# Evaluating downside risks in reliable networks

Megha Sharma
Diptesh Ghosh

**W.P. No. 2009-09-02**
September 2009



**INDIAN INSTITUTE OF MANAGEMENT**
**AHMEDABAD – 380015**
**INDIA**

# Evaluating downside risks in reliable networks

Megha Sharma[1]
Diptesh Ghosh[2]

**Abstract**

Reliable networks are those in which network elements have a positive probability of failing. Conventional performance measures for such networks concern themselves either with expected network performance or with the performance of the network when it is performing well. In reliable networks modeling critical functions, decision makers are often more concerned with network performance when the network is not performing well. In this paper, we study the single-source single-destination maximum flow problem through reliable networks and propose two risk measures to evaluate such downside performance. We propose an algorithm called COMPUTE-RISK to compute downside risk measures, and report our computational experience with the proposed algorithm.

Keywords: Network Flows; Reliable Networks; Maximum Flows.

## 1 Introduction

A network is a combinatorial structure in which one is given a finite node set $V = \{1, 2, \ldots, n\}$ and a finite arc set $A \subseteq V \times V$ connecting nodes in $V$ to other nodes in $V$. Arcs in a network conventionally have two properties; a strictly positive capacity value denoting the maximum amount of flow that can be sent along the arc, and a cost value denoting the cost of sending a unit of flow through the arc. A node $s \in V$ is designated as a source node and another node $t \in V$ is designated as a terminal node. Network flow problems are concerned with sending flow from the source node to the terminal node. These flows are called $s$-$t$ flows. A common network flow problem is the maximum $s$-$t$ flow problem, in which, given a network one needs to determine the maximum amount of $s$-$t$ flow that can be sent through the network. Ahuja et al. [1] provides a comprehensive treatment on networks and network flow problems including the maximum flow problem.

While modeling practical networks, one needs to account for the fact that network components can become non-functional from time to time. This is done by associating a reliability value to each arc. The reliability value denotes the probability that the arc would be functional at a given point in time. Networks which incorporate reliability values for their arcs are called reliable networks. In this paper we will assume that the arc reliabilities are independent of each other. A reliable network exists in one of $2^{|A|}$ network states, where in each state, only a subset of the arcs in $A$ are functional. The probability of the network being in a particular state is a function of the reliability values of the arcs and the set of functional arcs in the state. The maximum $s$-$t$ flow, differs from state to state, and hence is a random variable in reliable networks.

There is considerable literature on the evaluation of maximum $s$-$t$ flows in reliable networks. The most popular measure used is the expected maximum $s$-$t$ flow through the network (see, e.g., Ball et al. [2]). Another popular measure is obtained by defining a threshold value of flow, and finding

[1]P&QM Area, IIM Ahmedabad. Email: meghas@iimahd.ernet.in
[2]P&QM Area, IIM Ahmedabad. Email: diptesh@iimahd.ernet.in

the probability that the network will allow at least that much $s$-$t$ flow (see, e.g., Lee [3], Aggarwal et al. [4]). Both these measures have a limitation when evaluating the performance of networks that model critical processes. In such networks, one important aspect of network performance is its flow carrying capacity of the network when it is in a downside state, i.e., one in which many of the arcs are not functioning, and the network is not capable of carrying much flow. Neither of the measures mentioned above explicitly quantify the behavior of networks in their downside states. The only measure in the literature that deals with downside states is the 2-terminal connectivity measure, a connectivity measure which evaluates the probability that $s$ is connected to $t$ through a directed path in the network (see, e.g., Aggarwal et al. [5], Yarlagadda and Hershey [6]).

Alternate and equally meaningful measures for evaluating the performance of reliable networks modeling critical applications would be ones that evaluate the performance of the network in its downside states. We propose a *downside risk* (DsR) measure, in which one is given a value $p$, $0 \leq p \leq 1$, and the measure is the maximum value of $s$-$t$ flow attainable in the worst $100p\%$ of the network states. We also propose a *conditional downside risk* (CDsR) measure, in which one is given a value $p$, $0 \leq p \leq 1$, and the measure is the expected value of the maximum $s$-$t$ flow in the worst $100p\%$ of the network states. These measures are motivated by the value at risk (VaR) and conditional value at risk (CVaR) measures common in the financial literature (see, e.g., Jorion [7]).

Computing $100p\%$ DsR and $100p\%$ CDsR values of reliable networks requires the computation of the probability mass function of the maximum $s$-$t$ flow in the network for the bottom $100p\%$ of the network states. Existing algorithms generate the complete probability mass function either by pre-generating all possible minimal cuts in the network (see, e.g., Patra and Misra [8]) or by pre-generating all possible $s$-$t$ paths in the network (see, e.g. Patra and Misra [8]), or by complete enumeration of the network state space (see, e.g., Alexopoulos [9]). Since both the number of minimal cuts and the number of possible $s$-$t$ paths can grow exponentially with the number of arcs in a network (see Ball et al. [2]), these methods are not practical for network with more than a few arcs. In existing algorithms not much time saved in requiring the algorithm to compute a part of the probability mass function rather than the whole function. In addition, all papers that propose algorithms based on pre-generating all $s$-$t$ paths assume acyclic networks.

In Section 2 of this paper, we propose the COMPUTE-RISK algorithm which, given a value of $p$, $0 \leq p \leq 1$, computes the probability mass function of the maximum $s$-$t$ flow in the bottom $100p\%$ of the network states. It is based on state-space enumeration, but incorporates two enhancements. If $p$ is set to 1, the algorithm computes the complete probability mass function of the maximum $s$-$t$ flow. The time taken by this algorithm increases significantly with increasing values of $p$. In Section 3 we present the results of computational experiments with the COMPUTE-RISK algorithm. Section 4 summarizes the main contributions of the paper and presents directions for future research.

## 2 COMPUTE-RISK, an algorithm for evaluating risk measures

The COMPUTE-RISK algorithm to compute $100p\%$ DsR and $100p\%$ CDsR is a state-space enumeration algorithm. Given a reliable network, it first identifies the network states that can carry a maximum $s$-$t$ flow of zero units. It computes the probability that the network will be in one of these states. If the probability exceeds $100p\%$, then it terminates and outputs the value of $100p\%$ DsR and $100p\%$ CDsR as zero. Otherwise, it iterates until the probability that the network will be in one of the states identified by it is at least $100p\%$. In each iteration, it finds the minimum value of maximum $s$-$t$ flow among the as yet unidentified states, and identifies all states that have this value for maximum $s$-$t$ flow. Since in the process, it generates the probability mass function of the states that it identifies, we can calculate $100p\%$ DsR and $100p\%$ CDsR values.

We start the discussion of our algorithm by proposing an algorithm that finds the probability that a reliable network $N = (V, A, s, t)$ assumes a state in which the maximum $s$-$t$ flow is zero. To do this, we first introduce some terminology to describe the algorithm that we propose.

A reliable network is said to exist in one of many states. A *network state* of the reliable network $N$ is formally defined as a representation of its functional part at any point in time. It is represented by a network $N_S = (V, A_S, s, t)$ where $A_S \subseteq A$ is the set of arcs which are functional in the network state. The probability of observing $N$ in state $N_S$ is given by

$$\prod_{a_i \in A_S} p_i \prod_{a_j \in A \setminus A_S} (1 - p_j),$$

where $p_k$ is the reliability of arc $a_k$ for each $a_k \in A$.

A *Hasse diagram* for the network $N$ is a directed graph with $2^{|A|}$ nodes, where a node represents a state of the network, and an arc connects the node representing state $N_{S_1} = (V, A_{S_1}, s, t)$ to the node representing $N_{S_2} = (V, A_{S_2}, s, t)$ if and only if $A_{S_2}$ is obtained by adding one arc to $A_{S_1}$.

A *cut* in the network $N = (V, A, s, t)$ is defined as a minimal set of arcs $A_C \subseteq A$ such that there does not exist a path from $s$ to $t$ in the network $(V, A \setminus A_C, s, t)$. A *set of cuts* is a set whose elements are cuts. An *arc-disjoint set of cuts* is a set of cuts in which no two member cuts include the same arc.

**Example 1** Figure 1 shows a network with five arcs and its Hasse diagram. The node in the Hasse diagram corresponding to a state $N_S = (V, A_S, s, t)$ is labeled by the arc set $A_S$. Arcs $a$ and $b$ together form a cut of the network, as do arcs $a$ and $e$ together, and $d$ and $e$ together. Sets $\{\{a, b\}, \{d, e\}\}$ and $\{\{a, e\}\}$ are both arc disjoint sets of cuts for the network while the set $\{\{a, b\}, \{a, e\}\}$ is not. ∎

The following propositions concerning network states of reliable networks are immediate.

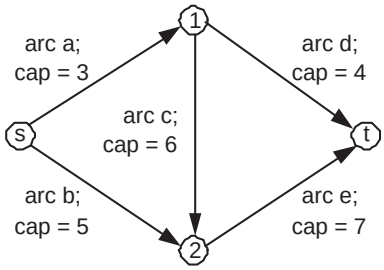
**Propositon 1** *Consider a sequence of states $N_1, N_2, \ldots, N_k$, in which for any pair of states $N_i = (V, A_i, s, t)$ and $N_j = (V, A_j, s, t)$, $i < j \iff A_i \subset A_j$. Then if $i < j$, the maximum $s$-$t$ flow in state $N_i$ cannot exceed the maximum $s$-$t$ flow in state $N_j$.*

**Proof:** The proof is trivial by contradiction. ∎

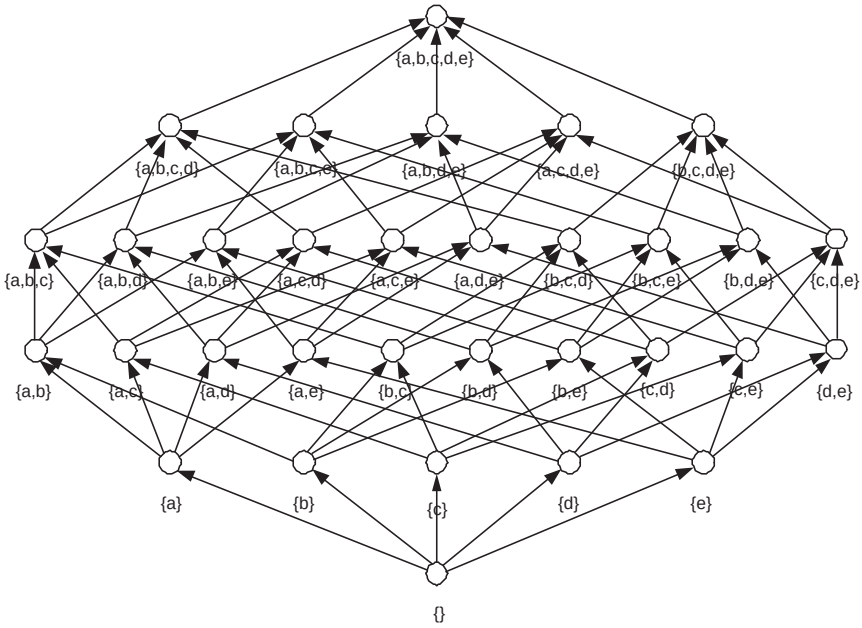
**Propositon 2** *Consider a reliable network and a Hasse diagram of the network. Let $T$ be the set of nodes in the Hasse diagram representing network states which cannot transmit flow from $s$ to $t$. Then there exists a tree in the Hasse diagram whose node set is $T$.*

**Proof:** This is a direct consequence of Proposition 1. ∎

Based on Proposition 2, it is easy construct an explicit enumeration algorithm to obtain the probability that the network exists in a state which does not allow a positive maximum $s$-$t$ flow. A naïve implementation of this algorithm is computationally expensive, since a large number of network states represented in the Hasse diagram will need to be evaluated to determine the maximum $s$-$t$ flow in them. We describe two modifications to the naïve implementation which, taken together, speed up the enumeration and associated probability computations.




(a) A five arc network




(b) The Hasse diagram of the network above

Figure 1: A network and its Hasse diagram

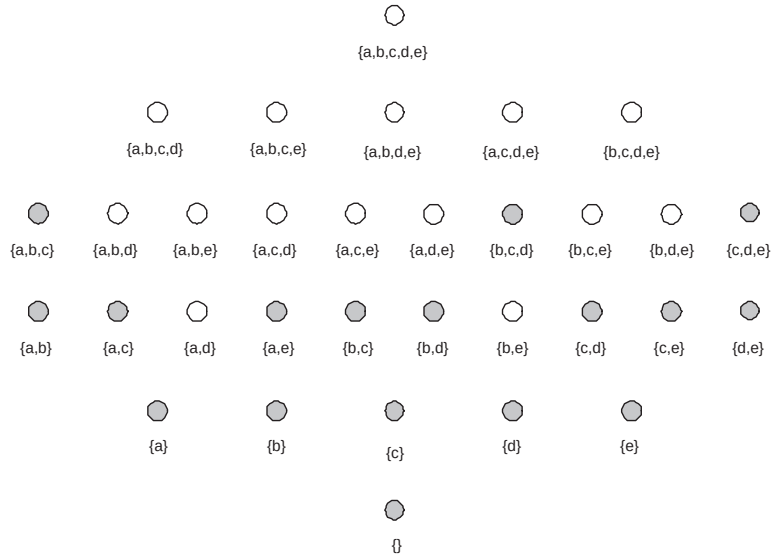
## Using cuts to reduce computation

Consider a cut $A_C = \{a_{[1]}, a_{[2]}, \ldots, a_{[k]}\}$ in a reliable network. Any network state that does not include a single arc from $A_C$ clearly does not carry any $s$-$t$ flow. Regardless of the number of such states, the aggregate probability of the network being in any one of those states can be computed as

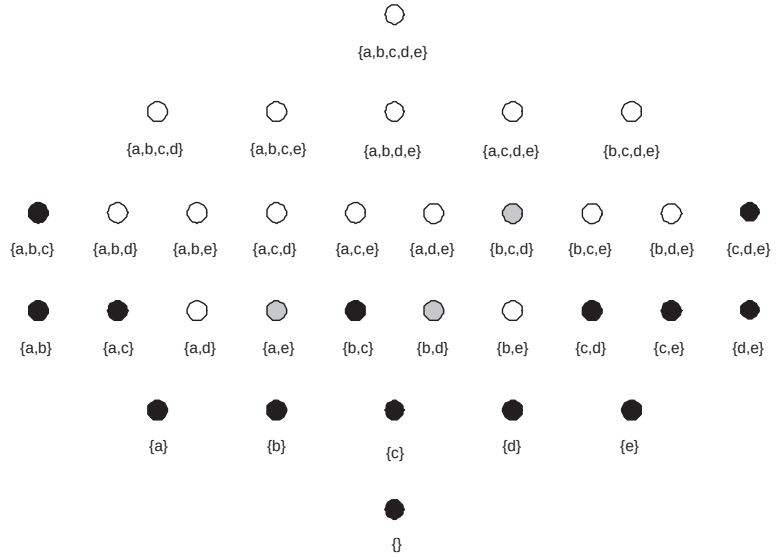
$$\prod_{1 \leq i \leq k} (1 - p_{[i]}).$$

If $\mathcal{C}$ is a maximal arc-disjoint set of cuts in the network, then the aggregate probability of the network being in a state that does not contain any arc included in at least one member of $\mathcal{C}$ can be easily computed using the inclusion-exclusion principle (see, e.g. Pitman [10]). This observation allows us to significantly reduce the time required to compute the aggregate probability of states represented in a Hasse diagram which do not carry a non-zero $s$-$t$ flow.

**Example 2** Consider the network shown in Figure 1(a). A maximal arc-disjoint set of cuts for this network is $\mathcal{C} = \{\{a, b\}, \{d, e\}\}$. The states in the Hasse diagram shown in Figure 1(b) which do not carry any $s$-$t$ flow are those represented by the gray colored nodes in Figure 2(a). The states that do not contain a single arc in $\mathcal{C}$ are represented by the nodes colored black in Figure 2(b). From the preceding discussion, we know that these states can be identified without solving maximum $s$-$t$ flow




(a) The network states corresponding to the nodes colored gray have zero $s$-$t$ flow




(b) Using cuts {{a,b},{d,e}} we know that the network states corresponding to the nodes colored black have zero $s$-$t$ flow

Figure 2: Use of cuts to speed computations

problems in them. The probability of the network being in one of these states can also be computed using the inclusion-exclusion principle. ∎

Note that the number of cuts in a maximal arc-disjoint set of cuts is bounded above by the number of arcs in the network. Hence speed ups using maximal arc-disjoint set of cuts are practical for reasonably sized networks. Note also that small cardinality cuts are better than large cardinality cuts in speeding up the probability calculations. The MAX-CUTSET algorithm (Algorithm 1) is a greedy algorithm to generate a maximal arc-disjoint set of cuts.

**Algorithm 1 MAX-CUTSET**: A greedy algorithm to generate a maximal arc-disjoint set of cuts in a network

**Input:** Network $N = \{V, A, s, t\}$.

**Output:** Maximal arc-disjoint set of cuts $\mathcal{C}$.

**Steps:**

Step 1: Set $\mathcal{C} \leftarrow \emptyset$. Set all arc capacities to 1 unit. Compute the minimum cut $A_C$ in $N$. Let $c \leftarrow$ capacity of the cut $A_C$. If $c = 0$, output $\emptyset$ and exit. Else go to Step 2.

Step 2: If $c \geq M$, output $\mathcal{C}$ and terminate. Else set $\mathcal{C} \leftarrow A_C$ go to Step 3.

Step 3: In the network $N$, replace the capacity of each arc in $A_C$ with $M$. Compute the minimum cut $A_C$ in $N$. Let $c \leftarrow$ capacity of the cut $A_C$. Go to Step 2.

**Algorithm 2 WARMSTART**: An algorithm to warmstart the calculation of maximum $s$-$t$ flow

**Input:** Network states $N_i = \{V, A_i, s, t\}$ and $N_j = \{V, A_j, s, t\}$, such that $A_i \subset A_j$, and $A_j \setminus A_i = \overline{A}$; maximum $s$-$t$ flow $f$ in $N_i$, and residual network $N_i^r = (V, A_i^r, s, t)$ after calculating the maximum $s$-$t$ flow in $N_i$.

**Output:** Maximum $s$-$t$ flow in $N_j$.

**Steps:**

Step 1: Set $maxflow \leftarrow f$; and set $N^r = (V, A_i^r \cup \overline{A}, s, t)$. Go to Step 2.

Step 2: If there is no augmenting path in$N_r$, output $maxflow$ and terminate. Else go to Step 3.

Step 3: Find an augmenting path in $N_r$, and set $f_a \leftarrow$ maximum amount of flow that can be routed from $s$ to $t$ through this path. Set $maxflow \leftarrow maxflow + f_a$. Update $N_r$ after routing this flow from $s$ to $t$. Go to Step 2.

## Warmstarting computations for network states

Consider two network states $N_i = (V, A_i, s, t)$ and $N_j = (V, A_j, s, t)$ of a reliable network $N = (V, A, s, t)$ such that $A_j = A_i \cup \overline{A}$, $\overline{A} \subset A$. Then the residual network obtained after computing the maximum $s$-$t$ flow in state $N_i$ can be used to compute the maximum $s$-$t$ flow in state $N_j$ efficiently. We call this process *warmstarting* and say that the calculation of maximum $s$-$t$ flow in $N_j$ has been warmstarted from $N_i$. The WARMSTART algorithm (Algorithm 2) is a procedure for warmstarting the calculation of maximum $s$-$t$ flow in state $N_j$ from $N_i$. Such warmstarting is particularly effective when one needs to compute the maximum $s$-$t$ flow in a large number of states of a reliable network whose arc sets have a large pair-wise intersection (see Sharma and Ghosh [11]).

Given the MAX-CUTSET and WARMSTART algorithms we construct the ZERO-FLOW algorithm (Algorithm 3) to compute the probability $p_0$ that a reliable network $N = (V, A, s, t)$ will be in a state which does not allow a positive $s$-$t$ flow. In Step 1 the algorithm uses MAX-CUTSET to generate a maximal set of cuts $\mathcal{C}$ in the network. Using the inclusion-exclusion principle, it computes the probability of states that do not have any of the member arcs of at least one cut in $\mathcal{C}$. It initializes $p_0$ with this value. In Step 2 of the algorithm, network states are enumerated as a tree rooted at $(V, \{\}, s, t)$. In this approach a child state is generated from a parent state by adding an arc to the set of arcs in the parent state, and backtracking is achieved by removing that arc from the child state. Apart from $p_0$, Algorithm ZERO-FLOW also outputs a set of network states $S$ which contain the network state allowing the least positive maximum $s$-$t$ flow, and the maximum $s$-$t$ flow

**Algorithm 3 ZERO-FLOW**: An algorithm to compute the probability that a reliable network will not permit a non-zero $s$-$t$ flow

**Input:** A reliable network $N = \{V, A, s, t\}$.

**Output:** Probability $p_0$ that $N$ will be in a state that does not allow positive $s$-$t$ flow; set $S$ of network states including one which has the minimum value of non-zero maximum $s$-$t$ flow.

**Steps:**

Step 1: Set $S \leftarrow \emptyset$. Set $\mathcal{C} \leftarrow$ MAX-CUTSET($N$). Use the inclusion-exclusion principle to find $p$, the probability that the network will be in a state that does not include any arc in a cut in $\mathcal{C}$. Set $p_0 \leftarrow p$. Go to Step 2.

Step 2: Generate an enumeration tree starting from the network state $(V, \{\}, s, t)$. For any state $N_S = (V, A_S, s, t)$ generated, fathom the state if

(a) there exists a cut in $A_C \in \mathcal{C}$ such that the state does not contain any arc in $A_C$; or

(b) A non-zero flow is possible from $s$ to $t$.

If $N_S$ is fathomed due to rule (b), and if $A_S$ is not a superset of the arc set of any other state in $S$, then $S \leftarrow S \cup \{N_S\}$. If after modifying $S$ in this manner, it is found that there exists a state $N'_S \in S$ whose arc set is a superset of any other state in $S$, then $S \leftarrow S \setminus \{N'_S\}$ If $N_S$ is not fathomed, then $p_0 \leftarrow p_0 + \prod_{a_i \in A_S} p_i \prod_{a_j \in A \setminus A_S} (1 - p_j)$.

Step 3: For each state $N_S \in S$, compute the maximum $s$-$t$ flow allowed in $N_S$ and associate it with $N_S$ in $S$.

Step 4: Output $(p_0, S)$ and terminate.

that they allow. In order to keep the cardinality of $S$ small, the set $S$ does not include any state whose arc set is a superset of that of any other state in $S$.

The ZERO-FLOW algorithm can, with minor modifications be used to compute the aggregate probability of states which allow the same maximum $s$-$t$ flow as a given state. We call this algorithm SAME-FLOW and describe it in Algorithm 4.

We now describe the COMPUTE-RISK algorithm (Algorithm 5) which takes a reliable network $N = (V, A, s, t)$ and a value $p$, $0 \leq p \leq 1$, as input, and outputs the $100p$% DsR and the $100p$% CDsR for $N$. The algorithm maintains two lists, called *distribution* and $frontier$. *distribution* stores the partial probability mass function of the maximum $s$-$t$ flow through the network. $frontier$ stores a short-list of network states to be examined by the algorithm. The states in $frontier$ are all capable of transmitting at least the amount of $s$-$t$ flow as is allowed by the state currently being examined by the algorithm. No state in $frontier$ has an arc set that is a superset of the arc set in any other state in $frontier$. $frontier$ also stores the maximum $s$-$t$ flow possible in each of its member states. The COMPUTE-RISK algorithm also maintains a number $p_c$ which stores the aggregate of the probability of states already accounted for in *distribution*.

In Step 1 of the COMPUTE-RISK algorithm, *distribution* and $frontier$ are both initialized to empty sets, and $p_c$ is initialized to 0. In Step 2, the probability of the network being in states that allow no positive $s$-$t$ flow is computed using ZERO-FLOW, and *distribution*, $frontier$, and $p_c$ are adequately updated. In Step 3, COMPUTE-RISK removes all network states from $frontier$ for which the value of the maximum $s$-$t$ flow is the minimum among all states in $frontier$. It adds these states to a separate set called $S^\star$. Let the maximum $s$-$t$ flow in all the states of $S^\star$ be $f_{S\star}$. In Step 4 the algorithm aggregates the probabilities of all states in $N$ whose arc sets are supersets of at least one of the members of $S^\star$. Since there may be states whose arc sets are supersets of

**Algorithm 4 SAME-FLOW**: An algorithm to compute the aggregate probability of network states in a Hasse diagram that have the same maximum $s$-$t$ flow as the state at the root of the diagram

**Input:** A reliable network $N = \{V, A, s, t\}$; a network state $N_{S_0}$; and a Hasse diagram with a node representing $N_{S_0}$ as root.

**Output:** Aggregate probability $p_{S_0}$ of states represented in the Hasse diagram that have the same maximum $s$-$t$ flow as $N_{S_0}$; set $S$ of network states including one which has the minimum value of maximum $s$-$t$ flow among states that do not have the same maximum $s$-$t$ flow as $N_{S_0}$.

**Steps:**

Step 1: Set $S \leftarrow \emptyset$. Let $N^r_{S_0} = (V, A^r_{S_0}, s, t)$ be the residual network obtained after computing the maximum $s$-$t$ flow in $N_{S_0}$. Replace each state $(V, A_j, s, t)$ represented in the Hasse diagram with $(V, A_j \setminus A_{S_0} \cup A^r_{S_0}, s, t)$ Set $\mathcal{C} \leftarrow$ MAX-CUTSET$(N^r_{S_0})$.

Step 2: Use the inclusion-exclusion principle to find $p$, the total probability of occurence of states in the Hasse diagram that do not include any arc in a cut in $\mathcal{C}$. Set $p_{S_0} \leftarrow p$. Go to Step 2.

Step 3: Generate an enumeration tree starting from the network state $N^r_{S_0}$. For any state $N_S = (V, A_S, s, t)$ generated, fathom the state if

(a) there exists a cut in $A_C \in \mathcal{C}$ such that $A_S$ does not contain any arc in $A_C$; or

(b) A non-zero flow is possible from $s$ to $t$ through $A_S$.

If $N_S$ is fathomed due to rule (b), and if $A_S$ is not a superset of the arc set of any other state in $S$, then $S \leftarrow S \cup \{N_S\}$. If after modifying $S$ in this manner, it is found that there exists a state $N'_S \in S$ whose arc set is a superset of any other state in $S$, then $S \leftarrow S \setminus \{N'_S\}$ If $N_S$ is not fathomed, then $p_0 \leftarrow p_0 + \prod_{a_i \in A_S} p_i \prod_{a_j \in A \setminus A_S} (1 - p_j)$.

Step 4: For each state $N_S \in S$, warmstart the computation of the maximum $s$-$t$ flow allowed in $N_S$ from $N_{S_0}$ and associate it with $N_S$ in $S$.

Step 5: Output $(p_{S_0}, S)$ and terminate.

those of more than one states in $S^\star$, care is exercised to prevent double counting. At the end of this step, the algorithm has computed the probability $p_{f_{S\star}}$ that the network exists in a state that allows a maximum $s$-$t$ flow of $f_{S\star}$. It has also discovered states of the network that are not members of $frontier$ but which allow more $s$-$t$ flow than $f_{S\star}$. It adds these states to $frontier$, and then removes any state from $frontier$ whose arc sets are supersets of arc sets of some other state in $frontier$. The algorithm then adds the tuple $(f_{S\star}, p_{f_{S\star}})$ to *distribution*. If it finds that the total probability accounted for in *distribution* is at least $p$, then it stops and outputs appropriately calculated $100p\%$ DsR and $100p\%$ CDsR values for $N$ in Step 6. Otherwise, it returns to Step 3 and generates a new $N_{S\star}$ set.

We illustrate the working of the COMPUTE-RISK algorithm in Example 3 on the network shown in Figure 1(a).

**Example 3** Consider the network in Figure 1(a). Let the reliability of each arc in the network be 0.8. Suppose we want to find the 20% DsR and 20% CDsR for this network. We input the network and a value of 0.2 to the COMPUTE-RISK algorithm. Step 1 of the algorithm initializes *distribution* and $frontier$ to empty lists and sets $p_c$ to 0. In Step 2 of the algorithm, it invokes ZERO-FLOW. ZERO-FLOW returns $p_0 = 0.109$ after evaluating the states corresponding to the states colored gray in Figure 2(a). It also returns the set $S$ which consists of the states with arc sets $\{a, d\}$, $\{b, e\}$, $\{a, c, e\}$, and $\{b, d, e\}$. (States with arc sets $\{a, b, d\}$, $\{a, b, e\}$, $\{a, c, d\}$, $\{a, d, e\}$, $\{b, c, e\}$, $\{a, b, c, d\}$, $\{a, b, c, e\}$, $\{a, c, d, e\}$, and $\{b, c, d, e\}$ are not in $S$ since their arc sets are supersets of the arc sets of

**Algorithm 5 COMPUTE-RISK**: An algorithm to compute the bottom $100p\%$ of the distribution of $s$-$t$ flow

**Input:** A reliable network $N = \{V, A, s, t\}$; a number $p$, $0 \leq p \leq 1$.

**Output:** The $100p\%$ DsR and $100p\%$ CDsR for $N$.

**Steps:**

Step 1: Set $distribution \leftarrow \emptyset$, $frontier \leftarrow \emptyset$, and $p_c \leftarrow 0$. Go to Step 2.

Step 2: Set $(p_0, S) \leftarrow$ ZERO-FLOW($N$); If $p_0 \geq p$, output $100p\%$ DsR $= 100p\%$ CDsR $= 0$ and terminate. Else set $distribution \leftarrow distribution \cup \{(0, p_0)\}$, $frontier \leftarrow S$, $p_c \leftarrow p_0$, and go to Step 3.

Step 3: Choose the set $S^\star$ of states for which the value of the maximum $s$-$t$ flow is minimum among states in $frontier$. Let $f_{S\star}$ be the value of the maximum $s$-$t$ flow in any $N_i \in S^\star$. Set $frontier \leftarrow frontier \setminus S^\star$. Go to Step 4.

Step 4: For each state $N_i \in S^\star$, create a Hasse diagram $H_{N_i}$. If there are states of $N$ which are present in more than one of the Hasse diagrams, then remove the nodes corresponding to these states from all but one of the Hasse diagrams. For each state $N_i \in S^\star$, let $(p_i, S_i) \leftarrow SAME{-}FLOW(N, N_i, H_{N_i})$. Set $distribution \leftarrow distribution \cup \{(f_{S\star}, \sum_i p_i)\}$, $frontier \leftarrow frontier \cup \{\cup_i S_i\}$, and $p_c \leftarrow p_c + \sum_i p_i$. Remove all sets from $frontier$ whose arc sets are supersets of the arc sets of some other states in $frontier$. Go to Step 5.

Step 5: If $p_c < p$ go to Step 3. Else, replace the the tuple $(f_{S\star}, \sum_i p_i)$ in $distribution$ with $(f_{S\star}, \sum_i p_i - (p_c - p))$ and go to Step 6.

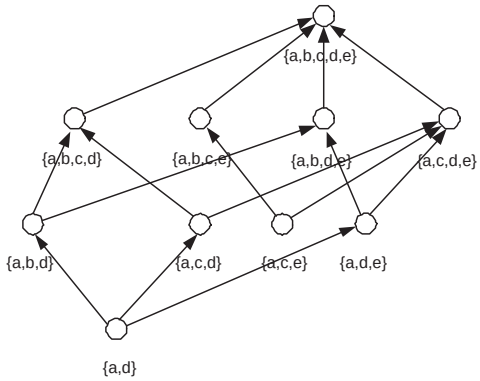
Step 6: Let $(f^\star, \cdot) = \max\{(f, \cdot) : (f, \cdot) \in distribution\}$. Output $f^\star$ as the $100p\%$ DsR for the reliable network $N$. Let $p^\star = \{p : (f^\star, p^\star) \in distribution\}$. Replace $(f^\star, p^\star)$ in $distribution$ with $(f^\star, p - p_c)$. Output

$$\sum_{(f_i, p_i) \in distribution} f_i \times p_i / p$$

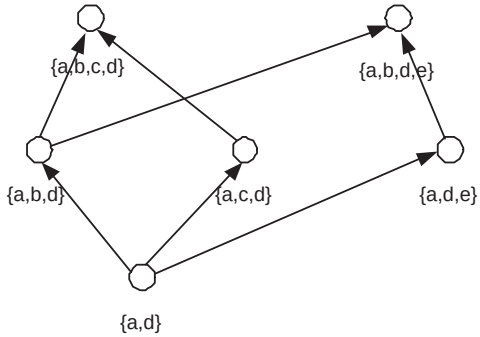
as the $100p\%$ CDsR for the reliable network $N$ and terminate.

other states in $S$.) Since $0.109 < 0.2$, the algorithm adds the tuple $(0, 0.109)$ to $distribution$, and all the states in $S$ are added to $frontier$. The minimum value of maximum $s$-$t$ flow among states in $frontier$ is 3 units. COMPUTE-RISK chooses the two states $\{a, d\}$ and $\{a, c, e\}$ in $frontier$ which allow a maximum $s$-$t$ flow of 3 units, removes them from $frontier$ and adds them to $S^\star$. In Step 4, the algorithm examines the maximum $s$-$t$ flow in the states in $S^\star$ and other sets whose arc sets are supersets of the arc sets of some state in $S^\star$, while making sure that no state is evaluated more than once. The union of the Hasse diagrams while examining all the states in $S^\star$ is shown in Figure 3(a) while the Hasse diagrams evaluated with base $(V, \{a, d\}, s, t)$ and $(V, \{a, c, e\}, s, t)$ are shown in Figures 3(b) and 3(c) respectively.

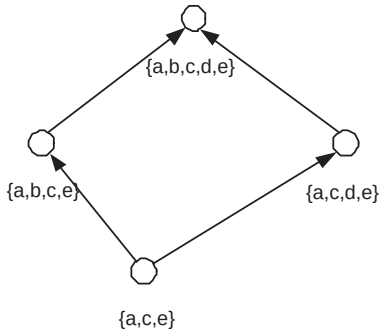
On examining these states, the COMPUTE-RISK algorithm observes that the probability of the network allowing a maximum $s$-$t$ flow of exactly 3 units is 0.225, and the new states that could be added to $frontier$ are those with arc sets $\{a, b, d, e\}$ and $\{a, b, c, d, e\}$. However, since the arc set of the state $(V, \{b, e\}, s, t)$ in $frontier$ is a subset of the arc sets of both these states, these states do not finally become members of $frontier$. So in Step 5, the algorithm adds the tuple $(3, 0.225)$ to $distribution$. At this point, $p_c = 0.334$, which is more than $p = 0.2$, and hence the algorithm moves to Step 6. In Step 6, the COMPUTE-RISK algorithm outputs 3 units as the 20% DsR. It replaces the tuple $(3, 0.225)$ in $distribution$ by $(3, 0.2 - 0.109)$, i.e., $(3, 0.091)$, and outputs the 20% CDsR as $(0 \times 0.109 + 3 \times 0.091)/0.2) = 1.365$ units. ∎




(a) Union of Hasse diagrams for states in $S^\star$




(b) Hasse diagram for $(V, \{a, d\}, s, t)$




(c) Hasse diagram for $(V, \{a, c, e\}, s, t)$

Figure 3: Hasse diagrams evaluated for states in $S^\star$

# 3 Computational experience

Papers in the literature that deal with performance measurement for reliable networks do not involve extensive computational experiments. Hence we could not find benchmark instances on which to evaluate the performance of the COMPUTE-RISK algorithm. We therefore generated three types of random networks to test the performance of this algorithm.

The first type of networks is called *completely random networks*. In these networks, a unit square is taken, and two nodes, designated as source and terminal nodes, are placed at the bottom left and top right corner of the square respectively. A pre-determined number of nodes are then randomly scattered on the square. Each node is connected to a random number of nodes among those which are relatively close to it. In order to ensure that the source and the terminal nodes are connected,

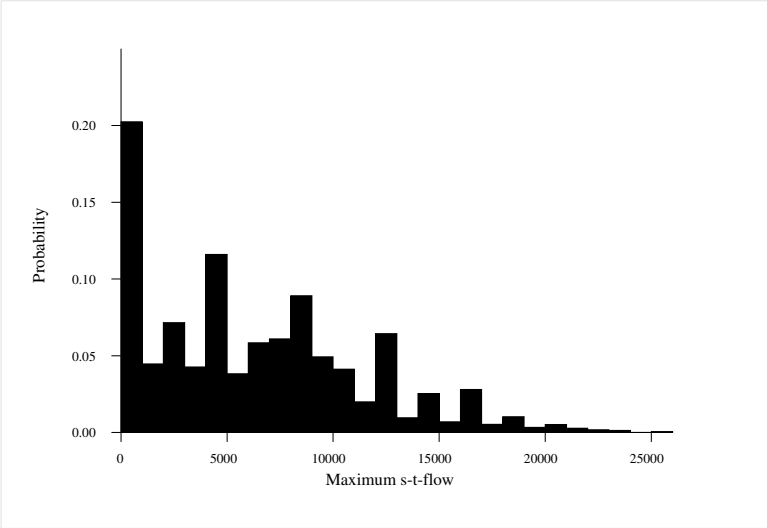



Figure 4: The probability mass function for the maximum $s$-$t$ flow in a random layered network with 24 arcs

a path is artificially provided from the source to the terminal node, passing through some of the other nodes. The second type of networks is called *random layered networks*. In these networks, the nodes apart from the source and the terminal nodes are arranged in layers. The source node is connected to all nodes in the first layer, and all nodes in the last layer are connected to the terminal node. Each node in a layer except the last layer is connected to a random number of nodes in the next layer. The third type of networks is called *random grid networks*. In these networks, the nodes apart from the source and the terminal nodes are arranged in the form of a grid. The source node is connected to all the nodes in the first column of the grid and all the nodes in the last column of the grid are connected to the terminal node. Each node in the grid is connected to its adjacent nodes in the same column and to the nodes adjacent to it in the next column. A more detailed description of the method of constructing random layered and grid networks is available in Ahuja et al. [12].

Each arc in the networks thus formed is assigned an integer capacity randomly chosen from the interval $[500, 1000]$ if it is not connected to the source or the terminal node, and from the interval $[50000, 100000]$ if it connects either the source or the terminal node. The reliability of each arc is chosen randomly from the interval $[0.5, 0.9]$. An illustrative probability mass function for the maximum $s$-$t$ flow in these types of networks is given in Figure 4.

For our experiments we coded the COMPUTE-RISK algorithm in C, compiled it using the `gcc 4.4` compiler and run it on a Intel Core 2 Quad Q8300 processor with 3GB DDR2 SDRAM running Linux. The problem sizes considered for our experiments, measured in terms of the number of arcs in the network, are given in Table 1. For each combination of network type and problem size, we created 20 network instances. The results reported in this section are the averages of the results from all 20 instances.

Table 1: Problem sizes considered in the computational experiments

| Network type | Average number of arcs |
|---|---|
| Completely random networks | 21, 27, 33, 39, 52, 66, 77, 90, 104 |
| Random layered networks | 24, 30, 36, 48, 56, 68, 80, 92, 110 |
| Random grid networks | 24, 30, 36, 43, 54, 65, 76, 94, 110 |

In our first experiment, we examined our claim that DsR and CDsR values are indeed useful to decision makers while choosing among alternate reliable network configurations. We chose 20 instances each of completely random networks with 21 arcs, random layered networks with 24 arcs, and random grid networks with 24 arcs, and evaluated 15% DsR, 15% CDsR, expected maximum *s-t* flows, and the probability that the network allows a *s-t* flow which is more than 95% of the maximum *s-t* flow it will allow when all arcs in the network are functional. We hypothesised that the correlations between the downside risk measures and the other two measures are not very high; if they are, then it implies that the existing measures measure downside risks anyway. Our results with completely random network instances is shown in Table 2. From the table we see that the correlation values are indeed not very high, suggesting that existing measures do not measure downside risks effectively.

Table 2: Correlation between four different performance measures on completely random reliable networks with 21 arcs

| | 15% DsR | 15% CDsR | Expected max-flow | Upside flow probability |
|---|---|---|---|---|
| 15% DsR | 1.000 | | | |
| 15% CDsR | 0.379 | 1.000 | | |
| Expected max-flow | 0.591 | 0.427 | 1.000 | |
| Upside flow prob. | 0.361 | 0.146 | 0.199 | 1.000 |

We next ranked the 20 instances separately on the four measures, assigning a rank of 1 to the best instance according to that measure and a rank of 20 to the worst. We present the top three ranked instances as per the different performance measures in Table 3.

Table 3: The top three ranks among the 20 completely random reliable networks with 21 arcs based on four different performance measures

| Rank | 15% DsR | 15% CDsR | Expected max-flow | Upside flow probability |
|---|---|---|---|---|
| 1 | 6 | 20 | 6 | 6 |
| 2 | 2 | 6 | 15 | 13 |
| 3 | 4 | 18 | 4 | 17 |

It is interesting to see that apart from Instance 6 which appears among the top three ranks on all four measures, and Instance 4, which appears among the top three ranks in two of the measures, no other instance occupies any of the top three ranks based on more than one parameter. This clearly justifies the need for downside measures while evaluating reliable networks.

Our results for random layered networks and random grid networks are similar to the results shown above. The only difference between those results and the ones for completely random networks is that in both those types of networks, there was a very significant positive correlation between the DsR and CDsR values. However, the correlation between the downside risk measures and the other measures remains low, and sometimes becomes negative. We do not report those results here for the sake of brevity.

In our second experiment we computed $100p\%$ DsR values for smaller instances of the three types of networks for $p$ values of 0.05, 0.10, 0.15, and 0.20. In several of the instances evaluated, the $100p\%$ DsR values turn out to be 0, implying that in the bottom $100p\%$ of the network states in these networks, the source node $s$ is not connected to the terminal node $t$. Table 4 presents the number of instances out of the 20 generated, where this situation was observed.

Table 4: Number of instances in which the source and terminal nodes are disconnected in downside states

| Network type | Number of arcs | Instances in which $100p\%$ DsR $= 0$ | | | |
|---|---|---|---|---|---|
| | | $p = 0.05$ | $p = 0.10$ | $p = 0.15$ | $p = 0.20$ |
| Completely random networks | 21 | 17 | 9 | 5 | 5 |
| | 27 | 17 | 12 | 8 | 5 |
| Random layered networks | 24 | 20 | 20 | 16 | 6 |
| | 30 | 20 | 20 | 16 | 9 |
| Random grid networks | 24 | 20 | 19 | 16 | 13 |
| | 30 | 20 | 20 | 16 | 12 |

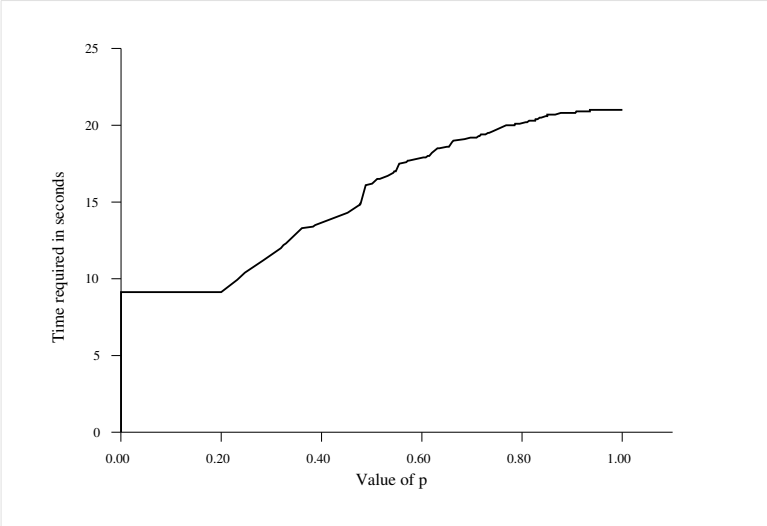



Figure 5: The variation of time required to compute $100p\%$ DsR and $100p\%$ CDsR with $p$ in a random layered network with 24 arcs

From Table 4 we see that the chances of the source and terminal nodes in random layered and random grid instances being disconnected is higher than for completely random instances. This leads us to conclude that in critical networks where downside risks are important, it is better not to choose layered or grid configurations if possible.

In our last experiment we computed the maximum value of $p$ for which the COMPUTE-RISK algorithm outputs $100p\%$ DsR and $100p\%$ CDsR values within 10 minutes of execution time. These $p$ values vary from instance to instance, so in Table 5 we present the average of the $p$ values over all 20 instances for different combination of network type and network size. The $p$ values obtained for different instances of completely random reliable networks were found to be more varied than for random layered and grid networks, since the topologies of the former type of networks were more varied than those of the latter.

From the table, we observe that within 10 minutes of execution time, the COMPUTE-RISK algorithm can find the complete probability mass function for only the smallest of the instances. Thereafter, the value of $p$ for which the COMPUTE-RISK algorithm can compute the $100p\%$ DsR and $100\%$ CDsR within 10 minutes decreases sharply with increasing values of $p$. We also observe that for a network instance of a given size, the time required to compute $100p\%$ DsR and $100p\%$ CDsR increases at an increasing rate as the value of $p$ increases (see Figure 5). From these runs we

Table 5: Percentage of the probability mass function of maximum $s$-$t$ flow computed by COMPUTE-RISK in 10 minutes

| Network type | Number of arcs | Percentage computed | Standard deviation |
|---|---|---|---|
| Completely random networks | 21 | 100.00% | 0.00% |
| | 27 | 85.97% | 34.28% |
| | 33 | 25.19% | 32.25% |
| | 39 | 6.15% | 5.62% |
| | 52 | 1.78% | 1.92% |
| | 66 | 0.94% | 0.87% |
| | 77 | 1.19% | 1.17% |
| | 90 | 1.57% | 2.00% |
| | 104 | 0.95% | 1.27% |
| Random layered networks | 24 | 100.00% | 0.00% |
| | 30 | 25.75% | 7.07% |
| | 36 | 5.99% | 2.00% |
| | 48 | 1.42% | 0.70% |
| | 56 | 1.28% | 0.45% |
| | 68 | 1.52% | 1.02% |
| | 80 | 1.67% | 0.99% |
| | 92 | 1.63% | 0.92% |
| | 110 | 0.46% | 0.41% |
| Random grid networks | 24 | 100.0% | 0.00% |
| | 30 | 25.10% | 7.32% |
| | 36 | 22.16% | 6.75% |
| | 43 | 5.87% | 2.37% |
| | 54 | 4.92% | 2.25% |
| | 65 | 4.53% | 1.82% |
| | 76 | 4.91% | 2.50% |
| | 94 | 1.74% | 0.99% |
| | 110 | 1.62% | 0.97% |

see that for most problem instances with more than 40 arcs, within 10 minutes of execution time, the COMPUTE-RISK algorithm was successful only in identifying network states which had zero $s$-$t$ flow. This leads us to conclude that meaningful downside risk information about problems of these sizes can only be obtained after more execution time. For these sizes of problems however, the COMPUTE-RISK algorithm can be used to evaluate the 2-terminal connectivity. If the COMPUTE-RISK algorithm returns the aggregate probability of zero maximum $s$-$t$ flow as $p_0$, the solution to the 2-terminal connectivity problem for that network is $(1 - p_0)$.

# 4 Summary and directions for future work

In this paper we introduced the concept of downside risk of reliable networks and proposed two measures, downside risk (DsR) and conditional downside risk (CDsR), to evaluate such risks. We argued that for reliable networks that model critical operations, it is important to factor in downside risks while evaluating alternative network configurations. We considered the problem of sending the maximum amount of flow from a pre-designated source node to a pre-designated terminal node in a reliable network, and explained why extensions of existing algorithms based on evaluating all possible

paths and all possible cuts in a network are not practical for computing downside risk measures in reasonable sized networks.

In Section 2 we proposed COMPUTE-RISK, a state-space evaluation based algorithm to compute DsR and CDsR values in reliable networks. Our algorithm is an enhancement of the naïve state-space enumeration algorithm in that it makes use of arc-disjoint cuts, and warmstarts maximum flow computation for one network state based on the residual network obtained after computing the maximum flow computation flows in another state.

In Section 3 we presented our computational experience with the COMPUTE-RISK algorithm. We first showed that the DsR and CDsR have low correlation with the existing measures of network performance. This leads us to conclude that downside risk information is not being properly captured while evaluating reliable network performance using conventional measures. We then showed that layered and grid networks in general have worse downside performance than completely random networks. We finally showed that the COMPUTE-RISK algorithm can compute DsR and CDsR values for reasonably sized reliable networks in reasonable time and the evaluate 2-terminal connectivity for larger networks.

The work in this paper can be extended in several directions. One immediate extension would be to study other network flow problems on reliable networks and develop similar algorithms which look at downside risk measures. Another research direction would be to form hybrid algorithms involving simulation which would compute reasonable accurate network performance measures for large networks. A third direction would be to develop algorithms that embed algorithms like COMPUTE-RISK to design reliable networks for critical applications.

# References

[1] R. Ahuja, T. Magnanti, J. Orlin, Network Flows: Theory, Algorithms and Applications, Prentice Hall, 1993.

[2] M. Ball, T. Magnanti, C. Monma, G. Nemhauser (Eds.), Handbooks in Operations Research and Management Science, Vol. 7, Network Models, Elsevier Science B.V., The Netherlands, 1996.

[3] S. Lee, Reliability evaluation of a flow network, IEEE Transactions on Reliability R-29 (1980) 24–26.

[4] K. Aggarwal, Y. Chopra, J. Bajwa, Capacity consideration in reliability analysis of communication systems., IEEE Transactions on Reliability 31 (1982) 177–180.

[5] K. Aggarwal, J. Gupta, K. Misra, A simple method for reliability evaluation of a communication system, IEEE Transactions on Communications 23 (5) (1975) 563–566.
URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1092838`

[6] R. Yarlagadda, J. Hershey, Fast algorithm for computing the reliability of communication network, . International Journal of Electronics 70 (1991) 549–564.

[7] P. Jorion, Value at Risk, 2nd Edition, Finance Series, McGraw-Hill International Edition, 2002.

[8] S. Patra, R. B. Misra, Evaluation of probability mass function of flow in a communication network considering a multistate model of network links, Microelectronic Reliability 36 (3) (1996) 415–421.

[9] C. Alexopoulos, A note on state space decomposition methods for analyzing stochastic flow networks, IEEE Transactions on Reliability 44 (1995) 354–357.

[10] J. Pitman, Probability, Springer-Verlag, New York, 1993.

[11] M. Sharma, D. Ghosh, Speeding up the estimation of expected maximum flows through reliable networks, Tech. Rep. IIMA Working Paper No. 2009-04-05, Indian Institute of Management Ahmedabad, India (2009).

[12] R. Ahuja, M. Kodialam, A. Mishra, J. Orlin, Computational investigation of maximum flow algorithm, European Journal of Operational Research 97 (1997) 509–542.