УДК 681.142.2

Д.А. КОЧКАРЬ[1], В.В. БОГОМОЛОВ[1], А.В. ОСТАПЧИК[2]

[1]Научно-производственное предприятие «Лесинформ», Украина
[2]Украинский НИИ лесного хозяйства и агролесомелиорации им. Г.Н. Высоцкого, Украина

АЛГОРИТМ ФОРМИРОВАНИЯ ПЛАНАРНОГО ГРАФА ПРИ ПОДГОТОВКЕ ЦИФРОВЫХ ЛЕСНЫХ КАРТ

Рассмотрен алгоритм преобразования неупорядоченного набора линейных векторных объектов, полученных в результате векторизации лесных растровых карт, в правильный планарный граф. Приведены особенности работы алгоритма с учетом специфики территориального деления лесного хозяйства.

планарный граф, геометрическая точность, близкие геометрические объекты, инцидентность

Введение

Лесные карты представляют собой картографическую основу для ведения лесного хозяйства. Базовой разновидностью лесных карт является лесоустроительный планшет масштаба 1:10000, на котором изображается территориальная организация лесохозяйственного предприятия или, другими словами, деление его территории на лесные уро-чища, кварталы и выделы.

В геометрическом отношении урочища, кварталы и выделы представляют собой многоугольники (полигоны). При этом полигоны кварталов полностью покрывают полигоны урочищ, а полигоны выделов – полигоны кварталов. Сторона полигона одного квартала (если она не совпадает с границей урочища) является одновременно сто-роной смежного полигона квартала. Аналогично, сторона полигона одного выдела является стороной другого выдела. Таким образом, стороны смежных полигонов, опирающиеся на вершины с оди-наковыми координатами, пространственно совпа-дают с границами кварталов или выделов. Другими словами, полигоны урочищ, кварталов и выделов можно представить через их границы. Каждая такая граница представляет собой различимый на местности линейные элемент, который тем или иным условным знаком вычерчивается на лесоустроительном планшете.

Совокупность границ кварталов и выделов в пределах урочища представляет собой такую структуру, которая позволяет проследовать от одной границы до любой другой. В теории графов подобные структуры называются планарными графами [1].

Цель работы: алгоритмизация задачи построения полигонов кварталов и выделов с помощью представления совокупности их границ в виде планарного графа и применения методов вычислительной геометрии [2].

Входные данные алгоритма

В качестве исходных данных для получения планарного графа используется набор ломаных (полилиний), полученных в результате векторизации растровой подложки в некой графической оболочке.

Графическая оболочка предоставляется тем или иным программным средством из разряда геоинформационных систем. Как показывает опыт работы по векторизации, ее производительность тем выше, чем больше свободы действий получает пользователь при обрисовке линий. Максимум

свободы действий достигается, когда пользователь обводит линии по принципу «что вижу, то и рисую», будучи избавлен от стыковки линий между собой, проверки перекрытия частей ломаных и согласованности всех линий как границ выделов. Одной из задач данного алгоритма и есть приведение произвольного набора полилиний в математическую модель планарного графа.

Таким образом, полилиния представляет собой упорядоченное множество *PolyLyne.*

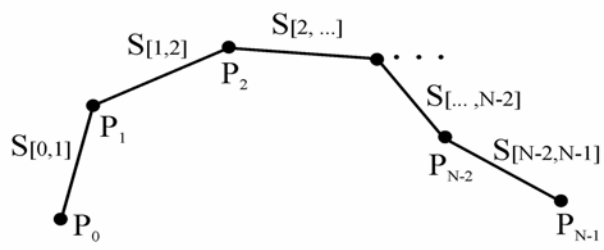

Рис. 1. Общий вид полилинии графа

$$PolyLine = \langle Rang, Style, Border, SetP \rangle \quad (1)$$

где: *Rang* – ранг полилинии, используется для обработки и преобразования входных полилиний;

Style – вид линии внутри одного ранга, то есть смысл линии на карте;

Border – признак свойства полилинии быть границей урочищ, кварталов, выделов или быть линией, имеющей другой смысл на карте;

SetP – упорядоченное множество точек плоскости, задаваемых декартовыми координатами (рис. 1).

$$P_i = \langle x_i, y_i \rangle, \quad (2)$$

$$SetP = \langle P_0, P_1, P_2, ...P_{N-2}, P_{N-1} \rangle, \quad (3)$$

Полилиния составлена из отрезков прямых линий, ограниченных парами последовательных точек плоскости из множества *SetP* :

$$PolyLine = \bigcup_{i=0}^{i=N-2} S_{i,i+1}, \quad (4)$$

$$S_{i,i+1} = [P_i, P_{i+1}], \quad (5)$$

$$SetPolyLine = \bigcup_{j=0}^{j=M-1} Polyline_j, \quad (6)$$

где *SetPolyLine* – множество произвольно расположенных на плоскости полилиний разного вида и ранга *SetPolyLine* и образуют данные, которые обрабатывает, преобразовывает и исправляет алгоритм.

Отождествление точек плоскости

При векторизации оператором растрового изображения проблемой является точная стыковка различных геометрических объектов: точек и сегментов полилиний.

Поскольку векторизация ведется с некоторой точностью, называемой геометрической точностью, которая намного больше видимой на экране монитора в масштабе работы оператора, то возникает большое количество ошибок крайне трудных для ручного исправления, поскольку требует просмотра всех элементов в большом увеличении.

Эта проблема исчерпывающим образом разрешается в алгоритме путем введения понятия близости точек с геометрической точностью, введения отношения эквивалентности на всем мно-жестве точек из входных данных и на основании этого отношения отождествление близких точек.

Будем считать две точки близкими, если расстояние между ними меньше геометрической точности, расширим это отношение до отношения эквивалентности путем введения транзитивности, то есть

$$\rho(P,Q) < \varepsilon \rightarrow P \sim Q, \quad (7)$$

$$\begin{cases} P \sim P, \\ P \sim Q \rightarrow Q \sim P, \\ P \sim Q \,\&\, Q \sim R \rightarrow P \sim R. \end{cases} \quad (8)$$

где *P, Q, R* – точки плоскости.

Это отношение рефлексивно, симметрично и транзитивно (7,8), поэтому все множество точек разбивается на классы эквивалентности, каждый из которых отождествляются с одной точкой плоскости. При этом пары точек стягиваются к средней точке, если принадлежат к полилиниям одного ранга, или точка с более низким рангом подвигается в точку с более высоким рангом. Перебирая все пары точек одного класса эквивалентности, стягиваем весь класс в одну точку. Таким образом, автоматически устраняются различные нестыковки (рис. 2) невидимые в обычном масштабе работы на экране монитора.

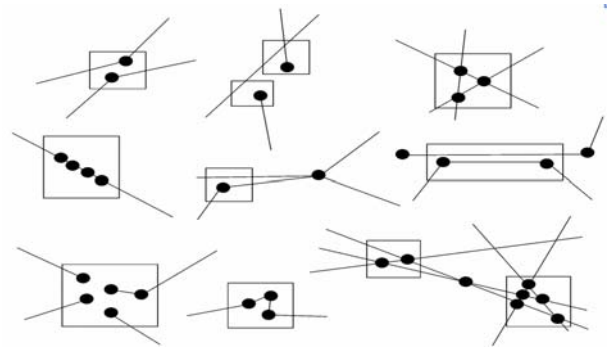

Рис. 2. Примеры нестыковок

После отождествления точек на полилинии могут появляться сегменты нулевой длины, то есть так называемые «слипшиеся» точки, такие сегменты удаляются.

Обработка точек, близких к сегментам полилиний. Пересечения сегментов полилиний

Если расстояние от вершины полилинии до сегмента той же или другой полилинии меньше заданной геометрической точности, то на сегменте образуется новая вершина в месте пересечения сегмента и перпендикуляра, опущенного из близкой вершины. Затем близкая вершина и новая отождествляются, то есть стягиваются навстречу друг другу, если ранг полилиний одинаков, или вершина, принадлежащая полилинии с низшим рангом подтягивается к вершине с более высоким рангом.

В результате отождествления близких с геометрической точностью вершин и подтягивания вершин к сегментам полностью решается задача точной стыковки геометрических объектов, причем близкие сегменты полностью налегают друг на друга. В этой ситуации на налегающих сегментах образуются новые вершины, соответствующие друг другу и из полилинии с меньшим рангом удаляется перекрытый сегмент и она разбивается на две полилинии. Следовательно, автоматически алгоритм без участия оператора исправляет все ошибки оцифровки, связанные с неточными стыковками, слипаниями точек и налегания полилиний друг на друга.

Исходные полилинии могут перемыкаться (рис. 3 и 4), то есть иметь несколько вершин, соответствующих одной и той же точки плоскости, могут пересекаться сегментами в месте, где может не быть вершин полилиний, могут слипаться. Оператор делающий оцифровку может рисовать сколь угодно сложные линии «одним росчерком» и пристыковывать линии друг к другу, не думая о необходимости образовывать пересечения. Описываемый алгоритм обрабатывает все пересечения автоматически.

При перемыкании двух полилиний или самоперемыкании одной полилинии эти полилинии разрезаются по вершине перемыкания.

Рис. 3 Взаиморасположение сегментов полилинии

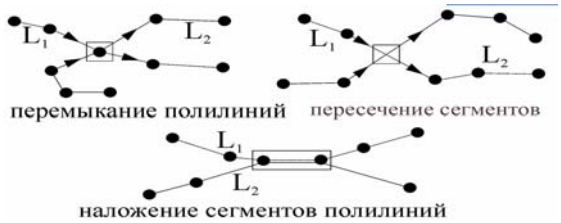

Рис. 4. Взаиморасположение сегментов двух полилиний

В случае пересечения сегментов одной полилинии или разных, на пересекающихся сегментах образуются новые вершины, и полилинии в этих новых вершинах разрезаются на две части.

В результате из исходного набора произвольно расположенных друг относительно друга полилиний получается набор новых полилиний, которые уже планарно вкладываются в плоскость, то есть нигде не накладываются, не перемыкаются и не пересекаются, а только могут примыкать в точках с координатами равными координатам начальных или конечных вершин полилиний. Полилинии, которые объявлены границами, но не имеющие продолжения, так называемые «висячие границы», удалятся как ошибочные, поскольку в принятой модели электронных карт таких границ не должно быть.

Инцидентность вершин полилиний

Рассмотрим множество вершин всего полученного множества полилиний (обрывков, полученных после обработки пересечений и перемыканий).

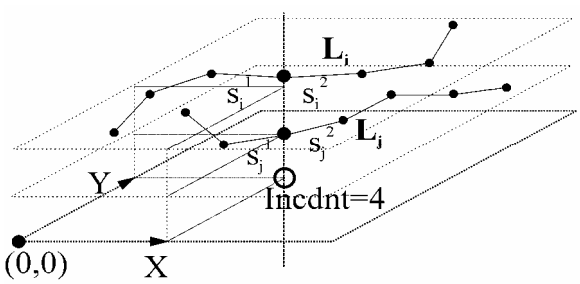

Рис. 5. Инцидентность вершин полилиний

В случае отдельной полилинии, каждая вершина является границей одного или двух сегментов (поскольку перемыкающих сегментов уже нет). Назовем это число внутренней инцидентностью вершины. Вершины разных полилиний могут соответствовать одной точке плоскости. Просуммировав инцидентность таких вершин, получим полную инцидентность этих вершин (рис. 5), то есть количество сегментов из всего множества полилиний, имеющих вершины с одинаковыми координатами. Обозначим полную инцидентность Incdnt. Теперь Incdnt может быть равна единице, двум или быть больше двух.

Вершины с Incdnt=1 – это концы полилиний, у которых нет продолжения во множестве обрывков полилиний. Такие обрывки считаются ошибочными, так называемые «висячие» полилинии. Удаление таких обрывков позволяет пристыковать линии простым перехлестом двух линий, алгоритм создает пересечение и удаляет «висячий» элемент.

Вершины с Incdnt=2 – это внутренние вершины полилинии, соединяющие два смежных сегмента, или, если это начало или конец обрывка – вершины, для которых есть продолжение, для таких обрывков отыскивается обрывок, имеющий также вершину с Incdnt=2 и совпадающую по координатам с вершиной первого обрывка. Оба обрывка сшиваются в одну полилинию с учетом следования вершин каждого из них. Эта операция продолжается до тех пор, пока в результате не получится полилиния, у которой нет начальной и конечной вершины с инцидентностью равной двум.

Если в результате полной сшивки получается полилиния, у которой начальная или конечная вершины имеют инцидентность единицу, они удаляются, как ошибки. Вершины с Incdnt>2 – это узлы правильного планарного графа, который и является конечной целью работы описываемого алгоритма. Все сшитые полилинии, если такая вершина не является начальной или конечной, разрезаются на две части по этой вершине. Таким образом, получается множество правильных полных полилиний – ребер графа, которые уложены в плоскость и примыкают друг к другу в узлах графа.

В полученном множестве полилиний присутствуют только ребра трех видов:

- простое ребро, которое начинается в одном узле графа и заканчивается в другом узле, не совпадающим с начальным;

- замкнутая полилиния, которая не соприкасается ни с одной другой полилинией множества – простой независимый цикл;

- простое ребро, которое начинается и заканчивается в одном и том же узле графа.

То есть мы получаем планарный граф, состоящий из простейших элементов, пригодный для дальнейшего анализа его топологических свойств.

Цикличность алгоритма. Описанный алгоритм имеет особенность, он производит анализ и исправления множества полилиний и построение планарного графа в несколько проходов. Это связано с тем, что некоторые ошибочные ситуации могут быть выявлены только на следующем проходе после исправления других ошибок. Кроме того, в процессе исправления ошибочных ситуаций могут появляться новые ошибки. Количество циклов алгоритма зависит от сложности входных данных. Эксперименты показали, что в среднем алгоритм полностью отлавливает все ошибки и исправляет их за 2 – 7 проходов. Планарный граф [3] считается правильным и контроль заканчивается тогда, когда на последнем проходе программы не было выявлено ни одной ошибочной ситуации и не было внесено ни одного изменения в данных (рис. 6).

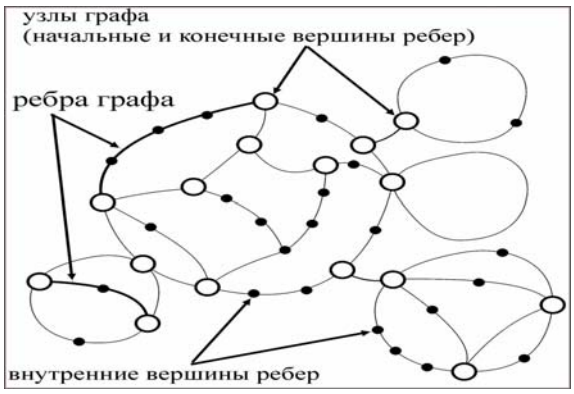

Рис. 6. Правильный планарный граф

Заключение

Представленный в статье алгоритм реализован в виде программы в среде Delphi 6.0. Программа автоматизирует большую часть ручной работы по оцифровке растровых изображений лесных карт и крайне трудоемкого процесса контроля и исправления ошибок, связанных с тем, что оцифровка ведется в масштабе гораздо более мелком, чем принятая геометрическая точность, что обусловлено ограниченностью разрешения компьютерных мониторов.

Данная программа может вызываться из меню любой ГИС, не требует никаких настроек, кроме возможности изменять геометрическую точность оцифровки и позволяет оператору переключиться на проверку не введенных линий и точек, а таксографических данных и растров, которые являются для него исходной информацией. Опыт эксплуатации данной программы в среде программы картографирования MapInfo показал, что производительность работы оператора существенно возрастает. Алгоритм тестировался на компьютере с процессором Pentium 4 (2Гц). Обработка одного лесоустроительного планшета (около 1000 выделов) выполняется за 5-30 сек.

Литература

1. Corbett, James P. Topological Principles in Cartography, Technical Paper 48, United States Department of Commerce, Bureau of the Census: Washington, D.C. – 1979. – P. 12-17.

2. Препарата, Шеймос Вычислительная геометрия.– М.: Мир, 1989. – С. 43-56.

3. Кристофидес Н. Теория графов. Алгоритмический подход. М.: Мир, 1978. – С. 81-116.

4. Скворцов А. В., Сарычев Д. С. Технология построения и анализа топологических структур для геоинформационных систем и систем автоматизированного проектирования // Вестник Томского гос. ун-та. – 2002. – № 275. – С. 60-63.

5. Gold C. Simple topology generation from scanned maps // Proceedings, Auto-Carto 13, ACM/ASPRS, Seattle. – 1997. – P. 337-346.

Поступила в редакцию 22.01.2008

Рецензент: канд. техн. наук, доцент А.В. Шостак Национальный аэрокосмический университет им. Н.Е. Жуковского «ХАИ», Харьков.