Uffe Møller

On Two Algorithms Finding the Maximal Flow in a Network

Preprint
February
1973

3

Institute of Mathematical Statistics
University of Copenhagen

Uffe Møller

On Two Algorithms Finding the Maximal Flow in a Network.

Preprint 1973 No. 3

INSTITUTE OF MATHEMATICAL STATISTICS

UNIVERSITY OF COPENHAGEN

February 1973

POLYTEKNISK FORLAG OG TRYKKERI, SØLVGADE 83, 1307 K. TLF. 11 98 22

Abstract:

The present paper first states the linear programming problem of finding the maximal flow in a network. Secondly, it presents a new algorithm for the solution of the problem, and thirdly it makes a remark on an existing erroneous algorithm.

Key-words: network, linear programming, maximum flow.

Description:

A network consists of a set of nodes $(N_i)_{i \in \{1,\ldots,n\}}$ and a set of arcs $(N_{ij})_{i,j \in \{1,\ldots,n\}}$ connecting the nodes. N_{ij} will here denote the directed arc from node N_i to node N_j.

For every arc N_{ij} we have a nonnegative real number b_{ij} which we call the capacity of the arc. This means that we allow a flow x_{ij} in the arc satisfying the condition $0 \leqq x_{ij} \leqq b_{ij}$. The connection between the flows in the different arcs will be:

1) We allow one node (N_1) to be a source, at which there will be an inlet of flow, and one node (N_n) to be a terminal which will be the outlet of flow. Some authors use the word sink for the terminal.
2) For any other node we must demand that the sum of flows to it is equal to the sum of flows from it.

This means that the inlet and the outlet of flow mentioned in 1) will be equal, and it is this flow we want to maximize.

The new algorithm presented in this paper follows the labeling method described by Hu [1] with the one exception that we allow

POLYTEKNISK FORLAG OG TRYKKERI, SØLVGADE 83, 1307 K. TLF. 11 98 22

noninteger capacities. In order to assure convergence to the true maximum we will consider all flows less than a given (small) number, say 10^{-6}, being zero.

Denoting the number of nodes by n we have in the new algorithm the capacities stored in an n x n array such that b_{ij} = cap(i,j). At the return from the procedure the resulting actual flow of the solution (or of one of the solutions) will be given as x_{ij} = flow(i,j), where flow is an n x n array.

It is assumed that the nodes are numbered from 1 to n and that the source is numbered 1 and the terminal n. This is not a serious restriction, and the procedure could easily be re-arranged to include two call parameters for the numbers of the source and the terminal. The reason for not doing it here is that transfer of parameters is often much time-consuming.

Operation:

For each increase of the flow the nodes are divided into two complementary subsets in the array chain, separated by the index pointer. The first subset (in [2] denoted by X) contains the nodes to which the flow can be increased from the source. The second subset (in [2] denoted by $\overline{X}$ and in the algorithm by CX) contains the rest of the nodes. Within X the nodes are stored in the same order as they were transferred to it from $\overline{X}$ beginning with the source, and as X is scanned sequentially for arcs to nodes in $\overline{X}$, it means that we use the so-called first-labeled first-scanned method mentioned in (2). This means further that any flow augmenting path we obtain is the one which contains the minimum number of arcs.

When a node, N_j, is transferred from $\overline{X}$ to X by means of a possible augment of flow from the node N_i, from [j] is set to i and plus [j] is set to the maximal possible augment of flow.

Tests:

The algorithm has been tested with numerous examples, ensuring test of all parts of the algorithm. Results have been compared with the results from algorithm 324 [3] showing that all deviations were due to the error in algorithm 324, which will be mentioned in the next section.

Remark on Algorithm 324:

The algorithm is not able to find a flow-increasing path from the source to the terminal such that the orientation of one or more of the arcs is directing to the source, i.e. such that the flow in this branch (these branches) has to be decreased. See [1] pp. 336-337. Thus the algorithm will in some cases produce erroneous results.

Example:

A network with 6 nodes have the following capacities:

$b_{12} = 21$, $b_{13} = 31$, $b_{14} = 71$, $b_{15} = 4$, $b_{23} = 31$,
$b_{43} = 24$, $b_{45} = 63$, $b_{36} = 71$, $b_{46} = 23$, $b_{56} = 44$,

POLYTEKNISK FORLAG OG TRYKKERI, SØLVGADE 83, 1307 K. TLF. 11 98 22

b_{ij} indicating the capacity of the arc from node i to node j. All other capacities are zero.

Algorithm 324 will give a solution with the following flows:

$f_{12} = 16$, $f_{13} = 31$, $f_{14} = 71$, $f_{15} = 4$, $f_{23} = 16$,

$f_{43} = 24$, $f_{45} = 24$, $f_{36} = 71$, $f_{46} = 23$, $f_{56} = 28$,

and the maximal flow 122.

But the flow may be increased by 5 through the arcs from node 1 to node 2 and from node 2 to node 3, then decreasing the flow node 4 to node 3 and again increasing the flow from node 4 to node 5 and from node 5 to node 6.

In this way the maximal flow is found to be 127:

$f_{12} = 21$, $f_{13} = 31$, $f_{14} = 71$, $f_{15} = 4$, $f_{23} = 21$,

$f_{43} = 19$, $f_{45} = 29$, $f_{36} = 71$, $f_{46} = 23$, $f_{56} = 33$.

References:

[1] Hadley, G.: Linear Programming. Addison-Wesley, Reading (Mass.), 1962. (5th Printing 1971, paperback).

[2] Hu, T.C.: Integer Programming and Network Flows. Addison-Wesley, Reading (Mass.), 1969.

[3] Bayer, G.: Algorithm 324, Maxflow, Comm. ACM. Vol. 11, No. 2. Feb. 1968.

POLYTEKNISK FORLAG OG TRYKKERI, SØLVGADE 83, 1307 K. TLF. 11 98 22

```


real procedure maxflow(n, max, flow, eps);
value                  n,               eps;
integer                n;    real       eps;
array                       max, flow;

begin

comment the procedure computes the maximal flow
in a network, where
n            is the number of nodes in the network.
             The nodes are numbered from 1 to n
             with 1 as the source and n as the
             terminal.
max(i,j)     is the maximal flow in the arc from
             node i to node j (oriented). It must
             be greater than or equal to zero.
flow(i,j)    becomes in the same way the flow - or
             one of the possible flows - in the
             solution.
eps          Flows less than eps will not be taken
             into consideration.
maxflow      becomes the resulting flow in the
             network from 1 to n.

The method is a modification of the method
described in T.C.Hu: Integer Programming and
Network Flows, pp.105-120. The method does not
take multiple solutions into account;

integer array from, chain(1:n);
real array plus(1:n);
real a, b, c, more, total;
integer i, j, k, m, x, nonx, pointer;
boolean nonterm;

comment
nonterm      is false when the node n is in the set
             X and true otherwise
from(i)      indicates the node from which the flow
             to node i was increased or to which
             the flow from node i was decreased.
chain        chain(1) to chain(pointer-1) are the
             nodes in the set X, the rest are the
             nodes of CX.
plus(i)      is the maximal increase of flow to the
             node i through the path indicated in
             from.
total        is the present flow from 1 to n.
more         is the present increase of flow from
             1 to n;
```

```
comment  the flow is set to zero;

for i := 1 step 1 until n do
for j := 1 step 1 until n do
  flow(i,j) := 0.0;

total := 0.0;

plus(1) := 1.0'600;   comment  the maximal number;
if eps<=3.0'-10 then eps := 3.0'-10;
      comment  the relative precision;

comment  start the iteration;

for m := 1,m+1 while -,nonterm do
  begin
  comment  the initial state: the source is the
    only node in X;
  pointer := 2;
  for k := 1 step 1 until n do chain(k) := k;
  for i := 1,i+1 while i<pointer and nonterm do
    begin
    comment a new node, x, in X is selected;
    x := chain(i);
    more := plus(x);
    for j := pointer,j+1 while j<=n do
      begin
      comment a node, nonx, in CX is selected;
      nonx := chain(j);
      a := max(x,nonx);
      b := max(nonx,x);
      comment check if a flow between x and nonx
        is possible;
      if abs(a+b)>eps then
        begin
        c := a - flow(x,nonx) + flow(nonx,x);
        comment  check if the flow may be increased;
        if c>eps then
          begin
          comment the flow may be increased:
            transfer nonx to X, note that the
            transfer took place be means of the
            arc from x and note the increase of
            the flow;
          from(nonx) := x;
          plus(nonx) := if c<more then c else more;
          chain(j) := chain(pointer);
          chain(pointer) := nonx;
          pointer := pointer + 1
          end  c>0;
        end  a+b>0;
      end  j-loop;
    nonterm := chain(n)=n and pointer<=n
    end  i-loop;
```

```
  comment  if a flow-augmenting path exists, the
    flow in the network will be increased;

  if -,nonterm then
    begin
    more := plus(n);
    total := total + more;
    for i := n,j while i>1 do
      begin
      j := from(i);
      comment  the path came from node j;
      a := flow(i,j) - more;
      comment  first try if the flow in the
        opposite arc may be decreased;
      if a>=0.0 then flow(i,j) := a
                 else
        begin
        comment  second increase the flow from
          j to i (a is negative);
        flow(i,j) := 0.0;
        flow(j,i) := flow(j,i) - a
        end;
      end  i-loop;
    end  increase of flow;
  end  m-loop;

maxflow := total

end  maxflow;
```