УДК 519.17(075.8)

М. Н. Кирсанов

Математический центр московского метро

Когда говорят о расположении какого-либо объекта в городе, принято вести отсчет от «центра». «Университет расположен в центре города» или «дом в 5 минутах от центра на метро». Так или иначе, понятие «центра» интуитивно ясно. Обычно это место расположения каких-либо важных объектов: исторических, культурных или торговых. Такое определение весьма субъективно, для одних важно одно, для других — иное. Ясность в вопросе о центре может внести математика. В последнее время бурно развивается ее большой раздел — дискретная математика, начало которой положил еще Л. Эйлер, решивший знаменитую задачу о Кенигсбергских мостах. Сейчас областью применения дискретной математики являются в основном компьютерные технологии. Теорию графов используют шире, в частности, для анализа различных сетей — транспортных, информационных, торговых. Поиск математического центра московского метро, выполненный в статье средствами Maple, является примером решения простой задачи анализа транспортной сети.

Задача

Нахождение центра транспортной сети.

Программные средства

Maple

Результаты

Найдены центральные станции Московского метрополитена. Приведены алгоритм решения задачи и Maple-программа.

Используя понятия и методы теории графов, найдем центр московского метро. Схему московского метро представим в виде графа, вершинами которого являются станции. Ветки метро образуют ребра, соединяющие вершины. Переходы между станциями также будем считать ребрами графа, никак не отличая их от путей.

Процедура определения центра графа проста. Сначала надо найти минимальные расстояния между всеми вершинами. Расстояние между вершинами примем для простоты равным числу ребер, соединяющих эти вершины. Найденные расстояния занесем в матрицу. Очевидно, матрица будет симметричной. Каждой станции (вершине графа) соответствует строка. Минимальный элемент строки всегда нуль — расстояние от вершины до самой себя (диагональный элемент матрицы расстояний). Максимальный элемент строки — эксцентриситет вершины. Если эксцентриситет вершины равен N, то это означает, что расстояние (в ребрах) до самой удаленной от нее вершины равно N. Эксцентриситеты вершин образуют вектор, максимальный элемент которого равен *диаметру* графа D, а минимальный — *радиусу* R. Вершины графа, эксцентриситет которых равен радиусу, образуют *центр* графа. Вершины с эксцентриситетом, равным диаметру, находятся на *периферии* графа.

Центральные станции московского метро: *Павелецкая (кольцевая), Добрынинская, Октябрьская (кольцевая), Парк Культуры (кольцевая), Серпуховская*.

Особенность расчетов в случае московского метро состоит в большой трудоемкости. Обозначим через n число вершин графа. Для $n = 164$ станций надо выполнить $(n^2 - n)/2 = 13366$ измерений расстояний между вершинами, разыскивая при этом каждый раз минимальные пути. Вручную это сделать затруднительно. Используем для исследования систему Maple, содержащую специальный пакет **networks** для работы с графами.

Maple-программа приведена в Приложении 1. Названия станций, занесенные в файл **stations.txt**, присваиваются одномерному массиву Станция. Для того чтобы файл **stations.txt** был прочитан, в третьей строчке программы надо указать полный путь к нему. В системе Maple допускаются русские буквы в именах, что упрощает чтение программ. Однако сами названия в файле **stations.txt** приходится набирать прописными буквами, так как строчная буква «я» системой неправильно воспринимается.

Оператор **new(G)** создает новый (пока еще пустой) граф G. Последовательно пронумерованные вершины добавляются в граф оператором **addvertex**. Затем соединяются вершины отдельных веток метро (оператор **connect**) и добавляются переходы между станциями. Оператор **allpairs** создает таблицу расстояний. Этот оператор является ключевым в программе. Зная расстояния между всеми парами станций, уже легко вычислить эксцентриситет каждой, а затем диаметр и радиус графа. Центральных станций московского метро в представленной модели оказывается всего пять: *Павелецкая (кольцевая), Добрынинская, Октябрьская (кольцевая), Парк Культуры (кольцевая)* и *Серпуховская*. Их эксцентриситет равен радиусу графа R, равному 14. Диаметр графа D равен 26. Две станции имеют такой эксцентриситет — *Алтуфьево* и *Битцевский парк*.

Результаты вычислений отображены на цветной схеме метро (рис. 1). В двойных числах первое соответствует кольцевой станции. Подчеркнем, что это именно схема, а не граф, так как здесь не отражены ребра графа — переходы между станциями, которые входят в вычисление центра метро. В полном графе (каждая вершина соединена со всеми остальными) радиус и диаметр равны 1 (недостижимая мечта пассажира!). Поэтому, чем меньше радиус и диаметр графа метро, тем оно удобнее. Используя приведенную программу, можно проводить теоретические эксперименты по оптимизации схемы метро. Так, если условно включить пригородную электричку в сеть метро, например, от *Выхино* (самая загруженная станция, см. http://www.metro.ru) до *Электрозаводской* и далее до *Комсомольской* (Казанский вокзал), то ни радиус, ни диаметр графа не уменьшатся, уменьшатся только эксцентриситеты трех конечных станций Таганской ветки. После введения в строй новой станции *Бульвар Дмитрия Донского* (165-я станция) Серпуховско-Тимирязевской линии радиус и диаметр не изменились. Центр, из которого вышли стации *Театральная* и *Третьяковская,* немного сместился на юг, а эксцентриситеты северных станций Калужско-Рижской и Филевской линий увеличились на 1. Для пересчета достаточно было в программе увеличить n до 165 и добавить одно ребро **connect(151,165,G)**.

В пакете **networks** есть специальный оператор вычисления диаметра графа **diameter(G)**. Однако, если кроме диаметра графа надо найти его радиус и эксцентриситеты отдельных вершин (как в нашем случае), лучше не использовать этот оператор, так как он сам задействует оператор **allpairs**, выполнение которого занимает значи-

▲ **Рис. 1.** Схема московского метро с указанием эксцентриситетов станций.

тельное время (около минуты на компьютере с частотой процессора 1200 МГц), и получается двойная трата времени — и на вычисление радиуса, и на вычисление диаметра.

Кроме того, в пакете **networks** имеется оператор изображения графа **draw(G)**. Этот оператор обладает одной особенностью — все вершины он расставляет по окружности, и для большого числа вершин граф становится неузнаваемым. Поэтому мы остережемся приводить схему московского метро в трактовке Maple, чтобы не пугать читателей.

Можно вычислить радиус графа метро и его центр, исходя из времени движения. Существуют схемы метро, на которых указано среднее время движения между отдельными станциями. Такой граф будет взвешенным. Вес ребра — время движения по нему. В системе Maple веса добавляются к графу вместе с ребрами. Вместо оператора **connect** надо использовать оператор **addedge**, например, в форме

```
> addedge({113,88},weights=2,G)
```

задающей время перехода между станциями *Китай-город* разных линий, равным 2 минутам. Определение диаметра и радиуса проводится по прежней схеме, с использованием оператора **allpairs**.

Алгоритм вычисления расстояний скрыт в операторе **allpairs** системы Maple. Для того что-

Автор:

Кирсанов Михаил Николаевич, профессор, доктор физико-математических наук, профессор кафедры теоретической механики; Московский энергетический институт, г. Москва

бы найти центр графа в другой математической системе или просто запрограммировать задачу на каком-либо алгоритмическом языке, необходимо как-то заменить этот оператор. Предлагаем следующий простейший алгоритм, основанный на известной теореме теории графов [1], согласно которой элемент a_{ij} матрицы $\mathbf{A} = \mathbf{M}^k$, где $\mathbf{M}$ — матрица смежности графа, есть число маршрутов от вершины i до вершины j длины k.

Матрица смежности состоит из нулей и единиц. При этом $a_{ij} = 1$, если вершина i соединена с вершиной j, в противном случае $a_{ij} = 0$. В Приложении 2 приведена Maple-программа алгоритма. Оператор **adjacency** задает матрицу смежности графа G. Матрица M0 в начале единичная, затем в цикле по k в ней накапливаются степени M. Нулевые элементы симметричной матрицы AP заменяются на число k, как только k-я степень матрицы смежности обнаруживает какое-либо число маршрутов между соответствующими вершинами. Вычисление продолжается, пока не найдутся все $n(n-1)/2$ расстояния. Недостатком такого алгоритма является его трудоемкость. Применительно к схеме московского метро он работает около 6 минут, что в 6 раз дольше алгоритма **allpairs**. Время счета определяет оператор **time()** (см. Приложение 1).

Пакет **networks** содержит оператор **isplanar(G)**, проверяющий планарность [1] графа. Время его работы —доли секунды. Граф петербургского метро оказывается планарным, а граф московского, как и следовало ожидать, не планарный.

Другие алгоритмы на графах на языке Pascal можно найти в [2, 3]. В качестве элементарного учебника по дискретной математике можно рекомендовать также [4].

В работе приняли участие студенты МГТУ «МАМИ» Г. Андреев, Н. Юткина, С. Артамонникова.

Литература

1. Судоплатов С. В., Овчинникова Е. В. Элементы дискретной математики.— М.:ИНФРА-М, Новосибирск: Изд-во НГТУ, 2002.— 280 с.
2. Иванов Б. Н. Дискретная математика. Алгоритмы и программы: Лаборатория Базовых Знаний, 2002.— 288 с.
3. Асанов М. О., Баранский В. А., Расин В. В. Дискретная математика: графы, матроиды, алгоритмы.— Ижевск: НИЦ «Регулярная и хаотическая динамика», 2001.— 288 с.
4. Показеев В. В., Матяш В. И., Черкесова Г. В. Курс лекций. Элементы дискретной математики.— М.: МГТУ «МАМИ», 2003.— 240 с.

Приложение 1. Maple-программа определения центра московского метро

```
> restart:
Открываем файл с названиями станций
    Файл:=fopen(«stations.txt»,READ,TEXT):
Считываем названия в массив «Станция»
    Станция:=readdata(Файл,string,1):
Закрываем файл
    fclose(Файл):
Загружаем библиотеку дискретной математики (графы)
    with(networks):
Создаем новый граф G
    new(G):
Задаем 164 вершины - все станции метро
    n:=164:
    addvertex({seq(i,i=1..n)},G):
Соединяем соседние станции
    Кольцевая ветка
    for i from 1 to 11 do connect(i,i+1,G) end:
    connect(1,12,G):
    Сокольническая ветка
    for i from 13 to 30 do connect(i,i+1,G) end:
    Замоскворецкая ветка
    for i from 32 to 50 do connect(i,i+1,G) end:
    Арбатско-Покровская ветка
    for i from 52 to 63 do connect(i,i+1,G) end:
    Филевская ветка
    for i from 65 to 76 do connect(i,i+1,G) end:
    Калужско-Рижская ветка
    for i from 78 to 100 do connect(i,i+1,G) end:
    Таганско-Краснопресненская ветка
    for i from 102 to 119 do connect(i,i+1,G) end:
    Калининская ветка
    for i from 121 to 126 do connect(i,i+1,G) end:
    Серпуховско-Тимирязевская ветка
    for i from 128 to 150 do connect(i,i+1,G) end:
    Люблинская ветка
    for i from 152 to 160 do connect(i,i+1,G) end:
    Каховская ветка
    for i from 162 to 163 do connect(i,i+1,G) end:

    Станции пересадок
connect(74,[63,1],G): connect(2,110,G): connect(3,38,G):
connect(4,136,G): connect(5,85,G): connect(9,43,G):
connect(59,[152,7],G): connect(7,152,G): connect(114,[126,8],G):
connect(10,141,G): connect(11,90,G): connect(12,25,G):
connect(20,87,G): connect(21,112,G): connect(6,18,G):
connect(61,[77,139],G): connect(138,[111,40],G):
connect(42,[89,127],G): connect(89,127,G): connect(46,162,G):
connect(88,113,G): connect(115,154,G): connect(125,153,G):
connect(8,126,G): connect(41,60,G): connect(22,[60,41],G):
connect(146,164,G): connect(23,[77,139,61],G):
t0:=time():
m:=Vector(n):
Расстояния:=allpairs(G):
for j to n do m[j]:=max(seq(Расстояния[i,j],i=1..n)): od:
Диаметр:=max(seq(m[i],i=1..n));
Радиус:=min(seq(m[i],i=1..n));
time()-t0;
for j to n do if m[j]=Радиус then print(Станция[j]) end; od;
for j to n do if m[j]=Диаметр then print(Станция[j]) end; od;
for j to n do Станция[j],m[j] od:
```

Приложение 2. Программа вычисления матрицы расстояний

```
M:=adjacency(G):
with(LinearAlgebra):
M0:=IdentityMatrix(n):
AP:=Matrix(n,shape=symmetric):
N:=0:
for k while N<>n*(n-1)/2 do
M0:=M0.M:
for i to n do
for j to i-1 do
if AP[i,j]=0 and M0[i,j]<>0 then
AP[i,j]:= k; N:=N+1:
end:
od:
od:
od:
Расстояния:=AP;
```