

# D-spectrum and Reliability of a Binary System With Ternary Components

**Ilya B. Gertsbakh<sup>1</sup>, Yoseph Shpungin<sup>2</sup>, R. Vaisman<sup>3</sup>**

<sup>1</sup>*Department of Mathematics, Ben-Gurion University  
P. O. Box 653, Beer-Sheva, 84105, Israel,  
elyager@bezeqint.net*

<sup>2</sup>*Software Engineering Department  
Sami Shamoon College of Engineering, Beer Sheva 84100 Israel,  
yosefs@sce.ac.il*

<sup>3</sup>*School of Mathematics and Physics,  
The University of Queensland, Brisbane 4072, Australia.  
r.vaisman@uq.edu.au*

## ABSTRACT

We consider a monotone binary system with ternary components. "Ternary" means that each component can be in one of three states: *up*, *middle (mid)* and *down*. Handling such systems is a hard task, even if a part of the components have no *mid* state. Nevertheless, the permutation Monte Carlo methods, that proved very useful for dealing with binary components, can be efficiently used also for ternary monotone systems. It turns out that for "ternary" system there also exists a combinatorial invariant by means of which it becomes possible to count the number  $C(r; x)$  of system failure sets which have a given number  $r$  and  $x$  of components in *up* and *down* states, respectively. This invariant is called ternary D-spectrum and

---

<sup>0</sup><sub>†</sub> This research was supported by the Australian Research Council under grant number CE140100049.

it is an analogue of the D-spectrum (or signature) of a system with binary components. Its value is the knowledge of system failure or path set properties which do not depend on stochastic mechanism governing component failures. In case of independent and identical components, knowing D-spectrum makes it easy to calculate system *UP* or *DOWN* probability for a variety of *UP/DOWN* definitions suitable for systems of many types, like communication networks, flow and supply networks. etc.

**Key words:** D-spectrum, signature, ternary components, binary system, path sets, failure sets, ternary importance spectrum.

## 1 Introduction

**1.1.** We consider a system containing  $n$  components numbered from 1 to  $n$ . Each component  $i$  can be in three states: *up*, *mid*, *down*. These states will be denoted by numbers: 2 for *up*, 1 for *mid* and 0 for *down*. The state of system's components state is described by a ternary vector

$$\mathbf{v} = (v_1, v_2, \dots, v_n),$$

where  $v_i = 2, 1$  or  $0$ , according to the component state. For example,  $\mathbf{v} = (2, 0, 1, 1, 2)$  means that components 1 and 5 are in state *up*, components 3,4 - in *mid*, and component 2 is *down*.

The system has only two states, i.e. is binary: *UP* (denoted by 1) and *DOWN* (denoted by 0). The system state is determined by a binary structure function

$$\varphi = \varphi(\mathbf{v}).$$

We make the following standard assumptions regarding the dependence of system state on component states [1]:

- (i)  $\varphi(2, 2, \dots, 2) = 1$ ;  $\varphi(0, 0, 0, \dots, 0) = 0$ .
- (ii) if  $\mathbf{v} > \mathbf{y}$ , then  $\varphi(\mathbf{v}) \geq \varphi(\mathbf{y})$ . ( $\mathbf{v} > \mathbf{y}$  means that  $v_i \geq y_i$  for all  $i$  but there is at least one  $j$  such that  $v_j > y_j$ ). (ii) means that the system is monotone.

Natural objects with ternary components are, for example, communication net-

works with three-level performance of their edges or nodes. System *UP* state can be defined as the situation when it is possible to maintain high-performance information exchange among the nodes of a given subset  $T_1$  of nodes **and** information exchange (on any level) of another subset of nodes  $T_2$ . The *DOWN* state then is an absence of at least one of these two connections modes. This is an extension of the standard T-terminal connectivity for a network with binary components.

Very often, the network components subject to failure are nodes, see for example [9], Chapters 12, 13. In modeling practical situations, it may be adequate to define three states for a node: *down* means deletion of all adjacent edges, *mid* means deletion of a part of edges adjacent to the node, and *up* means that all edges adjacent to the node remain operational. A standard issue for a large networks with randomly failing nodes is determining the size of the largest (connected) component. In our situation, the corresponding *UP* definition might be presence of a component having at least, say 80% of all nodes in it.

Another natural application can be found in stochastic flow networks with three levels of flow capacities for each network edge. The *UP* state of such network may be defined as the existence of source-terminal flow exceeding some given critical level  $L_{max}$ .

**1.2.** Our further exposition is as follows. In Section 2 we define the so-called ternary D-spectrum, which is the central issue of this paper. The ternary D-spectrum is system's combinatorial invariant. It depends only on its structure function  $\varphi(\cdot)$ . No information about component reliability is needed for calculating the ternary spectrum. Contrary to the D-spectra for binary systems considered in literature (also known as *signatures* or *internal distributions* [2], [4], [5], [6], [10], [11]), the ternary spectrum is not a single discrete cumulative distribution function (CDF), but a *collection* of such CDF's. Some of them are not proper, i.e. some of the CDF's are strictly less than 1. The cumulative D-spectrum of a binary system is a vector of dimension  $n$ , while the ternary D-spectrum is a set of  $k$  vectors,  $k < n$ , and all of them, except the first one, have dimension strictly less than  $n$ .

We show how to calculate system *DOWN* probability by means of this D-spectrum if the system has independent and identical components. A crucial role is played by a combinatorial formula (3) (see our Theorem 1) expressing, by means of the

D-spectrum, the number of failure (cut) sets in the system.

If we know that components are independent and identical, and know their state probabilities  $p_2 = P(up)$ ,  $p_1 = P(mid)$  and  $p_0 = P(down) = 1 - p_2 - p_1$ , then we can easily calculate system *DOWN* probability by means of the ternary D-spectrum, see formula (6) and Theorem 2. If the state probabilities depend on time, it is quite easy to calculate system "dynamic" reliability, see the example after Theorem 2 and formula (7).

Section 3 considers an example - a cubic network  $H_4$  with 16 nodes and 32 edges. This network is *UP* if the set  $T_1$  having six nodes is strongly connected and all nodes of  $H_4$  are weakly connected. We describe the ternary spectrum and present data on network reliability. In Section 4 we present a Monte Carlo algorithm for calculating the ternary spectrum. Section 5 is devoted to the so-called ternary importance spectrum by means of which we will be able to evaluate and compare component importance.

### 1.3. Some comments on the present state of art in computing reliability of complex networks.

Let us note here the important case of highly reliable systems with component failure probability, say of magnitude  $10^{-10}$ . Such reliability parameters as overall connectivity or *T*-terminal connectivity depends essentially only on the size and the number of minimal-size cut sets, or in our terminology, on the failure sets having minimal number of failed components, see Section 2. Such information is provided by the ternary D-spectrum. The complexity of calculating the ternary spectrum vs regular binary spectrum (signature) can be justified by the fact that ternary system is considerably more complex than the similar binary system with the same number of components. For example, hypercube of order 4 considered in Section 3 has 32 edges and total number of elementary states  $3^{32}$ . This is greater than the number of states of the same hypercube with binary edges by factor of about 400,000.

In certain situations, modeling ternary component may be partially achieved using a pair of binary components working in parallel. For example, in stochastic flow networks, an edge  $e = (a, b)$  with capacities  $0, D, 2D$  can be replaced by a pair of identical independent parallel binary edges with capacities  $0, D$ . This pair will provide  $P_e(up) = p^2$ ,  $P_e(down) = (1 - p)^2$  and  $P_e(mid) = 2p(1 - p)$ . This

method is often used in practice, see for example [9] and references there. The above replacement has however limited applicability. For example, it can not provide an adequate substitute for a ternary edge with capacities  $(0, 0.75D, D)$  and arbitrary  $(p_2, p_1, p_0)$ . In case that failed components in network are the nodes, it is not clear at all how to replace a ternary node by a pair of binary nodes.

Multi-state network reliability mainly dealt with a two-terminal reliability in flow network setting. A typical paper is [9] (see also references [11-17] there) which describes an algorithm for approximating network reliability with independent and possibly non-identical edges with variable capacities. Largest example presented in [9] deals with a network with 13 nodes and 20 edges having ternary capacities.

Signature or D-spectrum of a system with binary components has an important property first discovered by Samaniego [10]: the CDF of system life-time  $\tau_{syst}$  can be expressed via the CDF of its components as a linear combination of signature and respective order statistics. Similar property does not exist for ternary systems. A certain resemblance to it has formula (7) in Section 2 presenting the CDF of  $\tau_{syst}$  for the case that component states are described by independent and identical stochastic processes.

## 2 Ternary D-spectrum

Our object of main interest will be the properties of random permutations of system components. For our purposes, it will be convenient to write a permutation of component numbers together with the component states:

$$\pi = ((i_1; s(i_1)); (i_2, s(i_2)); \dots; (i_n, s(i_n))). \quad (1)$$

Here on the first position in each pair is component number ("name") and on the second - its state 2, 1 or 0 for *up*, *mid* and *down*, respectively. A permutation in form (1) determines in an obvious way the system state vector  $\mathbf{y} = \mathbf{y}(\pi)$ . For example, the permutation

$$\pi = ((5, 2); (1, 0); (4, 1); (3, 2); (2, 1))$$

means that components 5 and 3 are *up*, component 1 is *down* and components 2 and 4 are in *mid*. The corresponding system state vector will be

$$\mathbf{y} = \mathbf{y}(\pi) = (0, 1, 2, 1, 2),$$

where the  $j$ -th position is occupied by number 2,1,0 denoting the state of component number  $j$ ,  $j = 1, 2, \dots, n$ .

**Definition 1:** *random permutation of  $r$ -th type*,  $r = 0, 1, \dots, n - 1$ .

A random permutation of component numbers  $\{1, 2, \dots, n\}$  in which all components are in state *mid* is called a random permutation of zero-type. It has the following form:

$$\pi_0 = ((i_1, 1); \dots, ((i_2, 1); \dots; (i_n, 1)))$$

A random permutation  $\pi_r$  is called a permutation of  $r$ -th type,  $r > 0$ , if the components  $i_1, \dots, i_r$  on the first  $r$  positions are in state *up*, and the components  $i_{r+1}, \dots, i_n$  on the remaining  $(n - r)$  positions are in state *mid*:

$$\pi_r = ((i_1, 2); \dots, ((i_r, 2); (i_{r+1}, 1); (i_{r+2}, 1); \dots; (i_n, 1)))\#$$

Probability of obtaining a particular ordering of component numbers  $(i_1, i_2, \dots, i_n)$  in the  $r$ -th type permutation is  $1/n!$

**Definition 2:** *failure (cut) set* and  $(r; x)$ -*failure set*.

*Failure set* is a vector  $\mathbf{v} = (v_1, v_2, \dots, v_n)$  of component states such that

$$\varphi(\mathbf{v}) = \varphi(v_1, v_2, \dots, v_n) = 0.$$

Here  $v_i = 2, 1$  or  $0$  means that vector  $\mathbf{v}$  component  $i$  is in state *up*, *mid* or *down*, respectively.

A failure set which has  $r$  components in *up*,  $x$  components in *down*, and  $(n - r - x)$  components in *mid* is termed a  $(r; x)$ -*failure set*.#

**Definition 3:** *path set* and  $(r, x)$ -*path set*.

*Path set* is a vector  $\mathbf{w} = (w_1, w_2, \dots, w_n)$  of component states such that

$$\varphi(\mathbf{w}) = \varphi(w_1, w_2, \dots, w_n) = 1.$$

Here  $w_i = 2, 1$  or  $0$  means that vector  $\mathbf{w}$  component  $i$  is in state *up*, *mid* or *down*, respectively.

A path set which has  $r$  components in *up*,  $x$  components in *down*, and  $(n - r - x)$  components in *mid* is termed a  $(r; x)$ -*path set*.#

**Destruction process.** The destruction process has several stages denoted  $0, 1, \dots, n-1$ .

1. Stage  $r$  of destruction process consists of:

- (a) generating a random permutation of  $r$ -th type;
- (b) an initial check of system state, and
- (c) sequential destruction of its *mid* components (i.e. turning them from *mid* to *down*) by moving from left to right.

In (a), according to Definition 1, we generate a random permutation of components numbers, assign to the first  $r$  of them state *up* and to the remaining - state *mid*. For permutation of zero-type all components are in state *mid*.

(b) means checking system state when the  $r$  first components in the permutation are *up* and *all* remaining components are in *mid*. If for a particular  $r$ -permutation  $\pi_r$  the check reveals that the system is already *DOWN*, we say that this permutation has *anchor* equal zero.

Stage (c) is carried out only if the anchor is *not* zero. It consists of turning the components from *mid* to *down* by moving from left to right, and checking the system state after each component destruction.

**Definition 4:** *anchor of an  $r$ -type permutation*.

The anchor of a permutation  $\pi_r$  of  $r$ -th type denoted  $\delta(\pi_r)$  is the number of components which have been turned *down* when the system was for the first time discovered in state *DOWN*.#

If in an  $r$ -type permutation, the system is already *DOWN* when no component has been turned from *mid* to *down*, then this permutation has anchor equal zero.

**Remark 1.** There might exist such  $r$  for which the permutation has no anchor.

It means that after destruction of all  $n - r$  *mid* components the system remains in *UP*. If this happens for a particular  $r^*$ , then by monotonicity property of the system, there will be no anchor for all  $r > r^*$ .

It is important to note that by the same monotone property of the system, for any permutation there might be at most a single transition  $UP \rightarrow DOWN$ , i.e. at most a single anchor. This property is very important for designing an fast search procedure for locating the anchor.#

**Example 1.** Suppose that the system has 7 components. Consider a 3-rd type random permutation

$$\pi_3 = ((2, 2); (1, 2); (7, 2); (6, 1); (4, 1); (5, 1); (3, 1))$$

In this permutation, the first three positions occupy components 2,1 and 7 which are in state 2, i.e. *up*. The remaining components are in state 1: component 6 is on the fourth position (from left to right), component 5 is on sixth position, etc. An initial check of this permutation reveals that the system is *UP*. Suppose that we start turning into *down* the components in *mid* by moving from left to right. Suppose that after doing this to components 6 and 4, the system remained in *UP*. When we turned *down* component 5 the system went *DOWN*. Then the anchor of this permutation  $\delta(\pi_3) = 3$  while the permutation now took the form:

$$\pi_3 = ((2, 2); (1, 2); (7, 2); (6, 0); (4, 0); (5, 0); (3, 1))\#$$

Let  $f_r(y)$  be the probability that the anchor of  $r$ -th type permutation equals  $y$ ,  $y = 0, 1, 2, \dots, n - r$ :

$$f_r(y) = P(\text{In } \pi_r, \text{ the anchor } \delta(\pi_r) = y).$$

Since the total number of permutations for each  $r$  is  $n!$ , and all permutations are equally probable,

$$f_r(y) = \frac{\text{the number of } r\text{-permutations with } \delta(\pi_r) = y}{n!}, y = 0, 1, \dots, n - r. \quad (2)$$



**Definition 5:** *cumulative  $r$ -spectrum.*

$$F_r(x) = \sum_{y=0}^x f_r(y), x = 0, 1, 2, \dots, n - r,$$

is called cumulative  $r$ -spectrum. Obviously,  $F_r(x) \leq 1$ .#

**Remark 2**

Let us assume that all  $r$ -type permutations have anchor equals zero. It means that the numerator of  $f_r(0)$  in (2) will be equal  $n!$  and therefore  $f_r(0) = 1$ ,  $f_r(j) = 0$  for  $j = 1, \dots, n - r$  and  $F_r(0) = 1 = F_r(j)$ ,  $j = 1, \dots, n - r$ .

**Definition 6:** *ternary  $D$ -spectrum.*

The collection of all cumulative  $r$ -spectra  $\mathcal{T}sp = \{F_r(x)\}$  for  $0 \leq r < n$  is called *ternary  $D$ -spectrum*.#

Denote by  $C(r; x)$  the number of all  $(r; x)$  - failure sets in the system. (Let us remind that an  $(r; x)$  -failure set has  $r$  components *up*,  $x$  components *down* and the remaining components in *mid*.)

**Theorem 1**

$$C(r; x) = F_r(x) \cdot \frac{n!}{r!x!(n - r - x)!}. \quad (3)$$

**Proof**

Let us count all  $r$ -type permutations having anchor  $j$ ,  $j \leq x$ . After completing the destruction process, there will be  $n!f_r(j)$  such permutations. Each of them has the following form:

$$((i_1, 2); (i_2, 2); \dots (i_r, 2); (i_{r+1}, 0); \dots; (i_{r+j}, 0); (i_{r+j+1}, 1); \dots; (i_n, 1)).$$

Now let us turn into *down* the components which are on the positions  $r + j + 1, r + j + 2, \dots, r + x$ . We will obtain permutations which have the following form:

$$((i_1, 2); \dots (i_r, 2); (i_{r+1}, 0); \dots, (i_{r+j}, 0); \dots; (i_{r+x}, 0); (i_{r+x+1}, 1); \dots, ((i_n, 1)), (*)$$

i.e. the first  $r$  positions occupy components in *up*, next  $x$  positions - components

in *down* and the remaining - components in *mid*. Obviously, all these permutations are  $(r; x)$ -failure sets. Therefore, the  $r$ -permutations produce

$$n![f_r(0) + f_r(1) + \dots + f_r(x)] = n!F_r(x)$$

$(r; x)$ -failure sets. But each particular  $(r; x)$ -failure set will be repeated  $r!x!(n - r - x)!$  times. Therefore, the number of different  $(r; x)$  - failure sets produced in the above process is

$$F_r(x) \cdot \frac{n!}{r!x!(n - r - x)!} = C_1$$

and therefore  $C(r; x) \geq C_1$ . On the other hand each  $(r; x)$ -failure set has its "representatives" in permutations (\*). Therefore  $C(r; x) = C_1$ . #

**Remark 3.**

Suppose that in the set of  $n$  system components we choose randomly  $r$  components and turn them into *up*. Then we choose randomly  $x$  components out of  $n - r$  remaining and turn them into *down*. Finally, we turn the remaining  $(n - r - x)$  components into *mid*. Call the set resulting from this operation set  $Q$ . There are  $n!/(r!x!(n - r - x)!)$  different and equiprobable ways to create a  $Q$ -type set. Some of these sets will be failure sets. Theorem 1 states that the probability that the set  $Q$  is a failure set equals  $F_r(x) = f_r(1) + \dots + f_r(x)$ . This fact is true no matter what is the stochastic mechanism governing system components transition from *up* to *mid* and from *mid* to *down*.#

Denote by  $V(j; x)$  the number of *path* sets having  $j$  components *up*,  $x$  components *down* and the remaining components in state *mid*. The following Corollary was suggested to the authors by the anonymous Reviewer.

**Corollary.**

$$V(j; x) + C(j; x) = \frac{n}{j!x!(n - j - x)!}. \quad (4)$$

The proof is straightforward. Choose randomly  $j$  components out of  $n$  and turn them into *up*. Chose randomly  $x$  components out of existing  $(n - j)$  components and term them into *down*. Turn the remaining  $(n - j - x)$  into *mid*. The state vector obtained in this way is either a path vector or a failure (cut) vector. Since

the number of  $(j; x)$ -failure sets is  $F_r(x) \cdot \frac{n!}{r!x!(n-r-x)!}$ , the number of  $(j; x)$ -path sets is  $V(j; x) = (1 - F_r(x)) \cdot \frac{n!}{r!x!(n-r-x)!}$ , which proves the corollary. #

**Example 2:**  $s - t$  flow in a triangular network.

The network has three edges  $e_1 = (s, v)$ ,  $e_2 = (v, t)$  and  $e_3 = (s, t)$ . Each edge can be in three states *up*, *mid* and *down*. These states correspond to edge flow capacity 2, 1 and 0, respectively. The system is *UP* if the  $s - t$  flow  $L \geq 2$ . Otherwise the network is *DOWN*. It is easy to see, for example, that for the state vector  $\mathbf{y}_0 = (1, 1, 1)$  (all components are in *mid*), system is *UP*. For example, for  $\mathbf{y}_1 = (2, 0, 1)$  the flow is 1, and therefore the system is *DOWN*.

Let us find the ternary D-spectrum. Note that for three components numbered 1, 2 and 3 we have  $3!=6$  different permutations of their numbers. Below is the list of them ( component states are not given):

$$\pi_1 = ((1, -); (2, -); (3, -)), \pi_2 = ((1, -); (3, -); (2, -)),$$

$$\pi_3 = ((2, -); (1, -); (3, -)), \pi_4 = ((2, -); (3, -); (1, -)),$$

$$\pi_5 = ((3, -); (1, -); (2, -)), \pi_6 = ((3, -); (2, -); (1, -)).$$

Consider now the permutations of 0-type. Recall that all components now are in *mid*. Obviously, initially the system is *UP*. For example,  $L = 2$  for  $\pi = ((1, 1), (2, 1), (3, 1))$ . The initial check reveals that no permutation has anchor equal zero. Let us start the destruction process.

If we destruct the component on the first position in any of the permutations  $\pi_1, \dots, \pi_6$ , the system will go *DOWN*. Therefore the anchor of any  $\pi_i$  is 1, and therefore  $f_0(1) = 1$ . Obviously,  $f_0(2) = f_0(3) = 0$ , and  $F_0(1) = F_0(2) = F_0(3) = 1$ . By Theorem 1,

$$C(0; 1) = 3!/(0!1!2!) = 3; C(0; 2) = 3!/(0!2!1!) = 3; C(0, 3) = 3!/(0!0!3!) = 1.$$

Permutations of  $(r = 1)$ -type have on the first position a component in state *up*. So, now in each  $\pi_i$  the first position is occupied by a component with capacity 2, and next two positions - by components with capacity 1. Obviously, for each  $\pi_i$  the system is initially *UP*. Now destruct the component on the second position in

each of the  $\pi_i$ 's. It is easy to see that in the first four permutations, the system goes *DOWN* because the  $s - t$  flow becomes 1. For  $\pi_5$  and  $\pi_6$ , which have on their first position component  $e_3 = (s, t)$ , the system remains *UP*. Therefore, the anchor equals one with probability  $f_1(1) = 4/6$ . It is seen that for  $\pi_5$  and  $\pi_6$  destruction of the component on third position does *not* bring the system *DOWN*. Therefore,  $f_1(2) = 0$ . Thus  $F_1(1) = 4/6 = F_1(2)$ . Now

$$C(1; 1) = (2/3) \cdot 3!/(1!1!1!) = 4. C(1, 2) = (2/3) \cdot 3!/(1!2!) = 2.$$

All permutations of 2-type have two components in *up* on their first two position. It is easy to check that it guarantees that the system is *UP*. Therefore, our ternary spectrum is a collection of two vectors:

$$T_{sp} = \{(F_0(1), F_0(2), F_0(3)); (F_1(1), F_1(2))\} = \{(1, 1, 1); (2/3, 2/3)\}.\#$$

Theorem 1 opens way to compute system *DOWN* probability for the case that all components are independent and have identical probabilities  $(p_2, p_1, p_0)$  for *up*, *mid* and *down* states, respectively. In that case each  $(r; x)$  - failure set has probability  $p_2^r p_1^{(n-r-x)} p_0^x$  and therefore the probability weight of all such sets equals

$$C(r; x) \cdot p_2^r p_1^{(n-r-x)} p_0^x. \quad (5)$$

## Theorem 2

$$P(DOWN) = \sum_{\{x \geq 0, r \geq 0: 0 \leq r+x \leq n\}} C(r; x) \cdot p_2^r p_1^{(n-r-x)} p_0^x. \quad (6)$$

Formula (6) can be rewritten in an equivalent "dynamic" form to include the time factor. Suppose we have  $n$  independent and identically distributed stochastic processes  $\{\chi_i(t), t > 0, i = 1, \dots, n\}$ . Each  $\chi_i(t)$  is a decreasing, left continuous process with three states: 2, 1 and 0. State 0 is absorbing. Each trajectory of  $\chi_i(t)$  starts at  $t = 0$  in state 2, jumps into state 1 and later gets absorbed in state 0. At any

time instant  $t > 0$ ,  $\chi_i(t)$  is in one of its three states with probabilities  $p_2(t), p_1(t)$  and  $p_0(t)$ , respectively. Obviously

$$p_0(t) + p_1(t) + p_2(t) = 1, t > 0.$$

Let  $\tau_2(i)$  be the sojourn time of  $\chi_i(t)$  in state 2. Then  $P(\tau_2(i) \leq t) = 1 - p_2(t)$ . Let  $\tau_1(i)$  be the sojourn time of  $\chi_i(t)$  in state 1. Then the event  $\{\tau_2(i) + \tau_1(i) \leq t\}$  means that at time  $t + 0$ ,  $\chi_i(t)$  has already left state 1, and is therefore in state 0, i.e.

$$P(\tau_2(i) + \tau_1(i) \leq t) = p_0(t).$$

Note that if at time instant  $t$  the system is *DOWN*, then its failure-free operation time  $\tau_{up}$  does not exceed  $t$ . Then we can write that

$$P(\tau_{up} \leq t) = \sum_{\{x \geq 0, r \geq 0: 0 \leq r+x \leq n\}} C(r; x) [p_2(t)]^r [p_0(t)]^x [1 - p_0(t) - p_2(t)]^{(n-r-x)}. \quad (7)$$

### 3 Example: Hypercube Network

We consider fourth order hypercube  $H_4$ . It has 16 nodes and 32 edges, see Figure 1. Each edge, independently of other edges, can be in three states: *up*, *mid* and *down* with respective probabilities  $p_2, p_1$  and  $p_0$ ,  $p_2 + p_1 + p_0 = 1$ . States *up* and *mid* provide high and medium connection speed, respectively. Edge *down* state means loss of connection. We say that node set  $T_1$  is *strongly connected* if any pair of nodes from this set is connected by a path consisting only of edges providing high speed connection. We say that node set  $T_2$  is *weakly connected* if any pair of nodes in this set can be connected by a path of operational edges. We define *UP* state of our system as presence of strong connection between nodes  $T_1 = \{0, 2, 4, 6, 10, 14\}$  and weak connection between all 16 nodes of  $H_4$ , (In our example  $T_2 = V$ , the set of all nodes). System *DOWN* state is absence of strong connection for  $T_1$  or absence of weak connection for  $T_2$ , or both.

Our main tool for computing  $P(DOWN)$  is the ternary D-spectrum. For this purpose we used Monte Carlo procedure, the algorithmic details of which will be

described in the next section. Now let us mention some general properties of the ternary spectrum. First, it is easy to check from Figure 1 that the minimal number of edges providing strong connection for  $T_1$  equals 5. Therefore, if we have  $r \leq 4$  edges *up* and other edges in *mid*, the system is *DOWN*. Thus the permutations of  $(r; x)$ -type have for  $r = 0, 1, 2, 3, 4$  the anchor equal zero. In the course of simulation it was revealed that if  $r \geq 29$ , the system can not fail. Therefore,  $F_r(x) \equiv 0$  for these  $r$  values. Figure 2 presents a sample of graphs of  $\{F_i(x)\}$ .

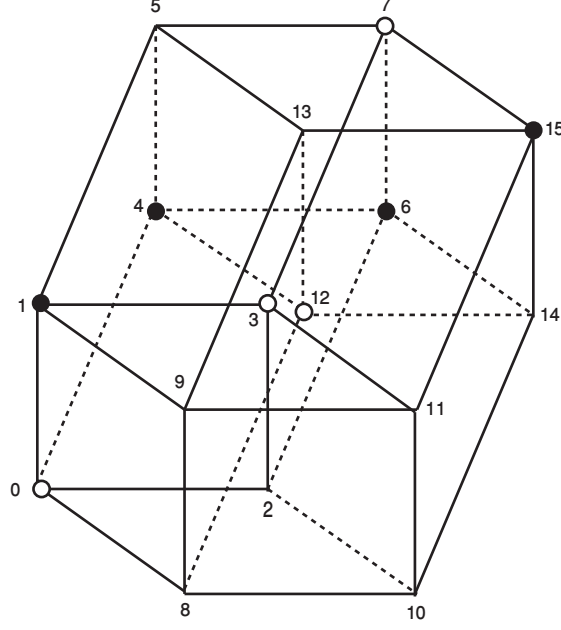
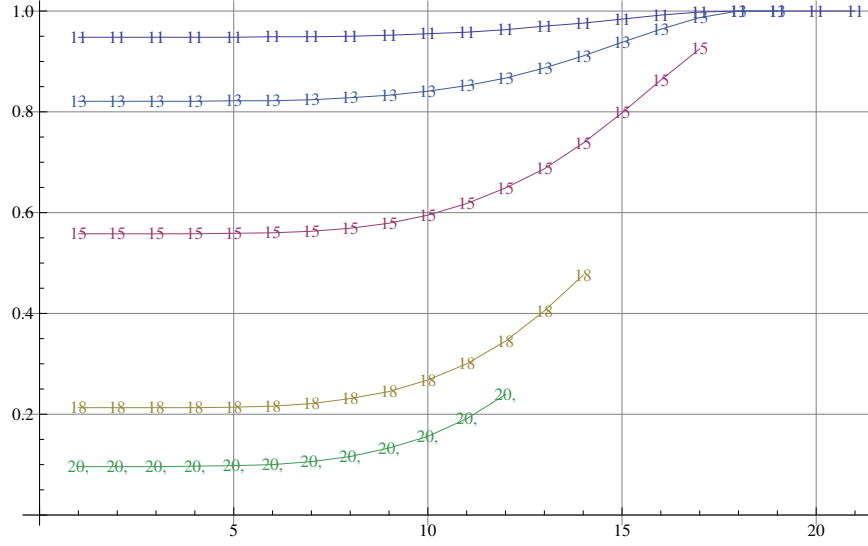


Figure 1: Hypercube of order four.  $T_1 = \{0, 2, 4, 6, 10, 14\}$ ,  $T_2 = V$ .

The D-spectrum has an ordering property following from monotonicity of the system:  $F_r(x) > F_{r+1}(x)$ .

Table 1: Network *DOWN* Probability

$p_2$	$p_1$	$p_0$	$P(DOWN), M = 10^5$	$P(DOWN), M = 5 \cdot 10^5$	CMC, $N = 10^7$
0.5	0.1	0.4	0.57733	0.57769	0.57775
0.5	0.2	0.3	0.49718	0.49802	0.49783
0.5	0.3	0.2	0.46520	0.46616	0.46605
0.5	0.4	0.1	0.45787	0.457881	0.45847
0.6	0.1	0.3	0.25952	0.25954	0.20937
0.6	0.2	0.2	0.20896	0.20935	0.20937
0.6	0.3	0.1	0.19737	0.19779	0.19762
0.7	0.1	0.2	0.07360	0.07336	0.07347
0.7	0.2	0.1	0.05923	0.05917	0.05911
0.8	0.1	0.1	0.01156	0.01148	0.01155

Figure 2: Sample of  $F_i(x)$ . From top to bottom:  $F_{11}, F_{13}, F_{15}, F_{18}, F_{20}$ .

The data on system *DOWN* probability is presented in Table 1. The fourth and fifth columns present the calculation results obtained from formula (6) by using the simulated D-spectrum based on  $M$  runs,  $M = 100,000$  and  $M = 500,000$ . The CPU time for simulating the D-spectrum was 11.5 and 57.2 sec, respectively.

We see that for  $p_2 \leq 0.6$  the system has low reliability -  $P(DOWN) \geq 0.2$ . To provide reliability of about 0.9 or higher,  $p_2$  should be near 0.7 or higher. To assess

the accuracy of the simulation, we made a crude Monte Carlo (CMC) simulation of  $P(DOWN)$  by using  $N = 10^7$  simulation runs for each of 10 combinations of  $(p_2, p_1, p_0)$  shown in Table 1. The CPU time of each CMC run was about 28 sec.

Assuming that the CMC provides a result with negligible error, we see from Table 1 that  $M = 100,000$  provides an absolute error within the limits  $(80 - 10) \cdot 10^{-5}$ , which is quite satisfactory for practical reliability purposes. Let us note that for systems with high reliability (last two rows in Table 1), the absolute error is about 0.0001. The increase of  $M$  leads to overall reduction of estimation error. The absolute errors for  $M = 500,000$  lie in the limits  $[7 \cdot 10^{-5}, 34 \cdot 10^{-5}]$ .

## 4 Calculating the ternary spectrum

The Monte Carlo procedure for calculating the ternary D-spectrum is rather straightforward.

(a) Simulate a random permutation of  $r$ -th type,  $0 \leq r < n$ . Start with  $r = 0$  and proceed by increasing  $r$ . Turn into *up* the first  $r$  components, and into *mid* the remaining components. For each  $r$ , carry out the destruction process by turning into *down* one *mid* component after another by moving from left to right. Check system state after each destruction. If the system is found *DOWN* on the  $k$ -th step, increase by one the  $k$ -th counter:  $M_k \Leftarrow M_k + 1$ . If the initial check reveals *DOWN*, set  $M_0 \Leftarrow M_0 + 1$ . On the other hand, if we completed the  $r$ -th type permutation simulation procedure by turning *all mid* components into *down* and no anchor was discovered, the D-spectra counters should not be updated at all.

(b) Repeat step (a)  $M$  times; compute fractions  $\hat{F}_r(x) = \frac{\sum_{i=0}^x M_i}{M}$ .

(c) Repeat steps (a), (b) for  $r$  running from 0 to  $n - 1$ . Take  $\hat{F}_r(x)$  as the estimate of  $F_r(x)$ .

The most CPU time consuming part of the above procedure is checking system *UP* state after each destruction step. Essential acceleration here can be achieved by appropriate use of so-called DSS-disjoint set structures, see [4], page 30. The question of principal interest is the choice of  $M$ . It depends essentially on the number of components  $n$  and desired estimation accuracy. For practical purposes estimation of system *DOWN* probability with relative error 0.1-0.2 % is quite satisfactory.



Our calculations reveal that for  $M = 100,000$  the relative error lies within these limits. This has been checked by crude Monte Carlo based on  $10^7$  replications. It is worth noting that having  $M = 100,000$  for estimating each  $F_r(x)$  in our example in Section 3, consumes only 11.5 sec of CPU time!

Now several general comments about the above described Monte Carlo procedure. This is a typical *counting* algorithm. The purpose of it is estimating the D-spectrum by means of which it is possible to count failure sets of the system with given structure (number of components in *up*, *mid* and *down*). The fact that due to the monotone property of our system each permutation can have at most one anchor considerably simplifies the calculation algorithm.

Since  $M_i/M$  are just fractions, their estimation in the above algorithm is unbiased and consistent. So are the estimates  $\hat{F}_r(x)$  of  $F_r(x)$ . Therefore the estimation of  $P(DOWN)$  by (6) is also unbiased and consistent.

Let us make some comments regarding the complexity and efficiency of the above Monte Carlo procedure. A permutation can be stored in array of size  $n$ . So, the space complexity is  $O(n)$ . For each  $r \in [0, n - 1]$  we generate  $M$  permutations. Having in mind that permutation generation can be performed in linear time ( $O(n)$ ) and that the anchor of any permutation can be found using binary search procedure in  $O(\log(n))$  time, we conclude that the overall time complexity is equal to  $O(nM \log(n))$ .

Regarding the efficiency, let us note that in this paper we consider an extension to the well known binary terminal network reliability problem. The binary version of this problem belongs to  $\#P$  complexity class [7]. As far as we know, there exists an efficient approximation scheme (FPRAS) for the *all* terminal network reliability but not for the general terminal reliability problem like that considered in this paper. Our general recommendation is to use the estimators of the variance to control the relative error for sample size  $M$ .

It is also worth noting that our D-spectrum method has the following valuable property: in estimating very small  $P(DOWN)$  probabilities by (6) (the rare event case), the phenomenon of variance increase as  $P(DOWN) \rightarrow 0$  is avoided.  $P(DOWN) \rightarrow 0$  takes places for very reliable systems if  $p_2 \rightarrow 1$ . But the variance of estimating the coefficients  $C(r; x)$  in formula (6) *does not depend* in

our spectrum algorithm on probabilistic properties of system components.

## 5 Component Importance and Importance Spectrum

### 5.1 Component importance

In this section we present a very useful reliability characteristic - component importance. We will assume that the system consists of independent components. In binary case when each component has two states *up* and *down*, the most known is the so-called the Birnbaum Importance Measure (BIM). For component  $m$ , it is defined ([1], Chapter 1) as

$$BIM_m = \frac{\partial R(p_1, \dots, p_n)}{\partial p_m}. \quad (8)$$

Using pivotal decomposition, it is easy to obtain that

$$\begin{aligned} BIM_m &= R(p_1, \dots, 1_m, \dots, p_n) - R(p_1, \dots, 0_m, \dots, p_n) = \\ &= G(p_1, \dots, 0_m, \dots, p_n) - G(p_1, \dots, 1_m, \dots, p_n), \end{aligned} \quad (9)$$

where  $R(p_1, \dots, 1_m, \dots, p_n)$  is the reliability of a system with the component  $m$  being *up*, and  $R(p_1, \dots, 0_m, \dots, p_n)$  is the reliability of the system with the component  $m$  being *down*. For convenience we represented  $R(\cdot)$  as  $1 - G(\cdot)$ .

$BIM_m$  has transparent probabilistic meaning: it is the gain in system reliability received from replacing a *down* component  $m$  by an absolutely reliable one.  $BIM_m$ , being partial derivative, gives an approximation to the system reliability increment  $\delta R$  resulted from reliability increment of component  $m$  by  $\delta p_m$ . This increment equals  $\delta R(\cdot) \approx BIM_m \cdot \delta p_m$ . In binary situation, increment of component *up* probability  $p$  by  $\Delta$  means necessarily *decrease* by  $\Delta$  the *down* probability  $q = 1 - p$ . In ternary case there is a complication: suppose we increase  $p_2$  by  $\Delta$ . Then we have to reduce  $p_1$  together with  $p_0$  by the same quantity, and this can be done in several ways. For example, we can reduce  $p_0$  by  $\Delta$  and leave unchanged  $p_1$ . Or we can decrease  $p_1$  by  $\Delta$  and leave  $p_0$  unchanged, or we can reduce both  $p_1$  and  $p_0$  by  $\Delta/2$ , and so on.

It seems to us that the most natural and simple is the first option. Our goal is to derive a formula for ternary importance measure for the case of independent and identical components. This means that each component has the same state distribution vector  $\mathbf{p} = (p_2, p_1, p_0)$ , where  $p_2, p_1$  and  $p_0$  are the probabilities that a component is in *up*, *mid* or *down*, respectively.

**Definition 7:** *ternary importance measure (TIM) of component  $m$ .*

$$TIM_m = G(\mathbf{p}, \dots, 0_m, \dots, \mathbf{p}) - G(\mathbf{p}, \dots, 2_m, \dots, \mathbf{p}). \quad (10)$$

The problem in calculating the component importance is that usually the reliability function is not available in closed analytic form. The *TIM-spectrum* which we define below allows to evaluate TIM's by means of Monte Carlo simulation.

## 5.2 TIM-spectrum

This notion is closely related to the  $r$ -permutation defined in Section 2. Let us remind that in the process of component destruction, this permutation has on its first  $r$  positions the numbers of components in state *up*, the next  $x$  positions occupy component numbers which are in state *down*, and the components on the remaining  $(n - r - x)$  positions are in state *mid*.

Denote by  $Z(r; x, m)$  the number of  $r$ -permutations satisfying the following two conditions:

- (i) If  $x$  elements in the permutation are *down*, then the system is *DOWN*;
- (ii) component  $m$  is among the  $x$  *down* components.

Denote by  $Y(r; x, m)$  the number of  $r$ -permutations satisfying the above condition (i) and the following condition

- (iii): component  $m$  is among the  $r$  *up* components.

**Definition 8:** *ternary importance measure spectra (TIM- spectra).*

The collection  $\{z(r; x, m)\} = \{Z(r; x, m)/n!\}$ ,  $x = 1, \dots, n; m = 1, 2, \dots, n$  is called the *TIM - down -spectrum* of the system.

Similarly, the collection  $\{y(r; x, m)\} = \{Y(r; x, m)/n!\}$ ,  $x = 1, \dots, n; m = 1, 2, \dots, n$  is called the *TIM - up - spectrum* of the system. #

Note that by definition,  $Z(r; x, m)$  permutations produce

$$\frac{Z(r; x, m)}{r!x!(n-r-x)!}$$

$(r; x)$  - failure sets in which component  $m$  is among the *down* components.

Similarly,  $Y(r; x, m)$  permutations produce

$$\frac{Y(r; x, m)}{r!x!(n-r-x)!}$$

$(r; x)$  - failure sets in which component  $m$  is among the *up* components. Then we arrive at the following

**Theorem 3**

$$TIM_m = G(\mathbf{p}, \dots, 0_m, \dots, \mathbf{p}) - G(\mathbf{p}, \dots, 2_m, \dots, \mathbf{p}), \quad (11)$$

where

$$G(\mathbf{p}, \dots, 0_m, \dots, \mathbf{p}) = \sum_{j=0}^{n-1} \sum_{x=1}^{n-j} \frac{z(j; x, m)n!}{j!x!(n-j-x)!} p_2^j p_0^{x-1} p_1^{(n-j-x)}. \quad (12)$$

and

$$G(\mathbf{p}, \dots, 2_m, \dots, \mathbf{p}) = \sum_{j=0}^{n-1} \sum_{x=1}^{n-j} \frac{y(j; x, m)n!}{j!x!(n-j-x)!} p_2^{(j-1)} p_0^x p_1^{(n-j-x)} \# \quad (13)$$

**Example 3.**

Let us consider a small network with three edges  $a, b$  and  $c$ . Edges  $a = (s, v)$  and  $b = (v, t)$  are in series, and are parallel to edge  $c = (s, t)$ . Each edge can be in three states: *up*, *mid* and *down* with respective capacities 2, 1 and 0. System is *UP* if the maximal  $s - t$  flow is at least 2. Otherwise the system is *DOWN*. Below is the list of all 13 *DOWN* states:

$$\mathbf{v}_1 = (0, 0, 0), \mathbf{v}_2 = (0, 0, 1), \mathbf{v}_3 = (0, 1, 0), \mathbf{v}_4 = (0, 2, 0), \mathbf{v}_5 = (1, 0, 0),$$

$$\mathbf{v}_6 = (2, 0, 0), \mathbf{v}_7 = (0, 1, 1), \mathbf{v}_8 = (0, 2, 1), \mathbf{v}_9 = (1, 0, 1), \mathbf{v}_{10} = (2, 0, 1),$$

$$\mathbf{v}_{11} = (1, 1, 0), \mathbf{v}_{12} = (1, 2, 0), \mathbf{v}_{13} = (2, 1, 0).$$

Let us calculate the *TIMs* of all edges using the Definition 8.

By symmetry  $TIM_a = TIM_b$ , and

$$TIM_a = (p_0^2 + 2p_0p_1 + p_1^2 + p_0p_2 + p_1p_2) - (p_0^2 + 2p_0p_1) = p_1^2 + p_0p_2 + p_1p_2.$$

$$TIM_c = p_2^2 + 2p_0p_1 + 2p_0p_2 + p_1^2 + 2p_1p_2.$$

Let us now demonstrate the calculation of  $TIM_a$  using the Importance spectrum and Theorem 3. In total, there are the following six permutations of all components:  $per1=(a,b,c)$ ,

$$per2 = (a, c, b), per3 = (b, a, c), per4 = (b, c, a), per5 = (c, a, b), per6 = (c, b, a).$$

Suppose that  $a$  is *down*. Let  $r = 0$ . Then the following permutations satisfy **(i)** and **(ii)** of the definition of  $Z(r; x, m)$ :  $per1$  and  $per2$ , for  $x = 1$ . Then  $Z(0; 1, a) = 2$ .  $per1, per2, per3, per5$  satisfy **(i)**, **(ii)** - for  $x = 2$ , and then  $Z(0; 2, a) = 4$ . Finally, all permutations satisfy the same conditions for  $x = 3$ . Then  $Z(0; 3, a) = 6$ .

Now let  $r = 1$ . The above conditions are satisfied by  $per5$  for  $x = 1$ , by  $per3$  and  $per4$  for  $x = 2$ . Therefore,  $Z(1; 1, a) = 1$ ,  $Z(1; 2, a) = 2$ .

For  $r = 2$ , all permutations give the *UP* state.

Suppose now that it is given that  $a$  is *up*. We have the following permutations satisfying **(i)**, **(iii)** of the definition of  $Y(r; x, m)$ .

The case  $r = 0$  is not relevant. Let  $r = 1$ . For  $per1, per2$  and  $x = 1$  we have  $Y(1; 1, a) = 2$ . For  $x = 2$ , we get  $per2$  and  $per2$ , and  $Y(1; 2, a) = 2$ . For  $r = 2$  we have *UP* in all six permutations.

Now by Theorem 3 we obtain

$$TIM_a = \left( \frac{2}{0!1!2!}p_1^2 + \frac{4}{0!2!1!}p_0p_1 + \frac{6}{0!3!0!}p_0^2 + \frac{1}{1!1!1!}p_2p_1 + \frac{2}{1!2!}p_2p_0 \right) -$$

$$(\frac{2}{1!1!1!}p_0p_1 + \frac{1}{1!2!}p_0^2) = p_1^2 + p_1p_2 + p_0p_2. \#$$

For systems having more than four components, manual calculations of the *TIMs* becomes too complicated. There is however an efficient way of numerical estimation of ternary importance spectrum by means of Monte Carlo simulation. Let us outline briefly how this can be done. Look at step (a) of the procedure described in Section 4. It contains the simulation of an  $r$ -permutation and subsequent destruction process. Suppose that this process reveals the system in *DOWN* state after exactly  $x$  steps. Then check whether a fixed component  $m$  is among these  $x$  destructed components. If "Yes", add 1 to the counter of  $Z(r; x, m)$ . The calculation of  $Y(r; x, m)$  goes along similar lines. In general, this reminds the calculation of the BIM-spectra for binary system, see the description in [4], Chapter 10. Summing up, the calculation of the ternary importance spectrum is carried out by means of rather small modification of the basic algorithm for computing the ternary spectrum.

### Acknowledgments

The authors express their sincere gratitude to the Reviewer for his/her valuable and constructive critical remarks and suggestions.

### References

- [1] Barlow, R.E. and Frank Proschan, (1975). *Statistical Theory of Reliability and Life Testing*. Holt, Rinehart and Winston, Inc.
- [2] Elperin, T., Gertsbakh, I.B. and M.Lomonosov. (1991). Estimation of network reliability using graph evolution models. *IEEE Transactions on Reliability*, **40**(5), 572 - 581.
- [3] Gertsbakh, I., Rubinstein, R., Shpungin Y. and R. Vaisman. (2014). Permutational methods for performance analysis of stochastic flow networks. *Probability in Engineering and Informational Sciences*, 28(01),21-38.
- [4] Gertsbakh, I. and Shpungin, Y. (2009). *Models of Network Reliability: Analysis, Combinatorics and Monte Carlo*, CRC Press.
- [5] Gertsbakh, I. and Y. Shpungin. (2011). *Network Reliability and Resilience*, Springer Briefs in Electrical and Computer Engineering, Springer.
- [6] Gertsbakh, I. and Y. Shpungin.(2012). Spectral approach to reliability eval-

uation of flow networks, *Proceedings of the European Modeling and Simulation Symposium*, 68-73.

[7] Karger,D.R.(1996). A randomized fully polynomial scheme for all terminal network reliability problem. In *SIAM Journal on Computing*, 11-17.

[8] Newman, M.E.J.(2010). *Networks. An Introduction*. Cambridge University Press.

[9] Ramirez-Marquez, Jose and David W. Coit.(2005). A Monte Carlo simulation approach for approximating multi-state two-terminal reliability. *Reliability Engineering and System Safety*, **87**, Issue 2, 253-264.

[10] Samaniego, F.J. (1985). On closure under IFR formation of coherent systems. *IEEE Transaction on Reliability*, **34**: 69-72.

[11] Samaniego, F. J. (2007). *System Signatures and Their Application in Engineering Reliability*, Springer.