ISSN 1407-8015

RĪGAS TEHNISKĀS UNIVERSITĀTES
ZINĀTNISKIE RAKSTI

SCIENTIFIC PROCEEDINGS
OF RIGA TECHNICAL UNIVERSITY

6. SĒRIJA

MAŠĪNZINĀTNE UN TRANSPORTS
TRANSPORT AND ENGINEERING
Dzelzceļa transports
Railway Transport

30. SĒJUMS

RTU IZDEVNIECĪBA

RĪGA 2008

ISSN 1407-8015 TRANSPORT AND ENGINEERING
MAŠĪNZINĀTNE UN TRANSPORTS 2009-8015

RAILWAY TRANSPORT
DZELZCEĻA TRANSPORTS

INTELLECTUAL LOGISTIC SYSTEM FOR PASSENGER RAILWAY TRANSPORTATIONS IN TOWN

INTELEKTUĀLA LOĢISTISKĀ SISTĒMA DZELZCEĻA PASAŽIERU PĀRVADĀJUMIEM PILSĒTAS ROBEŽĀS

ИНТЕЛЛЕКТУАЛЬНАЯ ЛОГИСТИЧЕСКАЯ СИСТЕМА ДЛЯ ПАССАЖИРСКИХ ЖЕЛЕЗНОДОРОЖНЫХ ПЕРЕВОЗОК В ГОРОДЕ

Pavel Gavrilov, Master of Science in transport and traffic engineering, Ph.D student
Riga Technical University
Faculty of Transport and engineering
Institut of Railway transport
Address: 8a Indrika Street, Riga, Latvia
E-mail: pavels.gavrilovs@rtu.lv

Leonids Ribickis, Professor, Dr.Hab.sc.ing.
Riga Technical University
Faculty of Energetic and electrical engineering
Address: 1 Kalku Street, Riga, Latvia,
E-mail: leonids.ribickis@rtu.lv

Peteris Balckars, Professor, Dr.sc.ing.
Riga Technical University
Faculty of Transport and engineering
Institut of Railway transport
Address: 8a Indriķa Street, Riga, Latvia
E-mail: peteris@dzti.edu.lv

Anatolijs Levcenkovs, Professor, Dr.sc.ing.
Riga Technical University
Faculty of Energetic and electrical engineering
Address: 1 Kalku Street, Riga, Latvia
E-mail: anatolijs.levcenkovs@rtu.lv

Ключевые слова: железнодорожный транспорт, пассажирские перевозки, орграф сети, алгоритм решения, минимальное время

Введение

Для пассажира наиболее удобно, когда транспорт обеспечивает доставку “от двери до двери”. На небольших расстояниях эта проблема успешно решается при помощи использования автотранспорта. Но рост количества автомобилей привел к перенасыщению транспортом автомагистралей и особенно улиц городов. Образуются пробки, в результате снижается скорость движения, увеличивается загрязнённость окружающей среды.

Развитие сетевой инфраструктуры рельсового транспорта предлагается по следующим направлениям:

- интеграция рельсового транспорта в единую систему с другими видами транспорта;
- развитие системы городских сообщений по типу «трамвай – автобус - троллейбус - поезд», которая предусматривает использование всех видов транспорта в городской среде. Использование концепции «трамвай – автобус - троллейбус - поезд» дает возможность решить многие проблемы внутригородских перевозок столицы. Переход с линии одной системы на участок другой совершается в строго определенных стыковочных пунктах по соединительным путям;
- улучшение условий пересадки между видами транспорта с согласованием расписаний, созданием единых станций — пересадочных узлов, позволит снизить нагрузку на узлы и сократить перемещение пассажиров, как по времени, так и находится, в «пробках», в часы пик.

В плане г. Риги должна быть предусмотрена реновация зон отчуждения железнодорожных линий, учитывая идущий процесс включения сети железной дороги и станций в структуру городских транспортных систем, прежде всего в составе многофункциональных транспортных комплексов в районе железнодорожных станций: вокзал, ст. Земитане, Браса, Саркандаугава, Мангали, Земельблазма, Вецдаугава, Чиекуркалнс, Югла, Торнякалнс, Засулаукс, Золитуде, Иманта, вагонный парк, Яняварти, Шкиротава, Даугмале, Атгазене.

Для решения данной задачи рассматривается алгоритм, который определяет минимальное время в пути между вершинами в простом орграфе с неотрицательными весами. К таким орграфам сводятся многие типы графов. Если граф не является простым, его можно сделать таковым, отбрасывая все петли и заменяя каждое множество параллельных ребер кратчайшим ребром (ребром с наименьшим весом) из этого множества; каждое неориентированное ребро заменяется парой ориентированных ребер. Если граф не взвешен, то можно считать, что все ребра имеют один вес.

Пусть $\Gamma = (X, U, \Phi)$ — простой орграф, для каждого ребра $u \in U$ определен вес $w(u) > 0$. Найдем кратчайший путь по времени между выделенными вершинами x_0 и z (рис. 1). Несуществующие ребра будем считать ребрами с бесконечными весами. Сумму весов ребер в пути будем называть весом иди длиной пути. Обозначим W_{ij} — вес ребра $u = (x_i\, x_j)$.

Алгоритм поиска кратчайшего пути по времени, начиная из вершины x_0, просматривает граф в ширину, помечая вершины x_j значениями — метками их времени от x_0. Временная метка вершины x_j — это минимальное время x_0 до x_j, когда в определении пути на графе учитываются не все маршруты из x_0 до x_j. Окончательная метка x_j — это минимальное время на графе от x_0 и x_j. Таким образом, в каждый момент времени работы алгоритма некоторые вершины будут иметь окончательные метки, а остальная их часть — временные. Алгоритм заканчивается, когда вершина z получает окончательную метку, т.е. время от x_0 и z.

Вначале вершине x_0 присваивается окончательная метка 0 (нулевое время до самой себя), а каждой из остальных |X| — 1 присваивается временная метка ∞ (бесконечность). На каждом шаге одной вершине с временной меткой присваивается окончательная и поиск продолжается дальше. На каждом шаге метки меняются следующим образом.

1. Каждой вершине x_j, не имеющей окончательной метки, присваивается новая временная метка — наименьшая из ее временной и числа (w_{ij} + окончательная метках x_i), где x_i — вершина, которой присвоена окончательная метка на предыдущем шаге.
2. Определяется наименьшая из всех временных меток, которая и становится окончательной меткой своей вершины. В случае равенства меток выбирается любая из них.

Циклический процесс п.1 + п.2 продолжается до тех пор, пока вершина z не получит окончательной метки. Легко видеть, что окончательная метка каждой вершины — это минимальное время от этой вершины до начала x_0.

Рассмотрим пример поиска минимального времени на графе, представленном на рис. 1.

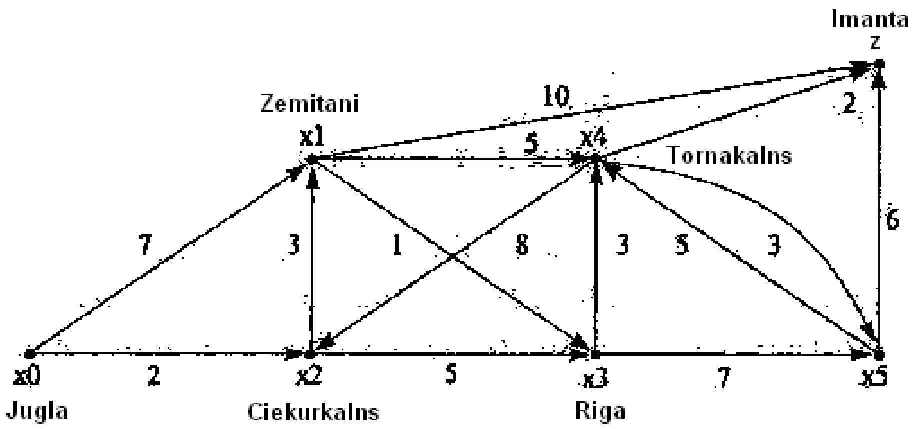

Рис. 1. Простой взвешенный орграф

Процесс назначения меток вершинам графа на каждом шаге удобно представить в виде следующей таблицы.

	$\boxed{x0}$	x1	x2	x3	x4	x5	z
0	0	∞	∞	∞	∞	∞	∞
1		7	$\boxed{2}$	∞	∞	∞	∞
2		$\boxed{5}$		7	∞	∞	∞
3				$\boxed{6}$	10	∞	15
4					$\boxed{9}$	13	15
5						12	$\boxed{11}$

Квадратами выделены окончательные метки, т.е. время от них до x_0. По такой таблице легко восстановить путь перемещения от z к x_0, который отмечен ломаной кривой.

Реализация рассмотренной схемы поиска минимального времени представлена в алгоритме 1, где граф *Г = (X, U, Ф)* представляется матрицей весов *We* = [w_{ij}], веса несуществующих ребер полагаются равными $+\infty$. Вектор *Mark*[x] меток вершин устанавливает принадлежность вершины $x \in X$ постоянной *(TRUE)* или временной *(FALSE)* метке. Вектор *Dist*[x] в алгоритме фиксирует текущие значения меток вершин. Вектор *Prex*[x] позволяет восстановить в обратной последовательности вершины минимального времени.

Алгоритм 1. *Алгоритм минимального времени в пути на орграфе*

Шаг 1

```
for x ∈ X do begin
        Mark[x] = FALSE; Dist[x] = ∞;
end;
```

Шаг 2

```
    y = x0; Mark[x0] = TRUE; Dist[x0] = 0;
```

Шаг 3

```
    while not Mark[z] do begin
        for x ∈ X do
                if not Mark[x] and dist[x] > dist[y] + w[y, z] then begin
                        dist[x] = dist[y] + w[y, x];
                        prev[x] = y;
                end;
```

Шаг 4

{Поиск новой вершины y ∈ X с минимальной временной меткой}

```
        dist[y] = min dist[x];
                    x ∈ X and Mark[x] = FALSE
        Mark[y] = TRUE;
end.
```

Prev[x] указывает на вершину с окончательной меткой, ближайшую к вершине *x*. Последовательность вершин минимального времени будет имеет следующий вид:

Z, prev[z], prev[prev[z]], prev[prev[prev[z]]], ... , x_0,

а значение *Dist[z]* составит время в пути из x_0 в z. Очередная новая вершина, претендующая на постоянную метку, обозначается через y.

Сложность алгоритма. Алгоритм обращается к телу цикла *while* не более [X] — 1 раз, и число операций, требующихся при каждом таком обращении, равно *0(\X\)*. Тогда сложность алгоритма составит $0(|X|^2)$.

Интересно заметить, что если требуется найти минимальное время в пути от x_0 до всех вершин графа, то в алгоритме 1 условие цикла ***while not*** *Mark*[z] ***do begin*** надо заменить на условие ***while not*** *Mark[x]* ***do begin***. При этом сложность алгоритма остается прежней.

Программная реализация алгоритма поиска минимального времени в пути представлена в алгоритме 2 на Pascal'e, который близко соответствует множественному описанию алгоритма 1.

Алгоритм 2. *Программа минимального времени в пути на орграфе*

Шаг 1

```
 Program Short; {Кратчайшие пути на графе)
 uses CRT,DOS; Const nVertex=50; {Максимальное количество вершин}
 Type
   TypeMark=array[O..nVertex] of Boolean; TypeDist=array[O..nVertex] of LongInt;
   TypePrev=array[O..nVertex] of Integer; TypeWeight=array[O..nVertex, 0..nVertex] of Integer;
```

Шаг 2

```
Var
   f   :Text;      { Текстовый файл } nX   :Integer;    { Количество вершин в графе } Mark
   :TypeMark; {Признаки временных и постоянных меток} Dist :TypeDist;    { Значения текущих
```

```
    меток вершин (расстояния)} Prev :TypePrev;     { Указатель на ближайшую вершину }   We
    :TypeWeight;   (Матрица весов ребер графа }  xO   : Integer;     { Вершина начала пути }  z
    :Integer;      { Вершина конца пути }  y    :Integer;  {Последняя вершина с постоянной меткой}
```
Шаг 3

```
Var
    i, j, x  :Integer; weight    :LongInt;
```
Шаг 4

```
begin
    Assign (f, 'Short.in'); Reset (f) ; {Файл открыт для чтения}
    {Ввод исходных данных)
    Read (f, x0); {Начальная вершина пути} Read (f, z); {Конечная вершина пути} Read(f, nX);
{Количество вершин в графе} nX: = nX-1; (* X={0,1,2,...,nX} - множество вершин *)
    for i:= 0 to nX do begin
           for j:=0 to nX do begin
           Read (f, We[i,j]); { Ввод матрицы весов }
           if We [i,j] = 0 then We [i,j]: = $7fff; {+бесконечность}
           end;
   end;
   Close (f); Assign (f,'Short.out');  Rewrite(f); {Файл открыт для записи}
```
Шаг 5

```
   for x:=0  to  nX do begin
      Mark[x]:=FALSE; Dist[x]:=$7fffffff;
  end;
    y: = xO; {Последняя вершина с постоянной меткой} Mark[y] := TRUE; Dist[y]:= 0;
```
Шаг 6

```
     while  not Mark[z]  do  begin
        {Обновить временные метки}
        for x := 0 to nX do if not Mark[x] and ( Dist[x] > Dist[y] + We[y,x] ) then begin
           Dist[x] :== Dist[y] + We[y,x]; Prev[x] := y;
        end;
        {Поиск вершины с минимальной временной меткой)
        weight := $7fffffff;
        for x:=0 to nX do if not Mark[x] then if weight > Dist[x] then begin weight := Dist[x];
           y := x; Mark[y]:=TRUE;
     end;
```
Шаг 7

```
     Write (f,'Вершины пути=');
     x:=z; while x<>x0 do begin Write (f,x:2); x:=Prev[x];
     end;
     WriteLn (f, x : 2); WriteLn (f,'Длина пути= ‘, Dist[z]); Close (f);
end.
```

Рассмотрим пример расчета по программе алгоритма 2 поиска минимального времени в пути на графе, показанном на рис. 1.

Исходные данные графа представляются матрицей весов его ребер в текстовом файле Short.in со следующей структурой:

- в первой строке определяется номер начальной вершины пути x_0;
- во второй строке определяется номер конечной вершины пути z;
- в третьей строке указывается количество *nX* вершин в графе;
- в следующих *nX* строках определяются строки матрицы весов $[w_{ij}]$ графа.

0	7	2	0	0	0	0
0	0	0	1	5	0	10
0	3	0	5	0	0	0
0	0	0	0	3	7	0
0	0	8	0	0	5	2
0	0	0	0	1	0	6
0	0	0	0	0	0	0

Результаты расчетов сохраняются в выходном файле Short.out со следующей структурой:
Вершины пути = 6 4 3 1 2 0
Время в пути = 34 (мин).

Выводы

Внедрение железнодорожного транспорта в транспортную сеть города позволит снизить нагрузку на транспортные узлы и сократить время перемещения пассажиров и нахождения их в «пробках» в часы пик.

Литература

1. Praude V., Beļčikovs J. Loģistika. Vaidelote, 2003.
2. Urbahs A., Cerkovņuks A. Intermodālie konteineru pārvadājumi. – R.: RTU Izdevniecība, 2003.- 496 lpp.
3. Krampe H., Lucke H.J.; Grundlagen der Logistik; Huss-Verlag, München, 1993.
4. Arnold D.; Meterialflusslehre; Braunschweig, Wiesbaden: Vieweg, 1995.
5. Koether R.; Technische Logistik; München; Wien: Hanser, 1993.
6. Логистика; под ред. Б.А. Аникина; Москва, ИНФРА-М, 1997.
7. Промышленная логистика; под ред. А. Колобова; Москва, 1997.

Gavrilovs P., Ribickis L., Balckars P., Levčenkovs A. Intelektuāla loģistiskā sistēma dzelzceļa pasažieru pārvadājumiem pilsētas robežās

Automobiļu daudzuma pieaugums ir radījis automaģistrāļu un, it īpaši pilsētu, ielu pārsātinājumu.Uz ielām veidojas automobiļu sastrēgumi, kas samazina kustības ātrumu un palielina apkārtējās vides piesārņojumu. Šo problēmu varētu risināt, iekļaujot dzelzceļa transportu pilsētas transporta sistēmā. Šīs problēmas risinājumam ir piedāvāts algoritms, kurš dod iespēju noteikt minimālo laiku ceļā starp dažādām dzelzceļa stacijām, kas iekļautas pilsētas transporta tīklā.

Gavrilovs P., Ribickis L., Balckars P., Levcenkovs A. Intellectual logistic system for passenger railway transportations in town

Growth of amount of cars resulted in supersaturating of motorways a transport and especially streets of cities. Constantly congestions appear on the streets of cities, the rate of movement goes down as a result, muddiness of environment is increased. This problem can it would be decided, plugging a railway transport in a city transport system. For the decision of this task an algorithm is examined, using which it is possible to define minimum time on the way between railheads, plugged in a city transport network.

Гаврилов П., Рибицкис Л., Балцкарс П., Левченков А. Интеллектуальная логистическая система для пассажирских железнодорожных перевозок в границах города

Рост количества автомобилей привел к перенасыщению транспортом автомагистралей и особенно улиц городов. Постоянно образуются заторы на улицах городов, в результате снижается скорость движения, увеличивается загрязнённость окружающей среды. Эту проблему можно было бы решить включив железнодорожный транспорт в городскую транспортную систему. Для решения этой задачи рассматривается алгоритм, используя который можно определить минимальное время в пути между железнодорожными станциями, включенными в городскую транспортную сеть.