### [Academia de Studii Economice din Bucuresţi](http://www.ase.ro/)

Facultatea de Cibernetică, Statistică și Informatică Economică

PROIECT PROGRAMARE EVOLUTIVA SI ALGORITMI GENETICI

Tema lucrării: Determinarea unui drum optim intr-un graf ponderat conex

Student, Profesor coordonator,

Drăghici Alice Florentina Mogos Radu

Grupa 1044

Anul universitar: 2017-2018 

Semestrul II

**CUPRINS**

1. **Enunt**
2. **Introducere**
3. **Structura generala a algoritmului**
4. **Functii implementate**
5. **Apelare**
6. **Enunt**

**15.** Problema determinării unui drum optim într-un graf ponderat conex – exprimată în termeni de distanțe între puncte de interes (orașe, puncte de aprovizionare etc.).

1. **Introducere**

Se numeste graf ponderat un graf in cadrul caruia fiecarui arc ii este asociata o valoare. Valoarea asociata arcului are semnificatia de "cost" a legaturii intre cele 2 noduri sau de "distanta" intre noduri.

Fie G=(N,A) un graf conex in care fiecare arc (u,v) apartinand lui A are costul atasat c(u,v). Un arbore de acoperire al lui G este arborele liber care cuprinde toate nodurile din N si este un subgraf al lui G. Costul unui arbore de acoperire este suma costurilor tuturor arcelor cuprinse in arbore.

Costul minim al drumului dintre doua noduri este minimul dintre costurile drumurilor existente intre cele doua noduri. Pentru a calcula drumul optim, vom folosi algoritmul lui Dijkstra care calculeaza drumurile minime de la o sursa catre celelalte noduri.

1. **Structura generala a algoritmului**

Algoritmul lui Dijkstra este un algoritm care calculeaza drumurile minime de la un nod al unui graf la toate celelalte noduri din graf.

Grafurile pe care poate lucra algoritmul lui Dijkstra sunt, in general, ponderate si orientate – arcele sunt orientate de la un nod la alt nod si au un anumit cost de care se va tine seama in aflarea drumului minim.

1. Fie s nodul de start – acesta se marcheaza ca vizitat.
2. Ideea gasirii drumului minim de la s la un un alt nod este cautarea treptata: se presupune ca drumul minim este alcatuit dintr-un singur arc (arcul direct intre s si nodul tinta, d, care poate sa nu existe, costul sau fiind infinit in acest caz), apoi se cauta drumuri mai scurte alcatuite din 2 arce, apoi din 3, etc. – un drum minim nu poate avea mai mult de N-1 arce, deci algoritmul are N-1 pasi (al N-lea este banal)
3. Dupa pasul k (1 ≤ k ≤ N-1), tabloul L va contine lungimile drumurilor minime de la nodul s la celelalte noduri, toate aceste drumuri fiind alcatuite din maxim k arce.
4. Deci, dupa pasul k, au fost gasite numai drumurile minime alcatuite din maxim k arce – abia la finalul algoritmului (dupa pasul N-1) drumurile minime obtinute sunt definitive, ele fiind drumuri minime alcatuite din maxim N-1 arce
5. La inceputul fiecarui pas k avem un set de k-1 noduri marcate (in cadrul pasilor precedenti) – nodurile marcate sunt cele pentru care se cunoaste drumul minim (initial, doar nodul de start este marcat deoarece doar pentru el se cunoaste drumul minim)
6. In cadrul pasului k trebuie executate 3 operatiuni:

* se gaseste acel nod d nemarcat care are L minim (acesta este singurul dintre nodurile nemarcate pentru care se poate spune sigur ca drumul minim are lungimea L[d]) – pentru celelalte noduri nemarcate valoarea corespunzatoare din tabloul L s-ar putea sa nu reprezinte lungimea drumului minim ci un drum minim intermediar (alcatuit din maxim k-1 arce) care poate fi imbunatatit in cadrul pasilor viitori ai algoritmului
* nodul d se marcheaza ca vizitat

1. **Functii implementate**

**Functia dijkstra**

function [ e L ] = dijkstra(A,s,d)

% A = graful ; s = sursa ; d = destinatia

if s==d

e=0;

L=[s];

else

A = grafic(A,inf,1);

if d==1

d=s;

end

A=interschimbareNod(A,1,s);

lengthA=size(A,1);

W=zeros(lengthA);

for i=2 : lengthA

W(1,i)=i;

W(2,i)=A(1,i);

end

for i=1 : lengthA

D(i,1)=A(1,i);

D(i,2)=i;

end

D2=D(2:length(D),:);

L=2;

while L<=(size(W,1)-1)

L=L+1;

D2=sortrows(D2,1);

k=D2(1,2);

W(L,1)=k;

D2(1,:)=[];

for i=1 : size(D2,1)

if D(D2(i,2),1)>(D(k,1)+A(k,D2(i,2)))

D(D2(i,2),1) = D(k,1)+A(k,D2(i,2));

D2(i,1) = D(D2(i,2),1);

end

end

for i=2 : length(A)

W(L,i)=D(i,1);

end

end

if d==s

L=[1];

else

L=[d];

end

e=W(size(W,1),d);

L = listaDijkstra(L,W,s,d);

end

end

**Functia interschimbareNod**

function G = interschimbareNod(G,a,b)

% interschimbare elementului de pe coloana a cu cel de pe coloana b

aux=G(:,a);

G(:,a)=G(:,b);

G(:,b)=aux;

% interschimbare elementului de pe linia a cu cel de pe linia b

aux=G(a,:);

G(a,:)=G(b,:);

G(b,:)=aux;

end

**Functia grafic**

function G = grafic(G,b,s)

if s==1

for i=1 : size(G,1)

for j=1 :size(G,1)

if G(i,j)==0

G(i,j)=b;

end

end

end

end

if s==2

for i=1 : size(G,1)

for j=1 : size(G,1)

if G(i,j)==b

G(i,j)=0;

end

end

end

end

**Functia listaDijkstra**

function L = listaDijkstra(L,W,s,d)

index=size(W,1);

while index>0

if W(2,d)==W(size(W,1),d)

L=[L s];

index=0;

else

index2=size(W,1);

while index2>0

if W(index2,d)<W(index2-1,d)

L=[L W(index2,1)];

L=listaDijkstra(L,W,s,W(index2,1));

index2=0;

else

index2=index2-1;

end

index=0;

end

end

end

1. **Apelare**

>> G=graph([1 1], [2 3]);

>> G = addedge(G,2,3)

G =

graph with properties:

Edges: [3x1 table]

Nodes: [3x0 table]

>> G=addedge(G,2,4)

G =

graph with properties:

Edges: [4x1 table]

Nodes: [4x0 table]

>> G=addedge(G,2,5)

G =

graph with properties:

Edges: [5x1 table]

Nodes: [5x0 table]

>> G=addedge(G,3,4)

G =

graph with properties:

Edges: [6x1 table]

Nodes: [5x0 table]

>> G=addedge(G,4,6)

G =

graph with properties:

Edges: [7x1 table]

Nodes: [6x0 table]

>> G=addedge(G,4,7)

G =

graph with properties:

Edges: [8x1 table]

Nodes: [7x0 table]

>> G=addedge(G,5,3)

G =

graph with properties:

Edges: [9x1 table]

Nodes: [7x0 table]

>> G=addedge(G,5,4)

G =

graph with properties:

Edges: [10x1 table]

Nodes: [7x0 table]

>> G=addedge(G,5,6)

G =

graph with properties:

Edges: [11x1 table]

Nodes: [7x0 table]

>> G=addedge(G,5,7)

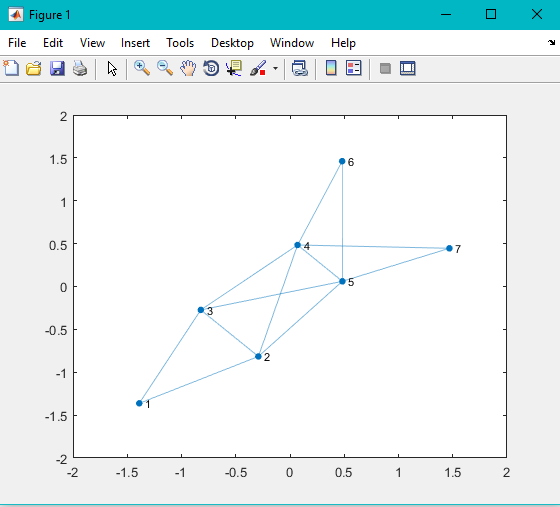
G =

graph with properties:

Edges: [12x1 table]

Nodes: [7x0 table]

>> plot(G)

****

>> G = [0 3 9 0 0 0 0;

0 0 0 7 1 0 0;

0 2 0 7 0 0 0;

0 0 0 0 0 2 8;

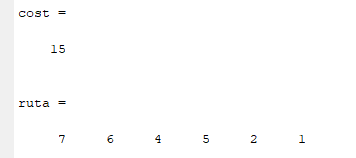
0 0 4 5 0 9 0;

0 0 0 0 0 0 4;

0 0 0 0 0 0 0;

];

>> [cost ruta] = dijkstra(G,1,7)

****