УДК 519.688

АЛГОРИТМЫ ПОСТРОЕНИЯ ТРИАНГУЛЯЦИИ С ОГРАНИЧЕНИЯМИ

А. В. Скворцов[1]

В работе рассматривается задача построения триангуляции с ограничениями и приводится ряд алгоритмов для ее конструирования. Обсуждается проблема вычислительной устойчивости алгоритмов триангуляции. Предлагается устойчивая модификация алгоритма построения триангуляции Делоне с ограничениями.

Ключевые слова: триангуляция Делоне, вычислительная устойчивость, вычислительная геометрия, машинная графика, геоинформационные системы, итеративные алгоритмы, выпуклая триангуляция.

Введение. Триангуляция является одной из базовых структур вычислительной геометрии [3], широко используемой в машинной графике и геоинформационных системах для моделирования поверхностей и решения пространственных задач.

В задаче построения триангуляции необходимо по заданному набору точек плоскости создать такой планарный граф, который бы разбивал плоскость на треугольники и внешнюю бесконечную фигуру.

Задача построения триангуляции является неоднозначной, и поэтому на практике используют несколько ее основных видов. Наиболее часто применяется триангуляция Делоне, т.к. она обладает рядом оптимальных свойств, относительно проста в анализе и может быть построена в среднем за линейное время [1, 4].

Однако во многих случаях на структуру триангуляции часто накладываются дополнительные ограничения. Типичным примером является триангуляция внутренности некоторого многоугольника.

В общем случае в задаче триангуляции с ограничениями задается множество отрезков, с которыми ребра триангуляции не пересекаются, а только проходят по ним. Такая общая постановка используется в геоинформатике при моделировании поверхностей, когда в качестве исходных точек берутся высотные отметки на поверхности, а ограничениями выступают так называемые структурные линии рельефа — обрывы, границы водоемов, хребты гор и т.д. (рис. 1).

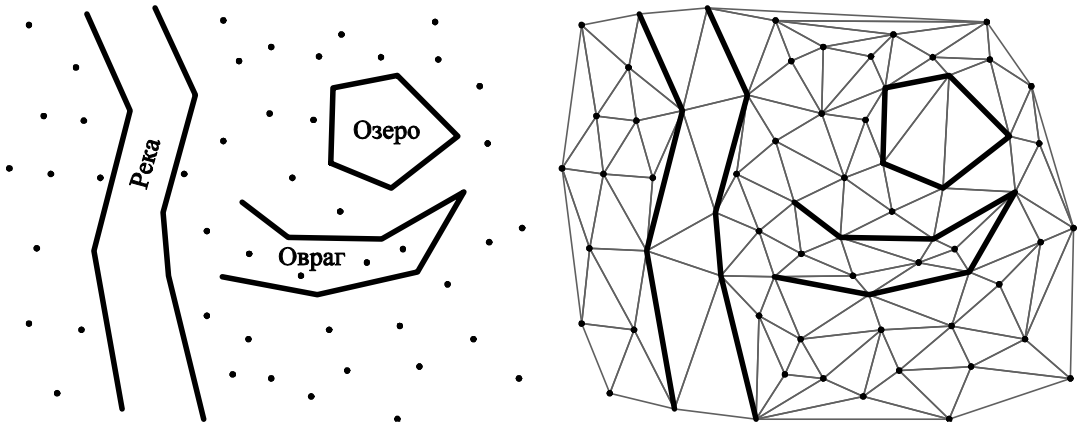

Рис. 1. Пример триангуляции с ограничениями (а — исходные данные, б — триангуляция)

Задача построения триангуляции с ограничениями также является неоднозначной, поэтому возникает вопрос выбора типа такой триангуляции. Поскольку почти все виды триангуляции трудоемки и, тем самым, не эффективны для практических реализаций, более предпочтительной оказывается триангуляция

[1] Томский государственный университет, факультет информатики, пр. Ленина, д. 36, 634050, г. Томск; e-mail: skv@csd.tsu.ru

Делоне или ее отдельные приближения. При ее использовании либо добавляются некоторые дополнительные точки в триангуляцию (чтобы выполнялось условие Делоне), либо условие Делоне игнорируется вдоль исходных отрезков-ограничений.

В настоящей работе сделан обзор наиболее распространенных алгоритмов построения триангуляции с ограничениями в соответствии со следующей схемой:

1. Жадный алгоритм построения жадной триангуляции.
2. Переборный алгоритм (метод ветвей и границ) построения оптимальной триангуляции.
3. Цепной алгоритм построения триангуляции.
4. Итеративные алгоритмы построения триангуляции Делоне с ограничениями:
   - 4.1. “Строй, разбивая”.
   - 4.2. “Удаляй-и-строй”.
   - 4.3. “Перестраивай-и-строй”.

В разделе 1 статьи представлены определения и некоторые алгоритмы, применимые при небольшом количестве исходных данных. В разделе 2 описывается один из первых алгоритмов, применимых для больших размерностей. В разделе 3 приводится группа наиболее часто используемых итеративных алгоритмов построения триангуляции Делоне с ограничениями и предлагаются различные варианты их реализации. В разделе 4 обсуждаются вопросы программной реализации алгоритмов триангуляции, а также предлагается усовершенствованный алгоритм вставки структурных отрезков в триангуляцию.

1. Виды триангуляций с ограничениями. Для дальнейшего обсуждения введем несколько определений [1, 3, 4].

Определение 1. Планарной *триангуляцией* называется разбиение плоскости на M фигур, из которых одна является внешней бесконечной, а остальные — треугольниками.

Определение 2. *Задача построения триангуляции с ограничениями.* Пусть даны множества точек $\{P_1, \ldots, P_k\}$ и отрезков $\{Q_1, \ldots, Q_l\}$. Необходимо на множестве точек $\{P_1, \ldots, P_k\}$ и концов отрезков Q_i построить триангуляцию таким образом, чтобы все отрезки проходили по ребрам триангуляции. Исходные отрезки Q_i называются структурными линиями.

Определение 3. Ребра триангуляции с ограничениями, по которым проходят исходные структурные линии, называются *структурными ребрами* (фиксированными, неперестраиваемыми).

Среди различных видов триангуляций с ограничениями выделяют три основных вида триангуляций: жадную, оптимальную и триангуляцию Делоне с ограничениями.

1.1. Жадная триангуляция. *Жадный алгоритм построения триангуляции с ограничениями* может быть представлен последовательностью следующих шагов.

Шаг 1. Во множество исходных точек помещаются концы всех структурных отрезков, затем генерируется список всех возможных отрезков, соединяющих пары исходных точек; полученный список сортируется по длинам отрезков.

Шаг 2. Выполняется вставка отрезков в триангуляцию от более коротких до длинных. Вначале вставляются все структурные отрезки, а затем вставляются остальные отрезки. Если отрезок не пересекается с другими ранее вставленными отрезками, то он вставляется, иначе он отбрасывается. *Конец алгоритма.*

Заметим, что если все возможные отрезки имеют разную длину, то результат работы этого алгоритма однозначен, иначе он зависит от порядка вставки отрезков одинаковой длины.

Условием правильной работы жадного алгоритма является отсутствие взаимных пересечений исходных структурных отрезков. Если таковые имеются, то от них надо избавиться до начала работы жадного алгоритма разбиением этих отрезков на части.

Определение 4. Триангуляция с ограничениями называется *жадной*, если она построена жадным алгоритмом.

Трудоемкость приведенного алгоритма составляет $O(N^3)$ относительно общего количества исходных точек и отрезков. Она складывается из квадратичного количества $O(N^2)$ всех возможных отрезков, каждый из которых при вставке должен быть проверен на пересечение со всеми ранее вставленными отрезками.

Данный алгоритм может быть несколько улучшен за счет ускорения процедуры проверки вставляемого отрезка на пересечение с другими. В [6] описана такая процедура на основе дерева отрезков, позволяющая выполнять одну проверку за время $O(\log N)$. Однако эта процедура достаточно громоздка в описании и реализации и поэтому используется редко. Логически более простым способом является помещение всех отрезков в R-дерево [7]. Тогда проверка пересечения будет выполняться в худшем случае за $O(N)$ операций, а в среднем — за $O(\log N)$ операций.

Таким образом, трудоемкость работы жадного алгоритма при некоторых его улучшениях составляет $O(N^2 \log N)$ [3] без учета предварительного этапа удаления пересекающихся отрезков. Если предварительный этап принимается во внимание, то сложность оценивается как $O(N^4 \log N)$, поскольку в худшем случае может образоваться $O(N^2)$ меньших отрезков. В связи со столь большой трудоемкостью на практике такой алгоритм применяется редко.

1.2. Оптимальная триангуляция. Дадим определение этого вида триангуляции.

Определение 5. Триангуляция с ограничениями называется *оптимальной*, если сумма длин всех ребер минимальна среди всех возможных триангуляций с ограничениями, построенных на тех же исходных данных.

Задача построения оптимальной триангуляции, по-видимому, является NP-полной [3], т.е. имеет сложность $O(e^N)$, а поэтому на практике она почти не применяется.

Тем не менее, иногда возникает необходимость ее построения. Для этого используется алгоритм полного перебора в глубину всех возможных триангуляций с элементами метода ветвей и границ. В его основе лежит жадный алгоритм, поэтому если прервать его работу, то получится некоторая триангуляция, суммарная длина ребер которой будет гарантированно не больше, чем у жадной триангуляции. Это свойство позволяет получать в условиях ограничений на время работы алгоритма некоторое приемлемое приближение.

Алгоритм полного перебора построения оптимальной триангуляции состоит из следующих шагов.

Шаг 1. Во множество исходных точек помещаются концы всех структурных отрезков. Затем генерируется список всех возможных отрезков $(r_1, r_2, ..., r_M)$, соединяющих пары исходных точек; полученный список сортируется по длинам отрезков. Выполняется вставка всех структурных отрезков в триангуляцию и вычисляется суммарная длина всех ребер S в триангуляции. Затем устанавливается минимальная достигнутая сумма всех ребер ($S_{\max} := \infty$).

Шаг 2. Выполняются шаги 3 – 5 алгоритма, начиная с первого элемента списка, т.е. при $k = 1$ (шаги 3 – 5 по сути образуют одну рекурсивную процедуру).

Шаг 3. Выполняется попытка вставки отрезка r_k в триангуляцию. Если отрезок не пересекается с другими ранее вставленными отрезками, то выполняем следующее. Отрезок вставляется в триангуляцию и вычисляется текущая суммарная длина ребер $S := S + s_k$, где s_k — длина ребра r_k. Если $S < S_{\max}$ и $k < M$, то выполняется шаг А. Если $S < S_{\max}$ и $k = M$, то выполняется шаг Б.

Шаг 4. Теперь отрабатывается вариант, когда отрезок r_k не включается в триангуляцию. Для этого рекурсивно выполняется шаг 3 при $k := k + 1$.

Шаг 5. Найденная триангуляция, дающая $S = S_{\max}$, принимается за искомую и работа алгоритма завершается.

Шаг А. Рекурсивно выполняется шаг 3 при $k := k + 1$. Затем полагается $k := k - 1$, $S := S - s_k$ и выполняется возврат к точке вызова данного шага.

Шаг Б. Запоминается текущая найденная триангуляция, полагается $S_{\max} := S$ и выполняется возврат к точке вызова данного шага. *Конец алгоритма.*

Данный алгоритм в худшем случае имеет трудоемкость $O(e^N)$; однако в среднем на реальных данных он показывает значительно меньшую трудоемкость, что позволяет применять его на относительно небольших размерах исходных данных.

Для построения оптимальной триангуляции при большом количестве исходных данных на практике обычно используются приближенные методы [2]. Для этого в качестве основы берется некоторая другая триангуляция, которая может быть построена достаточно быстро (например, жадная триангуляция или триангуляция Делоне), а затем она последовательно улучшается. Улучшение обычно производится локальным перестроением пар соседних треугольников либо их небольших групп. Условием выбора любого перестроения является уменьшение суммарной длины ребер.

1.3. Триангуляция Делоне с ограничениями. Введем следующее

Определение 6. Триангуляция заданного набора точек называется *триангуляцией Делоне с ограничениями*, если условие Делоне выполняется для любой пары смежных треугольников, которые не разделяются структурными ребрами.

Для построения триангуляции Делоне с ограничениями могут быть обобщены некоторые из алгоритмов построения обычной триангуляции Делоне [4, 5]. Наиболее хорошо для такого обобщения подходят итеративные алгоритмы триангуляции. Алгоритмы триангуляции слиянием с этой точки зрения хуже, т.к. в них не всегда возможно деление множества исходных объектов на непересекающиеся части (например, когда есть длинные структурные линии, проходящие через всю триангуляцию). Алгоритмы прямого построения триангуляции подходят лучше, но они должны быть модифицированы путем добавления в

процедуру поиска очередного соседа Делоне дополнительной операции проверки пересечения со структурными линиями. Большинство эффективных двухпроходных алгоритмов построения триангуляции в данном случае использовать нельзя, т.к. они либо являются неприменимыми алгоритмами слияния, либо строят огромное количество узких вытянутых треугольников, которые почти всегда приходится перестраивать.

2. Цепной алгоритм построения триангуляции с ограничениями. Один из первых эффективных алгоритмов построения триангуляции с ограничениями основан на процедуре регуляризации планарного графа и триангуляции монотонных многоугольников [3]. Трудоемкость этого алгоритма составляет $O(N \log N)$, где N — количество исходных отрезков.

Исходными данными для цепного алгоритма является множество непересекающихся отрезков на плоскости, образующих по сути планарный граф. Если в триангуляцию необходимо поместить также отдельные точки, то их следует добавить в триангуляцию уже после работы данного алгоритма (например, итеративным способом).

Цепной алгоритм построения триангуляции с ограничениями состоит из следующих шагов.

Шаг 1. Из множества исходных структурных отрезков формируется связанный планарный граф (рис. 2 (а)).

Шаг 2. Выполняется *регуляризация графа*, т.е. добавляются новые ребра, не пересекающие другие, так что каждая вершина графа становится смежной, хотя и с одной вершиной выше нее и одной ниже. Регуляризация выполняется в два прохода с помощью вертикального плоского заметания [3]. В первом проходе снизу вверх последовательно находятся все вершины, из которых не выходят ребра, ведущие вверх. Например, на рис. 2 (б) такой является вершина B. Проводя горизонтальную линию, обнаруживаем ближайшие пересекаемые ею слева и справа ребра графа AD и EF. Затем в четырехугольнике $DEHG$ находим самую низкую вершину и проводим в нее ребро из B. Аналогично выполняется второй проход сверху вниз (рис. 2 (в)). В результате работы этого шага каждая область планарного графа становится монотонным многоугольником.

Шаг 3. Каждую область графа необходимо разбить на треугольники. Для этого можно воспользоваться алгоритмом невыпуклого слияния двух триангуляций (рис. 2 (г)). *Конец алгоритма.*

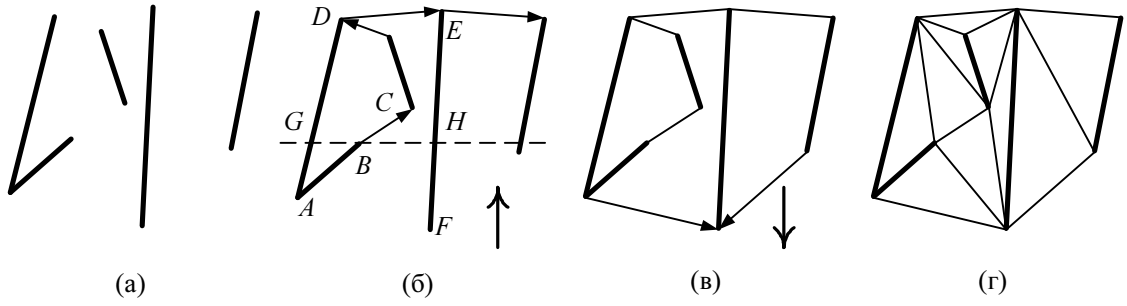

Рис. 2. Схема работы цепного алгоритма триангуляции (а — исходные отрезки, б — проход снизу вверх регуляризации графа, в — проход сверху вниз, г — триангуляция монотонных многоугольников)

Недостатком цепного алгоритма является то, что о форме получаемой триангуляции ничего заранее сказать нельзя. Это не оптимальная триангуляция, не жадная и не триангуляция Делоне с ограничениями. Кроме того, в цепном алгоритме могут получаться очень длинные вытянутые треугольники.

Для улучшения качества полученной триангуляции можно проверить все пары смежных треугольников, не разделенных структурным ребром, на выполнение условия Делоне и при необходимости произвести перестроения. В результате будет получена триангуляция Делоне с ограничениями.

3. Итеративный алгоритм построения триангуляции Делоне с ограничениями. За основу итеративного алгоритма построения триангуляции Делоне с ограничениями может быть взят любой итеративный алгоритм построения обычной триангуляции Делоне. Однако наиболее удобно использовать алгоритм динамического кэширования [4, 5], т.к. после окончания его работы будет дополнительно создана структура кэша, которая может быть использована для последующей быстрой локализации точек в триангуляции.

Итеративный алгоритм построения триангуляции Делоне с ограничениями состоит из следующих шагов.

Шаг 1. Вначале выполняется построение обычной триангуляции Делоне на множестве всех исходных точек и точек, входящих в состав структурных линий.

Шаг 2. Выполняется вставка отрезков структурных линий в триангуляцию (заметим, что к началу этого шага концы вставляемых отрезков уже вставлены в триангуляцию как узлы). *Конец алгоритма.*

Второй шаг этого алгоритма на практике может быть реализован по-разному. Рассмотрим различные варианты процедуры вставки отрезков.

3.1. Вставка структурных отрезков "Строй, разбивая". Алгоритм вставки структурных отрезков *"Строй, разбивая"* является наиболее простым в реализации и устойчивым в работе. В нем необходимо, последовательно переходя по треугольникам вдоль вставляемого отрезка, находить точки его пересечения с ребрами триангуляции (рис. 3 (а)). В этих точках пересечения нужно поставить новые узлы триангуляции, разбив существующие ребра и треугольники на части (рис. 3 (б)). После этого все вновь построенные треугольники проверяются на выполнение условия Делоне и при необходимости перестраиваются без участия фиксированных ребер.

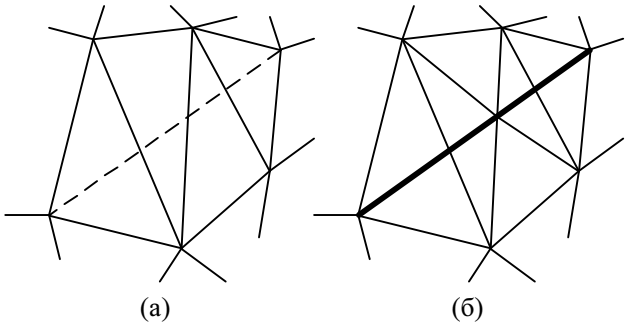

Рис. 3. Вставка структурных отрезков "Строй, разбивая"

В некоторых случаях недостатком данного алгоритма вставки может быть создание большого числа дополнительных узлов и ребер триангуляции. В то же время в других случаях этот недостаток становится преимуществом, поскольку не происходит образования длинных узких треугольников, что особенно ценится при моделировании рельефов.

Другое преимущество этого алгоритма вставки проявляется при попытке вставки структурного отрезка в триангуляцию, в которой среди пересекаемых им ребер есть фиксированные. Такие ребра, как и все остальные, просто разбиваются на две части.

3.2. Вставка структурных отрезков "Удаляй-и-строй". В алгоритме вставки структурных отрезков *"Удаляй-и-строй"* необходимо, последовательно переходя по треугольникам вдоль вставляемого отрезка, найти все пересекаемые треугольники (рис. 4 (а)) и удалить их из триангуляции (рис. 4 (б)). При этом в триангуляции образуется дырка в виде некоторого многоугольника. После этого в триангуляцию вставляется структурный отрезок, делящий многоугольник-дырку на две части (левую и правую), каждая из которых затем заполняется треугольниками (рис. 4 (в)).

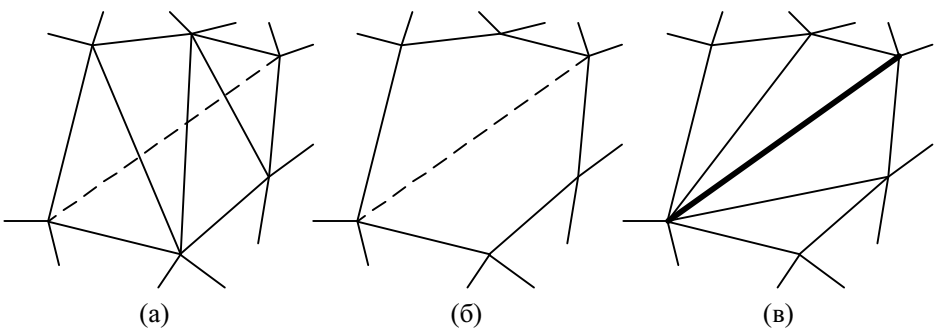

Рис. 4. Вставка структурных отрезков "Удаляй-и-строй"

Заполнение левой и правой частей дырки треугольниками можно выполнить, проходя вдоль ее гра-

ницы и анализируя тройки последовательных точек границы. Если на месте этой тройки точек можно построить треугольник, то он строится и граница укорачивается. Такой цикл повторяется, пока количество точек на левой и правой границах больше двух.

После этого все вновь построенные треугольники должны быть проверены на выполнение условия Делоне и при необходимости перестроены, не затрагивая фиксированные ребра.

Отдельным представляется случай, когда при вставке среди множества пересекаемых ребер находятся фиксированные ребра. Для избежания такой ситуации можно заранее, еще до первого шага работы алгоритма построения триангуляции Делоне с ограничениями, найти все точки пересечения всех структурных отрезков и разбить этими точками отрезки на части. Такую операцию можно выполнить, например, с помощью алгоритма заметания плоскости [3].

Другим вариантом учета пересекаемых фиксированных ребер является следующий алгоритм. Пусть при вставке очередного структурного отрезка AB обнаружено пересечение с некоторым фиксированным ребром CD в точке S (рис. 5 (а)). Тогда необходимо разбить пересекаемое ребро CD на две части CS и SD, также разбив смежные треугольники CDE и CDF на две части: $\triangle CSE$, $\triangle SDE$ и $\triangle CSF$, $\triangle SDF$ соответственно (рис. 5 (б)). После этого задача вставки исходного отрезка AB сводится к двум вставкам ребер AS и SB (рис. 5 (в)).

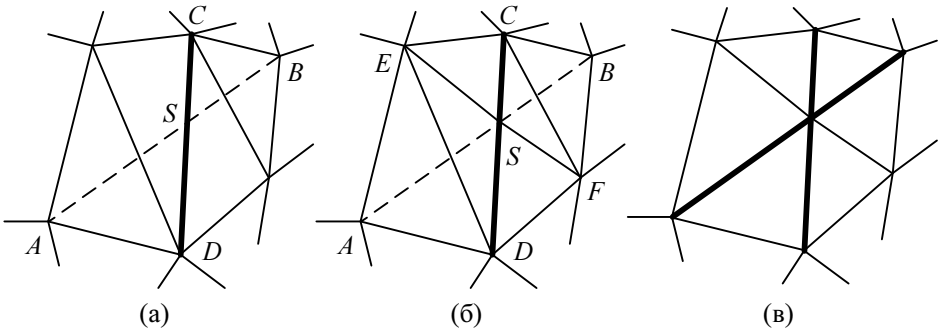

Рис. 5. Пересечение вставляемого структурного отрезка с ранее вставленным фиксированным ребром

Теперь остановимся на вопросе влияния структуры данных на реализацию алгоритма вставки “Удаляй-и-строй”. Наиболее сложной частью является удаление треугольников, временное запоминание границы области удаленных треугольников и последующее ее заполнение. Наиболее просто эта задача выполняется на структуре “Узлы, ребра и треугольники”, в которой запоминается список граничных ребер [4].

На структуре “Узлы, простые ребра и треугольники” [4], часто используемой для построения триангуляции с ограничениями, ребро представляется в неявном виде как треугольник и номер образующего ребра, т.к. при непосредственной записи ребра ссылки на смежные треугольники не формируются.

В данном алгоритме вставки возможны ситуации, когда при удалении треугольников необходимо сохранить некоторые образующие их фиксированные ребра. Например, на рис. 6 (а) изображена ситуация, когда необходимо вставить структурный отрезок AB при условии, что вблизи находится ранее вставленное фиксированное ребро KL. После удаления всех пересекаемых треугольников должно остаться ребро KL (рис. 6 (б)), после чего необходимо заполнить треугольниками многоугольник $ABKLK$ слева от ребра AB (рис. 6 (в)).

Рассматривая вариант с висячим ребром, отметим возможность ситуации, когда это ребро вообще не будет связано с границей области удаленных треугольников (рис. 7 (а), (б)). При этом процедура заполнения области треугольниками, безусловно, существенно усложняется. По сложности сама эта задача в целом эквивалентна задаче построения триангуляции Делоне с ограничениями внутри заданного региона.

Для решения столь сложной ситуации нужно временно удалить мешающее фиксированное ребро KL из триангуляции, заполнить очищенную область треугольниками, а затем повторно вставить ранее удаленные фиксированные ребра (рис. 7 (в)).

3.3. Вставка структурных отрезков “Перестраивай-и-строй”. Основная идея этого алгоритма вставки заключается в попытке уменьшения количества пересекаемых ребер за счет перестроений пар соседних треугольников до тех пор, пока не останется ни одного пересекаемого треугольника (рис. 8 (а) – (в)), т.е. вставляемый структурный отрезок не станет коллинеарным некоторому ребру триангуляции (рис. 8 (г)).

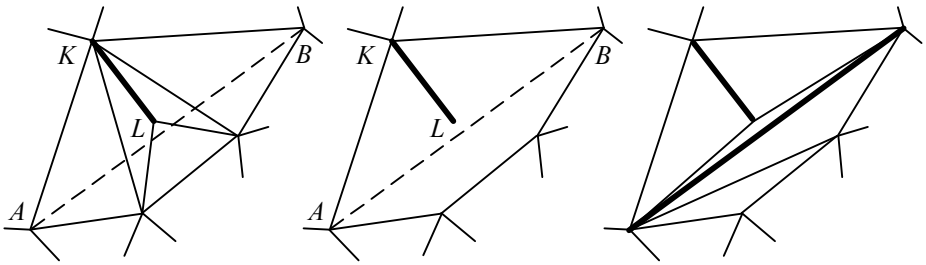

Рис. 6. Сохранение фиксированного ребра при удалении треугольников в алгоритме вставки "Удаляй-и-строй"

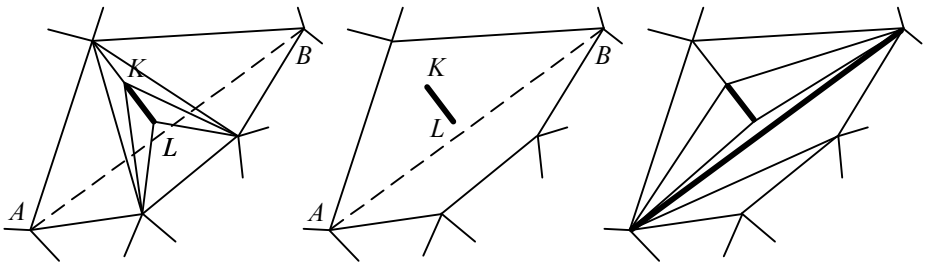

Рис. 7. Фиксированное ребро, несвязанное с границей области удаленных треугольников

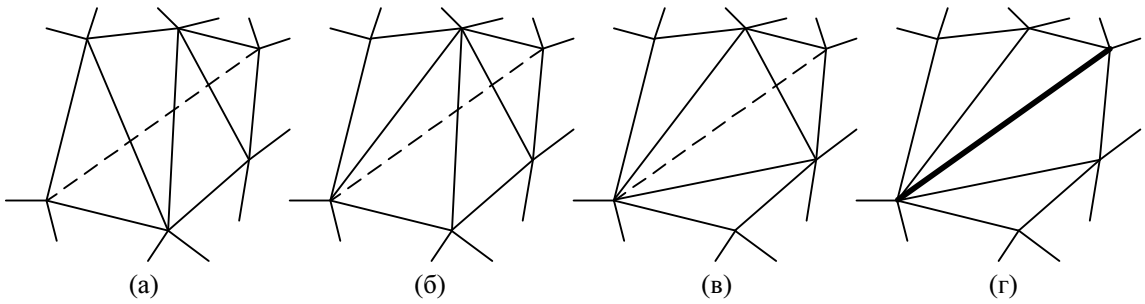

Рис. 8. Простое перестроение треугольников в алгоритме вставки структурных отрезков "Перестраивай-и-строй"

Главным недостатком этого алгоритма является возможность образования тупиковых ситуаций, когда дальнейшие перестроения пар соседних треугольников не уменьшают числа пересекаемых треугольников. Пример такой ситуации приведен на рис. 9 (а). Для разрешения ситуаций, когда нельзя выполнить перестроение треугольников с уменьшением числа пересечений, необходимо перестраивать те пересекаемые пары треугольников, у которых общая сторона образует максимальный угол со вставляемым структурным ребром. Эта операция выполняется до тех пор, пока не образуются пары смежных треугольников, перестроения которых не уменьшат число пересечений ребер. На рис. 9 (а) таким ребром (среди тех, у которых смежные треугольники можно перестроить) является AB, поэтому оно и перестраивается (рис. 9 (б)). После этого выполняются обычные перестроения с уменьшением пересечений (рис. 9 (в) – (з)).

Случай, когда вставляемый структурный отрезок пересекает уже ранее вставленное фиксированное ребро, обрабатывается так же, как и в предыдущем алгоритме вставки "Удаляй-и-строй".

Трудоемкости всех трех рассмотренных алгоритмов вставки составляют в худшем случае $O(M^2)$, где M — количество узлов в триангуляции после завершения работы алгоритма триангуляции с ограни-

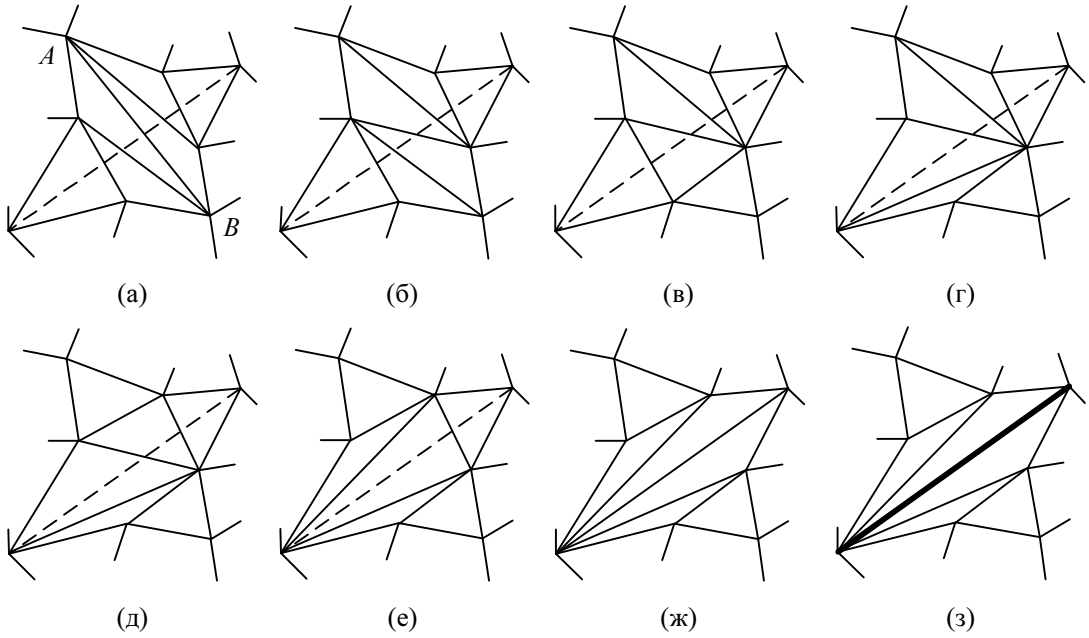

Рис. 9. Модифицированное перестроение треугольников в алгоритме вставки структурных отрезков "Перестраивай-и-строй"

чениями. Заметим, что при большом количестве взаимных пересечений структурных линий эта оценка составляет $O(N^4)$, где N — количество исходных точек и вершин исходных структурных линий.

Оценка трудоемкости в среднем очень сильно зависит от распределения структурных линий. Если их количество невелико и они мало пересекаются между собой, то общая оценка трудоемкости может составить в среднем $O(N)$.

4. Особенности реализации алгоритмов триангуляции. Проблема вычислительной устойчивости является одной из основных при решении большинства задач вычислительной геометрии. Многие внешне простые алгоритмы требуют учета многочисленных особых случаев, без чего алгоритм на практике оказывается неработоспособным [3].

4.1. Вычислительная устойчивость алгоритмов триангуляции. За внешней относительной простотой описанных выше алгоритмов триангуляции с ограничениями в действительности скрываются многочисленные детали реализации, от которых существенно зависит устойчивость работы алгоритмов. В число таких подзадач входят: а) проверка совпадения двух заданных точек; б) проверка взаимного расположения двух точек относительно прямой; в) проверка коллинеарности трех заданных точек; г) проверка взаимного расположения точки и треугольника; д) проверка порядка обхода трех заданных точек; е) проверка выполнения условия Делоне для двух заданных смежных треугольников; ж) локализация точки в триангуляции; з) поиск точки пересечения двух прямых.

В теории все эти задачи прекрасно решаются методами аналитической геометрии. Однако в силу ограниченной точности вычислений операции с компьютерными числами не обладают свойствами вещественных чисел (например, для них не выполняется свойство ассоциативности из-за потери точности в промежуточных вычислениях). Небольшими потерями точности в некоторых случаях можно пренебречь, но при построении триангуляции этого делать нельзя.

Одной из наиболее критичных к точности вычислений является операция вычисления точки пересечения двух прямых, возникающая при вставке структурного ребра в триангуляцию, пересекающего другое ранее вставленное ребро, и при дроблении ребер на части.

В данном случае первой проблемой является то, что определяемая точка пересечения в силу ограниченности точности вычислений, как правило, не лежит на ранее существовавшем ребре и не лежит на новом. Возможно, что эта точка лежит даже не в смежных с ребром треугольниках, поэтому в результате разбиения старого ребра на части образуются новые "вывернутые" треугольники, разрушающие структуру триангуляции. На рис. 10 приведен пример вставки ребра AB в триангуляцию, приводящей к

пересечению с существующим ребром в точке S (вертикальными и горизонтальными линиями размечена дискретная координатная сетка). В результате округления точка пересечения окажется немного выше реальной — в узле дискретной координатной сетки.

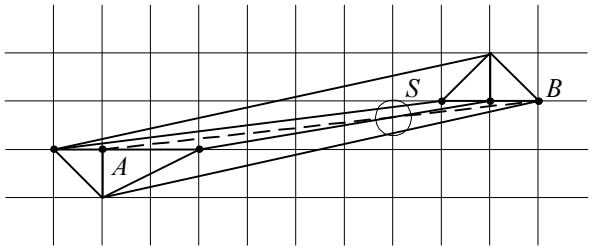

Рис. 10. Пример возможного "выворачивания" треугольников

Несмотря на кажущуюся экзотичность приведенного сценария возникновения ошибки, вероятность его велика уже при попытке вставить в триангуляцию порядка нескольких десятков взаимно пересекающихся структурных ребер. Вдоль структурных ребер образуются многочисленные узкие вытянутые треугольники, которые и создают указанную критическую ситуацию.

Вторая проблема в задаче поиска точек пересечения связана непосредственно с самим используемым способом вычислений. Обычно точка пересечения (x, y) находится следующим образом:

$$a = (x_1 - x_3)(y_4 - y_3) - (y_1 - y_3)(x_4 - x_3), \quad b = (x_4 - x_3)(y_2 - y_1) - (y_4 - y_3)(x_2 - x_1),$$
$$x = x_1 + a(x_2 - x_1)/b, \quad y = y_1 + a(y_2 - y_1)/b.$$

Здесь сложности возникают при нахождении точки пересечения двух "почти" коллинеарных отрезков. В результате потери точности возможно значительное смещение найденных координат от реального значения. Кроме того, из-за потери точности мы можем предполагать, что отрезки пересекаются, хотя это не так. Тогда попытка вычисления их пересечения может привести к значительному удалению найденной точки от самих отрезков (для "почти" коллинеарных отрезков).

Таким образом, большинство поставленных проблем связано с потерей точности внутренних вычислений.

Первый способ контроля точности при помощи стандартных вещественных типов данных, предлагаемых большинством распространенных языков программирования, весьма сложен. Как правило, в современных компьютерах поддерживаются стандарты ANSI-представления вещественных чисел, однако даже 10-байтовый тип extended позволяет хранить не более 20 значащих цифр. В то же время, операция проверки условия Делоне требует для корректных промежуточных вычислений в четыре раза большей точности. Это означает, что реальная достижимая точность построения триангуляции Делоне составляет не более $20/4 = 5$ знаков в задании координат исходных данных.

Другой способ заключается в использовании в явном виде вычислений с фиксированной точкой. В этом случае можно точно контролировать все потери точности.

Более простым является переход к целочисленному представлению координат исходных точек. Например, используя обычные 32-битные целые числа, можно обеспечить точность представления в 9 значащих цифр, что является уже приемлемым в большинстве ситуаций. Для реализации алгоритма построения триангуляции необходимо реализовать несколько дополнительных функций, оперирующих с 32-, 64- и 128-битными числами. Тогда, используя целочисленный способ представления исходных данных совместно с дополнительными операциями над длинными целыми числами, можно четко решить поставленные задачи.

Вернемся к проблеме поиска точек пересечения и разбиения вставляемых структурных ребер на части.

Несмотря на то, что мы можем выполнять вычисления с помощью дополнительных функций практически без потери точности, все равно точка пересечения двух прямых в общем случае будет иметь нецелые координаты, которые мы будем вынуждены округлять, т.е. в общем случае точка пересечения двух прямых не будет лежать на этих прямых.

Таким образом, возникает необходимость модификации всех алгоритмов вставки структурных линий так, чтобы учитывать возможные нарушения структуры триангуляции и устранять их. Рассмотрим такой обобщенный алгоритм вставки.

Обобщенный алгоритм вставки структурных линий в триангуляцию с ограничениями. Пусть L — сортированный по длине список еще не вставленных структурных отрезков. Тогда обобщенный алгоритм вставки можно представить в виде последовательности следующих шагов.

Шаг 1. Вначале в L заносим все отрезки исходных структурных линий.

Шаг 2. Последовательно в цикле извлекаем (с удалением) из L самый длинный отрезок AB и пытаемся вставить этот отрезок в триангуляцию любым алгоритмом вставки, описанным выше. Если обнаруживается, что вставляемый отрезок пересекает некоторые ранее вставленные структурные ребра (рис. 11 (а)), то их необходимо пометить как обычные нефиксированные ребра (рис. 11 (б)). При этом надо найти все точки пересечения C_i, $i = \overline{1,n}$, нового отрезка со вставленными ребрами E_iF_i. Далее нужно вставить новые точки C_i в триангуляцию (рис. 11 (в)). Затем надо в список L поместить все отрезки C_iC_{i+1}, где $i = \overline{0,n}$, $C_0 = A$, $C_{n+1} = B$. После этого необходимо поместить в список L все отрезки E_iC_i и C_iF_i, $i = \overline{1,n}$. Если вставляемый в список отрезок имеет нулевую длину, то его вставлять не надо. Цикл вставки продолжается, пока список L не пуст (рис. 11 (г)). *Конец алгоритма.*

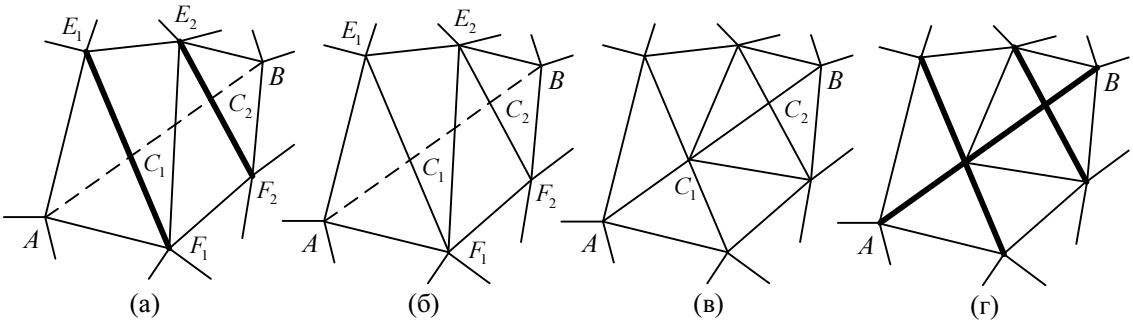

Рис. 11. Обобщенный алгоритм вставки структурных линий в триангуляцию с ограничениями

В такой реализации алгоритма вставки на практике достаточно часто возникает ситуация, когда небольшое количество исходных структурных линий приводит к значительному разрастанию списка L в процессе работы. Например, задав пять "почти" коллинеарных отрезков в качестве исходных структурных линий, можно в конце концов получить тысячи структурных ребер. Происходит это вследствие того, что каждая пара отрезков после нахождения их пересечений образует четыре новых отрезка, также являющихся "почти" коллинеарными остальным отрезкам. Такое дробление идет до тех пор, пока размеры отрезков не станут сравнимыми с размерами единицы координатной сетки, когда маленькие отрезки становятся "совсем не" коллинеарными или размер отрезков становится настолько малым, что его дальнейшее деление невозможно.

Но и на этом микроуровне возможны проблемы. Например, пусть построена некоторая "плотная" триангуляция, когда в каждом узле координатной сетки имеется по узлу триангуляции, и требуется вставить отрезок AB, который пересекается с ранее вставленным структурным ребром CD (рис. 12). В результате найденная точка пересечения AB и CD будет округлена, например, до точки A. В итоге в список L попадут отрезки AC, AD и опять AB. Так как мы извлекаем на каждом шаге из списка самое большое ребро, то список будет бесконечно разрастаться за счет постоянной вставки ребер AC и AD.

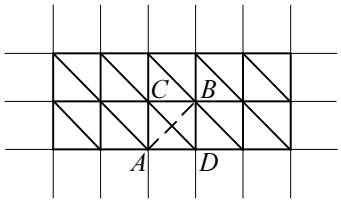

Рис. 12. Попытка вставки микроребра

Таким образом, в этом варианте алгоритм вставки также не годится для практической работы. Во избежание пересечения пар "почти" коллинеарных отрезков и проблем на микроуровне нужны еще дополнительные модификации алгоритма.

Во-первых, при вставке очередного отрезка необходимо найти все узлы триангуляции, лежащие вблизи от отрезка на расстоянии не более некоторого ε_1. Тогда если такие точки найдены, вставляемый отрезок разобьем этими точками на части, которые поместим в список L.

Во-вторых, найдя точку пересечения структурных ребер, прежде чем вставлять новый узел попробуем найти в окрестности радиуса ε_2 другой, ранее уже вставленный узел триангуляции.

На практике работа алгоритма значительно улучшается уже в тех случаях, когда значения указанных констант не меньше трех. Их увеличение приводит к сокращению размера списка L, но несколько увеличивает время выполнения дополнительного поиска точек в окрестностях. Наиболее приемлемыми с точки зрения быстродействия и качества являются значения $\varepsilon_1 = \varepsilon_2 = 10$.

В заключение отметим, что использование целочисленного представления исходных чисел позволяет, с одной стороны, явно контролировать точность вычислений, с другой — повысить скорость работы алгоритма построения триангуляции за счет отказа от вещественных операций.

Тем не менее, простой переход от вещественных вычислений к целочисленным приводит к другому неприятному эффекту — значительному росту количества структурных ребер в триангуляции. Во избежание этого приходится несколько усложнять алгоритм, вводя дополнительные проверки на наличие совпадающих узлов триангуляции с некоторой заранее заданной точностью.

СПИСОК ЛИТЕРАТУРЫ

1. *Делоне Б.Н.* О пустоте сферы // Изв. АН СССР, ОМЕН. 1934. 4. 793–800.
2. *Костюк Ю.Л., Фукс А.Л.* Приближенное вычисление оптимальной триангуляции // Геоинформатика. Теория и практика. Вып. 1. Томск: Изд-во Томского гос. ун-та, 1998. 61–66.
3. *Препарата Ф., Шеймос М.* Вычислительная геометрия: Введение / Пер. с англ. М.: Мир, 1989.
4. *Скворцов А.В.* Обзор алгоритмов построения триангуляции Делоне // Вычислительные методы и программирование. 2002. **3**. 14–39 (http://num-meth.srcc.msu.su).
5. *Скворцов А.В., Костюк Ю.Л.* Эффективные алгоритмы построения триангуляции Делоне // Геоинформатика. Теория и практика. Вып. 1. Томск: Изд-во Томского гос. ун-та, 1998. 22–47.
6. *Gilbert P.N.* New results on planar triangulations. Tech. Rep. ACT-15. Coord. Sci. Lab., University of Illinois at Urbana. Urbana, 1979.
7. *Guttmann A., Stonebraker M.* Using a relational database management system for computer aided design data // IEEE Database Engineering. 1982. **5**, N 2. 21–28.

Поступила в редакцию
17.02.2002