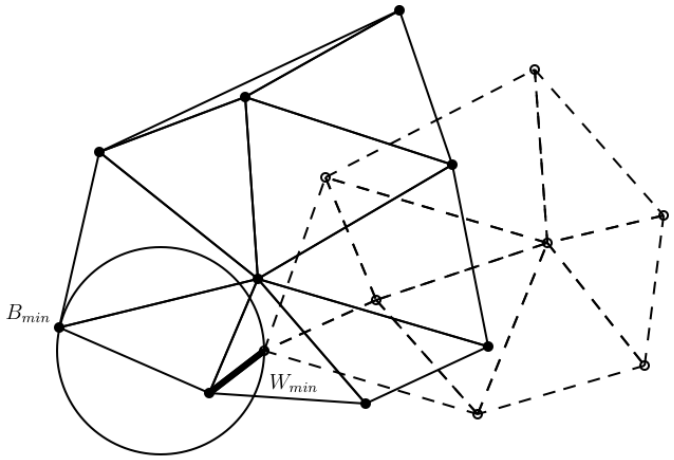

Рис. 8: Поиск первого стартера

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_min — самая левая точка множества pointsB
3:     w_min — самая левая точка множества pointsW
4:     starter_0 := w_min            ▷ Не ограничивая общности, b_min лежит левее w_min
5:     R_min := Inf
6:     for p ∈ pointsB do
7:         r — радиус окружности с центром на горизонтальном луче из точки w_min
   влево, инцидентной сайтам p и w_min
8:         if r < R_min then
9:             R_min := r
10:            starter_1 := p                    ▷ Объявить p концом стартера
11:        end if
12:    end for
13:    return [starter_0, starter_1]
14: end procedure
```

Время поиска первого стартера складывается из вычисления самых левых точек множеств **B** и **W** и перебора точек из множества **B** (или **W**) по методу из леммы 3, то есть линейно по общему числу сайтов, и равно $O(n_1 + n_2)$.

5.3.2 Минимальные остовные деревья

Задача поиска последующих стартеров основана на использовании минимальных остовных деревьев (МОД) исходных триангуляций Делоне. МОД исходных триангуляций вычисляются один раз, поэтому это действие можно отнести к этапу предобработки, предшествующем непосредственному слиянию исходных триангуляций. Теоретическая оценка трудоемкости задачи построения

минимального остова составляет $O(n \log n)$. Поскольку известно, что МОД является подграфом триангуляции Делоне, и на ее основе МОД может быть построен за линейное время (рис. 9). Такой вычислительной сложностью обладает алгоритм Чериτона-Тарьяна [6, стр. 226-230].

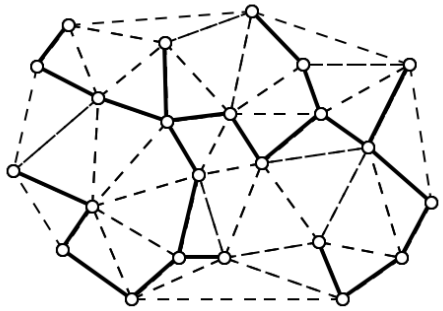

Рис. 9: Евклидово минимальное остовное дерево триангуляции Делоне

Поиск последующих стартеров основан на том, что при построении разреза в триангуляции Делоне нарушается ее связность, поэтому 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.*

Доказательство. Предположим противное: пусть AB —- ребро МОД и внутрь окружности с диаметром AB попадает точка C. В $\triangle ABC$ угол $\angle C$ тупой, поэтому $|AC| < |AB|$ и $|BC| < |AB|$. Тогда если ребро AB исключить из МОД, то дерево распадется на две связные компоненты, в одной из которых находится вершина C. Если C оказалась в одной компоненте с A, то включаем в дерево ребро CB, которое короче AB. Аналогично, если C находится в одной компоненте с B, меняем ребро AB на AC. В обоих случаях получаем дерево с длиной ребер меньшей, чем у исходного, что противоречит принадлежности AB к МОД. ■

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

Доказательство. Предположим противное. Пусть AA_1 и BB_1 —- ребра МОД с серединами A_0 и B_0 такие, что $|A_0B_0 \leq AA_0$ (рис.10). Тогда их окружности влияния пересекаются в некоторых точках C и D. Не нарушая общности, предположим, что $AB \geq A_1B_1$. Из точки C проведем диаметры CE и CF кругов A_0 и B_0 соответственно. В $\triangle CDE$ угол $\angle CDE$ прямой, а в $\triangle CDF$ угол $\angle CDF$ тоже прямой, следовательно, EF проходит через точку D.

Из леммы 4 следует, что точка A лежит на дуге ED, а точка B – на дуге DF. Углы $\angle EAD$ и $\angle DBF$ тупые, поскольку дуги ED и DF меньше полуокружностей. Следовательно, $AB < AD + DB < ED + DF = EF = 2A_0B_0$, т.е. $AB < AA_1$. Поскольку $A_1B_1 \leq AB$, то и $A_1B_1 < AA_1$. Покажем, что в

этом случае AA_1 не может быть ребром МОД. Действительно, если удалить из покрывающего дерева ребро AA_1, то одна из точек A или A_1 окажется в одной компоненте с ребром BB_1. Если это точка A, то заменяем AA_1 на A_1B_1 и получаем покрывающее дерево с меньшей длиной ребер. Если это точка A_1, то заменяем AA_1 на AB с тем же результатом. Полученное противоречие доказывает утверждение леммы.

■

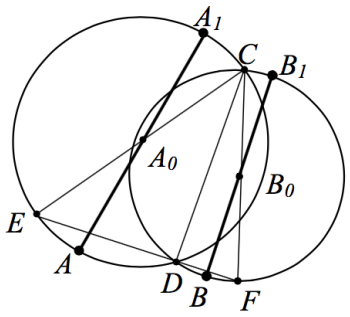

Рис. 10: Пояснение к лемме 5

Идея использования МОД для поиска стартеров основана на том очевидном факте, что если при образовании разреза в ТД нарушается ее связность, то обязательно среди разрушенных ребер окажется ребро МОД этой ТД.

5.3.3 Поиск последующих стартеров

Мостами будем называть ребра исходных триангуляций, входящие в МОД, разрушенные при построении очередного разреза (при выполнении функции $deleteWrongeEdges$). При построении разрезов удалялись ребра AC_1 и BD_1, переставшие удовлетворять лемме 1, при этом сайты A и B будем называть *закрепленными*, потому что они имеют разноцветные инцидентные ребра, а значит, они прикреплены к объединенной триангуляции, а C_1 и D_1 будем называть *свободными*. 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.*

Доказательство. Рассмотрим мост, инцидентный свободному сайту. По определению он является ребром МОД одной из исходных триангуляций и тогда, согласно лемме 4, внутри его окружности влияния нет сайтов одного с ним цвета. Но поскольку это ребро было разрушено при построении разреза, это означает, что внутрь этой окружности попали сайты противоположного цвета. Следовательно, для свободного сайта выполнено условие леммы 3, что и доказывает утверждение. ■

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

существуют 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.*

Доказательство. Любая пустая окружность, инцидентная сайту A, принадлежит объединению максимальных пустых окружностей сайта A. Поэтому если сайты A и B образуют ребро Делоне, то инцидентная им пустая окружность лежит также в этом объединении. ■

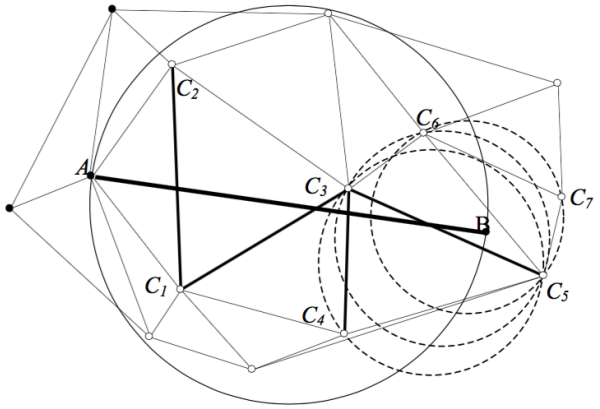

Рис. 11: Поиск последующих стартеров

Пусть AB — мост, сайт A является закрепленным, сайт B — свободным (рис. 11). 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. Задачу определения местоположения свободного сайта относительно триангуляции, в которую уже включен закрепленный сайт, будем называть *локализацией моста.* Для этого найдем точки C_1 и C_2 (рис.11, такие, что AB находится между AC_1 и AC_2, а угол $\angle AC_1C_2$ минимален. При этом ребро C_1C_2 пересекает мост AB. Для определенности будем считать, что сайт C_1 находится слева от AB, C_2 — справа. Точка C_3 является следующей в направлении по часовой стрелке после C_2 в пучке C_1 и предыдущей в направлении обхода против часовой стрелки после C_1 в пучке C_2, то есть $C_3 = neighbours[C_1].next(C_2) = neighbours[C_2].prev(C_1)$.

Из ребер C_1C_3 и C_2C_3 выбираем то, которое пересекает AB, и продолжаем трассировку. Если ни одно из этих ребер не пересекает AB, значит, локализация сайта B закончена, мы имеем либо грань, в которой лежит точка B, либо ребро, определяющее полуплоскость, в которой лежит B.

Найденная грань имеет описанную окружность, которая содержит свободный сайт B, но эта окружность может быть не минимального радиуса. Для поиска окружности минимального радиуса с центром на мосту AB, проходящей через точку B, переберем точки, лежащие в пучках вершин грани (полуплоскости), являющейся решением задачи локализации моста. То есть последовательно будем строить окружность по двум точкам и центру на AB, затем искать минимальный радиус. Сайт, минимизирующий радиус окружности, будет являться концом стартера.

Таким образом, поиск стартера на основе свободного сайта и инцидентного ему моста может быть осуществлен алгоритмом 4:

Алгоритм 4 Поиск последующих стартеров

```
1: procedure findNextStarter(bridge)
2:     open — открытый сайт ребра bridge, fix — закрепленный сайт моста
3:     c_1 — ближайшая к bridge точка слева, c_2 — справа
4:     c_3 := neighbours[c_1].next(c_2)
5:     while ( doTrue)
6:         if c_3 ≠ neighbours[c_2].prev(c_1) then
7:             checkPoints := {neighbours[c_1], neighbours[c_2]}
8:             break
9:         end if
10:        if c_1c_3 пересекает bridge then
11:            c_2 := c_3
12:        else if c_2c_3 пересекает bridge then
13:            c_1 := c_3
14:        else
15:            checkPoints := {neighbours[c_1], neighbours[c_2], neighbours[c_3]}
16:            break
17:        end if
18:    end while
19:    R_min := Inf
20:    for p ∈ checkPoints do
21:        r — радиус окружности с центром на bridge, инцидентной сайтам p и open
22:        if r < R_min then
23:            R_min := r
24:            starter_1 := p
25:        end if
26:    end forreturn [open, starter_1]
27: end procedure
```

Пусть AB — мост, сайт A является закрепленным. Последовательным перебором смежных с A сайтов находим такие, которые лежат по разные стороны от моста AB, на рисунке 11 это сайты C_1 и C_2, и точка пересечений C_1C_2 с AB лежит на отрезке AB.

Затем локализуем сайты C_1 и A в пучке сайта C_2. Если A предшествует C_1 в обходе против часовой стрелки, то дальнейшее движение от сайта C_2 нужно продолжать в том же направлении против часовой стрелки, иначе обход брать по часовой стрелки. Движение по пучку сайта C_1 будет производиться в обратном направлении, относительно направления движения в пучке сайта C_2.

Далее производим движение от сайтов C_1 и C_2 в полученном направлении. Проверяем, какое из ребер C_1C_3 и C_2C_3 пересекает мост AB. На рисунке это ребро C_1C_3, значит, дальнейшее движение будем производить от сайтов C_1 и C_3.

Повторяя предыдущий шаг, мы получим, что в некоторый момент ни одной из ребер после поворота не будет пересекать мост. Это означает, что мы нашли либо грань, которой принадлежит свободный сайт B, либо полуплоскость, которой принадлежит свободный сайт.

5.4 Общая структура алгоритма

В предыдущих пунктах был подробно описан каждый шаг алгоритма слияния перекрывающихся триангуляции. Обобщим полученные результаты и напишем формальную процедуру слияния (алгоритм 5).

Алгоритм 5 Слияние триангуляций

```
1: procedure makeConcatenation()
2:     makeMST()
3:     starter := findFirstStarter()
4:     Добавить starter в триангуляцию
5:     bridges — множество всех мостов, пополняется при удалении ребер МОД
       функцией deleteWrongeEdges
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
```

6 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. Построение МОД относится к предобработки, и время его построения не учитывается при подсчете сложности алгоритма слияния триангуляций. Однако известно, что сложность алгоритма построения МОД Черитона-Тарьяна составляет $O(n)$, что доказано в [6, стр.226-230]. Вычислительную сложность других частей алгоритма рассмотрим более подробно.

6.1 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 $O(1)$. Пусть разрезана одна из исходных триангуляций, например, $Del(\mathbf{W})$ и разрез разбил ее на две компоненты. Очевидно, что в таком случае хотя бы одно из ребер МОД окажется разрушенным. Следствием лемм 4 и 5 является тот факт, что внутри окружности влияния моста не могут находиться концевые и серединные точки других мостов. Это позволяет оценить меру пересечения одного моста с окружностью влияния другого моста.

Лемма 8. *Мост может вырезать из окружности влияния другого моста дугу размером не более* $60°$.

Доказательство. Пусть A_1A_2 и B_1B_2 — мосты с серединами A_0 и B_0 соответственно (рис. 12), а D_1 и D_2 — точки пересечения A_1A_2 и окружности влияния моста B_1B_2. Как следует из лемм 4 и 5 точки A_0, A_1 и A_2 лежат вне окружности, с центром в B_0. Следовательно, отрезок D_1D_2 целиком лежит либо на A_1A_0, либо на A_0A_2. Пусть для определенности $D_1D_2 \subseteq A_1A_0$. Тогда $D_1D_2 \leq A_1A_0$. В треугольнике $\triangle A_1A_0B_0$ сторона A_1A_0 не превосходит по длине стороны A_1B_0 и A_0B_0. Следовательно, $\angle A_1B_0A_0 \leq 60°$. Но тогда и $\angle D_1B_0D_2 \leq 60°$, что и доказывает утверждение леммы. ■

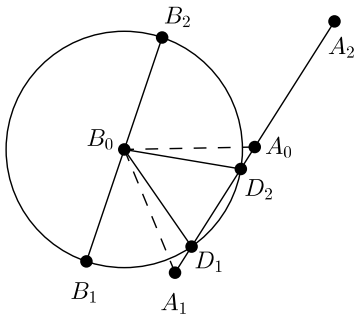

Рис. 12: Пояснение к лемме 8

Лемма 9. *Если два моста* AA_1 *и* BB_1 *одной ТД пересекают ребро* PQ *другой ТД, и концевая точка* P *ребра попадает внутрь обеих окружностей влияния мостов, то разность углов* $\angle APA_1$ *и* $\angle BPB_1$ *не меньше* $60°$.

Доказательство. С окружностью влияния моста BB_1 мост AA_1 имеет две точки пересечения D_1 и D_2 (рис. 13). Поскольку угол $\angle BPB_1$ лежит внутри $\angle APA_1$,

имеем $\angle APA_1 = \angle BPB_1 + \angle APB + \angle A_1PB_1$. Очевидно, что $\angle APB \geq \angle D_1PB \geq 0.5\widehat{D_1B}$ и $\angle A_1PB_1 \geq \angle D_2PB_1 \geq 0.5\widehat{D_2B_1}$. Тогда $\angle APB + \angle A_1PB_1 \geq 0.5(\widehat{D_1B} + \widehat{D_2B_1}) = 0.5(180^\circ - \widehat{D_1D_2})$. Но согласно лемме 8 имеем $\widehat{D_1D_2} \leq 60^\circ$. Поэтому $\angle APB + \angle A_1PB_1 \geq 0.5(180^\circ - 60^\circ) = 60^\circ$. Следовательно, $\angle APA_1 \geq \angle BPB_1 + 60^\circ$, что и требовалось доказать. ■

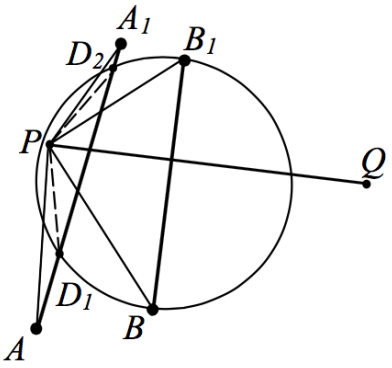

Рис. 13: Пояснение к лемме 9

Лемма 10. *Если ребро ТД пересекает несколько мостов другой ТД, то концевая точка ребра может попасть внутрь окружностей влияния не более чем двух мостов.*

Доказательство. Пусть ребро PQ пересекает несколько мостов, и точка P попадает внутрь окружностей влияния трех мостов AA_1, BB_1 и CC_1 (рис.14). Тогда согласно лемме 9 имеем: $\angle APA_1 \geq \angle BPB_1 + 60^\circ \geq \angle CPC_1 + 120^\circ$. Но $\angle APA_1 \leq 180^\circ$, и поэтому $CPC_1 \leq 60^\circ$. Поскольку угол $\angle CPC_1$ опирается на диаметр окружности влияния CC_1, то точка P лежит внутри этого круга. Следовательно, он должен быть не менее 90°. Полученное противоречие показывает, что точка P не может оказаться внутри окружности влияния третьего моста CC_1, что и требовалось доказать. ■

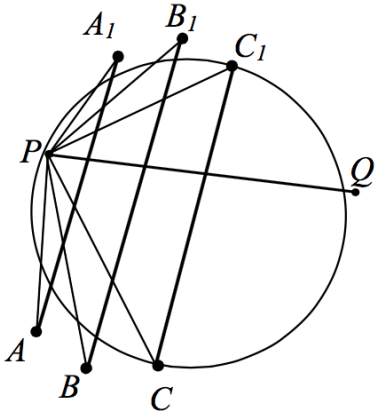

Рис. 14: Пояснение к лемме 10

Теперь можно оценить количество мостов ТД, которые может разрушить ребро другой ТД при их объединении.

Теорема 1. *При объединении двух ТД каждое ребро пересекается не более чем с четырьмя мостами, которые оно разрушает.*

Доказательство. Непосредственно следует из леммы 10. Поскольку разрушение моста ребром происходит лишь при попадании конца ребра внутрь окружности влияния моста, а по лемме 10 один конец ребра может попасть лишь в две такие окружности, общее число разрушенных одним ребром мостов не превышает четырех. ■

На основании теоремы 1 можно теперь оценить общее время поиска всех стартеров. При локализации моста осуществляется анализ всех пересечений моста с ребрами объединенной триангуляции. При этом сам мост является одноцветным ребром одной из исходных триангуляций, а пересекаемые им ребра являются либо разноцветными, либо одноцветными, но противоположного цвета с мостом. Если два отрезка пересекаются и каждый из них является ребром Делоне в своей триангуляции, то в объединенную триангуляцию один из них не должен входить. Это прямо следует из леммы 1. Следовательно, либо мост разрушает ребро, либо ребро разрушает мост. Те ребра, которые мост разрушает, имеют одно пересечение с одним мостом, так как к моменту анализа следующего моста эти ребра уже разрушены. Те ребра, которые сами разрушают мост, могут разрушить не более четырех мостов каждое. Следовательно, общее количество пересечений мостов с ребрами не может превысить пятикратное число ребер во всех триангуляциях, т.е. составляет $O(n)$. Таким образом, справедлива следующая теорема:

Теорема 2. *Вычислительная сложность алгоритма поиска всех стартеров составляет* $O(n)$.

6.2 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 разноцветного 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, с учетом теоремы 2 нами доказана теорема:

Теорема 3. *Предложенный алгоритм объединяет две неразделенные триангуляции Делоне за время* $O(n)$.

Очевидно, что размер памяти, используемой при работе алгоритма, составляет также $O(n)$.

7 Заключение

В данной статье был предложен вычислительно эффективный алгоритм объединения двух перекрывающихся триангуляций Делоне. Оценка вычислительной сложности алгоритма была доказана, предложенный подход был теоретически обоснован. Предложенный метод позволяет эффективно производить сравнение $3D$ поверхностей, при этом построение двух исходных ТД и МОД производится один раз, и относится к этапу предобработки, объединение триангуляций на каждом шаге подгонки поверхностей будет производиться за $O(n)$. Таким образом, общая оценка вычислительной сложности алгоритма равна $O(n \log n) + O(mn)$, где n — общее число точек двух исходных триангуляций, m — число итераций.

Работа выполнена при поддержке Российского Фонда фундаментальных исследований (РФФИ), проект 14-01-00716-а.

Список литературы

[1] D.T. Lee, B.J. Schachter. Two algorithms for constructing a Delaunay triangulation // International Journal of Computer and Information Science. vol. 9, no.3. 1980. С. 219–242.

[2] N. Dyshkant. Comparison of Point Clouds Acquired by 3D scanner // Lecture Notes in Computer Science: Discrete Geometry for Computer Imagery. vol.7749. 2013. С. 47–58.

[3] D.G. Kirpatrick. Efficient computation of continuous skeletons // Proceedings of the 20th Annual IEEE Symposium on FOCS. 1979. С. 18–27.

[4] B. Chazelle. An optimal algorithm for intersecting three-dimensional convex polyhedrons // SIAM J.Comput. vol.21. 1992. с. 671–696.

[5] M.I. Shamos, D. Hoey. Closest-point Problems // Proceedings of the 16th Annual IEEE Symposium on FOCS. 1975. С. 151–162.

[6] Preparata Franco P., Shamos Michael I. Computational Geometry: An Introduction. New York, NY, USA: Springer-Verlag New York, Inc., 1985.

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