УДК 625.711.84

М.В. Пискунов

Северный (Арктический) федеральный университет

Пискунов Максим Владимирович родился в 1985 г., окончил в 2007 г. Архангельский государственный технический университет, ассистент кафедры автомобильных дорог Северного (Арктического) федерального университета. Имеет 1 печатную работу в области оптимизации дорожной сети.
E-mail: avdor@agtu.ru, mackkss@mail.ru

АЛГОРИТМ ПОСТРОЕНИЯ ОПТИМАЛЬНОЙ СЕТИ ЛЕСНЫХ ДОРОГ

Предложен алгоритм построения оптимальной сети лесных дорог с использованием электронной базы данных. С помощью запросов на языке SQL в базе формируются такие представления данных (таблицы), с помощью которых можно получить список путей, формирующих оптимальную дорожную сеть на некоторой лесной территории.

Ключевые слова: проектирование лесовозных дорог, оптимизация дорожной сети, применение географических информационных систем.

Строительство лесовозных дорог является одним из самых капиталоемких процессов в освоении лесных ресурсов. Европейский Север России располагает огромными запасами древесины, но в очень отдаленных и малозаселенных районах со слаборазвитой сетью лесных дорог и дорог общего пользования. В силу истощения лесных ресурсов на территориях, прилежащих к крупным транспортным магистралям, проблема освоения лесов в отдаленных районах становится все более актуальной. Строительство дорог на новых (как и на уже освоенных) территориях должно осуществляться максимально эффективно как с экономической, так и с экологической точек зрения.

Идея построения оптимальной сети лесных дорог с использованием электронно-вычислительных средств представлена в работе А.П. Мохирева [2]. Им предложена методика проектирования сети лесных дорог и определения экономической доступности ресурсов с учетом транспортной инфраструктуры и динамики лесного фонда. Оптимальная сеть при этом строится на основе алгоритма минимального покрывающего дерева из теории графов дискретной математики. Территория лесозаготовительного предприятия представляется в виде связного взвешенного графа, в котором вершинами являются участки лесного фонда, а ребрами – возможные транспортные пути между ними, характеризуемые соответствующими стоимостями строительства дорог.

Автором предложена реализация алгоритма минимального покрывающего дерева в терминах языка SQL (Structured Query Language). Целью настоящих разработок является автоматизация процесса оптимизации сети лесных дорог при помощи серии структурированных запросов к релятивистской базе данных.

В общем случае задачу построения минимального покрывающего дерева можно сформулировать так. Пусть дан связный неориентированный граф $G(V, E)$, в котором V – множество вершин, а E – множество их возможных попарных соединений (ребер). Пусть для каждого ребра (u,v) однозначно определен их вес $w(u,v)$ (например, стоимость строительства дороги).

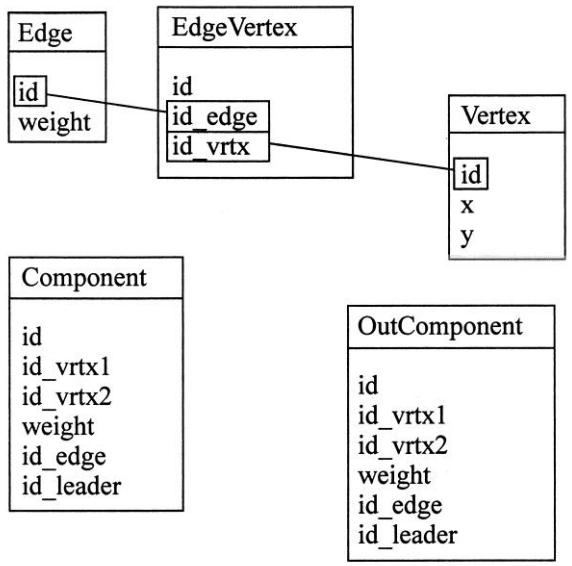

Рис. 1. Схема базы данных

Задача состоит в нахождении такого связного ациклического подграфа *T*, принадлежащего *G* и содержащего все вершины, при котором суммарный вес его ребер будет минимален.

Так как *T* связен и не содержит циклов, он является деревом и называется *остовным* или *покрывающим деревом*. Остовное дерево *T*, у которого суммарный вес его ребер минимален, называется *минимальным остовным* или *минимальным покрывающим деревом.*

Искомый остов строится постепенно. Алгоритм использует некоторый ациклический подграф *A* исходного графа *G*, который называется *промежуточным остовным лесом*. Изначально *G* состоит из *n* вершин-компонент, не соединенных друг с другом (*n* деревьев из одной вершины). На каждом шаге в *A* добавляется одно новое ребро. Граф *A* всегда является подграфом некоторого минимального остова. Очередное добавляемое ребро $e = (u,v)$ выбирается так, чтобы не нарушить этого свойства. $A \cup \{e\}$ тоже должно быть подграфом минимального. Такое ребро называется *безопасным* [3].

Используя функционал современной электронной базы данных, можно решить эту задачу наиболее эффективным способом. В такой базе должны храниться координаты всех географических объектов, необходимые для расчета минимального покрывающего дерева сети дорог на определенной территории. Схема базы данных представлена на рис. 1.

В таблице *Vertex* (вершина) будут храниться вершины графа, которые располагаются в центре тяжести лесозаготовительных участков (их геометрических центрах или в центре тяжести по объему запасов древесины), а также координаты этих вершин, либо абсолютные (географические), либо относительные (например, относительно пункта доставки древесины). Соответственно поля *x* и *y* будут хранить такие координаты.

Таблица *Edge* (ребро) соответственно будет заполнена данными о тех транспортных путях, которые возможны между вершинами графа. Эти пути (ребра) имеют определенный вес, т. е. некоторую стоимость строительства дорог между лесозаготовительными участками, значение которой хранится в поле *weight.*

Таблица *EdgeVertex* используется для связи таблиц *Edge* и *Vertex.*

Для построения минимального остова в созданном графе используется алгоритм Борувки [3], который для реализации SQL-запросов наиболее приемлем. Для примера взят простой случай, когда исходный граф состоит всего из пяти вершин и семи ребер, соединяющих эти вершины (рис. 2). Каждому ребру назначается вес, т. е. стоимость строительства дороги между двумя точками, которые соединяет данное ребро. Соответственно вес ребра 1 будет равняться 200 000, 2 – 500 000, 3 – 300 000, 4 – 400 000, 5 – 700 000, 6 – 900 000, 7 – 950 000 р.

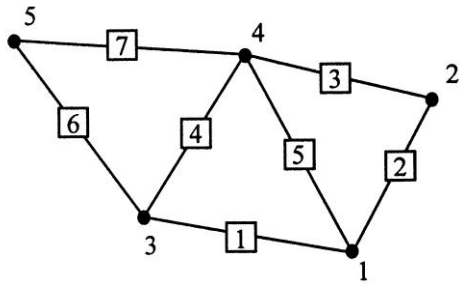

Рис. 2. Исходный граф для расчета

Вначале для каждой вершины определяются ребра с минимальным весом. Сделать это можно при помощи SQL-запроса, объединяющего две таблицы *EdgeVertex* и *Edge*. Функция min() используется для поиска минимального значения веса ребра, инцидентного каждой вершине (исходящего из каждой вершины). Запрос группируется по идентификаторам вершин. В каждом запросе создается представление (CREATE VIEW) для удобства записи последующих запросов.

```
CREATE VIEW MinWeightForVrtx AS SELECT EdgeVertex.id_vrtx AS mwv_id_vrtx,
min(Edge.weight) AS mwv_min_weight FROM EdgeVertex LEFT JOIN Edge ON
Edge.id = EdgeVertex.id_edge GROUP BY mwv_id_vrtx.
```

Таблица 1

Представление *MinWeightForVrtx*

mws_id_vrtx (вершина графа)	*mwv_min_weight* (минимальный вес ребра среди всех ребер, инцидентных данной вершине)
1	200 000
2	300 000
3	200 000
4	300 000
5	900 000

Результатом запроса (SELECT * FROM MinWeightForVrtx) будет табл. 1.

Далее необходимо знать, какое именно ребро обладает таким весом. Поэтому на основе представления *MinWeightForVrtx* создается новое – *VrtxAndMinWeightEdge*, в которое добавляется поле, содержащее идентификатор соответствующего ребра с минимальным весом.

Для каждой вершины представленной в первой графе табл. 1, необходимо получить также идентификаторы смежных им вершин относительно ребра с минимальным весом.

```
CREATE VIEW VrtxEdgeAndSecondVrtx AS SELECT
VrtxAndMinWeightEdge.vmwe_id_vrtx AS vesv_id_vrtx,
VrtxAndMinWeightEdge.vmwe_min_weight AS vesv_min_weight,
VrtxAndMinWeightEdge.vmwe_id_edge AS vesv_id_edge,
EdgeVertex.id_smt AS vesv_scnd_vrtx FROM VrtxAndMinWeightEdge
LEFT JOIN EdgeVertex WHERE EdgeVertex.id_edge =
VrtxAndMinWeightEdge.vmwe_id_edge AND EdgeVertex.id_vrtx !=
VrtxAndMinWeightEdge.vmwe_id_smt.
```

Результатом запроса к представлению *VrtxEdgeAndSecondVrtx* являются данные табл. 2.

Представление *VrtxEdgeAnd SecondVrtx* группируется по графе *vesv_id_edge*. Получается, что одному ребру исследуемого графа соответствует одна строка представления *VrtxEdgeAndSecondVrtx*. Для этого запрос создания представления необходимо немного откорректировать:

Таблица 2

Представление *VrtxEdgeAndSecondVrtx*

vesv_id_vrtx (вершина графа)	*vesv_min_weight* (минимальный вес ребра среди всех ребер, инцидентных вершине *vesv_id_vrtx*)	*vesv_id_edge* (ребро с минимальным весом, инцидентное вершине *vesv_id_vrtx*)	*vesv_scnd_vrtx* (вершина, смежная вершине *vesv_id_vrtx* относительно ребра *vesv_id_edge*)
1	200 000	1	3
2	300 000	3	4
3	200 000	1	1
4	300 000	3	2
5	900 000	6	3

```
CREATE VIEW VrtxEdgeAndSecondVrtx AS SELECT
VrtxAndMinWeightEdge.vmwe_id_vrtx AS vesv_id_vrtx,
VrtxAndMinWeightEdge.vmwe_min_weight AS vesv_min_weight,
VrtxAndMinWeightEdge.vmwe_id_edge AS vesv_id_edge,
EdgeVertex.id_vrtx AS vesv_scnd_smt FROM VrtxAndMinWeightEdge
LEFT JOIN EdgeVertex WHERE EdgeVertex.id_edge =
VrtxAndMinWeightEdge.vmwe_id_edge AND EdgeVertex.id_vrtx !=
VrtxAndMinWeightEdge.vmwe_id_vrtx GROUP BY vmwe_id_edge.
```

С помощью представления *VrtxEdgeAndSecondVrtx* (табл. 3) уже можно получить некоторые части искомого минимального остова, так называемые *компоненты связности*, т. е. множества ребер, взаимосвязанных друг с другом (рис. 3).

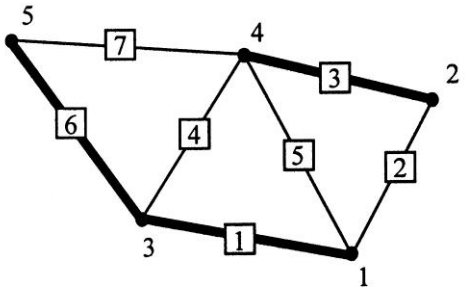

Рис. 3. Компоненты связности

Таблица 3

Представление *VrtxEdgeAndSecondVrtx*, сгруппированное по графе *vesv_id_edge*

vesv_id_vrtx	*vesv_min_weight*	*vesv_id_edge*	*vesv_scnd_vrtx*
3	200 000	1	1
4	300 000	3	2
5	900 000	6	3

Информация об этих компонентах будет храниться в таблице *Component* (табл. 4). Для этого необходимо запрограммировать определенный алгоритм, который будет заполнять эту таблицу данными. Алгоритм можно реализовать как прикладную программу, взаимодействующую с базой данных, либо используя функциональность самой системы управления базами данных (СУБД).

Поля табл. 4 означают следующее: *id_vrtx1* – вершина графа; *id_vrtx2* – вершина, смежная с вершиной *id_vrtx1* по ребру с минимальным весом; *weight* – вес ребра; *id_edge* – ребро с минимальным весом; *id_leader* – ребро-лидер для определенной компоненты связности (т. е. строки таблицы, имеющие одинаковое значение в графе *id_leader*, принадлежат одной компоненте связности).

Таблица 4

Таблица *Component*

id	*id_vrtx1*	*id_vrtx2*	*weight*	*id_edge*	*id_leader*
1	3	1	200 000	1	1
2	4	2	300 000	3	3
3	5	3	900 000	6	1

Таблица 5

Таблица *OutComponent*

id	*id_vrtx1*	*id_vrtx2*	*weight*	*id_edge*	*id_leader*
1	3	4	500 000	2	1
2	1	4	400 000	4	1
3	1	2	700 000	5	1
4	5	4	950 000	7	1

Аналогичным образом создается таблица *OutComponent* (табл. 5), в которую записываются ребра, не принадлежащие компонентам связности. В графе *id_leader* хранятся идентификаторы лидеров тех компонент связности, к которым примыкают ребра из этой таблицы.

Для получения списка безопасных ребер создается представление *OutCompMinWeight*, где производится группировка по графе *id_leader* с поиском минимального значения в графе *weight*:

```
CREATE VIEW OutCompMinWeight AS SELECT
min(OutComponent.weight) AS omw_min_weight,
OutComponent.id_leader AS omw_id_leader FROM OutComponent
GROUP BY omw_id_leader.
```

Далее создается новое представление *OutCompMinWeightVrtx*, где к *OutCompMinWeight* добавляется поле *omws_id_vrtx1*, содержащее идентификаторы вершин, принадлежащих безопасным ребрам. На основе *OutCompMinWeightVrtx* создается представление *OutCompMinWeightEdge*, в котором присутствует еще одно поле *omwe_id_edge*, которое как раз и содержит идентификаторы этих безопасных ребер.

```
CREATE VIEW OutCompMinWeightEdge AS SELECT
OutCompMinWeightVrtx.omws_id_sm1 AS omwe_id_vrtx1, Edge.id AS
omwe_id_edge, OutCompMinWeightVrtx.omwv_min_weight AS
omwe_min_weight, OutCompMinWeightVrtx.omwv_id_leader AS
omwe_id_leader FROM OutCompMinWeightVrtx LEFT JOIN Edge ON
Edge.weight = OutCompMinWeightVrtx.omwv_min_weight AND Edge.id
IN (SELECT EdgeVertex.id_edge FROM EdgeVertex WHERE
EdgeVertex.id_smt = OutCompMinWeightVrtx.omwv_id_vrtx1).
```

Таким образом, множества ребер из таблицы *Component* и представления *OutCompMinWeightEdge* и будут минимальным покрывающим деревом в исследуемом графе (рис. 4).

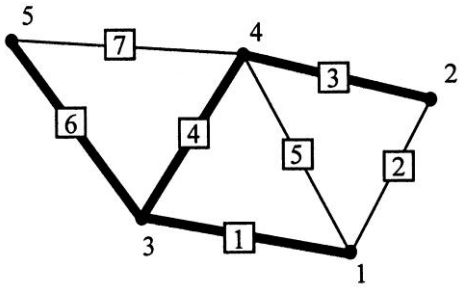

Рис. 4. Минимальное покрывающее дерево в исследуемом графе

В заключение стоит отметить, что плюсом релятивистских СУБД является их огромная распространенность во всем мире, подавляющее большинство всей информации хранится именно в таких базах. Эти базы отличаются высокой быстротой поиска, т. е. очень эффективны при работе с огромными массивами информации. Например, для точного построения оптимальной сети лесных дорог необходимо разбивать район ее построения на более мелкие участки. В результате количество вершин исходного графа может достигнуть нескольких миллионов. Таким образом, хранение данных и расчет сети дорог с применением релятивистских СУБД в этом смысле оправданны и перспективны.

СПИСОК ЛИТЕРАТУРЫ

1. *Домнин Л.С.* Элементы теории графов: учеб. пособие. Пенза: Изд-во ПГУ, 2004. 139 с.

2. *Мохирев А.П.* Обоснование проектирования сети лесных дорог на примере предприятий Нижнего Приангарья: дис. … канд. техн. наук. Красноярск, 2007. 174 с.

3. *Рыбаков Г.* Минимальные остовные деревья [Электронный ресурс]. Электрон. дан. 2005. Режим доступа: http://rain.ifmo.ru/cat/view.php/theory/graph-spanning-trees/mst-2005, свободный. Загл. с экрана.

Поступила 25.01.11

M.V. Piskunov

Northern (Arctic) Federal University

Algorithm of Building Optimal Forest Road Network

The algorithm of building optimal forest road network is offered by using the digital data base. Based on the requests in SQL the data representations (tables) are formed allowing to get a list of ways forming the optimal road network on some forest territory.

Keywords: forest roads design, road network optimization, application of geographical information systems.