Yugoslav Journal of Operations Research
5 (1995), Number 1, 95–109

SENSITIVITY ANALYSIS OF NETWORK OPTIMIZATION PROBLEMS

Dimiter IVANCHEV

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

Abstract: Sensitivity analysis deals with the problem of finding an optimum solution of a given problem on a network if some input data are not known or can be changed. Such problems arise for example by laying out a network, by dropping out some network elements, by changing the resources of the economical situation etc. We present some of our results on sensitivity analysis of network flow and network connectivity problems like maximum flows, shortest paths, minimum spanning trees and most vital links and nodes.

Keywords: Network design, parametric analysis, k most vital elements, complexity.

1. INTRODUCTION

In the real world network problems some of input data are not fixed or not well known. One can look for the optimum solution but how to find it among all feasible solutions if their set is unknown because of uncertainty or missing information. If we know the boundaries of changing of some parameters and if we allow the parameters to vary between them it will be a class of problems of certain type and obviously the desired optimum solution will also vary in dependence of all input data changes. Of course one can say why to find an optimum solution if there are some uncertainties or if there is not enough information, why not to obtain information first and then to solve the well known problem? The reason is that if we know in advance what the optimum solution will be for some input data we will know the consequences and if they are not satisfactory we can avoid the situation choosing another input data. For instance, why to lay out the traffic system in a city first and after seeing that it is not good enough to reconstruct it instead of calculating first its capacity per time unit in dependence of all link widths and choosing them large enough (but not too large because of expenses) in order to obtain the desired properties of the network.

On the other hand topological changes are also possible. What will happen if some links or nodes of the network are damaged or switched off? Which k links (or nodes) of the network to switch off in order to reduce the network properties we are interested in to the necessary level? Or just the opposite: how to design the network so that the removal of a certain number of its elements causes minimum destruction of the network in the worst case?

Sensitivity analysis of networks deals with such a kind of questions. We start with some notations and definitions. Let $G = (V, A)$ be a connected directed graph with a set of nodes $V = \{ V_1, \ldots, V_n \}$ and a set of arcs $A = \{ A_1, \ldots, A_m \}$. Each arc A_k can be presented as (V_i, V_j) or simply (i,j). We exclude parallel arcs and loops without loss of generality. We assume depending on the considered problem, some arc weights like capacity, length etc. Sometimes we shall denote the arc A_k also by A_{ij}. For each problem we give one Pascal–like algorithm for solving it. For paper length reasons, we give up illustrative examples and figures.

2. THE MAXIMUM FLOW PROBLEM IN NETWORK WITH GAINS (P1)

Let us assume two special nodes V_s and V_t called start node and target node, respectively, and three arc weights c_{ij}, d_{ij} and g_{ij} for each arc A_{ij}. A flow in a parametric capacity bounded network with gains is a vector x with coordinates corresponding to the arcs, such that

$$\sum_{(i,j)\in A} x_{ij} - \sum_{(k,i)\in A} g_{ki}\, x_{ki} = \begin{cases} 0 & \text{for } i \neq s,t \\ v_s(x) & \text{for } i = s \\ -v_t(x) & \text{for } i = j \end{cases} \qquad (1)$$

$$0 \le x_{ij} \le c_{ij} + \lambda\, d_{ij} \text{ for all arcs } (i,j), \qquad (2)$$

where λ a real number is varying in a parameter set Λ (let us assume without loss of generality $\Lambda = [0, \lambda_r]$). $v_s(x)$ and $v_t(x)$ are called start flow value and target flow value, respectively. The problem we are interested in is to find a flow with a maximum target value $v_t(x)$ and since there are maybe many flows with this property among all of them to find one with a minimum start value $v_s(x)$. Such a flow is called sometimes t–maximal and s–optimal. The motivation for this formulation is, for example, the currency problem: each node presents one currency, g_{ij} is the rate from V_i to V_j; which is the minimum amount of the start currency V_s and how to exchange it in order to maximize the amount $v_t(x)$ of the desired target currency V_t?

The mathematical formulation of the problem is:

$$\text{lexmax } \{ (v_t(x), -v_s(x)) \mid (1) \text{ and } (2) \text{ for all } \lambda \text{ from } \Lambda \}, \qquad (P1)$$

where lexmax means that a lexicographical maximum ($v_t(x)$ is more important than $-v_s(x)$).

Let C be a cycle in G with a fixed direction. We denote by C^+ and C^- the sets of arcs from C with positive and negative direction respectively. Let x be a feasible flow for a fixed λ. C is called x, t–generating (x, s–generating) cycle, if (i), (ii) and (iii) hold:

(i) $g(c) := \prod_{(i,j)\in C^+} g_{ij} / \prod_{(i,j)\in C^-} g_{ij} > 1$,

(ii) $((i,j) \in C^+ \Rightarrow x_{ij} < c_{ij} + \lambda d_{ij}) \wedge ((i,j) \in C^- \Rightarrow x_{ij} > 0)$,

(iii) there exists a flow augmenting path from some node of C to V_t (to V_s).

THEOREM 1. x is an optimum solution of (P1) for some fixed λ if and only if there exists a V_s and V_t separating cut (X, Y) such that 1) and 2) hold:

1) $((i,j) \in (X,Y) \Rightarrow x_{ij} = c_{ij} + \lambda d_{ij}) \wedge ((i,j) \in (Y,X) \Rightarrow x_{ij} = 0)$,

2) there is no x,t–generating cycles in Y and no x,s–generating cycles in X.

This theorem summarizes Theorems 1 and 2 from [5].

Let x be an optimum solution of (P1) for any fixed λ_0 and let M be a sufficiently large number. We define for each arc (i,j)

$$\Delta l_{ij}^0 := \begin{cases} 0 & \text{if } x_{ij}^0 = 0 \\ -M & \text{if } x_{ij}^0 > 0 \end{cases} \tag{3}$$

$$\Delta c_{ij}^0 := \begin{cases} d_{ij} & \text{if } x_{ij}^0 = c_{ij} + \lambda_0 d_{ij} \\ M & \text{if } x_{ij}^0 < c_{ij} + \lambda_0 d_{ij} \end{cases} \tag{4}$$

We define the analogous of restriction (2)

$$\Delta l_{ij}^0 \le x_{ij} \le \Delta c_{ij}^0 \qquad \text{for each arc } (i,j) \tag{2'}$$

To find an optimum solution of (P1) for $\lambda > \lambda_0$ we investigate the derived problem

$$\min \{ v_s(\Delta x) \mid (1), (2'), \Delta x \text{ is } t\text{–maximal} \}, \tag{dP}$$

which is for a well known type (no parameter λ) and can be solved for instance using the results from [5].

Assuming a feasible flow Δx^o of (dP) for a fixed λ_0 we partition the set of arcs A into four disjunct subsets $E_1 - E_4$ as follows:

$$E1 := \{ (i,j) \mid \Delta x_{ij}^0 \ge 0, \Delta x_{ij}^0 > d_{ij} \}, \quad E2 := \{ (i,j) \mid \Delta x_{ij}^0 < 0, \Delta x_{ij}^0 \le d_{ij} \}$$

$$E3 := \{ (i,j) \mid \Delta x_{ij}^0 < 0, \Delta x_{ij}^0 > d_{ij} \}, \quad E4 := \{ (i,j) \mid \Delta x_{ij}^0 \ge 0, \Delta x_{ij}^0 \le d_{ij} \}$$

THEOREM 2. [7] Let x_o be an optimal solution of (P1) for any fixed λ_0 and Δx^o an optimal solution of (dP). Then $x^o + \Delta\lambda \Delta x^o$ is an optimal solution of (P1) for all $\Delta\lambda$ from the interval $[0, \Delta\Lambda]$, where

$$\Delta\Lambda := \min_{k=1,2,3} \min_{(i,j)\in E_k} \{ Q_k^0(i,j) \} \tag{5}$$

$$Q_1^0(i,j) := (c_{ij} + \lambda_0 d_{ij} - x_{ij}^0) / \Delta x_{ij}^0 - d_{ij}) \tag{6}$$

$$Q_2^0(i,j) := -x_{ij}^0 / \Delta x_{ij}^0 \tag{7}$$

$$Q_3^0(i, j) := \min(Q_1^0(i, j), Q_2^0(i, j)) \tag{8}$$

This theorem proves the following algorithm for solving (P1).

ALGORITHM (A1)

Put $\lambda_0 := 0$; $k := 1$; *Nofeasible* := false;
repeat
  x^0 – solution of (P1) for fixed $\lambda = \lambda_0$;
  if (there is not feasible solution of (dP) for λ_0) then
    begin *Nofeasible* := true; $\lambda_0 := \lambda_T$ end
  else
    begin
      denote by Δx^0 a feasible solution of (dP);
      compute $\Delta\Lambda$ according to (5) – (8);
      $x^k := x^0 + \Delta\Lambda\,\Delta x^0$; $k := k + 1$;
      $\lambda_0 := \min(\lambda_r, \lambda_0 + \Delta\Lambda)$
    end
until $\lambda_0 = \lambda_r$.

This algorithm partitions $[0, \lambda_r]$ into subintervals and finds for the k–th of them an optimum solution x^k of (P1). The optimum target flow value function $v_t(x)$, $k = 1, 2, \ldots$ is obviously convex and piece–wise linear in dependence on λ. If at any iteration the logical variable *Nofeasible* becomes true it means that the restrictions (1) and (2') are in contradiction on the rest of the interval.

In [11] we have applied this approach in the case if some target nodes with stochastic demands are given. Assuming that the i–th demand node V_i has a known probability p_i^j to demand j units we add a new artificial demand node V_t and the arc (i, t). Then the following condition (9) will be a measure of the satisfaction of all stochastic demands:

$$\sum_{(i,t)\in A} \sum_{j} (x_{it} - j)^2 p_i^j \to \min \tag{9}$$

and the problem is

$$\min \{ v_s(x) \mid (1), (2), (9) \}.$$

The algorithm for solving it is based on the results from [1] and [11].

2. SHORTEST PATH PROBLEM (P2)

Sometimes it is necessary to use the shortest path problem as a subroutine in order to solve another one. If it is a parametric network problem then the question arises how to find the shortest path between a special node called start point (V_1 or V_s) and all other nodes reachable from it.

Now we assume that for each arc $A_k = (i,j)$ the arc length $l_k(\lambda) = l_{ij}(\lambda)$ is known for all λ from Λ, where the functions $l_k(\lambda)$ are continuous and non negative.

It is easy to adapt the Dijkstra's algorithm for finding the shortest routine between two vertices for this case. We also assign two labels $(L_i(\lambda), S_i)$ to each vertex V_i: $L_i(\lambda)$ is the temporal length of the "best" path from V_1 to V_i for the current iteration and S_i is the number of the last arc of this path. When this label becomes final we denote it by $(L_i^*(\lambda), S_i^*)$.

ALGORITHM (A2)

Put $L_1^*(\lambda) := 0$ for all λ; $S_1^* := m+1$; $k := 0$;
repeat (* until $\Lambda = \varnothing$ *)
$k := k + 1$;
pick any λ_0 such that for small $\varepsilon > 0$ either $\lambda_0 + \varepsilon$ or $\lambda_0 - \varepsilon$ belongs to Λ;
(* if both put first $\lambda_0 := \lambda_0 + \varepsilon$*); $i := 1$;
put $L_j(\lambda) := 0$ and $S_j := m + 1$ for all $j = 2, 3, \ldots, n$ and *break* := false;
repeat (* until break *)
$\Lambda_k := \Lambda$;
for each arc $A_k = (i,j)$ with temporal label S_j investigate :
if $L_i^*(\lambda_0) + c_k(\lambda_0) < L_j(\lambda_0)$ then
begin
denote by Λ_0 the solution set of the inequality
(*) $L_i^*(\lambda) + c_k(\lambda) < L_j(\lambda)$;
put $L_j(\lambda) := L_i^*(\lambda) + c_k(\lambda)$; $S_j := k$; $\Lambda_k := \Lambda_k \cap \Lambda_0$
end;
if all nodes are finally labelled then *break* := true else
begin
find a node V_p with the smallest temporal label $L_p(\lambda_0)$
and denote by Λ_0 the solution set of
(**) $L_p(\lambda) \le L_q(\lambda)$;
$\Lambda_k := \Lambda_k \cap \Lambda_0$;
if $S_p = m + 1$ then *break* := true
(*some nodes are not reachable from the start point *)
else begin label V_p final by $L_p^*(\lambda)$ and S_p^*; $i := p$ end
end
until break;
$\Lambda := \Lambda \setminus \Lambda_k$
until $\Lambda = \varnothing$.

REMARK. If in (**) the equality holds for some indices take that p for which the solution set is "largest" in order to make the iterative procedure faster.

THEOREM 3. The algorithm (A2) solves the problem (P2) for all parameter values in a finite number of steps.

PROOF. At each iteration of the first repeat loop the algorithm finds a tree with a root in the start point containing one path to each reachable node which is the shortest one for all λ from Λ_k (Dijkstra's algorithm for a fixed λ). Since Λ_k is the solution set of the system (*) and (**) this tree will not appear any more at any further iteration. The finiteness of (A2) follows from the finiteness of the number of trees in the graph with this property.

3. OPTIMUM SPANNING TREE RELATED PROBLEMS (P3)

We begin this section with an example. Given are four points in the plane with their coordinates as follows: $P_1(0, -1)$, $P_2(2, -1)$, $P_3(1, 1)$ and $P_4(2, 2)$. Find a point P_5 on the straight line connecting the points $(0, 0)$ and $(2, 0)$ and link it with all other points so that the total Euclidean length of the line be minimal. The line can fork only in the points $P_1 - P_5$.

If we link each two points we can find the lengths of the edges in the complete graph, i.e. the distance between P_1 and P_2 is 2, between P_1 and P_3 is $5^{1/2}$ but since the coordinates of P_5 are $(\lambda, 0)$ for λ from $[0, 2]$ the distance between P_1 and P_5 will be $(\lambda^2+1)^{1/2}$. In this way we obtain the problem for finding a minimum spanning tree (MST) in a graph with edge lengths depending on a parameter.

We consider an undirected connected graph $G = [V, A]$ with edge weights $c_k(\lambda)$ as continuous functions of λ from Λ for all $k = 1, \ldots, m$. Let $\Lambda = (-\infty, \infty)$ without loss of generality. The problem is to find for each λ a minimum spanning tree $G^* = [V^*, A^*]$ of G. We call it (P3). This problem is related to some other connectivity problems. Let V' be a given node subset. If the desired minimum subgraph of G has to contain V' and if each path in G can fork only in nodes from V', we have the so called modified minimum spanning tree problem. We get (P3) from this problem formulation if we put $V' = V$. If each path in G^* can fork everywhere we obtain the Steiner tree problem in a graph with basic nodes V' and Steiner points set the rest of the nodes. If $|V'| = 2$ we have the shortest path problem discussed in the previous section as (P2).

Let us consider first the linear case for $c_k(\lambda)$, i.e. let

$$c_k(\lambda) = a_k\lambda + b_k \text{ for each edge } A_k, \qquad k = 1, \ldots, m. \tag{10}$$

We order the edges like $A_{k_1}, A_{k_2}, \ldots, A_{k_m}$ so that for each $j = 1, 2, \ldots, m-1$ the following relation will be fulfilled

$$(a_{k_j} > a_{k_{j+1}}) \vee (a_{k_j} = a_{k_{j+1}} \wedge b_{k_j} \le b_{k_{j+1}}). \tag{11}$$

It follows from (10) and (11) that

$$c_{k_1}(\lambda) \le c_{k_2}(\lambda) \le \ldots \le c_{k_m}(\lambda) \tag{12}$$

for all λ from $(-\infty, \lambda_0]$ for any λ_0 holds.

LEMMA 1. Let k be an integer with $1 < k < m$ and λ, μ from Λ, such that

$$c_1(\lambda) \le c_2(\lambda) \le \ldots \le c_i(\lambda) \le c_{i+1}(\lambda) \le \ldots \le \\ \le c_{i+k}(\lambda) \le c_{i+k+1}(\lambda) \le \ldots \le c_m(\lambda) \tag{13}$$

and

$$c_1(\mu) \le c_2(\mu) \le \ldots \le c_i(\mu) \le c_{j_1}(\mu) \le \ldots \le \\ \le c_{j_k}(\mu) \le c_{i+k+1}(\mu) \le \ldots \le c_m(\mu) \tag{14}$$

where $j_1, j_2, \ldots j_k$ is a permutation of $i+1, i+2, \ldots, i+k$. Let us denote by G_1^* and G_2^* the minimum spanning trees of G founded by the greedy algorithm of Kruskal for the edge weights (13) and (14) respectively. Then for all p with $0 < p < i+1$ or $i+k < p < m+1$ the edge A_p belongs (or does not belong) to both spanning trees G_1^* and G_2^*.

PROOF. Let us note how the algorithm of Kruskal runs: it orders first the edges according to their weights like (12) or (13) and starting from the edge with the smallest weight (in this case A_1) it adds it to G^* if and only if G^* will be an acyclic subgraph of G. In this way G^* is a forest growing iteratively and the algorithm stops if G^* becomes a tree. If we stop the algorithm earlier i.e. after checking the first $i+k$ edges in (13) or (14) we shall obtain two spanning forests G_1^* and G_2^* with p components of connectivity (trees) and $n-p$ edges. Let us assume that A_r is the first edge in (13) or (14) with $r > i+k$ such that A_r belongs to G_1^* but not to G_2^*. If we now drop all edges in (13) and (14) after A_r then G_1^* will obtain $n-p+1$ edges but G_2^* one less – $n-p$, which is a contradiction since G_1^* and G_2^* are spanning forests of the graph $[V, \{A_1, \ldots, A_r\}]$ and they should have the same number of links.

Using Lemma 1 we can make the procedure given below much faster.

ALGORITHM (A3)

Order $c_i(\lambda)$ according to (11) and (12);
$i := 1$; $\lambda_i = -\infty$; $\lambda' := -\infty$; denote by G_i^* the MST for fixed λ near λ_i;
repeat (* until $\lambda'' = \infty$ *)
solve the system (12) for $\lambda \ge \lambda'$ and denote the solution by $[\lambda', \lambda'']$;
if $(\lambda'' < \infty)$ and $c_{k_j}(\lambda'') < c_{k_{j+1}}(\lambda'') = c_{k_{j+2}}(\lambda'') = \ldots = c_{k_{j+p}}(\lambda'') < c_{k_{j+p+1}}(\lambda'')$
then
begin
invert the places of $c_{k_{j+1}}, c_{k_{j+2}}, \ldots, c_{k_{j+p}}$ in (12) like
$c_{k_{j+p}}, c_{k_{j+p-1}}, \ldots, c_{k_{j+1}}$;
$A_{i+1}^* := A_i^*$; drop $A_{k_{j+1}}, A_{k_{j+2}}, \ldots, A_{k_{j+p}}$ from A_{i+1}^*;
for $l := j+p$ downto $j+1$ do
if $[V, A_{i+1}^* \cup \{A_{k_l}\}]$ is cycleless then $A_{i+1}^* := A_{i+1}^* \cup \{A_{k_l}\}$;
if $A_{i+1}^* \ne A_i^*$ then
begin denote $[V, A_{i+1}^*]$ by G_{i+1}^*; $i := i+1$; $\lambda_i := \lambda''$ end;

$\lambda' := \lambda''$

end

else put $\lambda_{i+1} := \infty$

until $\lambda'' = \infty$.

THEOREM 4. Let $G_1^*, G_2^*, \ldots, G_r^*$ be the set of subgraphs constructed by (A3) and $[\lambda_1, \lambda_2], \ldots, [\lambda_r, \lambda_{r+1}]$ the founded intervals. Then for each λ from $[\lambda_i, \lambda_{i+1}]$ G_i^* is a MST of G, where $\lambda_1 = -\infty$ and $\lambda_{r+1} = \infty$.

PROOF. Obviously G_1^* is a MST for all λ from $[\lambda_1, \lambda_2]$, since the order of the edge weights in (12) is the same at least for the left part of the interval. For the rest of it the edge order in (12) can be changed but no other MST has been found. According to the Lemma 1 G_j^* and G_{j+1}^* do not differ for the first i edges and for the last $m-i-p$ edges so that the next MST G_{j+1}^* is correctly found using G_i^* and the edges which differ in the order for G_i^* and G_{i+1}^*.

The finiteness of the interval number r follows from the finiteness of the cross points of the graphs of the functions $c_k(\lambda)$, $k = 1, \ldots, m$. The procedure starts with putting $\lambda_1 := -\infty$ and stops when $\lambda_{r+1} = \infty$, it means the intervals cover $(-\infty, \infty)$.

In the case if $c_k(\lambda)$ are continuous but not linear functions some changes in (A3) have to be made. We have to start with some λ_1 and G_1^* but the solution set Λ_1 of (12) cannot be an interval. Then we put $\Lambda := \Lambda \setminus \Lambda_1$, pick any λ_2 from Λ and find the next MST G_2^* and continue the procedure until $\Lambda = \varnothing$.

For the example we started this section with we can do the following: let us denote the edges like $A_1 = [P_1, P_2]$, $A_2 = [P_2, P_4]$, $A_3 = [P_3, P_4]$, $A_4 = [P_1, P_3]$, $A_5 = [P_1, P_5]$, $A_6 = [P_2, P_5]$, $A_7 = [P_4, P_5]$ and $A_8 = [P_3, P_5]$. The edges lengths are $c_1 = 2$, $c_2 = 3$, $c_3 = 2^{1/2}$, $c_4 = 5^{1/2}$, $c_5 = (\lambda^2+1)^{1/2}$, $c_6 = ((\lambda-2)^2 + 1)^{1/2}$, $c_7 = ((\lambda-2)^2 + 4)^{1/2}$ and $c_8 = ((\lambda-1)^2 + 1)^{1/2}$ which is the nonlinear case for $\Lambda = [0, 2]$.

At the first iteration we find $G_1^* = [V, A_1^*]$ with $A_1^* = \{A_5, A_8, A_3, A_1\}$ for all λ from $[0, 2-3^{1/2}]$.

The second iteration gives an optimum solution with $A_2^* = \{A_5, A_8, A_3, A_6\}$ for all λ from $[2-3^{1/2}, 3^{1/2}]$ and the third iteration – a solution with $A_3^* = \{A_6, A_8, A_3, A_1\}$ for the rest part of the interval.

If we compare the total weights of G_1^*, G_2^* and G_3^* we will find the minimum of the objective function for $\lambda = 1$, which means that the optimum for P_5 is in the middle of the interval and to link P_5 with P_1, P_2 and P_3 and P_3 with P_4.

4. MOST VITAL ARCS AND NODES (P4)

In this section we shall consider the possible changes not only in the input data but in the graph topology as well. For example which will be the consequences in connection with the maximum flow in a capacity constrained network if k of its arcs have to be removed? Or, which are those k arcs in the network whose removal minimizes the value of the maximum flow? These arcs are called k most vital (or

foremost) arcs of the network. Such a problem is solved in [9]. In [2] we have solved the problem for a directed graph if the arc capacities depend linearly on a parameter. In [8] we have solved a similar problem if the graph is planar and not directed and if we are interested in two network properties: flow value and reliability of the network. In [3] and [4] we solve a similar problem when k nodes and their adjacent arcs have to be dropped.

In this section we consider again the case of a planar graph but a directed one with two arc weights – capacity and reliability (both depending on a parameter). The question is to find k arcs in dependence on the parameter the removal of which leads to minimization of the maximum flow value and to minimization of the probability of existence of at least one path between start and target node. The optimization is in a lexicographical sense and the first criterion has more importance. If we change the sign of the second criterion, the interpretation will be the following: we want to reduce the capacity of the network and for this purpose to determine the minimum number k of the arcs to be switched off in order to reduce the maximum flow value to the desired level so that the reduced network is maximally reliable.

Let $G = (V, A)$ be a directed connected planar graph with nonnegative arc weights $c_k(\lambda)$ and $r_k(\lambda)$, λ from Λ, $k = 1, \ldots, m$. We denote the start node and target node by V_s and V_t, respectively. The flow in G is defined, as in the first section, as a vector $x = (x_1, \ldots, x_m)$ which satisfies the flow preservation conditions

$$\sum_{(i,j)\in A} x_{ij} - \sum_{(k,i)\in A} x_{ki} = \begin{cases} 0 & \text{for } i \neq s,t \\ v(x) & \text{for } i = s \\ -v(x) & \text{for } i = j \end{cases} \qquad (15)$$

and the flow arc restrictions

$$0 \le x_k \le c_k(\lambda), \qquad k = 1, \ldots, m. \qquad (16)$$

We denote by $CS(A)$ the set of all cuts in $G = (V, A)$ of type (X, Y) which separate start and target node, i.e.

$$(X, Y) := \{ (i,j) \mid (V_i \text{ from } X \text{ and } V_j \text{ from } Y) \text{ or } (V_i \text{ from } Y \text{ and } V_j \text{ from } X) \},$$

so that the start node belongs to X and target node to Y. If $B = (X, Y)$ we denote by B^+ the set of arcs from B starting from X and by B^- the arc subset of B starting from Y. We define the capacity of B as

$$c(B) := \Sigma \{ c_k(\lambda) \mid A_k \text{ from } B^+ \}$$

and the reliability of B as

$$r(B) := \max \{ r_k(\lambda) \mid A_k \text{ from } B^+ \}.$$

The maximum flow problem is defined as $\max \{ v(x) \mid (15), (16) \}$ and it is well–known as a max–flow min–cut theorem, that is

$$max \{ v(x) \mid (15), (16) \} = \min \{ c(B) \mid B \text{ from } CS(A) \}.$$

If B and B' are two cuts in G we call B lexicographically smaller than B' if $c(B) < c(B')$ or $c(B) = c(B')$ but $r(B) \le r(B')$ holds.

Let now k be an integer with $0 < k < m$. We denote by ε_k an arbitrary arc subset

of G containing exactly k arcs. The problem estimated in this section is

$$\text{lexmin}\ \{\ (c(B), r(B))\ |\ B \text{ from } CS(A \setminus \varepsilon_k), \text{ for all } \lambda\ \}, \qquad \text{(P4)}$$

i.e. to find the arc set ε_k of those k arcs in G, the elimination of which reduces the network to minimum capacity network and after that to minimum reliability of the minimum cut. The arcs from ε_k are called k bicriterial most vital in the parametric network. Of course the solution set ε_k will depend on the parameter λ.

Without loss of generality we assume that G does not contain nodes with valency 2. Since the graph is planar, let us draw it in the plane and let us also assume that start and target nodes belong to the border of the not bounded facet of G. Now we can add one artificial arc (V_n, V_1) to G and construct the dual graph $GD = (VD, AD)$ with start point s and target point t belonging to the neighbour facets with a common border arc which is the artificial one. We do not link s with t. Since there is (1, 1)-correspondence between both arc sets A and AD and between the cut set $CS(A)$ of G and the set of all chains from s to t in GD now we shall be interested in its elements. We denote it by $PS(AD)$ (Path Set).

Let B be from $PS(AD)$ (i.e. a chain linking s with t) and let B correspond to $B = (X, Y)$ from $CS(A)$. We define the direction of each edge from B as a forward edge and on this way the subset B^+ of forward edges of B as follows: $[i, j]$ from B (and hence from GD) belongs to B^+ exactly if its corresponding arc from G belongs to B^+. The rest of the edges from B have the opposite direction by definition. They define the subset B^- of backward edges of B. Since GD is a connected graph each edge $[i, j]$ belongs to at least one chain from $PS(AD)$. In this way it becomes directed and will be called arc (i, j). Hence GD becomes a directed graph.

LEMMA 2. The direction of each arc in the dual graph GD does not depend on the chain that has been used.

PROOF. Let $[i, j]$ be an arbitrary edge from the dual graph. The question is how to direct it. We assume that $[i, j]$ corresponds to the arc A_k from G. For an arbitrary cut (X, Y) in G for which A_k starts from X $[i, j]$ will be directed as (i, j), it means it will be a forward arc in the chain of GD (starting from s to t we shall pass first V_i and then V_j). If we pick any cut (X, Y) for which A_k is a backward arc, (i, j) will be a backward arc in the corresponding chain in GD, i.e. starting from s to t we shall pass first V_j and after that V_i. It means that the direction is also from i to j.

It follows that the dual graph GD is a correctly defined planar directed one. Instead of investigating (P4) in G we can reformulate it in GD.

If B is from $PS(AD)$ and B is the corresponding cut from $CS(A)$ we define the length of the chain and its reliability as

$$c(B) := c(B) \text{ and } r(B) := r(B), \text{ respectively,}$$

it means we take only the forward arcs in B.

The reformulation of (P4) is

$$\text{lexmin}\ \{\ (c(B), r(B))\ |\ B \text{ from } PS(AD \setminus \varepsilon_k) \text{ for all } \lambda\ \}. \qquad \text{(P4')}$$

Let us partition AD as $AD = E_1 + E_2 + \ldots + E_p$ and let Λ^* be a subset of Λ with the property

$$(e, e' \text{ from } E_i \Rightarrow r(e) = r(e')) \text{ and } (e \text{ from } E_i, e' \text{ from } E_{i+1} \Rightarrow r(e) < r(e'))$$

for all λ from Λ^*. Let denote by $E^i = E_1 + \ldots + E_i$ for $i = 1, 2, \ldots, p-1$. Let ε_k^q with $|\varepsilon_k^q| = k$ be an arc subset of AD, such that if we reduce the arc lengths of ε_k^q to zero the network $(VD, E^q \cup \varepsilon_k^q)$ will have minimum length of the shortest chain from s to t. We denote also the set of all shortest chains by $SPS(\varepsilon_k^q)$ and for B_k^q from $SPS(\varepsilon_k^q)$ let $B_k^q := B_k^q \setminus \varepsilon_k^q$.

LEMMA 3. Let B_k^q be from $SPS(\varepsilon_k^q)$ and B_k^{q-1} be from $SPS(\varepsilon_k^{q-1})$ with

$$c(B_k^p) = c(B_k^{p-1}) = \ldots = c(B_k^q) < c(B_k^{q-1}).$$

Then ε_k^q is a solution of (P4') for all λ from Λ^*.

PROOF. For each fixed λ from Λ^* it follows from Lemma 1 in [8].

Now we can formulate the algorithm for solving (P4') (and hence (P4)). The arcs of GD will be denoted as in G like $A_1, \ldots, A_m$. To each node we assign a label $L(i, K, \lambda)$ which we call temporal. At some iteration it will be labelled finally with $L^*(i, k, \lambda)$, which is the length of the shortest chain from V_s to V_i for a fixed λ if the lengths of no more than k arcs can be reduced to zero.

ALGORITHM (A4)

```
repeat (* until Λ = ∅ *)
denote by Λ* a subset of Λ and the partition E_1, ..., E_p as in Lemma 3;
Λ := Λ \ Λ*;
repeat (* until Λ* = ∅ *)
    pick any λ_0 from Λ*; put Λ' := Λ'' := Λ*;
    L*(s, K, λ) := 0 and L(i, K, λ) := ∞ for all V_i ≠ s, K = 1, ..., k and all λ;
    z(λ) := ∞ for all λ; break = false; q = p;
    repeat (* until break *)
        for K := 0 to k do
            begin
                i := s;
                for each arc A_k = (i,j) or (j, i) with temporal label L(j, K, λ) do
                    begin
                        if A_k = (i,j) then
                            if L(j, K, λ_0) > L*(i, K, λ_0) + c_k(λ_0)
                            and A_k belongs to E^q then
                                begin
                                    L(j, K, λ) := L*(i, K, λ) + c_k(λ) for all λ from the
                                    solution set of the inequality
                                    L(j, K, λ) > L*(i, K, λ) + c_k(λ);
```

$S(j, K) := i$

end

else

if $K \geq 1$ and $L(j, K, \lambda_0) > L^*(i, K-1, \lambda_0)$ and
A_k belongs to E^p then

begin

$L(j, K, \lambda) := L^*(i, K-1, \lambda)$ for all λ from the
solution set Λ'' of the inequality
$L(j, K, \lambda) > L^*(i, K-1, \lambda)$;

$S(j, K) := -i$

end

else ($^*A_k = (j, i)^*$)

if $L(j, K, \lambda_0) > L^*(i, K, \lambda_0)$ then

begin

$L(j, K, \lambda) := L^*(i, K, \lambda)$ for all λ from the
solution subset Λ'' of the inequality
$L(j, K, \lambda) > L^*(i, K, \lambda)$;

$S(j, K) := i$

end;

$\Lambda' := \Lambda' \cap \Lambda''$;

find a temporal labelled node V_j with minimum $L(j, K, \lambda)$
and if any then denote by Λ'' the parameter set for which
this minimum holds; put $\Lambda' := \Lambda' \cap \Lambda''$ and $i := j$

end

end;

if $L^*(t, k, \lambda_0) < z(\lambda_0)$ then put $z(\lambda) := L^*(t, k, \lambda)$ for all λ from Λ'

else

if $L^*(t, k, \lambda_0) > z(\lambda_0)$ then

begin

denote by Λ'' the solution set of $L^*(t, k, \lambda) > z(\lambda)$;

$\Lambda' := \Lambda' \cap \Lambda''$; *break* := true

end;

$q := q-1$; if $q = 0$ then *break* := true

until *break*;

$q := q + 1$; take all labels $S(j, k)$ from the last iteration if $q = 1$
and from the previous one in case $q > 1$; $\varepsilon_k^q := \varnothing$; $j := t$;

repeat (*until $j = s^*$)

let $i = S(j, k)$;

if $i < 0$ and $A_l(-i, j)$ then begin $\varepsilon_k^q := \varepsilon_k^q + \{A_l\}$; $j := -i$ end
else if $i > 0$ then $j := i$
until $j = s$;
(* the solution set of (P'4) is ε_k^q for all λ from Λ'*)
$\Lambda^* := \Lambda^* \setminus \Lambda'$
until $\Lambda^* = \varnothing$;
$\Lambda := \Lambda \setminus \Lambda^*$
until $\Lambda = \varnothing$.

REMARK. It is possible that $|\varepsilon_k^q| < k$. It means that there exists a chain from s to t containing less than k forward arcs. If we eliminate them the length of the chain will be reduced to zero.

THEOREM 5. (A4) solves (P4) on the whole parameter set Λ in a finite number of iterations.

PROOF. For any fixed λ this is a modification of the algorithm from [8] if G (and hence GD) is a directed graph, i.e. (P4) is correctly solved for any fixed λ.

The loop from type repeat(until $\Lambda = \varnothing$) partitions Λ into finite subsets (each denoted by Λ^*) since Λ^* is a solution of a system from type $r_{i_1}(\lambda) \le r_{i_2}(\lambda) \le \ldots \le r_{i_m}(\lambda)$ and there are no more than $m!$ such systems.

At each iteration of the loop from type repeat(until $\Lambda^* = \varnothing$) (A4) finds a subset Λ' for which all inequalities are fulfilled. It follows from here and from Lemma 3 that ε_k^q is the desired solution of (P4') and also of (P4) not only for the fixed λ_0 but for all λ from Λ'. But ε_k^q belongs to a chain from s to t and there is a finite number of such chains. It follows that Λ^* will be partitioned into a finite number of subsets from type Λ'.

At the end of this section let us consider briefly the case when k nodes have to be removed. This case can be transformed into the previous one if we use the well–known transformation: to each node V_i we assign a new one $V_{i'}$ and the arc (i, i'); each arc (i, j) will be substituted by the arc (i', j) in the new graph. Now the capacity and reliability of (i, i') can be defined as

$$\min \{ \Sigma \{ c_l(\lambda) \mid A_l \text{ enters } V_i \}, \Sigma \{ c_l(\lambda) \mid A_l \text{ leaves } V_{i'} \} \} \quad \text{and}$$

$$\min \{ \max \{ r_l(\lambda) \mid A_l \text{ enters } V_i \}, \max \{ r_l(\lambda) \mid A_l \text{ leaves } V_{i'} \} \},$$

respectively. The new graph is also a planar one and the problem is to solve (P4) in the new network. The solution will contain arcs of type (i, i') only, which define the k most vital nodes.

5. COMPLEXITY

Each of the problems investigated here is a generalization of the non–parametric case. It follows that each of them is not less difficult than the problem from the special case. All of the problems are polynomial. The only exception is maybe (P4) if G is not a

planar graph. We do not know any polynomial algorithm for solving it – the algorithm in [9] is a branch and bound one end hence with exponential complexity. We do not know any proof that the problem is exponential or polynomial.

Each of the parametric problems is exponential. In [12] the author refers that the objective function $v_t(\lambda)$ of (P1) has an exponential number of break points even when the arc capacities are linear functions of λ. Each break point corresponds to one iteration, i.e. their number grows exponentially.

We deal in (P2) with the shortest paths. For a fixed start and target point there are $(n-2)!$ paths between them in the complete graph. Let us denote all paths by W_1, W_2 etc. and let us partition Λ into subintervals (subsets) like Λ_1, Λ_2 etc. For each arc from W_i we define its length as 1 on Λ_i. For all arcs which do not belong to W_i we define the arc length as M – a large number. Obviously, W_i is the only shortest path in the network between start end target points for all λ from Λ_i. We do the same for W_{i+1} and Λ_{i+1}. Finally we change $c_k(\lambda)$ in the neighbourhood of the border point between neighbour subsets like Λ_i and Λ_{i+1} in order to make $c_k(\lambda)$ continuous and piece–wise linear. In this way each path will be the shortest on some parameter subset of the Λ. Hence (P2) has an exponential complexity.

In an analogous way we show that (P3) is an exponential problem since each of n^{n-2} spanning trees in the complete graph can be a minimal one.

In (P4) we deal with the shortest chains in the equivalent problem in the dual graph and hence it is also exponential.

ACKNOWLEDGMENT. Some of the results presented here were reported during the 2nd Balcan Conference on Operational Research, October 1993, Thessaloniki, Greece. The final version was prepared during our visit to the Aristotelian University and University of Macedonia, Thessaloniki, under the NATO Scientific Fellowship Programme (April–June, 1994). We are grateful to Professors B. Papathanassiou, B. Manos and K. Tsouros for the very good conditions of work and to the young colleague Dimitris Kydros for typing the paper.

REFERENCES

[1] Ivanchev,D., "An optimization problem in a network under risk conditions", *University Annual (Appl. Math.)*, Sofia, 15/4 (1979) 93–102.

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

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

[4] Ivanchev,D., "Finding the foremost nodes in parametric capacited network", *University Annual (Appl. Math.)*, Sofia, 22/2 (1986) 151–165.

[5] Ivanchev,D., "Stromoptimierung auf verallgemeinerten Netzen", Wissenschaftliche Berichte der TH Leipzig, 4, 1984, 35–46.

[6] Ivanchev D.,G.Naegler, *Network optimization*, TU of Sofia, 1993, p.206.

[7] Ivanchev D., and Ruhe,G., "Flows in generalized networks with parametric arc capacities", *Comptes rendus de l'Academie bulgare des Sciences*, Sofia, 37/4 (1984) 477–480.

[8] 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.

[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] Ruhe,G., "Characterization of all optimal solutions for parametric maximal flows in networks", *Mathematische Operationsforschung und Statistik, Series optimization* 16/1 (1985) 51–61.

[11] Ruhe,G., and Ivanchev,D., "The solution of generalized network flow problems with parametric capacities and stochastic demands", *Foundations of Control Engineering* 9/3 (1984) 133–139.

[12] Wagelmans,A., *Sensitivity analysis of combinatorial optimization*, Universiteits Drukkerej, Rotterdam, 1990, p.210.

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