УДК 519.683

А.В. СКВОРЦОВ

РАЗБИЕНИЕ МНОЖЕСТВА ОТРЕЗКОВ НА НЕПЕРЕСЕКАЮЩИЕСЯ ЧАСТИ НА ДИСКРЕТНОЙ ПЛОСКОСТИ

Рассматривается задача разбиения отрезков на пересекающиеся части на дискретной плоскости. Предлагаются алгоритм решения данной задачи и пути его дальнейшего улучшения. Анализируется трудоемкость.

Введение

Проблема поиска всех пересечений множества отрезков на плоскости приводится во многих книгах как пример простейшей задачи, решаемой, например, с помощью метода заметания (сканирования) плоскости [1,2]. Она возникает в самых различных приложениях, например при удалении невидимых линий [3], отсечении линий и многоугольников [2,3], построении триангуляции с ограничениями [4], в геоинформатике при построении покрытий [5] и во многих других случаях. Сформулируем её так.

Задача поиска пересекающихся пар отрезков**.** Дано множество отрезков $P = \{(x_j^A, y_j^A) - (x_j^B, y_j^B)\}$. Требуется найти все пары пересекающихся отрезков.

Данная задача в классической евклидовой геометрии имеет трудоемкость $O((r+n)\log n)$, где n – число исходных отрезков, а r – число пересекающихся пар. В худшем случае получается трудоемкость $\Theta(n^2)$, поскольку всего имеется $n(n-1)/2$ пар отрезков.

В практической постановке требуется не просто найти все пересекающиеся пары, но и найти точки пересечения пар отрезков, чтобы в итоге разбить все исходные отрезки на множество непересекающихся отрезков. Таким образом, возникает следующая родственная задача.

Задача разбиения отрезков на непересекающиеся части**.** Дано множество отрезков $P = \{(x_j^A, y_j^A) - (x_j^B, y_j^B)\}$. Требуется найти все точки пересечения отрезков множества P и разбить этими точками пересекающиеся отрезки так, чтобы полученные меньшие отрезки не пересекались (допускается только их касание концами).

Особенности задачи на дискретной плоскости

В классической евклидовой непрерывной геометрии вторая задача сводится к первой. Однако, как правило, реальные компьютерные вычисления выполняются только с ограниченной точностью, что приводит в рамках данной задачи к интересным эффектам. Например, найденная точка пересечения отрезков в силу ограниченности точности вычислений оказывается немного в стороне от реального вещественного значения. В результате два исходных пересекающихся отрезка после разбиение на две части точкой пересечения оказываются не лежащими на прямых, проходящих через исходные отрезки. В итоге это может привести к пересечению этих новых отрезков с какими-то другими ранее не пересекаемыми.

Данная ситуация иллюстрируется на рис. 1. Пунктирной сеткой размечена система координат, в которой каждый квадрат соответствует минимально возможной точности

представления координат точек (в частном случае использования целых чисел шаг сетки будет равен единице).

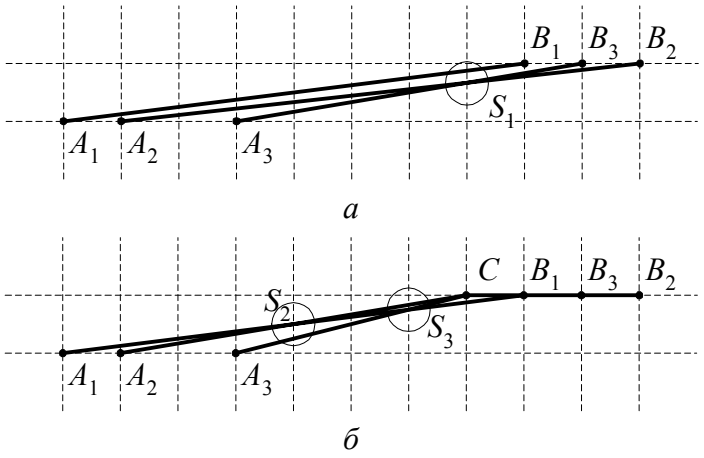

Рис. 1. Проблема поиска пересечений

На рис. 1,*а* изображены отрезки A_iB_i, $i=1,2,3$, причем второй и третий пересекаются в точке S_1.

После разбиения второго и третьего отрезков на части точкой их пересечения $C = округление(S_1)$ образуются четыре новых отрезка A_2C, A_3C, B_2C, B_3C, из которых первые два пересекаются с отрезком A_1B_1 в точках S_2 и S_3 соответственно.

Данная проблема часто возникает, если во множестве исходных отрезков имеется несколько коллинеарных, которые из-за ограниченной точности представления координат концов отрезков в действительности таковыми не являются и пересекаются.

Таким образом, при разработке алгоритма разбиения множества отрезков на непересекающиеся части необходимо учесть возможность появления новых пересечений и при необходимости разбивать в том числе и вновь образуемые отрезки.

Алгоритм решения

Поставленная задача в дискретной постановке, видимо, не может быть решена методом заметания, так как при обнаружении какого-либо очередного пересечения отрезков может существенно поменяться само множество отрезков и могут появиться новые точки пересечения, уже позади заметающей прямой.

Поэтому для решения поставленной задачи на дискретной плоскости автором предлагается следующий алгоритм.

Алгоритм разбиения множества отрезков на непересекающиеся части. Предполагается, что дано множество исходных отрезков $P = \{(x_j^A, y_j^A) - (x_j^B, y_j^B)\}$.

Шаг 1. Формируем множество узловых точек $N = \{n_i\}$, по очереди добавляя в него концы всех исходных отрезков; совпадающие точки отбрасываем. Искомое множество непересекающихся отрезков делаем пустым: $R = \varnothing$.

Шаг 2. Заносим в L – список еще не обработанных отрезков – все исходные отрезки в виде пар (n_j^A, n_j^B).

Шаг 3. Пока список L не пуст, извлекаем из него самый длинный отрезок (n^A, n^B) и

пытаемся вставить его во множество *R*. Если такой отрезок уже есть в *R*, то ничего не делаем. Если текущий отрезок не пересекает ни один ранее вставленный в *R* отрезок и не проходит через узлы из *N*, кроме n^A и n^B, то вставляем его. В противном случае, если имеет место последовательное прохождение через узлы $\tilde{n}^1,...,\tilde{n}^m \in N$, то заносим в список *L* новые отрезки: $(n^A,\tilde{n}^1),(\tilde{n}^1,\tilde{n}^2),...,(\tilde{n}^m,n^B)$. Иначе, если обнаружено последовательное пересечение с некоторыми ранее вставленными отрезками $r^1,...,r^m, r^i=(n_i^A,n_i^B)$ в точках $(\hat{x}^i,\hat{y}^i)$, то удаляем эти отрезки из *R* и заносим в список *L* новые отрезки $(n^A,\hat{n}^1),(\hat{n}^1,\hat{n}^2),...,(\hat{n}^m,n^B)$ и $(n_i^A,\hat{n}^i)$, $(\hat{n}^i,n_i^B)$, $i=\overline{1,m}$, где $\hat{n}^i$ – старые или вновь добавленные узлы в точке $(\hat{x}^i,\hat{y}^i)$. *Конец алгоритма*.

При моделировании работы данного алгоритма часто возникала ситуация, когда небольшое количество исходных отрезков приводит к значительному разрастанию списка *L* в процессе работы.

Например, возьмем 5 вложенных отрезков, лежащих примерно на одной прямой под углом почти 45°:

```
(1; 0)-(1 000 000; 1 000 128),
(100 001; 100 013)-(900 000; 900 114),
(200 001; 200 026)-(800 001; 800 102),
(300 000; 300 039)-(700 001; 700 089),
(400 000; 400 051)-(600 000; 600 077).
```

Будем считать, что координаты отрезков можно представлять только целыми числами. В итоге после выполнения данного алгоритма образуется 3420 непересекающихся отрезков. Кроме того, за время работы алгоритма в списке *L* побывало 13516 отрезков. Общее время вычислений составило около 1 мин на компьютере с процессором Pentium III 1 Гц.

Происходит это вследствие того, что каждая пара отрезков после расчета их пересечения образует 4 новых отрезка, также являющихся «почти» коллинеарными другим. 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 их размер становится настолько малым, что его дальнейшее деление невозможно.

Но и на этом микроуровне возможны проблемы. Например, пусть в каждом узле координатной сетки имеется по узлу из множества *N* и требуется добавить отрезок *AB*, который пересекается с ранее вставленным отрезком *CD* (рис. 2). 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 самый большой отрезок, то он будет бесконечно разрастаться за счет постоянной вставки отрезков *AC* и *AD*.

Таким образом, в таком варианте алгоритм все же не годится для практической работы.

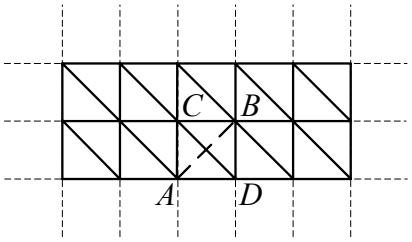

Рис. 2. Попытка вставки микроотрезков во множество *R*

Для избежания возникновения пересечения пар «почти» коллинеарных отрезков и проблем на микроуровне автором предлагается две модификации алгоритма.

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

Во-вторых, найдя точку пересечения отрезков, прежде чем вставлять новую, попробуем найти в окрестности радиуса ε_2 уже ранее вставленный узел из *N*.

В проведенном автором экспериментальном моделировании работы алгоритма значительное улучшение наблюдалось уже при значениях $\varepsilon \geq 1$. Увеличение ε приводит к сокращению размера списка *L*, но немного увеличивает время выполнения дополнительного поиска точек в окрестностях. Экспериментально автором были выбраны наиболее приемлемые с точки зрения быстродействия и качества значения $\varepsilon_1 = \varepsilon_2 = 10$.

Например, для приведенного выше случая уже при $\varepsilon_1 = 1,\ \varepsilon_2 = 0$ количество полученных отрезков составило 9, а не 3420, что и следовало бы ожидать, если бы исходные отрезки были коллинеарные.

Улучшения алгоритма

Остановимся на некоторых дополнительных возможностях по дальнейшему улучшению предложенного алгоритма.

Во-первых, для более эффективного поиска вершин в заданной окрестности можно поместить все вершины в бинарное дерево поиска [2]. Тогда поиск в указанной окрестности будет выполняться в среднем за логарифмическое, а не линейное время. Точно также, все ребра следует поместить в структуру *R*-дерева [6] для ускорения поиска возможных пересечений отрезков.

Такие улучшения позволяют в среднем достичь трудоемкости работы алгоритма $O((r+n)\log n)$, где n – число исходных отрезков, а r – число пересекающихся пар.

Во-вторых, после вычисления точек пересечения отрезков в список *L* помещаются отрезки, в чьи координаты входит некоторая ошибка округления. Когда эти отрезки в свою очередь будут пересекаться с другими, ошибка будет накапливаться. В результате точки пересечения могут уйти сколь угодно далеко от исходных отрезков. Особенно эта проблема является актуальной при $\varepsilon_1 \leq 1,\ \varepsilon_2 \leq 1$.

В связи с этим видится целесообразным для каждого отрезка в списках *L* и *R* дополнительно хранить координаты исходных отрезков. При этом при вычислении пересечения (x, y) каждой пары отрезков следует дополнительно вычислять вещественные координаты

пересечения исходных отрезков (x_0, y_0) и округление координат (x, y) производить в направлении (x_0, y_0).

Обратим внимание на то, что в данном случае может возникнуть соблазн вообще использовать в качестве точки пересечения (x, y) координаты (x_0, y_0). Но так нельзя делать, так как возможны случаи, когда точка пересечения исходных отрезков (x_0, y_0) может располагаться очень далеко от (x, y) и даже вообще не существовать, если исходные отрезки не пересекались.

Проведенное автором моделирование работы алгоритма с данной модификаций показало, что в ряде случаев действительно значительно уменьшилась степень отклонения результирующих отрезков от исходных.

СПИСОК ЛИТЕРАТУРЫ

1. Препарата Ф., Шеймос М. Вычислительная геометрия: Введение / Пер. с англ. – М.: Мир, 1989. – 478 с.
2. Ласло М. Вычислительная геометрия и компьютерная графика на С++ / Пер. с англ. – М.: Изд-во БИНОМ, 1997. –304 с.
3. Роджерс Д. Алгоритмические основы машинной графики / Пер. с англ. – М.: Мир, 1989. – 512 с.
4. Скворцов А.В., Костюк Ю.Л. // Геоинформатика. Теория и практика. Вып. 1. – Томск: Изд-во Том. ун-та, 1998. – С. 22-47.
5. Кошкарёв А.В., ТикуновВ.С. Геоинформатика. – М.: Картгеоцентр-Геодезиздат, 1993. – 213 с.
6. Guttmann A. // Proc. ACM SIGMOD Int. Conf. on Management of Data. – 1984. – P. 47-57.

Томский госуниверситет

Поступила в редакцию 01.12.01.