Yugoslav Journal of Operations Research
10 (2000), Number 1, 13-26

FINDING THE k MOST VITAL ELEMENTS OF AN s-t PLANAR DIRECTED NETWORK[*]

Dimiter IVANCHEV

Institute of Applied Mathematics and Computer Science
Technical University of Sofia
P.O. Box 384, 1000 Sofia, Bulgaria

Abstract: For a given s-t planar directed network with lower and upper arc capacities we find those k arcs the removal of which minimizes the flow value of the maximum flow. Such arcs are called the k most vital arcs of the network. Analogously we find the k most vital nodes. Further more, we find the k bicriterial lexicographical most vital elements, the removal of which minimizes the maximum flow value first, and on the other hand, is the "cheapest" variant. Such problems arise if one wants to know in advance what the consequences will be if some of the network elements terminate their function or have to be switched off.

Keywords: k most vital arcs/nodes, planarity of directed networks, multicriterial optimization on networks.

1. INTRODUCTION AND PROBLEM FORMULATION

We consider a finite, connected, directed and planar graph $G=(V,A)$ with a node set $V=(V_1,\dots,V_n)$, arc set $A=(A_1,\dots,A_m)$ and a facet set $F=(F_1,\dots,F_r)$. We assume the only source node $s=V_1$ and sink node $t=V_n$ to belong to the edge of one of the facets and let it be the unbounded one. Such networks are called s-t planar networks. To each arc A_p we assign two integer arc weights a_p and b_p called lower and upper arc capacities of the flow. The classical max-flow problem is formulated as follows: find an integer vector x, such that

[*] I am grateful to Prof. Helmut Viesel from the University of Karlsruhe, Germany, and to Dr. Dimitris Kydros from the T.E.I. of Serres, Greece for their indirect contribution. Many thanks to Miss Elena Radovanova for urging me to write this paper. Special thanks to the anonymous referees as well.

$$v(x) \to \max \tag{1}$$

subject to

$$\sum\{x_p / A_p \in CC_i^+\} - \sum\{x_p / A_p \in CC_i^-\} = \begin{cases} v(x), & i = 1 \\ 0, & i \neq 1, n \\ -v(x), & i = n \end{cases} \tag{2}$$

and

$$a_p \le x_p \le b_p, \quad p = 1, \ldots, m\,. \tag{3}$$

Here we denote by CC_i^+ the set of all arcs leaving V_i and by CC_i^- the set of those arcs entering it. The set of all arcs starting from V_i or ending in it is called a cocycle defined by V_i. We denote it by CC_i. Obviously, $CC_i = CC_i^+ \cup CC_i^-$, i.e. CC_i contains forward and backward arcs, those from CC_i^+ and CC_i^-, respectively. More general: for a given $V' \subset V$, we denote by $CC(V')$ the set of all arcs leaving or entering V' and we call it a cocycle defined by V'. It also contains forward arcs $CC^+(V')$ and backward arcs $CC^-(V')$. If in addition V' contains the source node s and not the sink node t, we call $CC(V')$ a cut in G. We denote the set of all cuts in $G = (V, A)$ by $CS(V, A)$. For a given $CC(V')$ we denote by

$$cap(CC(V')) := \sum\{b_p / A_p \in CC^+(V')\} - \sum\{a_p / A_p \in CC^-(V')\} \tag{4}$$

the so-called capacity of $CC(V')$. The well-known result is the max-flow min-cut theorem:

$$\max_x\{v(x)/(2),(3)\} = \min\{cap(CC(V'))/CC(V') \in CS(V, A)\}\,. \tag{5}$$

Now let k be a positive integer with $0 < k < m$. The problem (MVA^k) for finding the k most vital arcs in the network is:

$$\min_{A^k}\{\min_{V'}\{cap(CC(V'))/CC(V') \in CS(V, A \setminus A^k), A^k \subset A, |A^k| = k\}\}\,.$$

Analogously, the problem (MVN^k) for finding the k most vital nodes in G is:

$$\min_{N^k}\{\min_{V'}\{cap(CC(V'))/CC(V') \in CS(V \setminus N^k, A^*), N^k \subset V, |N^k| = k\}\},$$

for $0 < k < n$ and A^* - the set of remaining arcs.

Note that conditions (2) and (3) can be contradictable if some lower capacities in (3) are not zero, i.e. it is possible that no flow does exist in the network. In this case we assume by definition that the maximum flow value is zero. Hence, the removal of some network elements can reduce the maximum flow value to zero either if there exists a feasible flow in the network or if there is no feasible one.

We further we shall investigate (MVA^k) since (MVN^k) can be reduced to the previous one - see Section 5. Such a problem has been estimated in [10] for undirected planar networks with zero lower capacities. It has been solved in [1] for undirected networks by two criteria - arc capacities and reliabilities. In [9] it has been solved in directed non-planar graphs. We have solved it in [4, 5] in directed networks if the lower capacities are zero and the upper bounds in (3) depend on a parameter. In [2, 3] we have found the most vital nodes if all capacities do or do not depend on a parameter. The problem seems to be NP-complete if the graph is non-planar. The algorithms in [9, 4, 5] are exponential. We shall deal with the case of the planarity of a directed network with non-zero lower capacities in (3).

2. SOLUTION METHOD AND ALGORITHM

Without loss of generality we assume that there is a single inverse arc $A_m = (V_n, V_1) = (t, s)$ with $a_m = 0, b_m = M$ - a large number. Obviously, there are two possibilities for A_m to place it in the plane. We denote by CS the set of all simple cycles in G containing A_m as a foreward arc and by CCS the set of all simple cocycles which separate s and t and contain A_m as a backeward arc. Obviously, $CCS = CS(V, A)$ holds.

Now we define the dual graph $D(G) = (V^D, A^D)$ as in [5]. An algorithm with a time complexity $O(m)$ for finding it is described in [6]. We have proved some properties of the duality and have presented a PASCAL computer code for finding the dual graph. There are one-to-one correspondences between the sets of pairs from $G = (V, A)$ and $D(G) = (V^D, A^D)$ as follows:

V and F^D,

A and A^D,

F and V^D,

CS and CCS^D,

CCS and CS^D.

Here we denote analogously by F^D, CS^D and CCS^D the sets of all facets, all simple cycles containing A_m^D, and all simple cocycles containing A_m^D in the dual graph, respectively. A_m^D is the inverse arc in $D(G)$, which corresponds to A_m from G. The direction of the arcs A^D are defined in such a way that if CC and C^D are the corresponding cocycle from G and cycle from $D(G)$, the foreward arcs from CC correspond to the foreward arcs from C^D. Now we denote by $s^D = V_1^D$ the vertex of $D(G)$ which corresponds to the unbounded facet F_1 of G and by $t^D = V_r^D$ the vertex of $D(G)$ which corresponds to the neighbour facet F_r from G, so that $A_m^D = (t^D, s^D)$. To each arc A_p^D we assign the same two arc weights a_p and b_p and $D(G)$ becomes a $s^D - t^D$ directed planar network.

If C^D is a $s^D - t^D$ chain in $D(G)$ (particularly a cycle from CS^D) we define its length as

$$l(C^D) := \sum\{b_p / A_p^D \in C^{D+}\} - \sum\{a_p / A_p^D \in C^{D-}\} \tag{6}$$

Note that if C^D is a cycle from CS^D it is so oriented that A_m^D is a backward arc.

Theorem 1. Let $v_{\max}$ be the flow value of the maximum flow in G and let $l_{\min}$ be the length of the (simple) shortest path from s^D to t^D in $D(G)$ (particularly a cycle from CS^D). Then

$$v_{\max} = l_{\max}$$

holds.

Proof: It follows from the one-to-one set correspondence and from (4) and (6): $\text{cap}(CC) = l(C^D)$ if $CC \in CCS$ and $C^D \in CS^D$ are the corresponding cocycle and cycle from G and $D(G)$, respectively. (5) implies that

$$v_{\max} = \min\{\text{cap}(CC) / CC\} = \min\{l(C^D) / C^D\} = l_{\min} \qquad ❑$$

Corollary. (MVA^k) is equivalent to the following problem: which are those k arcs in the dual graph such that if we reduce both their weigths to zero it causes the minimization of the length of the shortest path from s^D to t^D, i.e. $(MVDA^k)$:

$$\min_{DA^k}\{\min_{C^D}\{l(C^D) / a_p = b_p = 0 \text{ for } A_p^D \in DA^k \subset A^D, |DA^k| = k\}\}.$$

We present below a PASCAL-like algorithm for solving $(MVDA^k)$ and hence for solving (MVA^k). It finds in $D(G)$ the shortest path between s^D and t^D if the weights of the k arcs are reduced to zero. It uses the DFS strategy for searching in a dual graph which has m arcs and r nodes since G contains m arcs and r facets.

2.1. The Algorithm MVAPDN (Most Vital Arcs in a Planar Directed Network)

(* Input:

s - t planar directed network $G=(V, A)$ with two arc weights a_p and b_p, $p=1,\dots,m$, k- integer, $0<k<m$;

Output:

Arc set List and Cycle, containing no more than k arcs of the dual graph (and hence of G) the removal of which either minimizes the maximum flow value or makes (2) and (3) contradictable.

Notations:

$L_j(K), S_j(K), F_j(K), C_j(K)$ - labels of V_j^D, $j=1,\dots,r$ if K arcs have to be removed, $K=0,1,\dots,k$, where:

$L_j(K)$ - the minimum length from s^D to V_j^D if K arcs can be removed;

$S_j(K)$ - the number of the last vertex before V_j^D in the shortest chain from s^D to V_j^D, such that if $S_j(K)=i>0$ then the last arc is $(i, j)^D$ and if $i<0$ then the last arc is $(j, i)^D$;

$F_j(K)$ - a logical variable which is true if the last arc in the chain $s^D-V_j^D$ has to be removed and false otherwise;

$C_j(K)$ - an integer variable which is -1, if there is a negative cycle in $D(G)$, containing V_j^D and 1 otherwise; it is initialized first as zero;

M - a large number in the whole algorithm.

***)**

Form the dual graph $D(G)$ using the algorithm DUAL from [6];

$List := \emptyset; Cycle := \emptyset$;

(*Main loop*)

```
for $K := 0$ to k do
    begin $i := 1$;
        if $j=1$ then
```

```
          begin $L_1(K) := 0; S_1(K) := M; F_1(K) := false; C_1(K) := -1$ end;
      for $j := 2$ to $r$ do
          begin $L_j(K) := M; S_j(K) := 0; F_j(K) := false; C_j(K) := 0$ end;
      while $i \neq M$ do
          begin
              find an arc $A_p^D = (i, j)$ (*Case 1*) or
                    $A_p^D = (j, i)$ (*Case 2*) such that
              if (Case 1) then
              begin
(*1*)                 if $L_j(K) > L_i(K) + b_p$ then
                  begin $L_j(K) := L_i(K) + b_p$; $S_j(K) := i$ end;
(*2*)                 if $K > 0$ and $L_j(K) > L_i(K-1)$ then
                  begin
                       $L_j(K) := L_i(K-1)$; $S_j(K) := i$; $F_j(K) := true$
                  end
              end;
              if (Case 2) then
              begin
(*3*)                 if $L_j(K) > L_i(K) - a_p$ then
                  begin $L_j(K) := L_i(K) - a_p$; $S_j(K) := -i$ end;
(*4*)                 if $K > 0$ and $L_j(K) > L_i(K-1)$ then
                  begin
                       $L_j(K) := L_i(K-1)$; $S_j(K) := -i$; $F_j(K) := true$
                  end
              end;
          if ((Case 1) or (Case 2)) then
              begin $i := j$;
(*5*)             if $C_i(K) < 0$ then
```

(*A negative cycle has been found in $D(G)$ with arcs in Cycle; List contains all K arcs the length of which have to be reduced to zero *)

```
      begin start := i;
          repeat
```

```
                $j := S_i(K)$ ;
                if $j > 0$ then
                begin
                    $Cycle := Cycle \cup \{(j, i)\}$ ;
                    if $F_i(K)$ then $List := List \cup \{(j, i)\}$
                end
                else (* $j < 0$ *)
                begin
                    $Cycle := Cycle \cup \{(i, -j)\}$ ;
                    if $F_i(K)$ then $List := List \cup \{(i, -j)\}$
                end;
            if $K > 0$ then $K := K - 1$ ;
            $i := |j|$
        until $i = start$ ; STOP.
        end
        else (* $C_i(K) \ge 0$ *) $C_i(K) := -1$
    end
    else (*neither Case 1 nor Case 2*)
        begin $C_i(K) := 1$ ; $|S_i(K)|$ end
    end (*of while*)
end; (*of main loop*)
```

(* Forming the arc set List from the dual graph with reduced lengths*)

```
 $j := r$ ;
 repeat
     if $F_j(k)$ and $k > 0$ then
         if $S_j(k) > 0$ then $List := List \cup \{(S_j(k), j)\}$
             else $List := List \cup \{(j, -S_j(k))\}$ ;
     $s := |S_j(k)|$ ;
     if $F_j(k)$ and $k := k - 1$; $j := s$
 until $j = M$ .
```

3. TWO ILLUSTRATIVE EXAMPLES

We shall give here in some illustrative examples in order to make the algorithm much clearer. In the original graph G all verticies are drawn with circle lines and the arcs - with continuous lines. In the dual graph they are given by rectangles and broken-lines, respectively.

Example 1. We shall now demonstrate how to find the two most vital arcs. Both graphs are given in Fig. 1 with the arc weights a_p / b_p as lower and upper bounds of the capacities of G and arc lengths of $D(G)$.

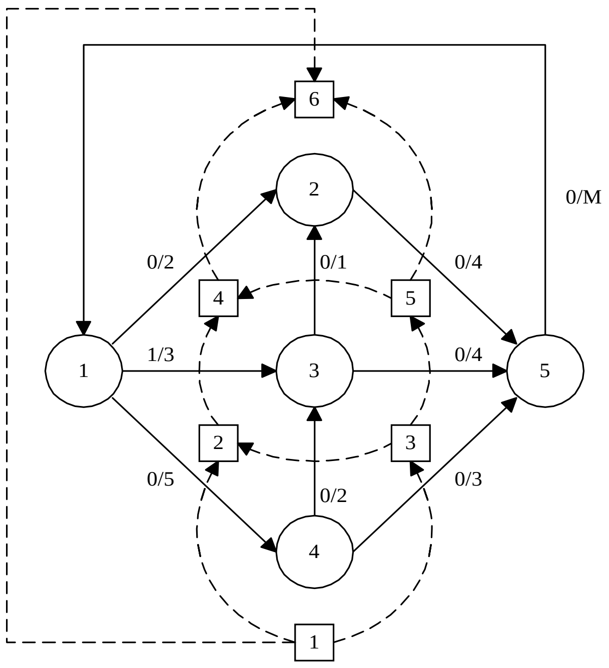

Figure 1.

First step $(K = 0)$. Initialization: $L_1(0) := 0$, $L_j(0) := M$, $j := 2,\ldots,6$; $S_1(0) := M$, $S_j(0) := 0$, $j := 1,\ldots,6$; $F_j(0) := \text{false}$, $j := 1,\ldots,6$; $C_1(0) := -1$, $C_j(0) := 0$, $j := 2,\ldots,6$; $List := \emptyset$; $Cycle := \emptyset$.

We shall start with $i := 1$ and find a dual arc (1, 2) such that $L_2(0) > L_1(0) + 5$ (Case 1). Now we modify $L_2(0) := 5, S_2(0) := 1, C_2(0) := -1$ and $i := 2$ and continue using the DFS strategy.

Starting now from V_2^D we find the arc (2, 4) (again Case 1) and put $L_4(0) := 8, S_4(0) := 2, C_4(0) := -1$ and $i := 4$.

For the next dual arc (4, 6) we put $L_6(0) := 10, S_6(0) := 4, C_6(0) := -1$, $i := 6$.

Now we find the backward dual arc (5, 6) (Case 2) and modify $L_5(0) := L_6(0) - a_{56} = 10 - 0 = 10$, $S_5(0) := -6$, $C_5(0) := -1$ and $i := 5$.

The next dual arc (3, 5) is also a backward one (again Case 2). We put $L_3(0) := 10, S_3(0) := -5, C_3(0) := -1$ and $i := 3$.

The only arcs adjacent to V_3^D are (3, 2) and (1, 3) but neither Case 1 nor Case 2 is fulfilled. We go back to V_5^D, i.e. $i := 5$, $C_3(0) := 1$ and investigate the arc (5, 4). Again neither Case 1 nor Case 2 is fulfilled; back to V_4^D, V_2^D, V_1^D and $C_4(0) := 1$, $C_2(0) := 1, C_1(0) := 1$.

For $i := 1$ we find as the next dual arc (1, 3) (Case 1) and modify $L_3(0) := L_1(0) + 3 = 3, S_3(0) := 1, C_3(0) := -1$. Then: $L_5(0) := 7, S_5(0) := 3, C_5(0) := -1, i := 5,$ back to V_3^D and $C_5(0) := 1, i := 3, C_3(0) := 1, i := 1$.

Since no changes are possible - there are three directions: (1, 2), (1, 3) and (6,1), we put $i := S_1(0) = M$ and the while loop has been completed for $K = 0$. As a result we have one shortest chain in $D(G)$ from V_1^D to V_6^D with nodes and arcs: V_6^D, (4, 6), V_4^D, (2, 4), V_2^D, (1, 2), V_1^D, since $S_6(0) = 4, S_4(0) = 2, S_2(0) = 1$. This chain corresponds to a minimum cut in G such as (1, 2), (1, 3), (1, 4) with a minimum capacity of 10 units, which is $L_6(0) = 10$.

Second step $(K = 1)$**.** Initialization: the same, $i := 1$.

The DFS strategy investigates sequentially the dual nodes V_i^D for $i = 1, 2, 4,$ 6, 5, 3, 5, 6, 4, 5, 3, 5, 4, 2, 1, 3, 1. The labels are modified as follows: $L_2(1) := 5$ and $S_2(1) := 1$ according to (*1*); $L_2(1) := 0$, $S_2(1) := 1$ and $F_2(1) := \text{true}$ according to (*2*); $i := 4, C_4(1) := -1, L_6(1) := 5$ and $S_6(1) := 4$ according to (*1*); $i := 6$ and $S_6(1) := -1$.

Now we have Case 2 for the dual arc (6, 5): $L_5(1) := 5$ and $S_5(1) := -6$ according to (*3*); $i := 5$ and $C_5(1) := -1$.

The next arc (3, 5) is also a backward one: $L_3(1) := L_5(1) - 0 = 3$, $S_3(1) := -5$. Since neither jumping to V_2^D nor to V_1^D is profitable we have to go back to V_5^D, etc.

After ten jumps this step finishes with the result: $L_1(1) := 0$, $L_2(1) := 0$, $L_3(1) := 0$, $L_4(1) := 3$, $L_5(1) := 3$, $L_6(1) := 5$, $S_1(1) := M$, $S_2(1) := 1$, $S_3(1) := 1$, $S_4(1) := 2$, $S_5(1) := 4$, $S_6(1) := 4$, $F_1(1) = F_3(1) = F_4(1) = F_6(1) := \text{false}$, $F_2(1) = F_5(1) := \text{true}$, $C_j(1) := 1$ for $j = 1, \dots, 6$.

In this step we have found one most vital arc - it is (1, 3) and it corresponds to (1, 2) in the dual graph, since the shortest reduced chain in $D(G)$ is: V_6^D, (4, 6), V_4^D, (2, 4), V_2^D, (1, 2), V_1^D and the only vertex with $F_j(1) = \text{true}$ is V_2^D. If we remove (1, 2) from G the maximum flow value in the reduced network will be $L_6(1) = 5$ units.

Third step $(K = 2)$**.** The DFS strategy investigates V_i^D for i = 1, 2, 4, 6, 5, 3, 5, 6, 4, 2, 3, 5, 3, 2, 1. The final result is: $L_j(2) = 0$ for $j = 1, \dots, 5$, $L_6(2) = 2$, $S_1(2) = M$, $S_2(2) = 1$, $S_3(2) = -2$, $S_4(2) = 2$, $S_5(2) = 3$, $S_6(2) = 4$, $F_j(2) = \text{false}$ for $j = 1,3,6$, $F_j(2) = \text{true}$ for $j = 2,4,5$, all $C_j(2)$ are 1. The interpretation is the following: if one drops two arcs from G the maximum flow value will be reduced to $L_6(2) = 2$ units; the shortest chain in the reduced graph $D(G)$ is V_6^D, (4, 6), V_4^D, (2, 4), V_2^D, (1, 2), V_1^D which corresponds in G to the cut set {(1, 2), (1, 3), (1, 4)}; the arcs in G^D whose length have to be reduced to zero are (2, 4) and (1, 2) since $F_4(2) = F_2(2) = \text{true}$. They correspond in G to (1, 3) and (1, 4) which are the two most vital arcs in G.

Example 2. Sometimes if we remove some arcs from G it is possible to make the constraints (2) and (3) contradictable. This is the case in this example for the graph in Fig. 2 if we remove one arc.

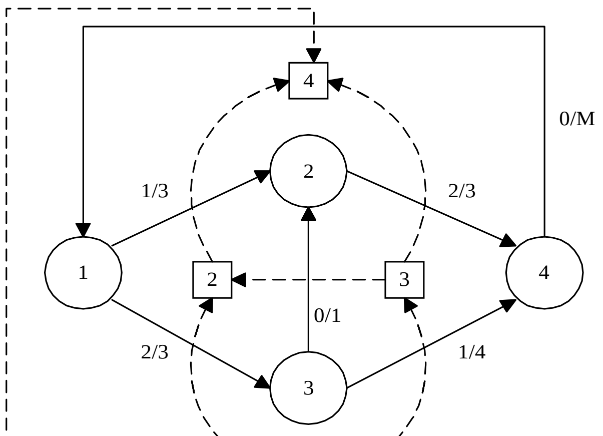

Figure 2.

The **First step** $(K=0)$ finishes with the result: $L_1(0)=0$, $L_2(0)=3$, $L_3(0)=3$, $L_4(0)=6$, $S_1(0)=M$, $S_2(0)=1$, $S_3(0)=-2$ and $S_4(0)=2$. $F_j(0)$ and $C_j(0)$ are not interesting in this example.

First step $(K=1)$. After initialization we put $i=1$. The DFS investigates the dual nodes with numbers 1, 2, 4, 3, 4, 2, 3, 1 and stops the execution of the algorithm since $C_1(1)=-1$. The final state of the labels is: $L_1(1)=-1$, $L_2(1)=L_3(1)=0$, $L_4(1)=3$, $S_1(1)=-3$, $S_2(1)=1$, $S_3(1)=-2$, $S_4(1)=2$, $F_2(1)=\text{true}$, $F_1(1)=F_3(1)=F_4(1)=\text{false}$, $C_1(1)=C_2(1)=C_3(1)=-1$, $C_4(1)=1$.

Since we have reached V_1^D again and $C_1(1)<0$, a negative cycle has been found in $D(G)$ after reducing some arc lengths to zero. Its arcs are in the area Cycle and the reduced arc from $D(G)$ is in List. We put $\text{start}:=i=1$ and it follows the execution of the repeat cycle. Finally it becomes $\text{Cycle}=((1, 3), (3, 2), (1, 2))$ and List = {(1, 2)}, i.e. if we drop (1, 3) in G (or reduce the length of (1, 2) in $D(G)$ to zero) the length of the cycle in $D(G)$ will be negative. The equivalent in G is that the cocycle CC_3 does not fulfil the conditions of Hoffman's circulation theorem. It implies a contradictability of (2) and (3).

4. JUSTIFICATION OF THE ALGORITHM

Theorem 2. The algorithm MVAPDN finds the k most vital arcs in a planar directed network correctly with time complexity $O(knm)$.

Proof: We form list for the dual graph with the algorithm from [6]. It takes $O(m)$ operations in the worst case ([6], Theorem 2). Then we start investigating the shortest chains from s^D to V_r^D according to the Corrolary. Induction on K follows.

At each iteration of the main loop we use the DFS strategy for searching in graphs which needs $O(m)$ operations. At the first iteration $(K=0)$ we find the shortest (s^D-t^D) chain with $O(nm)$ operations since some lengths are negative (see [8], p. 95). At each next iteration (for $K\geq 1$) we check in addition the conditions (*2*) or (*4*) in order to find possibly better combinations: let $L_j(K-1)$ be the shortest distance from s^D to V_j^D if we reduce to zero the lengths of $K-1$ arcs; conditions (*2*) and (*4*) mean that it would be better to reduce to zero the length of the last arc of the chain found from s^D to V_j^D (i.e. to remove the corresponding arc from G). It takes again $O(nm)$ operations to perform the main loop from the beginning to (*5*) after K iterations and $L_j(K)$ are the desired distances, $j=1,\dots,r$. The total complexity becomes $O(knm)$ for $K=0,1,\dots,k$. Obviously, the begin-end block after

(*5*) and repeat loop will be executed only once with time complexity $O(n)$ and it does not change the total complexity. ❑

If we develop a computer code for this algorithm we have to memorize $3(k+1)n$ integers and $(k+1)n$ logical variables for the node labels. It is possible to reduce this if we memorize the information from two neighbouring iterations of the main loop (for $K=1$ and for K only). If for some k in the last repeat loop the logical variable $F_j(k)$ is true we have to start the main loop again from $K=0$ to the current k and to look for the shortest chain from s^D to V_j^D. This modification of the algorithm needs obviously $O(k^2nm)$ operations in the worst case and to memorize $5n$ integers and n logical variables. This version of the algorithm follows below.

4.1. The modified algorithm MVAPDN

1. Initialization - the same as of MVAPDN;
2. Perform the main loop of MVAPDN with a given k and memorize $L_j(K)$, $S_j(K)$, $F_j(K)$, $C_j(K)$, $L_j(K-1)$ and $S_j(K-1)$ at each iteration after the first one for the current K in order to check (*2*) and (*4*); put $j:=r$;
3. Perform the last repeat loop and stop it if $j=M$ or k has been reduced to $k-1$; put $r=s$; if $j \neq M$ then go to 2. Else STOP.

The previous comments prove the following theorem.

Theorem 3. The Modified algorithm MVAPDN finds the k most vital arcs in a planar directed network correctly with time complexity $O(k^2nm)$ and needs to memorize $O(n)$ variables.

5. MOST VITAL NODES

The problem can be formulated analogously: which are those k nodes in the network the removal of which minimizes the maximum flow value of the network?

The problem can be reduced to the previous one if we use the well-known network transformation: to each vertex V_i from G we assign a new one V_i' and a new arc (i,i') from G' with arc weights (lower and upper bounds) $0/M$. The arc (i,j) from G corresponds to (i',j) from G'. The node V_i from G corresponds to the arc (i,i') from G'. Now to removing the node V_i from G is equivalent to removing the arc (i,i') from G' since it is the only outgoing arc from V_i and the only ingoing one to V_i'. Hence, finding the k most vital nodes in G is equivalent to finding the k most vital arcs in G'. Although G' has $2n$ nodes and $m+n$ arcs the complexity of the algorithms is the same - $O(knm)$ and $O(k^2nm)$, respectively.

6. MULTICRITERIAL MOST VITAL ELEMENTS

This is the case if we have to remove k arcs or/and nodes, but if we drop A_p or V_p we have to "pay" d_p units. This leads to a bicriterial optimum path problem in the dual network in the lexicographical sense. We denote by

$$d(DA^k) := \sum \{d_p / A_p^D \in DA^k \subset A^D, |DA^k| = k\}$$

and the corresponding problem is to find some solution DA^k of $(MVDA^k)$ which is the "cheapest" one, i.e. which minimizes $d(DA^k)$.

In order to solve this version of the problem we can combine the algorithms from the previous sections and the procedure from [7] (see [7], p. 83-85). For this purpose we denote by $D_j(K)$ the value of the "cheapest" path from s^D to V_j^D if K of their corresponding arcs have to be removed. The only changes in the algorithm are:

Initialization: $D_1(K) := 0, D_j(K) := M$ for all $j = 2, \dots, r$ and $K = 0,1,\dots,k$;

the begin-end block containing (*1*) and (*2*) is:

```
        begin
(*1'*)      if $(L_j(K), D_j(K)) \succ (L_i(K)+b_p, D_i(K))$ then
          begin $L_j(K) := L_i(K)+b_p$; $D_j(K) := D_i(K)$; $S_j(K) := i$ end;
(*2'*)      if $K > 0$ and $(L_j(K), D_j(K)) \succ (L_i(K-1), D_i(K-1)+d_p)$ then
          begin $L_j(K) := L_i(K-1)$; $D_j(K) := D_i(K-1)+d_p$; $S_j(K) := i$; $F_j(K) :=$ true end
        end;
```

Analogously the begin-end block containing (*3*) and (*4*) should be:

```
        begin
(*3'*)      if $(L_j(K), D_j(K)) \succ (L_i(K)-a_p, D_i(K))$ then
          begin $L_j(K) := L_i(K)-a_p$; $D_j(K) := D_i(K)$; $S_j(K) := -i$ end;
(*4'*)      if $K > 0$ and $(L_j(K), D_j(K)) \succ (L_i(K-1), D_i(K-1)+d_p)$ then
          begin $L_j(K) := L_i(K-1)$; $D_j(K) := D_i(K-1)+d_p$;
              $S_j(K) := -i$; $F_j(K) :=$ true end
        end.
```

Here we use the sign $\succ$ for the relation "lexicographically greater".

Analogously we can adapt the procedure for the case of more then two criteria even if they are from another type (see [7] and [1]).

7. CONCLUSIONS

In this paper we give two justificated algorithms for finding the k most vital arcs/nodes of s-t planar directed network if it is directed one and there are non-zero lower bounds of the arc capacities. For the purpose we use the concept for duality of directed planar graphs from [5] and [6]. In case there are more then one criteria we use the concept from [7] for multicriteria path optimization. In the dual graph we use a modification of the searching strategy DFS which deals with positive and negative arc lengths.

Our very last computer experiment shows that the main loop of the algorithm MVAPDN runs very fast. For 10 randomly generated networks with 100 nodes and 10000 (one thousand) arcs it takes less then one second.

REFERENCES

[1] Ivanchev, D., and Ruhe, G., "Finding the k bicriterial foremost edges in planar flow networks", Comptes rendus de l' Academie bulgare des sciences, Sofia, 39/9 (1986) 35-38.

[2] Ivanchev, D., "Establishment of foremost nodes in a network", Comptes rendus de l' Academie bulgare des sciences, Sofia, 38/11 (1985) 1469-1472.

[3] Ivanchev, D., "Finding the foremost nodes in parametric capacited network", University Annual (Applied Mathematics), Sofia 22/2 (1986) 151-165.

[4] Ivanchev, D., "Finding the foremost arcs in a network with parametric arc capacities", Optimization, Berlin, 16/6 (1985) 908-919.

[5] Ivanchev, D., "Sensitivity analysis of network optimization problems", YUJOR, 5/1 (1995) 95-109.

[6] Ivanchev, D., "Duality of s-t planar directed graphs and applications", Proceedings of the 4-th Balkan Conference on Operational Research, Thessaloniki, Greece (1997) - to appear.

[7] Ivanchev, D., and Kydros, D., "Multicriteria optimum path problems, YUJOR, 5/1(1995) 79-93.

[8] Naegler, G., and Stopp, F., Graphen und Anwendungen, Teubner Verlaggeselschaft, Stuttgart, Leipzig, 1996, S.192.

[9] Ratliff, D., Sicilia, T., and Lubore, H., "Finding the n most vital links in flow networks", Management Science, 21/5 (1975) 531-539.

[10] Wollmer, R., "Removing arcs from a network", Operations Research, 12/6 (1964) 934-940.