Transport and Telecommunication, 2011, Volume 12, No 1, 4–11
Transport and Telecommunication Institute, Lomonosova 1, Riga, LV-1019, Latvia

PREDISASTER DESIGN OF TRANSPORTATION NETWORK

Ilya Gertsbakh[1], Yoseph Shpungin[2]

[1]Department of Mathematics, Ben Gurion University
P.O.Box 653, Beer-Sheva, 84105, Israel
E-mail: elyager@bezeqint.net

[2]Software Engineering Department, Shamoon College of Engineering
Beer-Sheva, 84105, Israel
E-mail: yosefs@sce.ac.il

In this paper we consider transportation network optimal reinforcement problem. We mean by that network reliability improvement achieved by reinforcement of a certain number of its most important links (road segments), which are subject to failure (destruction). This reinforcement is made given a budgetary constraint, and its goal is to maximize the global network performance measure, which depends on the source-terminal reliability of a fixed set of origin-destination routes or network all-terminal connectivity.

The central role in the proposed optimization is played by a combinatorial Monte-Carlo algorithm for estimating the network $s-t$ reliability or the probability of all-terminal connectivity, and the gradient vector. The proposed optimization method is tested on an example described in [1] of a road network of Istanbul. The link failures are caused by natural disasters like earthquakes, and link e reinforcement means its replacement for a cost $c(e)$ by a more reliable one. Another example is a 4-terminal 34-link with 25 nodes network in which the reinforcement raises the link reliability up to 0.9 and the problem is stated as find the minimal cost reinforcement policy to guarantee the prescribed value of the probability of all terminal connectivity.

Keywords: network Monte Carlo, reliability gradient vector, network design, link failure, earthquakes, $s-t$ network reliability, all-terminal connectivity

1. Introduction

Optimal transportation network (TN) reliability design is not a well-defined notion. Its formulation demands first of all answering the following principal questions:

1) What are the network components subject to failure.
2) What is the definition of component failure.
3) How network reliability is connected or related to its component failure.
4) What is the whole network performance criterion (NPC) and how is it expressed via its reliability parameters.

After these issues are clarified, it is necessary to define the relevant resources which are at the disposal of the network designers. Then it will become possible to formulate the problem of finding the best policy of using these resources to achieve the maximal possible NPC under given constraints.

Following recently published papers [1,2], we make the following assumptions:

a) TN components subject to failure are the links (or arcs), i.e. the road segments connecting network nodes.

b) Link failure formally means that the link $e=(a,b)$ connecting nodes (a,b) becomes not operational, i.e. the connection between a and b via this link is completely disrupted. Physically it might be due a natural disaster (earthquake, flood), road accident or, speaking in military terms, an "enemy attack."

c) Each link e has its failure probability $q(e)$, and link failures are viewed as independent events.

d) The TN performance criterion is a function of the network ability to provide a connection between two sets of its nodes – the origins and destinations, or the connection between a special set of so-called terminal nodes. The simplest case of NPC will be a functional depending on the probability to provide connection between a "source" and "terminal", or the so-called probability of $s-t$ connectivity. The model considered in [1], singled out several most important $s-t$ pairs in the TN and assumed that the NPC depends on the set of $s-t$ connection probabilities of these pairs. Another version of the NPC is the value of the probability of all-terminal connectivity.

e) A natural way to formalize network improvement or reinforcement (termed in [1] as predisaster investment) is to carry out link *e* "repair" to reduce its failure probability from *q*(*e*) to $q_{\min}(e) < q(e)$ by investing *c*(*e*). In physical terms, this means reconstruction of the road segment and/or reinforcement of its weak elements, like bridges, tunnels, etc. Since there are always budget constraints, the total amount of money invested in the network component reinforcement must be limited by a certain budget *D*.

The above assumptions a)-e) allow formulating our problem in several ways: find the "best" set of reinforced components (links) in order to maximize the NPC subject to the budget constraint *D*, or to find the cheapest way to guarantee the desired level of network reliability, which is a formulation dual to the fisrt one.

The paper is organized as follows. In Section 2 we formulate the problem in formal terms and present our principal optimization algorithm. We suggest using a greedy-type semi-heuristic procedure. Its most important and difficult part is the estimation of network reliability and the reliability gradient vector for the selected network reliability criterion. We shortly describe this procedure in Appendix. Section 3 presents an account of our optimization results for an Istanbul network analyzed in [2]. Section 4 is an investigation of a 34-link network for which the problem is stated as achieving the desired level of all-terminal connectivity for minimal price. Section 5 presents a probabilistic description of network disintegration process based on so-called network D-spectrum.

2. Problem Formulation and Algorithm

By transportation network (TN) $N = (V, E, T)$ we denote an undirected graph with a node-set $V, |V| = n$, an edge-set $E, |E| = m$, and a set T $\subseteq$ V of special nodes called *terminals*. Instead of the term *edge* we will often use the word *link* or *arc*. Each network link *e* is associated with a probability p_e of being *up* and probability $q_e = 1 - p_e$ of being *down*. We postulate that link failures are mutually independent events. Link failure means its elimination (erasing) from *E*.

A. Let *s* and *t* be two distinct terminals, $s, t \in T$. The TN will be called $s-t$ (or source-terminal) connected if there is a path connecting *s* and *t*, with all its edges being in the *up* state. Denote by $R(s-t; \vec{p})$ the probability that the TN is $s-t$ connected. Here $\vec{p}$ denotes the vector of link *up* probabilities : $\vec{p} = (p_1, p_2, \ldots, p_m)$. In TN, typically there are several $s-t$ pairs of nodes whose connection is of special importance. We assume that there are *k* such pairs and denote them as $Z = \{s_1 - t_1, \ldots, s_k - t_k\}$.

Our first formulation of the NPC is *maximizing* the *minimal* reliability of the $s-t$ connectivity over the set of all source-terminal pairs *Z*:

$$G(\vec{p}) = \min_{i \in Z}[R(s_i - t_i; \vec{p})]. \quad (1)$$

Let us introduce two additional vectors: $p^* = (p_1^*, \ldots, p_k^*)$ and $c^* = (c_1, \ldots, c_k)$, where c_j is the cost of *increasing* link *j up* probability from p_j to p_j^*.

Let $\vec{B} = (b_1, \ldots, b_k)$ be a *k*-dimensional vector with binary 0/1 components. The set of all 2^k binary vectors we denote by Ω. Suppose we choose a particular vector $\vec{B} \in \Omega$ and decide to reinforce those links whose coordinates in $\vec{B}$ are not zeroes. Then the vector of link *up* probabilities will become

$$\vec{p}(\vec{B}) = (p_1 + (p_1^* - p_1) \cdot b_1, \ldots, p_k + (p_k^* - p_k) \cdot b_k).$$

The cost of this action will be

$$C(\vec{B}) = c_1 \cdot b_1 + \ldots + c_k \cdot b_k.$$

Suppose that the total reinforcement budget must not exceed the given value *B*.

Now we are ready to formulate our optimization problem in the following compact form: find such $\vec{B} \in \Omega$ which *maximizes*

$$G(\vec{p}(\vec{B})) \quad \text{subject to} \quad C(\vec{B}) \le D. \tag{2}$$

Now let us approach the maximization procedure. It will be based on an efficient Monte Carlo procedure of estimating network reliability gradient function. It does not demand the knowledge of the analytic form of $R(s_i - t_i; \vec{p})$. Let us postpone to the Appendix the details of the gradient estimation procedure and suppose that we have an accurate estimate of

$$\Delta_i R(s-t; \vec{p}) \approx \frac{\partial R(s-t; \vec{p})}{\partial p_i} \cdot (p_i^* - p_i).$$

Now we are ready describe a greedy type semi-heuristic solution algorithm for solving problem (2). Denote by *S* the set of all indices {1, 2, ..., k} and all *pairs* of indices $\{(i, j), 1 \le i < j \le k\}$.

Algorithm MinGradKnapsack

1. Set $X := 0$.

2. $Y := G(\vec{p})$. (This is the initial value of the NPC)

3. For each element $\alpha \in S$, estimate $\Delta_i G(\vec{p})$. [Calculate the increase of the minimal probability of $s-t$ connectivity as a result of the reinforcement of link (or links) α].

4. Calculate $\eta_\alpha = \frac{\Delta_i G(\vec{p})}{c_\alpha}$, for each $\alpha \in S$ and arrange them in *decreasing* order. Denote by *w* the index of the *maximal* η_i. [If α is a pair of indices, e.g $\alpha = (r, s)$, then put $c_\alpha = c_r + c_s$].

5. Put $X := X + c_w; S := S \setminus \alpha$. [Delete α from the set *S*; if $\alpha = j$, delete *j* and all pairs containing *j*. Act similarly if $\alpha = (r, s)$].

6. Recalculate p by replacing p_α by p_α^*. [If α is a pair (*r*,*s*), replace the *r*-th and the *s*-th component of *p* by p_r^*, p_s^*].

7. If $X > D$, then Stop. Otherwise GOTO 2. #

In simple words, the algorithm finds on each step the link or a pair of links whose reinforcement provides the maximal performance measure *increase per unit cost*, recalculates the reliabilities of all $s-t$ connections involved and the gradient vector after the links have been reinforced, repeats this procedure in a loop and stops when the budget becomes exhausted.

B. Our second network predisasted design approach is dual to the previous one. It consider the search for the minimal-cost link reinforcement to provide the given level of the reliability criterion, which in our example is defined as the probability of all-terminal connectivity. The optimization algorithm basically copies the above described heuristic Knapsack-type procedure: on each step, it chooses for reinforcent that link which provides the maximal reliability increase per unit cost.

3. Example 1: 30-link Network of Istanbul [1]

We describe in this section a road network of Istanbul, borrowed from the paper [1] by Srinivas Peeta, F.Sibel Salman, Dillek Gunnec, and Kannan Viswanath. The data for this example are summarized in Table 1 and are cited with the kind permission of the authors.

The network has 30 links and 20 nodes. The first and the fourth columns contain data on the links and the nodes they connect. Columns 2 and 5 are the link reinforcement costs; columns 3 and 6 give the link *up* probability. As suggested in [1], five $s-t$ routes are of primary importance: $a = (14,20)$, $b = (14,7)$, $f = (12,18)$, $g = (9,7)$, and $h = (4,8)$.

Our goal is to find the links which are to be reinforced in order to *maximize* the minimal reliability of the $s-t$ connectivity over the routes *a*, *b*, *f*, *g*, *h*. As assumed in [1], reinforcement of a link *e* raises its *up* probability from p_e to $p_e^* = 1$ for all *e*. We assumed the budget constraint $D = 1700$.

Our algorithm found that the following links have to be reinforced: 10, 17, 20, 21, 22, 23, the total cost of which is 1680. Our algorithm carried out 5 cycles. On each of the first four cycles, one link has been reinforced, and two links were selected on the last, fifth cycle.

The minimal reliability has route h: $R(h)$=0.686; the reliability of other $s-t$ pairs are as follows:

$$R(a) = 0.773, R(b) = 0.723, R(f) = 0.999, R(g) = 0.834.$$

In spite of the fact that [1] considers quite different NPC, a convex combination of functions of paths lengths for 5 selected pairs, our optimization results greatly coincide: [1] recommends the reinforcement of the links 10, 20, 21, 22, 23, 25. The cost of this replacement is smaller – 1640, the minimal $s-t$ probability 0.682 have routes b, h. The average reliability of the above 5 pairs differ only by 0.003 and is 0.800.

Table 1. Links, their repair costs and reliability for Example 1

Link e=(a,b)	c_e	p_e	Link e=(a,b)	c_e	p_e
1=(1,3)	80	0.80	16=(11,13)	940	0.55
2=(2,4)	80	0.80	17=(12,13)	300	0.70
3=(3,4)	320	0.80	18=(12,16)	520	0.60
4=(3,5)	260	0.70	19=(12,25)	40	0.80
5=(4,6)	160	0.80	20=(13,14)	800	0.55
6=(5,8)	420	0.60	21=(14,15)	40	0.80
7=(5,10)	160	0.80	22=(15,18)	160	0.70
8=(6,7)	620	0.60	23=(16,17)	40	0.80
9=(7,10)	120	0.80	24=(17,21)	620	0.60
10=(7,11)	340	0.70	25=(18,20)	260	0.70
11=(8,9)	940	0.55	26=(18,21)	780	0.60
12=(8,10)	160	0.80	27=(19,20)	800	0.55
13=(9,11)	620	0.60	28=(20,22)	120	0.80
14=(9,12)	180	0.50	29=(21,23)	220	0.70
15=(9,24)	40	0.80	30=(22,23)	500	0.60

4. Example 2: 34-link Network with 25 Nodes, 4 Terminals

Table 2. Links, costs and reliabilities for Example 2

Link e=(a,b)	c_e	p_e	Link e=(a,b)	c_e	p_e
1=(1,2)	1	0.6	18=(11,13)	2	0.8
2=(1,3)	2	0.7	19=(12,13)	3	0.6
3=(1,23)	3	0.8	20=(12,16)	4	0.7
4=(2,4)	4	0.6	21=(12,25)	1	0.8
5=(2,22)	1	0.7	22=(13,14)	2	0.6
6=(3,4)	2	0.8	23=(14,15)	3	0.7
7=(3,5)	3	0.6	24=(15,18)	4	0.8
8=(4,6)	4	0.7	25=(15,24)	1	0.6
9=(5,8)	1	0.8	26=(16,17)	2	0.7
10=(5,10)	2	0.6	27=(17,21)	3	0.8
11=(6,7)	3	0.7	28=(18,20)	4	0.6
12=(7,10)	4	0.8	29=(18,21)	1	0.7
13=(7,11)	1	0.6	30=(19,20)	2	0.8
14=(8,9)	2	0.7	31=(19,25)	3	0.6
15=(8,10)	3	0.8	32=(20,22)	4	0.7
16=(9,11)	4	0.6	33=(21,23)	1	0.8
17=(9,24)	1	0.7	34=(22,23)	2	0.6

Table 3. First iteration of the solution for Example 2

Link e=(a,b)	$\alpha = \frac{\partial R}{\partial p(e)}$	$\frac{\alpha \cdot (0.9 - p)}{c(e)}$	Link e=(a,b)	$\alpha = \frac{\partial R}{\partial p(e)}$	$\frac{\alpha \cdot (0.9 - p)}{c(e)}$
1=(1,2)	0.0618	0.018	18=(11,13)	0.1534	0.075
2=(1,3)	0.1430	0.0014	19=(12,13)	0.2524	0.0253
3=(1,23)	0.1632	0.0053	20=(12,16)	0.1638	0.0068
4=(2,4)	0.0709	0.0052	21=(12,25)	0.1005	0.010
5=(2,22)	0.0648	0.013	22=(13,14)	0.0868	0.013
6=(3,4)	0.0502	0.0153	23=(14,15)	0.0756	0.005
7=(3,5)	0.1526	0.0153	24=(15,18)	0.1405	0.0035
8=(4,6)	0.0856	0.0042	25=(15,24)	0.0946	**0.028**
9=(5,8)	0.0502	0.005	26=(16,17)	0.1362	0.0135
10=(5,10)	0.0708	0.010	27=(17,21)	0.1192	0.004
11=(6,7)	0.0852	0.0057	28=(18,20)	0.1235	0.0092
12=(7,10)	0.1325	0.0032	29=(18,21)	0.1720	**0.034**
13=(7,11)	0.1113	**0.033**	30=(19,20)	0.1008	0.005
14=(8,9)	0.0843	0.0085	31=(19,25)	0.1334	0.0133
15=(8,10)	0.1055	0.0037	32=(20,22)	0.1010	0.005
16=(9,11)	0.0775	0.0058	33=(21,23)	0.1538	0.015
17=(9,24)	0.0825	0.016	34=(22,23)	0.0528	0.008

The network has 34 links, 25 nodes, 4 of which are terminals (nodes 1, 18, 10, 12). Link reliabilities are presented in the third and sixth columns of Table 2. They range from 0.6 to 0.8. Reinforcement costs are given in the second and fourth column of Table 2. They range from 1 to 4. Contrary to the assumption made in the previous example, in this example we assume that the reinforced links have reliability less than 1, namely p*= 0.9.

Our goal is to reinforce edges to achieve $R^* = 0.85$ for minimal cost. The initial network reliability is $R_0 = 0.471$. Table 3 presents the first iteration of the solution by the Algorithm MinGradKnapsack**.** Note that in this example on each iteration we choose three "best" links.

The second and fifth columns give the values of the partial derivatives $\alpha = \frac{\partial R}{\partial p(e)}$. The third and sixth columns give the crucial ratio for reinforcement $\frac{\alpha \cdot (0.9 - p)}{c(e)}$. We see from the Table 3 that three links have the largest values of this ratio are 13, 25 and 29. According to the algorithm, we replace these links by more reliable, with the probability 0.9. This reinforcement raises the network reliability to $R_1 = 0.627$.

In the next iteration (not shown in this table) the links 7, 19 and 13 were chosen, and the network reliability becomes $R_2 = 0.719$. The next iteration gives links 1, 10, 17 and $R_3 = 0.783$. The last iteration adds links 2, 18, 22 and the final reliability is $R_4 = 0.848$.

In addition we can do the following remark about the accuracy of the result. The costs are rather approximate. Suppose that each cost c(e) = 1, 2, 3, 4, have possible fluctuations in the range [–0.5, +0.5].

Assuming that they are random, then approximately the sum of twelve such cost values will have fluctuations in the range $[-3.5 \cdot 0.5, +3.5 \cdot 0.5]$, i.e. in the range about [-2, +2]. This justifies our heuristic solution, not speaking about other sources of uncertainty like errors in determining p(e) and p*.

5. D- spectrum and Network Behaviour in the Process of Link Elimination

Imagine that there is an external process of "shocks", which destroy *in random* order, one link of the network after another. Such process may be a sequence of earthquakes, heavy road accidents, or even "enemy attacks". For the purpose of our exposition, the probabilistic description of this process is not essential, since we count "time" in the number of destroyed links.

We are interested in the probabilistic description of the process of gradual network disintegration into isolated components containing network terminals (which we call "clusters"). Theoretically, there are

several stages of this disintegration. On the first, the all-terminal connectivity is violated and the network falls apart into two clusters. When the links continue to fail, sooner or later there will appear three clusters, and finally each terminal will become isolated from all others.

Let us illustrate the first stage of this process on the example of 34-link network described in Section 4.

The central role in this description belongs to the so-called network *D-spectrum* which in our case coincides with so-called *signature* [4],[5]. Definition and some properties of the D-spectra are described in Appendix 2. For our exposition it is important only to mention that the D-spectrum is a discrete cumulative distribution function $H(x)$ of the random variable Y which is the number of sequentially destroyed links which cause network transition from state *UP* (all terminals are connected to each other) to the state *DOWN* defined as the loss of terminal connectivity. It means the following.

Suppose that links fail in random order. Let $h(i)$ be the probability that the network failure took place on the *i-th* step of this process, i.e. after exactly *i* links have failed, $h(i) = P(Y = i)$ and

$$H(x) = h(1) + h(2) + ... + h(x) = P(Y \le x), x = 1,2,..., n.$$

In the table below we present the D-spectrum $H(x)$ for the 34-link network of Example 2. $H(1) = H(2) = 0,\ H(24) = H(25) = ... = H(34) = 1,$ all other $H(x)$ values are shown in Table 3. These values are obtained as a result of a Monte Carlo experiment the detail of which are described in the Appendix 2.

From this table we see that the minimal size cut set of the network is 3. (More exactly, there are 12 such cuts). With probability about 0.93 the network fails after 7 or more links have failed. After 16 links fail, network is *DOWN* with probability close to 1. So, all process of loss of terminal connectivity takes place in rather narrow interval [7,16] of failed links. On the average, about 10 link failures cause the network to lose its terminal connectivity. In network analysis, the quantities which describe network *resilience* are expressed via the ratio of the number of failed links causing network failure to the total number of links.

In our example, the average resilience is $\frac{10}{34} = 0.31.$ This is a rather low number typical for sparse network. Indeed, our network is rather sparse, it has the average node degree $d = 34 \cdot \frac{2}{25} = 2.7$, which means that on the average there are less than 3 links incident to each node.

Table 4. Cumulative D-spectrum of the Network of Example 2

x	3	4	5	6	7	8	9
H(x)	.00216	.00991	.02790	.06464	.12748	.22590	.35821
x	10	11	12	13	14	15	16
H(x)	.51183	.66381	.79200	.88272	.93876	.97106	.98769
x	17	18	19	20	21	22	23
H(x)	.99509	.99816	.99934	.99984	.99996	.99999	1

6. Appendices

Appendix 1. *s – t* Reliability and Reliability Gradient

To estimate the $s-t$ connectivity reliability we suggest one of the Monte Carlo procedures described in [3], e.g. the turnip algorithm, see Chapter 9. For small networks, it might be even the crude Monte Carlo.

Let us describe an efficient approach to estimating the network reliability gradient vector. It is based on identifying so-called network *border states* [3], Chapter 5.

Definition A1.

Reliability gradient vector ∇R is defined as $\nabla R = (\frac{\partial R}{\partial p_1}, \ldots, \frac{\partial R}{\partial p_k}).\#$

Denote by a binary variable e_i the state of the *i*-th network component: 0 means *down*, 1 means *up*.

Definition A2.

Network state $V = (e_1, \ldots, e_k)$ is called *border state* if:

1. This state is a *DOWN* state;

2. There exists such state $W, W \in UP,$ that the states *V* and *W* differ in exactly one position (i.e. the Manhattan distance between vectors *W* and *V* equals 1). #

The following formula [3] binds together the partial derivative $\frac{\partial R}{\partial p_i}$ and the set of such border states that each of them may be transferred into the *UP* state by turning a single *down* element $e_i = 0$ into *up* state $e_i = 1$:

$$\frac{\partial R}{\partial p_i} = q_i^{-1} \{ \sum_{\{v \in BD, v+(0,\ldots,0,1_i,0,\ldots 0) \in UP\}} P(v) \}, \qquad i = 1, \ldots, k. \tag{3}$$

Here *BD* is the set of all border states, and *P*(*v*) stands for the static probability of the state *v*. In words: to calculate the *i*-th gradient's vector component, it is necessary to find the probability of all border states which become *UP* by "activating" a single component e_i.

The last formula is the key for Monte Carlo algorithm for computing reliability gradient. Note that this algorithm is rather efficient and avoids the so called *rare event phenomenon*. The algorithm works as follows.

Simulate permutation of the network elements. Suppose that initially all elements are *down*, and they are arranged in the order determined by this permutation. Start turning the elements from *down* to *up* from left to right. Suppose that after adding a new *up* element we arrive at a border state. Clearly, from this moment the network remains in border states until the moment it becomes *UP*. Calculate and sum up *P*(*v*) for all elements which may transfer the border state *v* into an *UP* state.

Appendix 2. Cumulative D-spectrum

Let us number the components of the network subject to failure by numbers $e_1, e_2, ..., e_n$, and consider an arbitrary permutation $\pi = (e_{i_1}, e_{i_2}, ..., e_{i_n})$ of these numbers. Assume that all components are *up* and we start turning them *down* moving from left to right along π. On each step of this process we check the network state and fix the step number $Y(\pi)$ on which network state becomes *DOWN*. Imagine that all *n*! permutations are equally probable and this procedure is repeated for each of these n! permutations. Let $h(k) = \frac{N(k)}{n!}$ be the ratio of permutations for which $Y(\pi) = k$ to all permutations, or the probability that *Y*=*k*. $\{h(k), k = 1, ..., n\}$ is a discrete density of random variable *Y*. This density was first considered in slightly different context by F. Samaniego in 1985 and called *signature*. In 1991 it was independently introduced by M. Lomonosov and termed *Internal Distribution* (*ID*). Later on, we used the term *D-spectrum* [3], with "D" for destruction process. The cumulative distribution of *Y*, $H(x) = h(1) + h(2) + ... + h(x)$ is called cumulative D-spectrum (or cumulative signature). It is important to stress that the D-spectrum is a structural parameter of the network which depends only on its topology and failure state, i.e. *DOWN* definition, and not on the random mechanism governing the real process of network component failures.

The D-spectrum has many interesting properties, the central of which is the following formula for network *DOWN* probability if the network components fail independently of each other and have failure probability *q*:

$$P(DOWN; q) = \sum_{x=1}^{n} H(x) q^x (1-q)^{n-x} \cdot \frac{n!}{x!(n-x)!}.$$

It follows from the definition of the D-spectrum that *H*(*x*) describes the probabilistic mechanism of network transition from *UP* state to *DOWN*, which in our example was defined as the loss of all-terminal

connectivity. Introducing several stages of network disintegration (appearance of two, three and four isolated components containing terminals, so-called clusters), we arrive at the definition of the first, second and third marginal D-spectra. This extension will not be considered here and the interested reader can find the details in [4].

The exact calculation of the D-spectrum is a NP computation problem. We will adopt a Monte Carlo approach to the approximation of D-spectrum. The corresponding Monte Carlo algorithm is based on *M* replications of the process of generating random permutation and on follow up of the network state in the process of turning *down* network components in accordance to the above described definition. Let $N(k)$ be the number of permutations, out of *M* generated, in which the network failure take place on the *k-th* step. Then the estimate of $H(k)$ will be the ratio $\frac{N(k)}{M}$. The details of this algorithm can be found in [3]. For networks of size similar to that considered in Example 2 it is enough to take $M = 10^5$, which takes just a few seconds of calculation on a PC computer.

References

1. Srinivas Peeta, F. Sibel Salman, Dillek Gunnec, Kannan Viswanath. Predisaster Investment Decisions for Strengthening a Highway Network, *Computers & Operations Research,* Vol. 37, 2010, pp. 1708–1719.
2. Sanchez-Silva, M., Daniels, M., Lleras, G., Patino, D. A Transport Network Reliability Model for Efficient Assignment of Resources, *Transportation Research, Part B*, Vol. 39, 2005, pp. 47–63.
3. Gertsbakh, Ilya and Yoseph Shpungin. *Models of Network Reliability: Analysis, Combinatorics, and Monte Carlo*. CRC Press, 2010. 217 p.
4. Gertsbakh, Ilya and Yoseph Shpungin. *Network Reliability and Resilience*. Springer Briefs, Springer, 2011. (In Press)
5. Samaniego, F. *System Signatures and Their Applications in Reliability Engineering*. New York, Berlin: Springer, 2007. 166 p.