ПРИМЕНЕНИЕ ТРИАНГУЛЯЦИИ ДЛЯ РЕШЕНИЯ ЗАДАЧ ВЫЧИСЛИТЕЛЬНОЙ ГЕОМЕТРИИ*

А.В. Скворцов, Ю.Л. Костюк

1. Введение

Структура триангуляции множества точек на плоскости может быть использована для решения различных практически важных задач вычислительной геометрии [1]. В настоящей работе рассматриваются следующие задачи:

1. Построение буферных зон.
2. Задачи алгебры карт: объединение, пересечение и разность полигонов.
3. Построение диаграмм Вороного и взвешенных зон близости.
4. Построение изолиний и изоконтуров.

Основная идея предлагаемых алгоритмов заключается в разбиении всей задачи на элементарные задачи, решаемые на одном треугольнике триангуляции с последующим объединением полученных результатов.

Для дальнейшего рассмотрения введём понятия *полиполилинии* – фигуры, состоящей из ненулевого числа ломаных, *полиполигона* – фигуры, состоящей из ненулевого числа полигонов, причём допустимы самопересечения и пересечения различных полигонов фигуры. При этом точки плоскости, принадлежащие k полигонам фигуры, считаются принадлежащими полиполигону тогда и только тогда, когда $k \equiv 1(\text{modulo } 2)$ (рис. 1). Такие фигуры на практике часто используются для представления, например линий с разрывами и полигонов с дырками внутри.

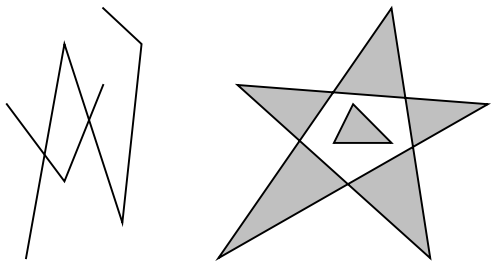

Рис. 1. Пример полиполилинии и полиполигона.

Для решения приведённых задач, кроме алгоритмов построения триангуляции Делоне, дополнительно необходимы следующие общие алгоритмы:

Алгоритм построения триангуляции с ограничениями.

Алгоритм классификации треугольников по попаданию в полиполигоны.

Алгоритм выделения полиполигонов.

2. Построение триангуляции с ограничениями

Определим задачу построения триангуляции следующим образом.

* Работа выполнена при финансовой поддержке РФФИ (грант № 98-05-03194)

Задача построения триангуляции с ограничениями. Пусть даны множества точек $\{P_1,...,P_k\}$, полиполилиний $\{Q_1,...,Q_l\}$ и полиполигонов $\{R_1,...,R_m\}$. Необходимо на множестве точек $\{P_1,...,P_k\}$, вершин полиполилиний и вершин полиполигонов построить триангуляцию таким образом, чтобы все отрезки полиполилиний и полиполигонов проходили по рёбрам триангуляции. Кроме того, если множество полиполигонов не пусто, то для всех построенных треугольников необходимо установление факта попадания в любой из заданных полиполигонов.

В такой постановке задача построения триангуляции наиболее часто используется при моделировании рельефа в геоинформационных системах. Задаваемые точки при этом определяют множество точек плоскости, в которых измерены высоты на поверхности, полиполилинии – проекции на плоскость структурных линий рельефа, а полиполигоны – области интересов [2].

В случае, когда множества точек и полиполилиний пусты, а заданы только полиполигоны, получается классическая задача построения триангуляции заданной замкнутой области [3].

Задача построения триангуляции, в том числе с ограничениями, является неоднозначной. Среди всех возможных классов триангуляций на практике чаще всего используется триангуляция Делоне [1,4]. Однако в случае дополнительного задания множества полиполилиний, которые должны пройти по рёбрам готовой триангуляции, построение триангуляции Делоне не всегда возможно. Поэтому введём следующее определение.

Определение. Триангуляция заданного набора точек будет называться *триангуляций Делоне с ограничениями*, если условие Делоне выполняется для всех рёбер, кроме некоторого набора фиксированных.

Авторами предлагается следующий алгоритм построения такой триангуляции, в котором за основу взят какой-либо итеративный алгоритм построения обычной триангуляции Делоне, например алгоритм динамического кэширования [4].

Алгоритм построения триангуляции Делоне с ограничениями. Пусть даны множества точек и полиполилиний.

Шаг 1. Строим суперструктуру, внутрь которой заведомо попадают все свободные точки и узлы полиполилиний.

Шаг 2. Цикл по полилиниям.

Шаг 2.1. Начальная точка полилинии добавляется в триангуляцию; треугольник, куда она попала, разбивается на три новых; смежные с новыми треугольниками проверяются на условие Делоне и при необходимости перестраиваются. Если

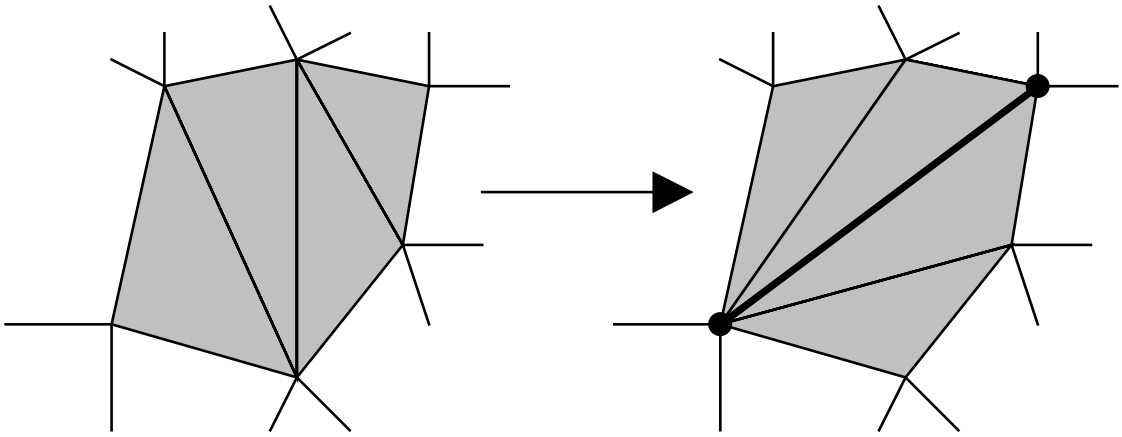

Рис. 2. Вставка фиксированного ребра.

ребро между смежными треугольниками фиксировано, то проверка не производится.

Шаг 2.2. Цикл по последующим точкам полилинии. Точка добавляется в триангуляцию, как на шаге 2.1. Отрезок между предыдущей точкой и только что добавленной проверяется на пересечение с существующими треугольниками триангуляции. Исключаем эти треугольники из триангуляции, но оставляем фиксированные рёбра. Оставшиеся фиксированные рёбра разбиваем на две части добавляемым отрезком, который также разбиваем на части. Строим новые фиксированные ребра триангуляции и делим образовавшиеся пустые области на треугольники каким-либо простым алгоритмом с постоянной проверкой условия Делоне для нефиксированных рёбер триангуляции (рис. 2). Заметим, что пустые области будут являться *почти* монотонными многоугольниками.

Шаг 3. В цикле добавляем в триангуляцию все свободные точки, как на шаге 2.1.

Пусть g – количество отрезков полилиний, n – число точек в триангуляции, а p – число точек взаимного пересечения отрезков полилиний. Тогда на первом шаге алгоритма будет проанализировано $(n+g)$ точек, на втором шаге построено $(g+p)$ узлов триангуляции, а на третьем будет достроена триангуляция с $(g+p+n)$ узлами. Таким образом, общая трудоёмкость данного алгоритма составляет (как в алгоритме динамического кэширования) в худшем:

$$T(g,n,p)=O(n+g)+O((g+p)^2)+O((g+p+n)^2)=O((g+p+n)^2), \quad (1)$$

а в среднем:

$$T(g,n,p)=O(n+g)+O(g+p)+O(g+p+n)=O(g+p+n), \quad (2)$$

если среднее число треугольников, пересекающихся с вставляемыми фиксированными ребрами, на каждом шаге ограничено константой.

3. Классификация треугольников

Теперь рассмотрим задачу классификации полученных треугольников триангуляции по признаку их попадания внутрь заданных полиполигонов.

В простейшем случае можно для каждого отдельно взятого треугольника взять любую точку внутри него и проверить её на попадание во все заданные полиполигоны. Трудоёмкость определения попадания точки в полиполигон составляет $O(m)$, где m – число точек в полиполигоне. Тогда общая трудоёмкость классификации составит $T(n,m)=O(nm)$.

Можно поступить по-другому. Пусть для каждого треугольника необходимо выставить признак $C_i=1$, если он попадает внутрь какого-либо полиполигона, и $C_i=0$, если нет. Предлагается использовать основу алгоритма растеризации произвольных полиполигонов, приведённого в [5]. Алгоритм выделяет из полиполигонов монотонные относительно вертикали части и, выполняя параллельный проход по всем частям сразу сверху вниз, разбивает весь полиполигон на набор простых треугольников или трапеций, которые в свою очередь разбиваются на сканирующие строчки. Особенностью алгоритма является то, что при отслеживании монотонных частей поддерживается информация о принадлежности точек плоскости слева и справа от линии к обрабатываемому полигону. Перекладывая данный алгоритм на задачу классификации треугольников, будем аналогичным образом выполнять параллельный проход по всем монотонным линиям, т.е. по рёбрам триангуляции. При этом мы будем сразу знать, с какой стороны от ребра находится внутренность обрабатываемого полиполигона. Остаётся выполнить методом затравки пометку всех треугольников, начиная с внутреннего, соседнего с текущим ребром.

Таким образом складывается следующий алгоритм.

Алгоритм определения попадания треугольников триангуляции во множество полиполигонов.

Шаг 1. Для каждого треугольника обнулить признаки $C_i=0$.

Шаг 2. Выполнить разложение всех контуров полиполигона на монотонные относительно вертикали линии в соответствии с алгоритмом растеризации полиполигонов. Выполнить спуск по монотонным линиям сверху вниз, отслеживая внутренность полиполигона. Проходя по каждой линии и, следовательно, по каждому ребру, начиная с его соседнего внутреннего в полиполигоне треугольника с $C_i=0$, методом затравки (как в алгоритме заливки растровой области методом затравки), обойти все треугольники внутри контура, исполь-

зуя в качестве границы помеченные рёбра, устанавливая пометки на всех пройденных треугольниках $C_i = 1$.

Трудоёмкость данного алгоритма состоит из сложности построения монотонных линий и прохода по ним, равной $O(m)$, и трудоёмкости затравки всех треугольников, равной $O(n)$. Так как $m \le n$, то общая трудоёмкость работы всего алгоритма классификации треугольников составляет $O(n)$.

4. Выделение полиполигонов

Задача выделения полиполигонов из триангуляции. Пусть мы имеем некоторую триангуляцию из m треугольников и для каждого треугольника в ней сопоставлен некоторый код C_i. Необходимо объединить все треугольники с одинаковыми кодами в полиполигоны (рис. 3).

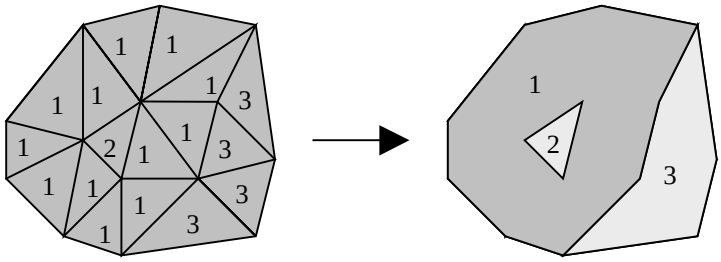

Рис. 3. Объединение треугольников в полиполигоны.

В данной работе предлагается следующий алгоритм решения поставленной задачи.

Алгоритм выделения полиполигонов из триангуляции.

Шаг 1. Для каждого треугольника T_i установить вспомогательный признак D_i в 0. Обнулить список готовых полиполигонов $L = \{\}$.

Шаг 2. Для каждого T_i с $D_i = 0$ выполнить шаги 3-4 в качестве текущего треугольника T_i. Закончить работу алгоритма и выдать L.

Шаг 3. Начиная с текущего треугольника T_i, методом затравки (как в алгоритме заливки растровой области методом затравки), построить список смежных треугольников S с кодом C_i. Для всех треугольников в S установить признак $D_i = 1$. Составить список R рёбер треугольников, не разделяющих два треугольника из этого списка. Взять произвольное ребро из списка и, используя структуру триангуляции, обойти по рёбрам из списка R контур. Обойдённые рёбра удалить из списка R. Пока список R не пуст, выполнять обходы для поиска оставшихся контуров.

Шаг 4. Проверить код C_i. Если в списке L уже есть полиполигон с таким кодом, то добавить к нему все выделенные на шаге 3 контуры.

Трудоёмкость работы третьего шага алгоритма определяется из трудоёмкости алгоритмов затравки и обходов. Пусть на третьем шаге алгоритм затрав-

ки выделил k треугольников. Следовательно, он работал время $O(k)$. Эти треугольники имеют $\theta(k)$ рёбер. Тогда обход займёт время $O(\theta(k))=O(k)$. Всего сложность третьего шага $O(k)$. Итого общая трудоёмкость алгоритма состоит из $O(m)$ на шаге 1 и $O(k_1)+O(k_2)+...+O(k_l)$ на шаге 2. А так как $k_1+k_2+...+k_l=m$, то она составляет

$$T(m)=O(m)+O(k_1)+O(k_2)+...+O(k_l)=O(m)+O(m)=O(m). \quad (3)$$

Так как число точек n и треугольников m в триангуляции линейно зависят друг от друга, то трудоёмкость приведённого алгоритма равна $O(n)$.

5. Построение буферных зон

Задача построения буферных зон требует определения всех точек плоскости, удалённых от множества объектов $\{a_i\}$ не более чем на расстояние $s_i=s(a_i)$. На практике рассматривают три вида объектов a_i: точки, ломаные и полиполигоны. На рис. 4 приведёны примеры буферных зон для этих видов объектов. Границы таких буферных зон могут состоять из множества сегментов двух видов: отрезков и дуг. Так как обработка дуг в дальнейшем очень неудобна, то их обычно аппроксимируют ломаными с некоторой заданной точностью. Поэтому, с некоторыми ограничениями, можно считать, что результатом построения буферных зон будет полиполигон.

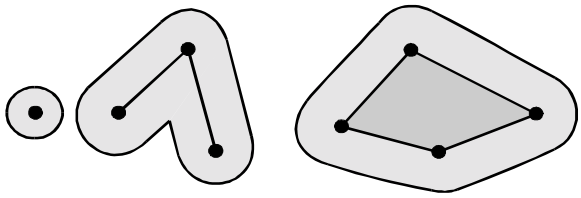

Рис. 4. Примеры буферных зон.

В данной работе предлагается следующий алгоритм решения поставленной задачи.

Алгоритм построения буферных зон.

Шаг 1. Для всех точечных объектов и вершин ломаных и полиполигонов строятся полигоны, аппроксимирующие вокруг них круговые буферные зоны. Выбор размера буферного полигона может осуществляться в виде правильного полигона, вписанного, описанного или равного по площади истинному буферному кругу, в зависимости от конечной цели.

Шаг 2. Для отрезков ломаных и полигонов вычисляются прямоугольники, которые в объединении с ранее вычисленными круговыми зонами полностью определяют буферные зоны отрезков и ломаных.

Шаг 3. Полученные круговые полигоны, прямоугольники и исходные полигоны передаются на вход алгоритма построения триангуляции с ограничениями.

Шаг 4. Все полученные треугольники классифицируются по признаку попадания в исходные полигоны.

Шаг 5. Все полученные положительно проклассифицированные треугольники объединяются в один полиполигон.

Самым трудоёмким шагом работы алгоритма является шаг 3. Пусть среди всего входного множества объектов было p точек l линий (всего t отрезков) и g полигонов (r рёбер). Аппроксимируя круговые буферные зоны с использованием s сегментов, будет построена триангуляция с $P = s \cdot (p + l + t + r) + 4t + r$ точками и таким же количеством отрезков. На основании (1)-(2) получаем общую трудоёмкость алгоритма в худшем:

$$T(p,l,t,g,r) = O((s \cdot (p + l + t + r) + 4t + r)^2) = O(s^2 \cdot n^2), \quad (4)$$

а в среднем:

$$T(p,l,t,g,r) = O(P) = O(s \cdot (p + l + t + r) + 4t + r) = O(sn), \quad (5)$$

где n – общее число точек и вершин полилилиний и полиполигонов.

Одним из вариантов построения буферных зон является буферизация со «взвешиванием», когда размер буфера является индивидуальным для каждого объекта. При этом аппроксимация круговых буферных зон может производиться полигонами с фиксированным числом вершин s либо с переменным, выбираемым на основании заданной точности. Чтобы во втором случае трудоёмкость алгоритма неограниченно не возрастала при увеличении размеров входных объектов, все индивидуальные s_i ограничивают сверху некоторым общим значением S.

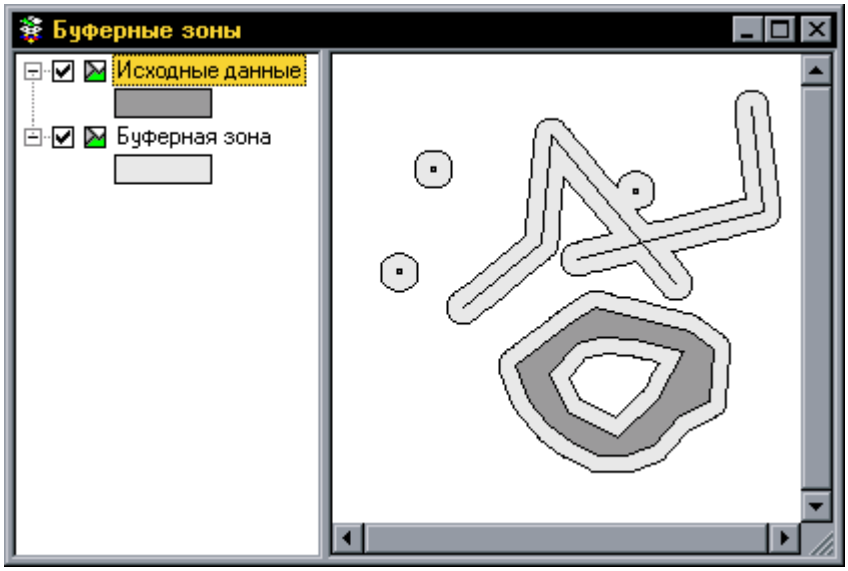

Рис. 5. Пример буферных зон.

На рис. 5 приведён пример построения буферных зон в системе ГрафИн.

6. Решение задач алгебры карт

Операции алгебры карт определяются на множествах A и B полиполигонов как задача нахождения их: 1) объединения; 2) пересечения; 3) разности; 4) симметрической разности [2]. Результат должен быть представлен в виде одного полиполигона.

В данной работе предлагается следующий алгоритм решения поставленной задачи.

Алгоритм выполнения операций алгебры карт.

Шаг 1. Всё множество полиполигонов передаём в алгоритм построения триангуляции с ограничениями в качестве замкнутых полиполилиний.

Шаг 2. Каждый треугольник T_i полученной триангуляции классифицируем в зависимости от выполняемой операции:

Вариант 1(объединение):

Если $T_i \in A$ или $T_i \in B$, то $C_i = 0$, иначе $C_i = 1$.

Вариант 2(пересечение):

Если $T_i \in A$ и $T_i \in B$, то $C_i = 0$, иначе $C_i = 1$.

Вариант 3(разность):

Если $T_i \in A$ и $T_i \notin B$, то $C_i = 0$, иначе $C_i = 1$.

Вариант 4(симметрическая разность):

Если $T_i \in A$ исключающее или $T_i \in B$, то $C_i = 0$, иначе $C_i = 1$.

Шаг 3. Выполнить алгоритм выделения полиполигонов. Выдать полученный полиполигон и завершить работу.

Заметим, что шаг 2 полностью аналогичен алгоритму классификации треугольников. Поэтому трудоёмкость шагов 1-2 равна сложности построения триангуляции с ограничениями, см. формулы (1)-(2). Третий шаг алгоритма имеет линейную трудоёмкость. Таким образом, общая трудоёмкость алгоритма выполнения операций алгебры карт составляет в худшем случае:

$$T(n) = O(n^2) + O(n) = O(n^2), \tag{6}$$

а в среднем:

$$T(n) = O(n) + O(n) = O(n), \tag{7}$$

где n – общее число вершин полиполигонов.

7. Построение зон близости

Задача построения зон близости требует определения всех точек плоскости, для которых расстояние s до объектов множества $\{a_i\}$ минимально.

В случае, когда все объекты точечные, данная задача определяется как задача построения диаграмм Вороного (полигонов Тиссена, ячеек Дирихле) [1]. Известно, что такая структура является дополнительной по отношению к триангуляции Делоне на том же наборе точек. Поэтому её построение на основе

триангуляции Делоне не представляет сложности. Пример построения зон близости в системе Граф-Ин приведён на рис. 6.

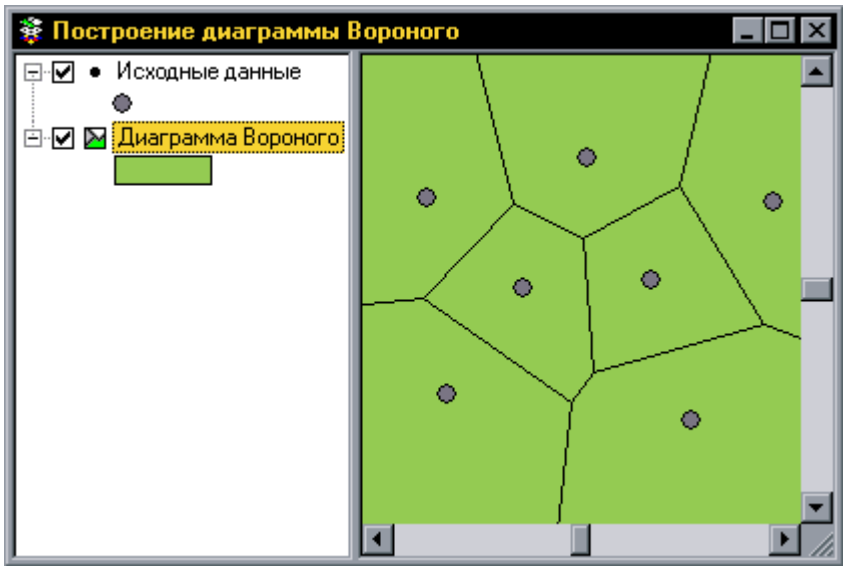

Рис. 6. Пример диаграммы Вороного.

На практике данная задача может использоваться, например, для нахождения зон скорейшего обслуживания (достижимости) из заданных базовых пунктов. Однако в действительности возможности базовых пунктов (скорость движения из них, удельные затраты на перемещение) могут быть разными.

Задача построения взвешенных зон близости требует определения всех точек плоскости, для которых расстояние s до объектов множества $\{a_i\}$, помноженное на веса $\{w_i \geq 0\}$, является минимальным. В случае если все веса равны нулю, будем считать (если трактовать веса, как скорость), что ни одна точка плоскости, кроме $\{a_i\}$, не достижима.

Для решения данной задачи рассмотрим случай двух точечных объектов a_1 и a_2 с весами w_1 и w_2. Будем считать, что эти веса ненулевые, иначе решение тривиально. Если $w_1 = w_2$, то решением являются две полуплоскости, разделённые прямой – срединным перпендикуляром к отрезку a_1, a_2. Иначе пусть $w_1 > w_2 > 0$; $e = \dfrac{w_2}{w_1}$. Тогда геометрическое место точек (x, y), ближайших ко второй точке, определяется следующим соотношением:

$$\sqrt{(x-x_1)^2+(y-y_1)^2}\cdot w_1 < \sqrt{(x-x_2)^2+(y-y_2)^2}\cdot w_2. \qquad (8)$$

Как видно, данное соотношение определяет круг, поэтому найдём уравнение определяющей его окружности в явном виде:

$$\left((x-x_1)^2+(y-y_1)^2\right)\cdot w_1^2 = \left((x-x_2)^2+(y-y_2)^2\right)\cdot w_2^2; \quad \Rightarrow$$

$$\left(x-\frac{(x_1-x_2e^2)}{(1-e^2)}\right)^2 - \frac{e^2(x_1-x_2)^2}{(1-e^2)^2} + \left(y-\frac{(y_1-y_2e^2)}{(1-e^2)}\right)^2 - \frac{e^2(y_1-y_2)^2}{(1-e^2)^2} = 0. \qquad (9)$$

Отсюда получаем координаты центра окружности (x_c, y_c) и радиус R:

$$x_c = \frac{(x_1 - x_2 e^2)}{(1-e^2)}, y_c = \frac{(y_1 - y_2 e^2)}{(1-e^2)}, R = \frac{e\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}}{(1-e^2)}. \quad (10)$$

Для решения задачи в случае многих объектов предлагается рассмотреть все возможные пары $\{(a_i, a_j) \mid w_i > 0, w_j > 0\}$, получить для них разбивающие плоскость линии $l_{i,j}$ (прямые или окружности). Теперь разобьём плоскость сразу всеми полученными линиями $l_{i,j}$. При этом заметим, что каждый получившийся элемент разбиения r_k целиком принадлежит какой-то одной зоне достижимости (иначе бы некоторые две точки из r_k принадлежали разным зонам, достижимым из некоторых a_i, a_j, но тогда они должны быть разделены линией $l_{i,j}$, т.е. принадлежать разным элементам разбиения). После этого необходимо проклассифицировать все элементы разбиения на принадлежность соответствующим зонам и объединить их в полиполигоны, соответствующие зонам. Таким образом получается алгоритм:

Алгоритм построения взвешенных зон близости.

Шаг 1. Из входного множества объектов $\{a_i\}$ удаляем все те, веса которых равны нулю. Если объектов не осталось, то выдать сообщение о некорректности задачи и закончить алгоритм. Если объект остался один, то получается одна зона достижимости в виде всей плоскости без входных точек, имеющих нулевые веса. Если все оставшиеся объекты имеют одинаковые веса, выполнить алгоритм построения диаграмм Вороного и закончить алгоритм.

Шаг 2. Пусть *B* является минимальным объемлющим все входные объекты прямоугольником. Получим из него прямоугольник *R* с помощью растяжения относительно центра в некоторое заведомо большое число раз (например, в 100 раз). Прямоугольник *R* будем считать областью интересов, т.е. областью, в которой нам необходимо получить результат. Для каждой пары объектов (a_i, a_j) необходимо найти $l_{i,j}$ (прямую или окружность). Выполнить её отсечение прямоугольников *R*. Если $l_{i,j}$ является окружностью, то аппроксимируем её полилинией. Количество точек аппроксимации *s* можно задать фиксированным либо вычислять для каждой окружности индивидуально, исходя из заданной точности построений.

Шаг 3. Все полученные на предыдущем этапе отрезки прямой и аппроксимирующие окружность полилинии необходимо подать в качестве структурных линий на вход алгоритма построения ограниченной триангуляции Делоне.

Шаг 4. Для каждого полученного треугольника необходимо выбрать из множества $\{a_i\}$ ближайший достижимый объект.

Шаг 5. Выполнить алгоритм выделения полиполигонов. Выдать полученные полиполигоны и завершить работу.

Трудоёмкость работы данного алгоритма складывается из линейной по числу входных объектов n: $O(n)$, квадратичной от n и линейной от s этапов 2-5: $O(sn^2)$. Итого общая трудоёмкость всего алгоритма составляет

$$T(n) = O(sn^2). \qquad (11)$$

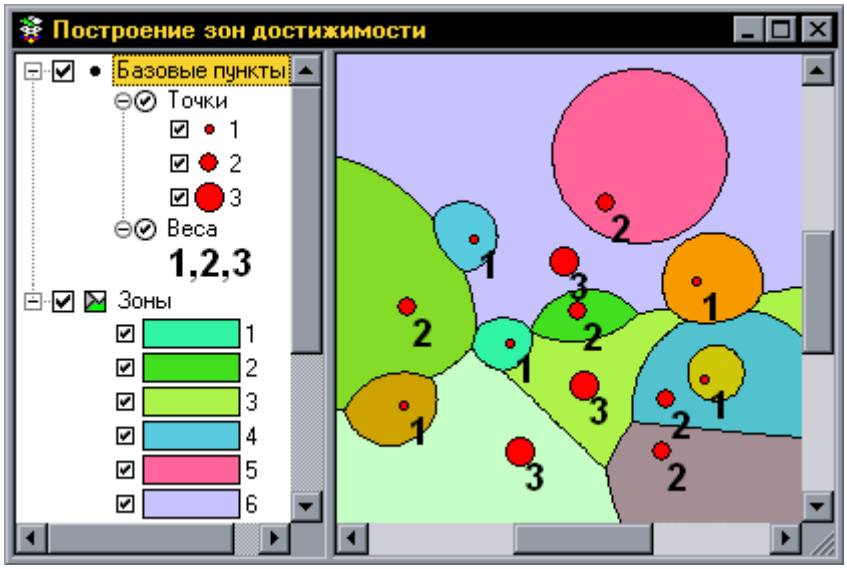

Рис. 7. Пример зон достижимости.

На рис. 7 приведён пример построения взвешенных зон достижимости в системе ГрафИн.

8. Построение изолиний и изоконтуров

Структура триангуляции на практике часто применяется для моделирования различного рода полей и поверхностей. При этом одной из наиболее распространённых операций при их анализе является построение различного рода изолиний, соединяющих точки региона или поверхности с некоторыми одинаковыми значениями, например высот (горизонтали), уклонов (изоклины), температуры (изотермы), давления (изобары) и т.п.

После построения изолиний вся обрабатываемая область оказывается разбитой полученными линиями на некоторые регионы, называемые изоконтурами. Они имеют смысл множеств точек, в которых анализируемая величина изменяется в диапазоне, определяемом ограничивающими изолиниями.

На практике построение изолиний в произвольно заданной области является относительно сложной задачей, но довольно просто выполняется на элементарных областях, например на треугольниках. Поэтому, предполагая известным алгоритм построения изолиний на треугольниках, общий алгоритм построения изолиний можно свести к построению триангуляции (если её ещё нет) в анализируемой области, локальному построению изолиний и вызову алгоритма выделения полиполилиний из триангуляции.

Построение изоконтуров можно также выполнить локально, получив разбиение каждого треугольника на полигоны, определяющие изоконтуры. При этом предлагается следующий общий алгоритм построения изоконтуров.

Алгоритм построения изоконтуров.

Шаг 1. В заданной триангуляции помечаем каждое ребро как фиксированное (не подлежащее перестроению). Все треугольники триангуляции запоминаем как первичные.

Шаг 2. Для каждого первичного треугольника выполняем его разбиение изолиниями на изоконтуры с помощью некоторого известного алгоритма. Вносим полученные изолинии в триангуляцию в качестве фиксированных рёбер. Все треугольники внутри текущего первичного треугольника необходимо проклассифицировать по принадлежности изоконтурам некоторым известным алгоритмом.

Шаг 3. Вызываем алгоритм выделения полиполигонов из триангуляции по признаку принадлежности различным изоконтурам.

Особенностью данного алгоритма является то, что дополнительно используется некоторый известный алгоритм принадлежности треугольника некоторому изоконтуру, так как по одним изолиниям это определить не всегда возможно, например в случае, когда изоконтур ограничивается изолиниями с одинаковыми значениями высот, уклонов и т.п.

9. Заключение

Триангуляция является универсальной структурой для решения самых разнообразных задач вычислительной геометрии и машинной графики. Её применение позволяет не только упростить решение возникающих задач, но и в ряде случаев существенно улучшить трудоёмкость.

ЛИТЕРАТУРА

1. Препарата Ф., Шеймос М. Вычислительная геометрия: Введение / Пер. с англ. – М.: Мир, 1989. – 478 с.
2. Кошкарёв А.В., Тикунов В.С. Геоинформатика. – М.: Картгеоцентр-Геодезиздат, 1993. – 213 с.
3. Ильман В.М. Алгоритмы триангуляции плоских областей по нерегулярным сетям точек // Алгоритмы и программы, ВИЭМС, вып. 10 (88). – М., 1985, с 3-35.
4. Скворцов А.В., Костюк Ю.Л. Эффективные алгоритмы построения триангуляции Делоне // Наст. книга, с. 22-47.
5. Роджерс Д., Адамс Дж. Математические основы машинной графики / Пер. с англ. – М.: Машиностроение, 1980. – 204 с.