Модел. и анализ информ. систем. Т. **19**, № **6** (2012) 112–126

УДК 519.17+510.52

О некоторых задачах локализации в триангуляциях Делоне

Дышкант Н.Ф.[1]

Московский государственный университет имени М.В. Ломоносова

e-mail: natalia.dyshkant@gmail.com

получена 15 сентября 2012

Ключевые слова: вычислительная геометрия, геометрический поиск, триангуляция Делоне, слияние перекрывающихся триангуляций, нерегулярная дискретная сетка, вычислительная сложность

Рассматриваются постановки задач локализации узлов в триангуляциях Делоне и методы их решения. Для задачи локализации множества узлов предлагается подход, основанный на прослеживании Евклидова минимального остовного дерева триангуляции Делоне. Приводятся и доказываются оценки сложности предложенных методов в среднем и худшем случаях.

Одной из известных задач геометрического поиска является *задача локализации узла в триангуляции Делоне* [1]: задан узел Q и триангуляция Делоне T. Требуется определить треугольник T, содержащий узел Q. Если узел Q совпадает с одним из узлов триангуляции, то можно указать любой из инцидентных данному узлу треугольников триангуляции. Если узел Q принадлежит одному из рёбер триангуляции, то можно указать любой из инцидентных данному ребру треугольников. Если узел заходит за границы триангуляции, то есть не принадлежит выпуклой оболочке множества узлов триангуляции, то можно определять ближайший к узлу треугольник.

Обзор существующих методов. Известные методы решения задачи геометрического поиска делятся на точные (использующие точные геометрические вычисления) и неточные, в которых на планарное разбиение (триангуляцию) накладывается равномерная прямоугольная сетка, и запрашиваемый узел локализуется с точностью до размера ячейки сетки.

Существуют методы (алгоритм Киркпатрика, [2]) решения задачи, работающие за время $O(logN)$ в худшем случае и использующие структуры данных размера $O(N)$, где N — количество узлов в триангуляции. Есть методы, работающие

[1]Работа поддержана РФФИ (проекты 12-07-31107-мол_а, 11-01-00783-а)

медленнее в худшем случае, но быстрее в среднем. В [3] показано, что не существует подходов, которые являлись бы оптимальными одновременно по всем параметрам: сложности предварительной обработки, использованию памяти и скорости локализации.

Точные методы решения задачи геометрического поиска:

1. Наивный алгоритм (перебор всех граней исходного графа).

2. Алгоритм Киркпатрика [2].

На этапе предобработки алгоритм создаёт иерархию разбиений, все грани которых являются треугольниками (иерархию триангуляций). В любой триангуляции можно за линейное время выделить независимое множество вершин наименьшей степени. Размер этого множества определяется по числу вершин в триангуляции. Для перехода на следующий уровень иерархии, вершины этого множества удаляются из триангуляции, и строятся триангуляции многоугольников, образовавшиеся в результате удаления вершин. Процесс повторяется логарифмическое число раз, после чего образуется «грубая» триангуляция с константным числом вершин. Во время выполнения запроса локализации узла в треугольнике, сначала происходит поиск треугольника в самой «грубой» триангуляции. Про этот треугольник известно, при построении триангуляции какого многоугольника он образовался. Удалённая из этого многоугольника вершина восстанавливается, далее происходит локализация заданного узла в инцидентных ей треугольниках. Время на выполнение одного запроса — $O(\log N)$.

3. Алгоритм последовательных переходов вдоль прямой («Walk along a line» strategy) [4].

Основная стратегия алгоритма — построение путей из попарно смежных треугольников к искомому треугольнику. Прослеживание по треугольникам может осуществляться следующими способами:

— прямое прослеживание (переходы по всем треугольникам, которые пересекают отрезок, соединяющий начальную точку с исходной);

— ортогональное прослеживание (переходы по треугольникам пути, параллельного одной из осей координат);

— прослеживание по обозримому пути (переход от одного треугольника к соседнему осуществляется через первое ребро треугольника, если оно отделяет начальную точку от исходной; в противном случае проверяется второе ребро, и т. д.);

— стохастическое прослеживание (аналогично предыдущему способу, только вместо первого ребра выбирается случайное ребро).

В отличие от алгоритма Киркпатрика, этот алгоритм не требует построения и поддержки дополнительных структур данных. Если вершины триангуляции выбирались случайно из равномерного распределения, среднее время работы $O(\sqrt{N}\,)$, время работы в худшем случае — $O(N)$.

4. Алгоритм прыжков и переходов (Jump and Walk) [5, 6]. Работает только для триангуляций Делоне.

Исходная триангуляция — триангуляция Делоне для N вершин. Случайным образом из них выбираются k вершин. При локализации узла Q из k вершин выбирается ближайшая к узлу Q точка M. Далее применяется алгоритм последова-

тельных переходов вдоль отрезка, соединяющего M и Q. Время работы в среднем — $O(k+\sqrt{N/k}\,)$, которое при k равном $O(N/3)$ является оптимальным — $O(N/3)$.

Позже в [7] было предложено улучшение данного алгоритма — алгоритм бинарного поиска и переходов (Binsearch and Walk). k выбирается равным $N/4$, для поиска используется бинарное дерево поиска. С помощью дерева находится ближайший узел, от которого применяется алгоритм последовательных переходов. Там же предложен алгоритм $2d$ поиска и переходов (2d Search and Walk), который использует сбалансированные k-d-деревья (k-d-trees). Для триангуляции Делоне на N вершинах, независимо и равномерно распределённых на плоскости, среднее время локализации $O(\log N)$.

5. Алгоритм опознавательных точек (Landmarks strategy) [3].

На этапе предобработки алгоритма выбираются опознавательные точки (точки-ориентиры), для которых решается задача локализации в триангуляции. Далее точки-ориентиры помещаются в иерархическую структуру данных, обеспечивающую быстрый поиск ближайшего соседа. Когда требуется локализовать новый узел Q, сначала находится ближайшая точка-ориентир Q_i, применяется стратегия последовательных переходов вдоль отрезка, соединяющего Q_i и Q. Теоретическая оценка сложности одного запроса локализации узла для алгоритма — $O(\log N)$.

Таким образом, наиболее быстрые алгоритмы решения задачи локализации узла имеют сложность $O(\log N)$ и требуют предобработки. Рассмотрим массовый запрос на решение задачи локализации узла — задачу локализации узлов сетки. Под *плоской двумерной сеткой* будем понимать набор взаимосвязанных геометрических элементов — узлы, рёбра, грани. *Неструктурированный массовый запрос* из N узлов может быть обработан за время $O(N \log N)$. Далее показывается, как можно использовать структуру локализуемой сетки (а именно триангуляцию Делоне) для получения более быстрого решения.

Для изложения предлагаемого метода рассмотрим более подробно стратегию локализации узла в триангуляции Делоне, основанную на упоминавшемся выше алгоритме последовательных переходов вдоль прямой при прямом прослеживании [4].

Метод локализации узла. Стратегия метода состоит в выборе некоторого начального узла M, локализация которого в триангуляции известна, и постепенном переходе от M к Q вдоль прямой (MQ). На каждом шаге осуществляется переход на смежный треугольник. Сложным случаем является ситуация, когда отрезок $[MQ]$ проходит через какой-либо узел триангуляции.

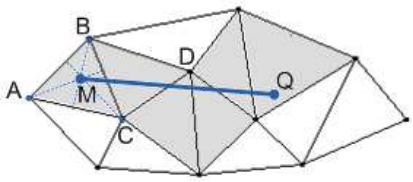

Рис. 1. Локализация узла в триангуляции

В процессе локализации узла строится путь из треугольников триангуляции,

каждый из которых (кроме начального) является смежным с предыдущим. На рис. 1 и 2 выделены треугольники, принадлежащие путям локализации.

Метод локализации узла в триангуляции состоит из следующих шагов:

1. Выбрать начальный треугольник — $\triangle ABC$. Присвоить ему статус текущего. Найти координаты точки $M = (M_x, M_y)$ пересечения медиан (центроида) этого треугольника: $M_x = (A_x + B_x + C_x) \,/\, 3, \quad M_y = (A_y + B_y + C_y) \,/\, 3$.

2. Определить, какую из сторон текущего треугольника пересекает отрезок $[MQ]$. Если данный пункт выполняется первый раз, то на предмет пересечения с отрезком $[MQ]$ проверяются все три стороны текущего треугольника, в противном случае достаточно проверить только две стороны и запомнить сторону, через которую перешли на текущий треугольник.

а. Пусть отрезок $[MQ]$ не пересекает ни одну из сторон или пересекает её в узле Q. Тогда текущий треугольник является искомым и работа алгоритма заканчивается.

б. Пусть отрезок $[MQ]$ пересекает одну из вершин текущего треугольника. Тогда осуществляется переход на треугольник, смежный с текущим по любой из сторон, содержащей данную вершину.

Пример. *На рис. 2 отрезок $[MQ]$ проходит через вершину C. При переходе на следующий треугольник можно перейти на любой из треугольников $\triangle BCD$ или $\triangle ACE$. Если перейти на треугольник $\triangle BCD$ ($\triangle ACE$), то на следующем этапе проверяются на предмет пересечения с $[MQ]$ отрезки $[BD]$ и $[CD]$ ($[EC]$ и $[AE]$), и переход к следующему треугольнику — $\triangle CDF$ ($\triangle CEG$) — будет однозначным.*

Треугольник, на который осуществлён переход, становится текущим.

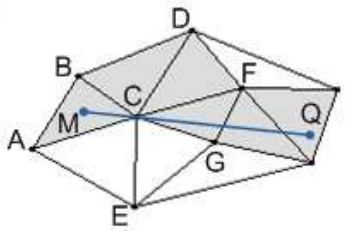

Рис. 2. MQ проходит через вершину триангуляции

Переходим в начало пункта 2.

в. Пусть отрезок $[MQ]$ пересекает одну из сторон треугольника во внутренней точке. Тогда переходим к треугольнику, смежному с текущим по данной стороне.

Пример. *На рис. 1 отрезок $[MQ]$ пересекает сторону BC начального треугольника. На следующем шаге переходим к треугольнику $\triangle BCD$.*

Треугольник, на который осуществлён переход, становится текущим. Переходим в начало пункта 2.

Пусть триангуляция Делоне представлена в виде структуры данных «Узлы с соседями» [8], в которой *пучок* каждого узла триангуляции представлен в виде двунаправленного циклического списка соседних узлов, упорядоченных по углу.

Пусть методы Найти_левого_соседа(X, V) и Найти_правого_соседа(X, V) возвращают соответственно левого и правого соседа узла X относительно узла V, в массиве $X.Triangle$ хранятся узлы треугольника, в котором локализован узел X. Тогда алгоритм локализации узла можно записать в следующем виде:

Алгоритм. Локализация узла в триангуляции.

Вход:

T — триангуляция Делоне с узлами $g[0], \ldots, g[N-1]$;

$\Delta(g[i]) = \{\Delta(g[i])[0], \ldots, \Delta(g[i])[i_k - 1]\}$ — упорядоченный по углу список узлов, смежных с $g[i]$ в T;

Q — узел для локализации.

Выход:

узлы A, B, C такие, что $\triangle ABC \in T$ и Q находится внутри или на границе $\triangle ABC$.

```
1: Ã := g[0]; B̃ := Δ(g[0])[0]; C̃ := Δ(g[0])[1];
2: M_x := (Ã_x + B̃_x + C̃_x)/3; M_y := (Ã_y + B̃_y + C̃_y)/3;
3: △ABC := Локализовать_узел(M, Ã, B̃, C̃, T, ИСТИНА);
4: вернуть A, B, C;
```

ПРОЦЕДУРА Локализовать_узел($Node\ M, Node\ A, Node\ B, Node\ C, Node\ Q, bool\ flag$)

```
1: если (flag = ИСТИНА) и ([MQ] и [AB] пересекаются) то
2:    R := Найти_правого_соседа(A, B);
3:    если (R = C) то
4:       L := Найти_левого_соседа(A, B);
5:       Локализовать_узел(M, A, B, L, Q, T, ЛОЖЬ);
6:    иначе
7:       Локализовать_узел(M, A, B, R, Q, T, ЛОЖЬ);
8: иначе если ([MQ] и [AC] пересекаются) то
9:    L := Найти_левого_соседа(A, C);
10:   если (L = B) то
11:      R := Найти_правого_соседа(A, C);
12:      Локализовать_узел(M, A, C, R, Q, T, ЛОЖЬ);
13:   иначе
14:      Локализовать_узел(M, A, C, L, Q, T, ЛОЖЬ);
15: иначе если ([MQ] и [BC] пересекаются) то
16:   R := Найти_правого_соседа(B, C);
17:   если (R = A) то
18:      L := Найти_левого_соседа(B, C);
19:      Локализовать_узел(M, B, C, L, Q, T, ЛОЖЬ);
20:   иначе
21:      Локализовать_узел(M, B, C, R, Q, T, ЛОЖЬ);
22: иначе
23:   вернуть △ABC;
```

Таким образом, процедура Локализовать_узел(M, A, B, C, Q, T, $flag$) локализует узел Q в триангуляции T, двигаясь вдоль отрезка $[MQ]$, причём точка M уже локализована в треугольнике $\triangle ABC$ триангуляции T. Индикатор $flag$ показывает, первый раз выполняется локализация узла или нет.

Трудоёмкость локализации одного узла определяется количеством расположенных вдоль отрезка $[MQ]$ треугольников и составляет $O(\sqrt{N})$ в среднем (при равномерном распределении узлов) [9] и $O(N)$ в худшем случае (когда $[MQ]$ пересекает все треугольники триангуляции).

Локализация узла в триангуляции: множественный запрос. *Задачей локализации узлов двумерной сетки g в триангуляции T* называется задача локализации каждого из узлов сетки g в данной триангуляции.

Предлагается алгоритм локализации узлов сетки, использующий Евклидово минимальное остовное дерево (далее — МОД), вершинами которого являются узлы данной сетки (рис. 3). Известно, что МОД триангуляции Делоне можно построить за линейное время. Локализация узлов организовывается таким образом, чтобы пути локализации проходили вдоль рёбер МОД.

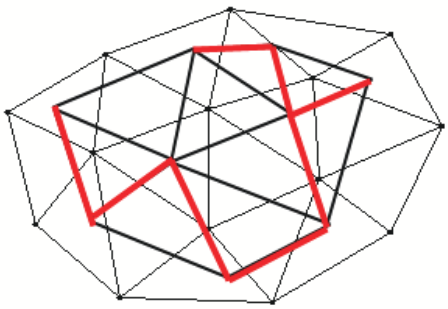

Рис. 3. Локализация узлов сетки в триангуляции с помощью МОД узлов сетки

Предлагаемый алгоритм локализации узлов сетки можно записать в следующем виде:

Алгоритм. Локализация узлов сетки в триангуляции.

Вход:

g_1 — сетка с узлами $g_1[0], \ldots, g_1[N_1 - 1]$ для локализации;

T_2 — триангуляция Делоне множества узлов $g_2 = \{g_2[0], \ldots, g_2[N_2 - 1]\}$;

MST_1 — МОД сетки g_1, состоящее из списка рёбер d с концевыми узлами $d.orig$, $d.dest$.

Выход:

заполненный массив $Triangle$ для всех узлов сетки g_1.

1: $A := g_2[0]$; $B := \Delta(g_2[0])[0]$; $C := \Delta(g_2[0])[1]$;
2: $M_x := (A_x + B_x + C_x)/3$; $M_y := (A_y + B_y + C_y)/3$;

3: $M.Triangle := \triangle ABC$;
4: Локализовать_узел_сетки($M, g_1[0], T_2, MST_1$);
5: **вернуть** g_1;

ПРОЦЕДУРА Локализовать_узел_сетки(*Node* M, *Node* Q, *Triangulation* T, *Tree* MST)

1: $Q.Triangle =$ Локализовать_узел(M, $M.Triangle[0]$, $M.Triangle[1]$, $M.Triangle[2]$, Q, T, ИСТИНА);
2: **если** (число рёбер инцидентных Q и принадлежащих $MST = 1$) **то**
3: **выход**;
4: **для всех** (рёбер $d \in MST$ таких, что $d.orig = Q$, а $d.dest$ ещё не локализована)
5: Локализовать_узел_сетки(Q, $d.dest$, T, MST);

Процедура Локализовать_узел_сетки(M, Q, T, MST) рекурсивная. В качестве первого параметра ей передаётся центроид любого треугольника триангуляции T; в качестве второго параметра — узел $g[0]$; третий параметр — это триангуляция T, в которой нужно локализовать сетку; четвёртый параметр — минимальное остовное дерево сетки g. Алгоритм вызывает процедуру для первого узла сетки $g[0]$. Далее с помощью процедуры Локализовать_узел выполняется локализация первого узла, затем рекурсивно вызывается для всех узлов, соседних с данным узлом в МОД. Таким образом, при каждом (кроме самого первого) вызове процедуры Локализовать_узел_сетки(M, Q, T, MST) её параметры будут такими, что MQ — ребро МОД сетки g, и пути локализации будут проходить вдоль рёбер минимального остова.

Так как по определению МОД не содержит в себе циклов и проходит через все узлы сетки g, алгоритм будет работать корректно: он не зациклится и произведёт локализацию всех N узлов сетки g.

Сложность алгоритма локализации узлов сетки. Оценим вычислительную сложность предложенного алгоритма локализации узлов сетки в триангуляции Делоне на основе минимального остова узлов данной сетки в среднем и в худшем случае.

Пусть координаты узлов исходных сеток g_1 и g_2 распределены равномерно и независимо по X и Y в прямоугольнике R. Внутри R выделим прямоугольник R' со сторонами a и b так, чтобы внутри R' не было «вытянутых» треугольников триангуляции T_2, которые могут располагаться вдоль выпуклой оболочки множества узлов g_2 (см. рис. 4). Пусть внутри прямоугольника R' содержится N_2' узлов сетки g_2. В таких условиях и при достаточно больших N_2 можно считать, что $N_2' \approx N_2$.

В [11] было получено значение средней евклидовой длины $\bar{l}$ ребра триангуляции Делоне T_2 для описанных условий:

$$\bar{l} = \frac{\pi\sqrt{5}}{6}\sqrt{\frac{ab}{N_2'}}. \tag{1}$$

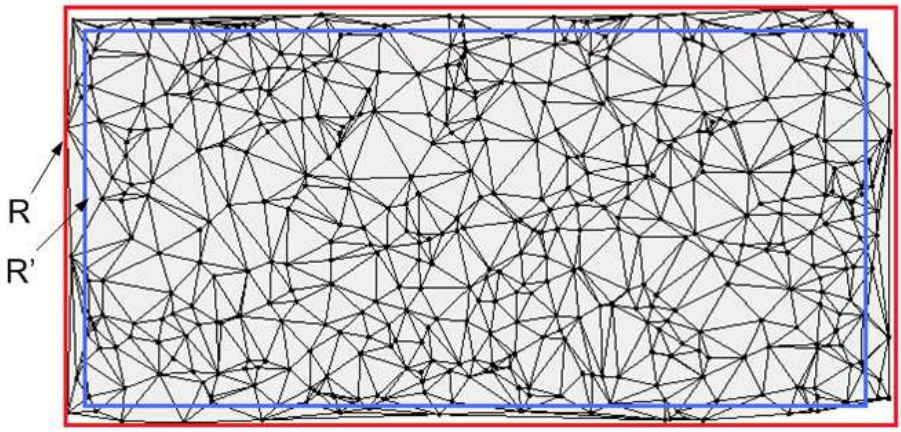

Рис. 4. Выбор прямоугольника R'

Пусть минимальный остов $MST_1 = MST(g_1)$ пересекает s рёбер триангуляции T_2, имеющих длины $l_1, \ldots, l_s$. Количество s пересекаемых рёбер не превосходит общего количества рёбер триангуляции, линейного по числу её узлов, поэтому $s = O(N_2)$. Однако МОД может пересекать какие-то рёбра несколько раз. Обозначим через Q количество пересечений минимального остова MST_1 и s пересекаемых им рёбер. Для определения вычислительной сложности алгоритма локализации требуется оценить Q.

В [12] доказана теорема, оценивающая среднее количество пересечений МОД и прямолинейного отрезка заданной длины:

Теорема 1 (Боуз, Девруа)**.** *Пусть X — множество из n точек $X_1, \ldots, X_n$, независимо распределённых с плотностью f на выпуклом компакте C на плоскости. Пусть для некоторых констант α, β на C выполнено: $0 < \alpha \leqslant f(x) \leqslant \beta < \infty$. Пусть L — прямолинейный отрезок длины l, содержащийся в C. Тогда существуют такие константы d и n_0, зависящие от α, β и C, что*

$$\bar{Q} \leqslant dl\sqrt{n}, \quad n \geqslant n_0, \tag{2}$$

где $\bar{Q}$ — среднее количество пересечений минимального остовного дерева множества X и отрезка L.

Для рассматриваемой задачи выполнены условия теоремы 1, поэтому для среднего количества $\bar{Q}_i$ пересечений минимального остова MST_1 и ребра триангуляции T_2 длины l_i справедлива оценка: $\bar{Q}_i = dl_i\sqrt{N_1}$. Складывая оценку (2) для s пересекаемых рёбер, получим среднее количество пересечений MST_1 с рёбрами T_2:

$$\bar{Q} = \sum_{i=1}^{s} \bar{Q}_i \leqslant d \sum_{i=1}^{s} l_i \sqrt{N_1}.$$

Используя значение для средней длины ребра (1), получим:

$$\bar{Q} \leqslant d\,\frac{\pi\sqrt{5}}{6}\, s \sqrt{\frac{ab}{N_2'}}\,\sqrt{N_1} = d\frac{\pi\sqrt{5ab}}{6}\, s \sqrt{\frac{N_1}{N_2'}}.$$

При $N_1/N_2' \leqslant c^2 = const$ получим, что

$$\bar{Q} \leqslant cd\,\frac{\pi\sqrt{5ab}}{6}\,s.$$

При примерно равной мощности сеток $N_1 \approx N_2$, можно полагать $c = 1$.

Учитывая то, что $s = O(N_2)$, такая же оценка будет справедлива для $\bar{Q}$: $\bar{Q} = O(N_2)$.

Таким образом, доказана следующая лемма:

Лемма 1. *Пусть узлы сеток g_1, g_2 с мощностями N_1, N_2 соответственно распределены равномерно в прямоугольнике и $N_1/N_2 \leqslant c = const$. Тогда среднее количество пересечений МОД множества узлов g_1 с рёбрами триангуляции Делоне, построенной на узлах g_2, линейно по N_2.*

Лемма 1 позволяет оценить трудоёмкость алгоритма локализации в среднем:

Теорема 2. *Алгоритм локализации множества узлов сетки g_1 мощности N_1 в триангуляции Делоне, построенной на множестве узлов g_2 мощности N_2, на основе МОД g_1 имеет линейную по* $\max(N_1, N_2)$ *вычислительную сложность в среднем при равномерном распределении узлов обеих сеток в прямоугольнике и условии ограничения отношения мощностей сеток $N_1/N_2 \leqslant c = const$.*

Доказательство. Локализация первого узла сетки g_1 производится методом последовательных переходов вдоль прямой. Количество треугольников в пути этой локализации не превосходит общее количество треугольников в триангуляции T_2, т.е. $O(N_2)$.

Далее алгоритм осуществляет проход вдоль рёбер минимального остова MST_1 (всего $N_1 - 1$ рёбер) по всем пересекаемым им треугольникам триангуляции T_2. При этом переход на следующий треугольник осуществляется за константное количество операций $O(1)$, позволяющих анализировать пересечения ребра минимального остова с рёбрами триангуляции. В соответствии с леммой 1, среднее количество таких пересечений в условиях теоремы есть $O(N_2)$.

Во время обхода MST_1 для каждого узла g_1 определяется треугольник триангуляции T_2, внутри которого содержится данный узел (константное количество операций на узел).

Таким образом, вычислительная сложность алгоритма в среднем будет линейной по количеству узлов в сетке б*о*льшей мощности $O\left(\max(N_1, N_2)\right)$.

Теорема доказана. ■

Худший случай для алгоритма локализации узлов сетки. В худшем случае время работы предложенного алгоритма будет квадратичным. Рассмотрим модельный контрпример для худшего случая.

Пусть узлы первой сетки, состоящей из N_1 узлов, расположены на двух вертикальных отрезках длины H с фиксированным шагом (см. рис. 5). На каждом из отрезков примерно одинаковое количество точек, расстояние между отрезками равно L, $L > H$. Таким образом, выпуклая оболочка множества точек первой сетки есть некоторый прямоугольник R со сторонами L и H.

Узлы второй сетки, состоящей из N_2 узлов, расположены на двух горизонтальных отрезках, расстояние между которыми равно H, содержащихся в прямоугольнике R, с фиксированным шагом так, что расстояние между двумя соседними узлами на отрезках превосходит расстояние между двумя ближними узлами из разных отрезков. Тогда МОД второй сетки будет состоять из отрезков, соединяющих узлы, лежащие на разных горизонтальных отрезках.

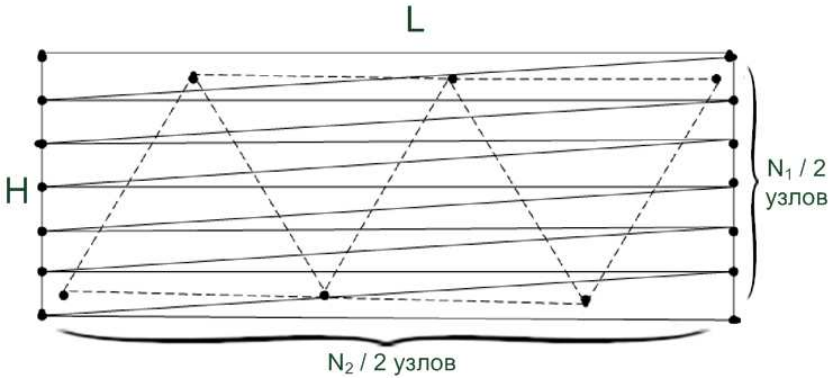

Рис. 5. Модельный пример для задачи локализации узлов сетки в триангуляции

Оценим количество операций при решении задачи локализации второй сетки в триангуляции Делоне первой сетки. Каждое их $(N_2 - 1)$ рёбер второго МОД пересекает $O(N_1)$ треугольников первой триангуляции. Поэтому общее число операций при локализации, равное $O(N_1\, N_2)$, будет квадратичным при $N_1 \approx N_2$.

Найдём соотношение между L и H, при котором реализуется рассматриваемый пример. Пусть расстояние между двумя соседними во второй триангуляции узлами, лежащими на разных горизонталях, равно a, а между двумя соседними узлами, лежащими на одной горизонтали — b; по условию примера $b > a$ (см. рис. 6). Тогда в $\triangle\, ABC$ угол $\angle BAC < 60^\circ$ и высота h, опущенная на AC из B, $h < a\frac{\sqrt{3}}{2}$. Будем считать, что $h \approx H$, например, $H = h(1+\alpha)$ для некоторого малого α. Тогда

$$H < (1+\alpha)a\frac{\sqrt{3}}{2} \;\Rightarrow a > \frac{2}{3}\sqrt{3}H\frac{1}{1+\alpha};$$

$$\frac{2L}{N_2} \approx b > a > \frac{2}{3}\sqrt{3}H\frac{1}{1+\alpha} \;\Rightarrow L > \frac{\sqrt{3}}{3}HN_2\frac{1}{1+\alpha}.$$

Таким образом, в рассматриваемом примере при $L > \frac{\sqrt{3}}{3}HN_2\frac{1}{1+\alpha}$ время работы алгоритма будет квадратичным.

Слияние неразделённых триангуляций Делоне. Триангуляции T_1 и T_2 называются *перекрывающимися (неразделёнными)*, если пересечение выпуклых оболочек множеств их вершин не пусто, то есть $Conv(g_1) \cap Conv(g_2) \neq \varnothing$.

В [13, 14] рассматривалась *задача слияния неразделённых триангуляций Делоне*, в которой требуется построить *объединённую триангуляцию Делоне* по двум исходным перекрывающимся триангуляциям. В указанных работах был предложен метод, позволяющий решить задачу за линейное время.

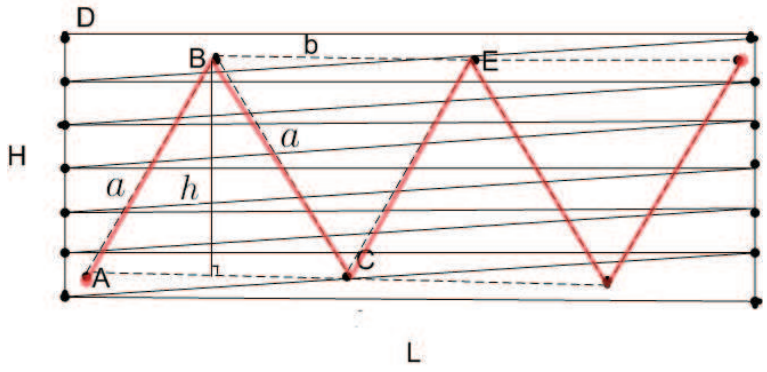

Рис. 6. Модельный пример для задачи локализации узлов сетки в триангуляции

При объединении триангуляций Делоне T_1 и T_2 некоторые рёбра и треугольники перейдут без изменений в объединённую триангуляцию T, а некоторые разрушатся. Таким образом, в T будут присутствовать «новые» рёбра и треугольники, соединяющие узлы из разных сеток. Будем называть такие рёбра и треугольники (грани) *интерфейсными*, так как они соединяют узлы, соответствующие разным триангуляциям.

Множество интерфейсных граней разбивается на несколько непересекающихся подмножеств, каждое из которых является *цепочкой* из смежных по рёбрам треугольников: либо *замкнутой* (циклической), в которой все интерфейсные рёбра являются внутренними рёбрами общей триангуляции T, либо *разомкнутой*, в которой крайние треугольники имеют хотя бы одну граничную сторону, т. е. сторону выпуклой оболочки $Conv(g)$ (см. рис. 7).

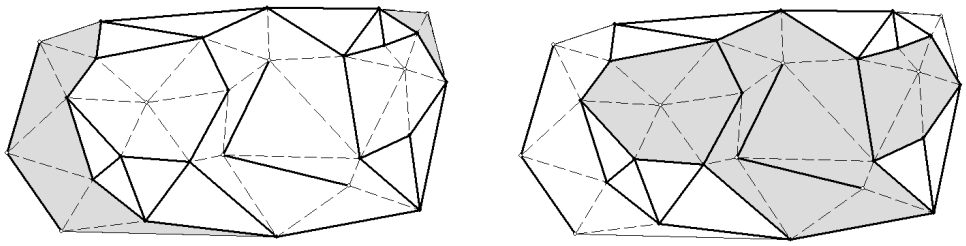

Рис. 7. Две разомкнутые (слева) и три замкнутые цепочки интерфейсных граней (справа)

Поэтому алгоритм выделения всех интерфейсных граней сводится к прослеживанию цепочек из таких граней. Таким образом, *задача прослеживания цепочек интерфейсных граней* состоит в построении списка всех таких цепочек по двум исходным триангуляциям.

Задача локализации лоскутов в триангуляции. Множество узлов и рёбер в триангуляции называется *связным*, если для любой пары входящих в него узлов существует цепь из попарно инцидентных узлов и рёбер. Множество рёбер и граней

называется связным, если для каждой пары входящих в него рёбер существует цепь из попарно инцидентных рёбер и граней. Максимальные связные подмножества узлов и рёбер, перешедшие в T из исходной триангуляции T_1 (или T_2) без изменений, будем называть *лоскутами* триангуляции T_1 (или T_2).

Лоскут является связным подграфом триангуляции. При объединении двух триангуляций Делоне, все рёбра, не вошедшие в лоскуты, разрушаются. Интерфейсные рёбра и грани «сшивают» лоскуты из разных триангуляций.

Лоскуты могут иметь простую форму: состоять только из одного узла или из цепочки узлов и рёбер, не содержащей циклов (см. рис. 8). Будем называть такие лоскуты, множество рёбер которых не образует ни одной грани исходной триангуляции, *простыми* лоскутами.

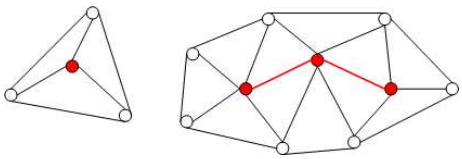

Рис. 8. Примеры простых лоскутов

На рис. 9 представлен лоскут более сложной формы, который содержит несколько треугольных граней исходной триангуляции и не является простым.

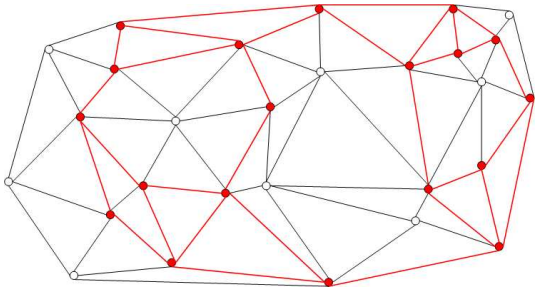

Рис. 9. Объединённая триангуляция Делоне, узлы лоскута закрашены

Будем называть упорядоченное замкнутое множество попарно инцидентных узлов и рёбер лоскута, инцидентных интерфейсным треугольникам, *контуром* лоскута. Каждому ребру лоскута, принадлежащему его контуру, соответствует одна или две цепочки интерфейсных граней.

Локализовать лоскут C в общей триангуляции T означает построить множество L всех интерфейсных треугольников триангуляции T, инцидентных каждому контуру этого лоскута. Будем говорить, что любой треугольник из L *участвует в локализации* лоскута C, множество всех треугольников из L *локализует* лоскут C, а лоскут C *локализуется* с помощью множества интерфейсных треугольников L.

Рассмотрим *задачу локализации лоскутов в триангуляции* T. Даны исходные триангуляции T_1, T_2, требуется локализовать все лоскуты обеих триангуляций.

Докажем справедливость следующего утверждения:

Утверждение 1. *Задача прослеживания цепочек интерфейсных граней линейно сводится к задаче локализации лоскутов в триангуляции.*

Доказательство. Пусть построен список всех цепочек интерфейсных граней. Докажем, что с помощью него можно за линейное время локализовать все лоскуты обеих триангуляций.

Заметим, что верно следующее:

— Если хотя бы одна интерфейсная грань из цепочки участвует в локализации лоскута, то все грани этой цепочки участвуют в локализации этого лоскута.

— Каждая замкнутая цепочка интерфейсных граней полностью локализует один лоскут.

— Любой простой лоскут однозначно локализуется с помощью одной цепочки интерфейсных граней: замкнутой, если лоскут не принадлежит выпуклой оболочке узлов общей триангуляции, и разомкнутой в противном случае.

Выбирая рёбра контура лоскута и соответствующие им цепочки интерфейсных граней, можно произвести локализацию всего лоскута.

При таком подходе каждый интерфейсный треугольник будет просмотрен ровно два раза, так как он соединяет одну вершину одной триангуляции и две вершины другой, а значит, участвует в локализации двух лоскутов, что доказывает утверждение. ■

Обозначим через N общее количество узлов в двух сетках g_1 и g_2. Рассмотрим задачу выделения всех интерфейсных граней триангуляции Делоне T. Используя идеи, аналогичные предложенным Л.М. Местецким и Е.В. Царик [14] для алгоритма слияния неразделённых триангуляций, автором был предложен метод решения задачи, имеющий линейную вычислительную сложность:

Теорема 3. *Алгоритм выделения всех интерфейсных граней имеет линейную сложность $O(N)$, где N — общее число узлов в объединённой сетке.*

Утверждение 1 и теорема 3 позволили получить следующий результат:

Теорема 4. *Локализация узлов сетки в триангуляции на основе списка интерфейсных граней может быть осуществлена за время $O(N)$ в худшем случае.*

Доказательства теорем 3, 4 приведены в работе [15].

Заключение. Рассмотренные задачи локализации узлов сетки в триангуляции Делоне возникают при решении задачи оценки сходства дискретных моделей однозначных поверхностей, заданных в узлах разных нерегулярных сеток. В [16] автором был предложен эффективный подход, позволяющий вычислять меры для таких поверхностей. Подход предполагает построение триангуляций Делоне и локализацию узлов каждой из сеток в триангуляции Делоне, построенной на узлах противоположной сетки. Результаты теорем 2 и 4 были проверены и подтверждены вычислительными экспериментами, проведёнными на реальных и модельных данных.

Список литературы

1. *Скворцов А. В. Обзор алгоритмов построения триангуляции Делоне // Вычислительные методы и программирование. 2002. № 3. С. 14–39.*

2. *Kirkpatrik D. G. Optimal search in planar subdivisions // SIAM J. Comput. 1983. Vol. 12, № 1. P. 28–35.*

3. *Haran I., Helperin D. An Experimental Study of Point Location in Planar Arrangements in Cgal // ACM Journal of Experimental Algorithms. 2009. Vol. 13, № 3. P. 1–31.*

4. *Devillers O., Pion S., Teillaud M. Walking in a triangulation // Internat. J. Found. Comput. Sci. 2002. 13. P. 181–199.*

5. *Mucke E., Saias I., Zhu B. Fast Randomized Point Location Without Preprocessing in Two- And Three-dimensional Delaunay Triangulations // Proceedings of the 11th Annual Symposium on Computational Geometry. 1996. P. 274–283.*

6. *Devroye L., Mucke E. P., Zhu B. A note on point location in Delaunay triangulations of random points // Algorithmica. 1998. Vol. 22. P. 477–482.*

7. *Devroye L., Lemaire C., Moreau J. Expected time analysis for Delaunay point location // Computational geometry. 2004. Vol. 29, № 2. P. 61–89.*

8. *Скворцов А. В., Костюк Ю. Л. Эффективные алгоритмы построения триангуляции Делоне // Геоинформатика. Теория и практика. Томск: Изд–во Томского ун-та, 1998. № 1. С. 22–47.*

9. *Shapiro M. A note on Lee and Schachter's algorithm for Delaunay triangulation // Inter. Jour. of Comp. and Inf. Sciences. 1981. Vol. 10, № 6. P. 413–418.*

10. *Cheriton D., Tarjan R. E. Finding minimum spanning trees // SIAM J. Comput. 1976. Vol. 5, № 4. P. 724–742.*

11. *Костюк Ю. Л. Графический поиск с использованием триангуляции и клеточного разбиения // Вестник Томского гос. ун-та. 2002. № 275. С. 147–152.*

12. *Bose P., Devroye L. Intersections with Random Geometric Objects // Computational Geometry: Theory and Applications. 1998. Vol, 10, № 3. P. 139–154.*

13. *Местецкий Л. М., Царик Е. В. Триангуляция Делоне: рекурсия без пространственного разделения точек // Труды международной конференции по компьютерной графике и машинному зрению ГрафиКон'2004. М.: МГУ, 2004. С. 267–270.*

14. *Местецкий Л. М., Царик Е. В. Слияние неразделённых триангуляций Делоне // Сложные системы: обработка информации, моделирование и оптимизация: Сборник научных трудов. Тверь: Тверской гос. университет, 2004. Вып. 2. С. 216–231.*

15. *Дышкант Н. Ф. Меры для сравнения дискретных моделей однозначных поверхностей // Вестник Московского университета. Серия 15. Вычислительная математика и кибернетика. 2011. Т. №4. С. 41–48.*

16. *Dyshkant N. Measures for Surface Comparison on Unstructured Grids with Different Density // Lecture Notes in Computer Science: Discrete Geometry for Computer Imagery. 2011. Vol. 6607. P. 501–512.*

About Some Localization Problems in Delaunay Triangulations

Dyshkant N.F.

Keywords: computational geometry, geometric search, Delaunay triangulation, merging of overlapping triangulations, unregular discrete mesh, computational complexity

We study some problems of nodes localization in a Delaunay triangulation and problem-solving procedures. For the problem of the set of nodes the computationally efficient approach that uses Euclidean minimum spanning tree of Delaunay triangulation is proposed. Efficient estimations for computational comlexity of the proposed methods in the average and in the worst cases are proved.

Сведения об авторе:
Дышкант Наталья Федоровна,
НИВЦ МГУ имени М.В. Ломоносова,
канд. физ.-мат. наук, научный сотрудник