CZU: 371.3:004

ASPECTE DIDACTICE ÎN PREDAREA ALGORITMULUI FORD-FULKERSON

Liubomir CHIRIAC, dr. hab., prof. univ.
Marina BOSTAN, drd.
Universitatea de Stat din Tiraspol

Rezumat. În prezentul articol este examinat algoritmul Ford-Fulkerson pentru determinarea fluxului maxim între două noduri. Sunt abordate metodele de rezolvare manuală, cu ajutorul softului specializat Maple și prin intermediul limbajului de programare Pascal.

Abstract. This paper examines the Ford-Fulkerson algorithm for determining the maximum flow between two nodes. Manual solving methods are approached, with the help of specialized Maple software and through the Pascal programming language.

Cuvinte cheie: graf orientat, flux de rețea, algoritmul Ford-Fulkerson.
Keywords: oriented graph, network flow, Ford-Fulkerson algorithm.

1. Ce reprezintă Flux în rețea?

Grafurile au multiple aplicații practice, fiind strâns legate de multe ramuri ale matematicii (cercetări operaționale, teoria grupurilor, teoria numerelor), dar sunt folosite și ca modele matematice în rezolvarea unor probleme tehnice, economice, etc. Astăzi, teoria grafurilor este folosită în domenii variate: fizică, chimie, biologie, sociologie, tehnologia comunicațiilor, rețelele de calculatoare, sisteme de transport etc. [3].

Rețelele de transport pot modela curgerea lichidului în sisteme cu țevi, deplasarea pieselor pe benzi rulante, deplasarea curentului prin rețele electrice, transmiterea informațiilor prin rețele de comunicare etc.

O problemă des întâlnită într-o rețea de transport este cea a găsirii fluxului maxim posibil prin arcele rețelei astfel încât:

1. să nu fie depășite capacitățile arcelor;
2. fluxul să se conserve în drumul său de la nodul sursă - *s* la nodul terminal – *t.*

Probleme de flux maxim apar în multe domenii, precum curgerea lichidelor, linii de asamblare, transportul curentului prin rețele de curent electric, cercetări operaționale ș.a.

O rețea de transport este un digraf (graf orientat) conex fără cicluri.

Un flux într-o rețea este o funcție care atribuie unui arc un anumit număr – ponderea arcului [2].

2. Descrierea algoritmului Ford-Fulkerson

Algoritmul Ford-Fulkerson determina fluxul maxim care poate fi introdus într-o rețea de transport și se mai cunoaște și sub denumirea de problema debitului maxim (max flow). Acest algoritm se bazează pe determinarea iterativă a unor drumuri de creștere a fluxului și acumularea acestora într-un flux total, până la apariția în rețea a unei tăieturi[1] , care separă sursa de stoc.

[1] Tăietură în graf – un set de muchii (arcuri), eliminarea cărora divide graful în două componente conexe

Problema cu debitul maximal a fost formulată în anul 1955 de către T.E. Harris, care a propus un model simplificat al fluxului de trafic feroviar. În scurt timp a fost demonstrată teorema fluxului maximal și tăieturii minimale a unui graf. În același an L. Ford și D. Fulkerson în premieră au elaborat algoritmul special preconizat pentru rezolvarea acestei probleme. Algoritmul lor a primit denumirea *„Algoritmul lui Ford-Fulkerson”*. Mai jos vom prezenta o metodă accesibilă de realizare a algoritmului respectiv.

Etape de realizarea algoritmului:

Pasul 1. Evidențiem (căutăm) toate lanțurile (căile, drumurile) de la sursa grafului la destinație;

Pasul 2. Fiecărui lanț (drum) i se atribuie un posibil flux mai mare de la sursă la destinație (îl scriem printr-o fracțiune cu ponderea arcului; în acest caz, fluxul (debitul) nu poate depăși ponderea arcului, dar poate fi egal cu acesta);

Pasul 3. Dacă fluxul devine egal cu ponderea arcului, atunci acest arc este saturat, adică este imposibil să treacă un flux prin acest arc atunci când se iau în considerare circuitele din graf;

Pasul 4. Deci alegem toate circuitele posibile până când devine imposibil să ajungem de la sursă la destinație;

Pasul 5. Fluxul în rețea va fi egal cu suma fluxurilor tuturor arcurilor saturate la fluxul grafului (trebuie remarcat faptul că suma fluxurilor tuturor arcurilor incidente la fluxul grafului este egală cu suma fluxurilor tuturor arcurilor incidente la sursa grafului).

3. Aplicarea algoritmului Ford-Fulkerson

Problemă. Avem o rețea de conducte prin care poate curge apă. În graful de mai jos, fiecare conducta are o capacitate ce determina cantitatea maxima de apă ce trece prin conducta respectivă într-o unitate de timp. Considerând că avem o sursa de unde se pompează apa și o destinație unde dorim să ajungă apa, se cere de aflat care este debitul maxim de apa (cantitatea de apă pe unitatea de timp) ce poate ajunge de la sursa 1 la destinația 8 folosind rețeaua de conducte existent?

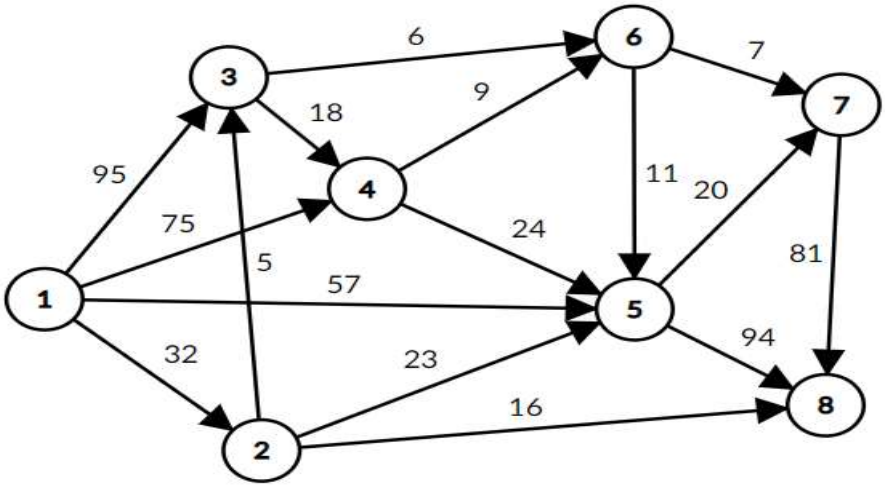


Figura 1. Rețeaua de conducte reprezentată prin graful orientat [7]

Soluție. Folosind algoritmul Ford-Fulkerson, găsim cel mai mare flux de la 1 la 8. Evidențiem toate drumurile posibile de la 1 la 8.

1) **1→3→6→7→8**
2) **1→3→4→5→7→8**
3) **1→3→4→5→8**
4) **1→4→5→8**
5) **1→4→6→7→8**
6) **1→4→5→7→8**
7) **1→4→5→8**
8) **1→5→8**
9) **1→5→7→8**
10) **1→2→8**
11) **1→2→5→8**
12) **1→2→5→7→8**

Pasul 1. Selectăm un flux arbitrar, din cele evidențiate mai sus, de exemplu, 1-3-6-7-8. Lățimea sa de bandă este egal cu minimul tuturor capacităților arcurilor incluse în acesta, adică 6. Reducem debitul arcurilor acestui flux cu 6, arcul saturat 3-6 se marchează.

1-(95)-3-(6)-6-(7)-7-(81)- 8;

1-(95/6)-3-(6/6)-6-(7/6)-7-(81/6)- 8;

1-(89)-3-(0)-6-(1)-7-(75)- 8;

Obținem, suma S=6.

Pasul 2. Alegeți un flux arbitrar, de exemplu, 1-4-5-8. Lățimea sa de bandă este egal cu minimul tuturor capacităților arcurilor incluse în acesta, adică 24. Reducem debitul arcurilor acestui flux cu 24, arcul saturat 4-5 se marchează.

1-(75)-4-(24)-5-(94)-8;

1-(75/24)-4-(24/24)-5-(94/24)-8;

1-(51)-4-(0)-5-(70)-8;

Obținem, S=6+24=30.

Pasul 3. Alegeți un flux arbitrar, de exemplu, 1-5-8. Lățimea sa de bandă este egal cu minimul tuturor capacităților arcurilor incluse în acesta, adică 57. Reducem debitul arcurilor acestui flux cu 57, arc saturat 1-5 se marchează.

1-(57)-5-(70)-8;

1-(57/57)-5-(70/57)-8;

1-(0)-5-(13)-8;

Obținem, S=6+24+57=87.

Pasul 4. Alegeți un flux arbitrar, de exemplu 1-2-8. Lățimea sa de bandă este egal cu minimul tuturor capacităților arcurilor incluse în acesta, adică 16. Reducem debitul arcurilor acestui flux cu 16, arcul saturat 2-8 se marchează.

1-(32)-2- (16)- 8;

1-(32/16)- 2- (16/16)- 8;
1-(16)- 2- (0)- 8;
Obținem, S=6+24+57+16=103.

Pasul 5. Selectați un flux arbitrar, de exemplu, 1-2-5-8. Lățimea sa de bandă este egal cu minimul tuturor capacităților arcurilor incluse în acesta, adică 16. Reducem debitul arcurilor acestui flux cu 23, arc saturat 5-8 se marchează.

1-(16)-2-(23)-5-(13)-8;
1-(16/13)-2-(23/13)-5-(13/13)-8;
1-(3)-2-(10)-5-(0)-8;
Obținem, S=6+24+57+16+13=116.

Pasul 6. Selectați un flux arbitrar, de exemplu 1-2-5-7-8. Lățimea sa de bandă este egal cu minimul tuturor capacităților arcurilor incluse în acesta, adică 3. Reducem debitul arcurilor acestui flux cu 3, arc saturat 1-2 se marchează.

1-(3)-2-(10)-5-(20)-7-(75)-8;
1-(3/3)-2-(10/3)-5-(20/3)-7-(75/3)-8;
1-(0)-2-(7)-5-(17)-7-(72)-8;
Obținem, S=6+24+57+16+13+3=119.

Pasul 7. Alegeți un flux arbitrar, de exemplu, 1-4-6-7-8. Lățimea sa de bandă este egal cu minimul tuturor capacităților arcurilor incluse în acesta, adică 1. Reducem debitul arcurilor acestui flux cu 1, arc saturat 6-7 se marchează.

1-(51)-4-(9)-6-(1)-7-(72)-8;
1-(51/1)-4-(9/1)-6-(1/1)-7-(72/1)-8;
1-(50)-4-(8)-6-(0)-7-(71)-8;
Obținem, S=6+24+57+16+13+3+1=120.

Pasul 8. Selectați un flux arbitrar, de exemplu, 1-4-6-5-7-8. Debitul său capacitatea este egală cu minimul capacității arcurilor incluse în ea, adică 8. Reduceți debitul arcurilor acestui flux cu 8, arc saturat Tăiați 4-6. Nu mai există căi.

1-(50)-4-(8)-6-(11)-5-(17)-7-(71)-8;
1-(50/8)-4-(8/8)-6-(11/8)-5-(17/8)-7-(71/8)-8;
1-(42)-4-(0)-6-(3)-5-(9)-7-(63)-8;
Debit total, S=6 + 24 + 57 + 16 + 13 + 3 + 1 + 8 = 128.

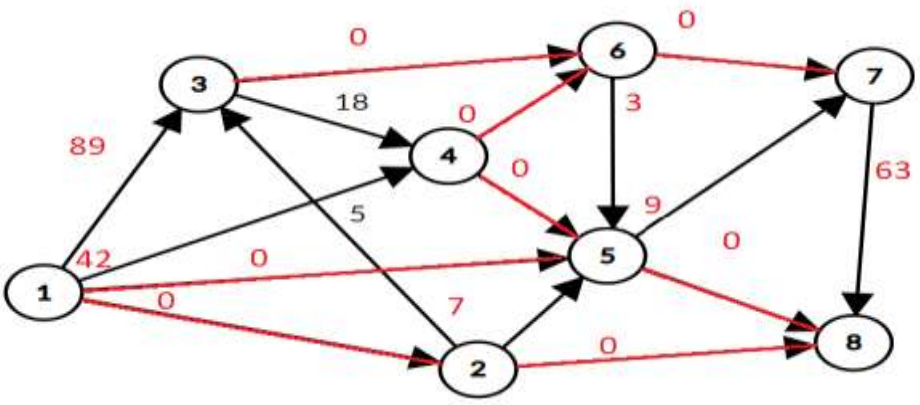


Figura 2. Fluxul de rețea obținut [7]

Inserăm marcajele. Nodul sursă este nodul 1 are marcajul 0. Din acest nod 1 către noduri 3 și 4 trec arce nesăturate (vezi Figura 2), respectiv marcăm nodurile cu +3 și +4. Alte marcaje nu pot fi plasate. Respectiv fluxul maximal a fost găsit și mulțimea nodurilor {2,5,6,7,8} (nodurile nemarcate) formează tăietura grafului. Se observă, că graful obținut nu mai are nici un lanț de la nodul sursă 1 la nodul terminal 8.

4. Rezolvarea în aplicația Maple [5]

```
restart : with(networks) :
new(G) : v :=$1 ..8 :
addvertex([v], G);
```

1, 2, 3, 4, 5, 6, 7, 8

```
v1 := 1 :#sursa
v2 := 8 :#destinatia
E := [[1, 2], [1, 3], [1, 4], [1, 5], [2, 3], [2, 5], [2, 8], [3, 4], [3,
    6], [4, 5], [4, 6], [5, 7], [5, 8], [6, 5], [6, 7], [7, 8]]
```

[[1, 2], [1, 3], [1, 4], [1, 5], [2, 3], [2, 5], [2, 8], [3, 4], [3, 6],
[4, 5], [4, 6], [5, 7], [5, 8], [6, 5], [6, 7], [7, 8]]

```
#Pondere
w := [32, 95, 75, 57, 5, 23, 16, 18, 6, 24, 9, 20, 94, 11, 7, 81]
```

[32, 95, 75, 57, 5, 23, 16, 18, 6, 24, 9, 20, 94, 11, 7, 81]

```
addedge(E, weights = w, G) :
flux = flow(G, v1, v2, ed)
```

flux = 128

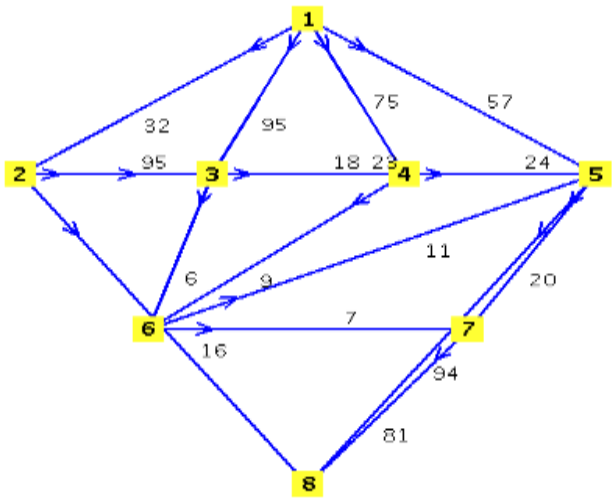
```
with(GraphTheory) :
G := Digraph({[[1, 2], 32], [[1, 3], 95], [[1, 4], 75], [[1, 5],
    57], [[2, 3], 95], [[2, 5], 23], [[2, 8], 16], [[3, 4], 18], [[3,
    6], 6], [[4, 5], 24], [[4, 6], 9], [[5, 7], 20], [[5, 8], 94], [[6,
    5], 11], [[6, 7], 7], [[7, 8], 81]})
```

G := Graph 6: a directed weighted graph with 8 vertices and 16 arc(s)

```
> IsNetwork(G)
```

{1}, {8}

> *DrawNetwork*(*G*)



MaxFlow(*G*, 1, 8)

$$128,\ \begin{bmatrix} 0 & 32 & 24 & 15 & 57 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 23 & 0 & 0 & 9 \\ 0 & 0 & 0 & 18 & 0 & 6 & 0 & 0 \\ 0 & 0 & 0 & 0 & 24 & 9 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 20 & 92 \\ 0 & 0 & 0 & 0 & 8 & 0 & 7 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 27 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Fluxul maximal, după cum se vede, este 128.

5. Rezolvarea în Pascal (consola Delphi) [6]

```
program Project2;
 {$APPTYPE CONSOLE}
uses
  SysUtils;
// Finding the Maximum Flow using Ford-Fulkerson Algorithm
const fi = 'MAXFLOW.INP';
      fo = 'MAXFLOW.OUT';
      max = 1000;
type TCapacities = array [1..max, 1..max] of integer;
var c: TCapacities;
    f: TCapacities;
    Trace: array [1..max] of integer;
    n, s, t: integer;

    procedure Enter;
    var m, i, u, v: integer;
        f2: text;
    begin
        assign(f2, fi); reset(f2);
        fillchar(c, sizeof(c), 0);
        readln(f2, n, m, s, t);
        for i:= 1 to m do readln(f2, u, v, c[u, v]);
        close(f2);
    end;
```

```
function FindPath: Boolean;
var u, v: integer;
    Queue: array[1..max] of integer;
    Front, Rear: integer;
begin
    fillchar(Trace, sizeof(Trace), 0);
    Front:= 1; Rear:= 1;
    Queue[1]:= s;
    Trace[s]:= n+1;
    repeat
            u:= Queue[Front];
            Inc(Front);
                        for v:= 1 to n do
                if (Trace[v] = 0) and (c[u, v] > f[u, v]) then
                begin
                    Trace[v]:= u;
                    if v = t then
                    begin
                            FindPath:= true;
                            exit;
                    end;
                     Inc(Rear);
                    Queue[Rear]:= v;
                end;
    until Front > Rear;
    FindPath:= false;
end;
procedure IncFlow;
var Delta, u, v: integer;
begin
    Delta:= MaxInt;
    v:= t;
    repeat
            u:= Trace[v];
            if c[u, v] - f[u, v] < Delta then Delta:= c[u, v] - f[u, v];
            v:= u;
    until v = s;

    v:= t;
    repeat
            u:= Trace[v];
            f[u, v]:= f[u, v] + Delta;
            f[v, u]:= f[v, u] - Delta;
            v:= u;
    until v = s;
end;
```

```
71      procedure PrintResult;
72      var u, v: integer;
73          m: integer;
74          f2: text;
75      begin
76          assign(f2, fo); rewrite(f2);
77          m:= 0;
78          for u:= 1 to n do
79              for v:= 1 to n do
80                  if f[u, v] > 0 then
81                  begin
82                     writeln(f2, 'f[', u, ', ', v, '] = ', f[u, v]);
83                     if u = s then m:= m + f[s, v];
84                  end;
85          writeln(f2, 'Max Flow: ', m);
86          for u:= 1 to n do
87          begin
88                  for v:= 1 to n do write(f2, f[u, v]:2, ' ');
89                  writeln(f2);
90          end;
91          close(f2);
92      end;
95  BEGIN
96          Enter;
97          fillchar(f, sizeof(f), 0);
98          repeat
99                  if not FindPath then break;
100                 IncFlow;
101         until false;
102         PrintResult;
103 END.
```

Fișierele cu datele de intrare și ieșire, după executarea programului:

MAXFLOW.INP — Блокнот

Файл Правка Формат Вид Справка

```
8 16 1 8
1 3 95
1 4 75
1 2 32
1 5 57
2 3 5
2 5 23
2 8 16
3 4 18
3 6 6
4 6 9
4 5 24
5 7 20
5 8 94
6 5 11
6 7 7
7 8 81
```

MAXFLOW.OUT — Блокнот

Файл Правка Формат Вид Справка

```
f[1, 2] = 32
f[1, 3] = 6
f[1, 4] = 33
f[1, 5] = 57
f[2, 5] = 16
f[2, 8] = 16
f[3, 6] = 6
f[4, 5] = 24
f[4, 6] = 9
f[5, 7] = 11
f[5, 8] = 94
f[6, 5] = 8
f[6, 7] = 7
f[7, 8] = 18
Max Flow: 128
 0 32  6 33 57  0  0  0
-32  0  0  0 16  0  0 16
-6  0  0  0  0  6  0  0
-33  0  0  0 24  9  0  0
-57 -16  0 -24  0 -8 11 94
 0  0 -6 -9  8  0  7  0
 0  0  0  0 -11 -7  0 18
 0 -16  0  0 -94  0 -18  0
```

Articol realizat în cadrul proiectului de cercetări științifice „Metodologia implementării TIC în procesul de studiere a științelor reale în sistemul de educație din Republica Moldova din perspectiva inter/transdisciplinarității (concept STEAM)”, inclus în „Program de stat” (2020-2023), Prioritatea IV: Provocări societale, cifrul 20.80009.0807.20, cu suportul financiar oferit de Agenția Națională pentru Dezvoltare și Cercetare

Bibliografie

1. MARUSIC, G.; FALICO, N.; KULEV, M. Aspecte algoritmice din teoria grafurilor privind fluxul maxim și drumurile minime (maxime): Indicații metodice la disciplinele Matematici speciale și Structuri de date și algoritmi. 2018. 48p. ISBN 978-9975-45-534-3.
2. BANG-JENSEN, J.; GUTIN, G.Z. Digraphs: Theory, Algorithms and Applications. Springer-Verlag, 2007. 754 p. ISBN 978-1-84800-998-1.
3. ХАГГАРТИ, Р., „Дискретная математика для программистов”, перевод с английского, Москва, Техносфера, 2003, 320 с., ISBN 5-94836-016-4.
4. BÂRZĂ, S.; MORGAN, L-M. Algoritmica grafurilor. București: Ed. Fundației România de Mâine, 2008. 148 p. ISBN 978-973-163-147-9 [vizitat 05.02.2018] http://www.ocpiilfov.ro/ocpi_ilfov/Man.pdf.
5. http://www.maplesoft.com/products/maple/new_features/maple18/Graph_Theory.aspx [vizitat 12.09.2021].
6. https://tosonnguyen.wordpress.com/2013/09/25/luong-cuc-dai-using-ford-fulkerson-algorithm/ [vizitat 12.09.2021].
7. https://csacademy.com/app/graph_editor/ [vizitat 10.09.2021].