Speeding Up the Estimation of Expected Maximum Flows Through Reliable Networks

Megha Sharma Diptesh Ghosh

W.P. No. 2009-04-05
April 2009

INDIAN INSTITUTE OF MANAGEMENT
AHMEDABAD – 380015
INDIA

Speeding Up the Estimation of Expected Maximum Flows Through Reliable Networks

Megha Sharma Diptesh Ghosh

Abstract

In this paper we present a strategy for speeding up the estimation of expected maximum flows through reliable networks. Our strategy tries to minimize the repetition of computational effort while evaluating network states sampled using the crude Monte Carlo method. Computational experiments with this strategy on three types of randomly generated networks show that it reduces the number of flow augmentations required for evaluating the states in the sample by as much as 52% on average with a standard deviation of 7% compared to the conventional strategy. This leads to an average time saving of about 71% with a standard deviation of about 8%.
Keywords: Network Flows; Reliable Networks; Cold Start; Warm Start; Reliable Network Evaluation Strategy.

1 Introduction

A network is defined as $G = \{V, A, s, t\}$, where V is the set of nodes; A is the set of arcs where each element $a \in A$ is of the form (i, j, w_{ij}), $i, j \in V$, $w_{ij} \in Z_+$; and $s \in V$ and $t \in V$ are the source and the terminal nodes respectively. For arc $a = (i, j, w_{ij})$, i, j, and w_{ij} are called the tail, the head, and the capacity of the arc respectively. Arc $a = (i, j, w_{ij})$ is also referred to as arc (i, j). Networks can be used to model a wide variety of real-world optimization problems (see e.g., Ahuja et al., 1993). An important problem on networks is the maximum flow problem (see e.g., Ahuja et al., 1993), in which one computes the maximum amount of flow that can be routed from s to t through the network. The maximum flow is a constant for a given network and a given pair of source and terminal nodes.

A reliable network is a network in which each arc has a certain probability of being functional. Arcs in reliable networks are represented as $a = (i, j, w_{ij}, p_{ij})$ where i, j and w_{ij} have their usual meanings, and p_{ij}, called the reliability of arc (i, j), is the probability that arc $(i, j) \in A$ is functional at any point in time. It is a more realistic representation of networks with imperfect elements, and is often used to model telecommunication networks. Reliable networks are observed in different states, where each state is a conventional network defined on a subset of arcs in the reliable network, which are functional at the point of interest. Formally defined therefore, a network state $S_i = \{V, A_i, s, t\}$ of a reliable network (V, A, s, t), is a conventional network where V refers to the node set of the network, $A_i \subseteq A$ refers to the set of arcs in A which are functional in this state, and s, t are the source and the terminal nodes respectively. A reliable network assumes each state with a probability that can be computed using the reliability values of the arcs present in the network. Since the maximum flows in different states are different, the maximum flow through a reliable network is a random variable and not a constant as in the case of conventional networks.

The problem of evaluating expected maximum flows in reliable networks occurs most often as a subproblem in the reliable network design problem. In a reliable network design problem, one is given a ground set $\mathcal{A}$ of imperfect arcs joining nodes belonging to V, a source node $s \in V$ and a terminal node $t \in V$, and is asked to choose a set of arcs $A \subseteq \mathcal{A}$ such that the resultant network is "best" in terms of sending flow from s to t. In this paper, the "best" network is considered to be one for which the expected value of maximum flow from s to t is the highest possible. One way of finding such a network is to generate a subset of the $2^{|\mathcal{A}|}$ candidate reliable networks, each defined on the node set V, and an arc set which is an element of the power set $P(\mathcal{A})$; computing the expected maximum flow from s to t through each of the candidate networks; and then choosing that network in which the expected maximum flow is the largest possible. Since a network design problem with a ground set of reasonable size requires one to compute the expected maximum flows in a very large number of candidate reliable networks, any speedup in computation of expected maximum flows is useful while solving reliable network design problems. In this paper, we present a technique that has been seen to speed up computation of expected maximum flows through reliable networks by up to 71%.

In this work, we consider reliable networks that have imperfect arcs and perfect nodes. Arc failures are assumed to be independent. We do not model imperfect nodes separately, since an imperfect node can be modeled using an imperfect arc between two perfect nodes (see, for example, Ball et al., 1996).

The paper is organized as follows. Section 2 provides a review of the literature on maximum flow evaluation in reliable networks. In Section 3, we describe a strategy for estimating the expected maximum flow in a reliable network. In Section 4 we report our computational experience with the proposed strategy using the conventionally used strategy for estimating maximum flow as a base. Section 5 provides a summary of the contributions of the paper and points out directions for future research.

2 Literature Review

Expected maximum flows are often computed in reliable networks in order to compare two or more reliable networks defined on the same node set, and having the same source and terminal nodes. So in this context, it is an example of a network evaluation problem. This problem is NP-hard (see Valiant, 1979; Provan and Ball, 1983), and its solution approaches can be broadly classified into two classes; exact approaches and estimation through Monte Carlo simulations. There is also literature on generating bounds for this problem.

Exact algorithms for this problem take time exponential in the number of arcs and therefore can only be used for networks with very small arc sets. Examples of such algorithms include application of factoring theorem (see Lee, 1980), path and cut enumeration based techniques (see Evans, 1976; Mirchandani, 1976). Polynomial time exact algorithms exist for networks with special structural properties, like the series parallel networks (see Satyanarayan, 1982), networks in which all but a few arcs have deterministic capacities (see Somers, 1982), and networks that are 1-critical i.e. networks that are minimal with respect to withstanding any single component failure (see Ball et al., 1995).

The majority of the network evaluation literature uses crude Monte Carlo and its more efficient variants like homogeneous Monte Carlo, homogeneous Monte Carlo with variance reduction, etc., to estimate the desired measure of performance. For details on various Monte Carlo techniques used in reliable network evaluation see Fishman (1986); Nel and Colbourn (1990); Buchsbaum and Mihail (1995). The procedure that all of these methods use to estimate the performance measure can be summarized as follows. Several states of the network are first generated based on the probability of

each arc in the reliable network being functional. The maximum flow from the source to the terminal node is determined for each of the network states generated using one of the existing maximum flow algorithms. These maximum flow values are then aggregated to estimate the expected maximum flow for the reliable network. All Monte Carlo simulation based methods for estimating expected maximum flow through reliable networks compute the maximum flow through each of the sampled states independently. Only one paper, Kashyap et al. (2006), makes use of a strategy for evaluating the maximum flow in a network by utilizing information obtained while computing the maximum flow in a related network.

Analytical bounds have also been proposed only for networks with special structural properties. Bounds proposed for general networks are Monte Carlo based, like the bounds proposed by Carey and Hendrickson (1984), the most probable states based bounds (see Li and Silvester, 1984) and their improvements (see Colbourn and Harms, 1993). A detailed review of the literature on reliable network evaluation problem and solution approaches can be found in Colbourn and Harms (1993) and Ball et al. (1996).

The work that we describe in this paper uses the maximum flow through a network state and its residual network to construct the maximum flow in a related state. In this respect, our work is most closely related to the work presented in Kashyap et al. (2006). However, the algorithm hinted in Kashyap et al. (2006) is similar to the easier of the two algorithms developed here; we propose another completely new algorithm which is integral to our speeding up process.

3 Speeding Up the Estimation of Expected Maximum Flows

We assume that the estimation is carried out using Monte Carlo simulations, where a set of network states of the reliable network are sampled, the maximum flow in each of the sampled states is obtained through some flow augmentation algorithm (see, e.g., Ahuja et al., 1997) and the expected maximum flow is computed as the average of the maximum flows through the sampled states. We use the following notation to describe our strategy of speeding up the estimation of expected maximum flow through a reliable network.

Definition (Cold Start)**.** *A state S_1 of a reliable network is said to be cold started if the maximum flow in state S_1 is computed without using information about any other state of the network.*

Definition (Cold Start Evaluation Strategy)**.** *The expected maximum flow of a reliable network is said to be estimated using a cold start evaluation strategy (CSES) if the estimation is done using Monte Carlo simulation, where all the sampled states are cold started.*

Definition (Warm Start)**.** *A state S_1 of a reliable network is said to be warm started from another state S_2 of the same network if the maximum flow in state S_1 is computed using the maximum flow in state S_2, and the residual network obtained after evaluating the maximum flow in S_2.*

Definition (Warm Start Evaluation Strategy)**.** *The expected maximum flow of a reliable network is said to be estimated using a warm start evaluation strategy (WSES) if the estimation is done using Monte Carlo simulation, where some of the sampled states are cold started, and the remaining are warm started from other states in the sample.*

Conventionally the estimation of expected maximum flow in a reliable network is done using a cold start evaluation strategy. Each network state is cold started using one of the well-known maximum flow

algorithms (see, e.g., Ahuja et al., 1997). In the warm start evaluation strategy, some of the network states are warm started from some other network states. The basic idea behind warm start is the following. Given two network states S_1 and S_2 of the same reliable network, S_2 can be obtained from S_1 by either removing some arcs from it, or by adding some arcs to it, or by a combination of both. Hence, if we have efficient algorithms for recomputing the maximum flow in a network after one or more arcs are added to it, and for recomputing the maximum flow after one or more arcs are removed from it, then we can design an efficient algorithm for warm starting state S_2 from state S_1.

In this section we first describe two algorithms, called ADD-ARC and DELETE-ARC. ADD-ARC re-computes the maximum flow in a network after one or more arcs are added to it. DELETE-ARC re-computes the maximum flow in a network after an arc is deleted from it. Both these algorithms use the residual network obtained after maximum flow computations in a network state to achieve their output. We then suggest the use of these two algorithms in the heuristic GENERATE-SEQUENCE to develop an efficient strategy of estimating the expected maximum flow through a reliable network.

3.1 The arc addition algorithm ADD-ARC

Consider two network states $S_1 = \{V, A_1, s, t\}$ and $S_2 = \{V, A_2, s, t\}$, $A_1 \subset A_2$. Clearly, state S_2 can be obtained by adding one or more arcs to state S_1. Given the maximum flow M_1 through state S_1 and its residual network $S_1^r = \{V, A_1^r, s, t\}$, the arc addition algorithm outputs the maximum flow M_2 through state S_2 and the corresponding residual network $S_2^r = \{V, A_2^r, s, t\}$. Algorithm ADD-ARC describes this algorithm.

Algorithm 1 ADD-ARC: The Arc Addition Algorithm

Input: Residual network S_1^r for $S_1 = \{V, A_1, s, t\}$, maximum s–t flow M_1 through S_1, state $S_2 = \{V, A_2, s, t\}$, $A_1 \subset A_2$.
Output: Maximum s–t flow M_2 through S_2, residual network S_2^r for S_2.
Steps:

Step 1: (Initialization) Set $S_2^r \leftarrow \{V, A_1^r \cup (A_2 \setminus A_1), s, t\}$, $M_2 \leftarrow M_1$. Go to Step 2.
Step 2: (Termination) If there is no augmenting path from node s to node t in S_2^r, then output S_2^r and M_2, and stop. Else go to Step 3.
Step 3: (Iteration) Augment the s–t path found in Step 2 and update S_2^r and M_2. Go to Step 2.

Theorem 1. *Given the residual network S_1^r of state S_1, the set $A_2 \setminus A_1$, and the maximum s–t flow M_1 through state S_1, ADD-ARC correctly determines the maximum s–t flow M_2 through state S_2.*

Proof. By contradiction, assume that the flow M_2 output by ADD-ARC is suboptimal. Then there must exist a flow augmenting path in the residual network S_2^r. If such a path exists, then the termination condition in Step 2 is not satisfied, and ADD-ARC will not terminate but send flow along the augmenting path. ■

3.2 The arc deletion algorithm DELETE-ARC

Consider two states $S_1 = \{V, A_1, s, t\}$ and $S_2 = \{V, A_2, s, t\}$ of a reliable network, where $A_2 = A_1 \setminus \{a\}$. This means that state S_2 can be obtained by deleting arc a from S_1. Given the maximum s–t

flow M_1 through state S_1 and the corresponding residual network $S_1^r = \{V, A_1^r, s, t\}$, the arc deletion algorithm determines the maximum s–t flow M_2 through state S_2. Algorithm DELETE-ARC describes this algorithm.

Algorithm 2 DELETE-ARC: The Arc Deletion Algorithm

Input: Residual network S_1^r for $S_1 = \{V, A_1, s, t\}$, maximum s–t flow M_1 through S_1, state $S_2 = \{V, A_s, s, t\}$, $A_1 \setminus A_2 = \{(i, j)\}$.
Output: Maximum s–t flow M_2 through S_2, residual network S_2^r for S_2.
Steps:

Step 1: (Initialization) Set $S_2^r \leftarrow \{V, A_1^r \setminus \{(i, j)\}, s, t\}$, $M_2 \leftarrow M_1 - f_{ij}$, where f_{ij} is the flow along arc (i, j) in S_1^r, excess$(i) \leftarrow f_{ij}$ and excess$(j) \leftarrow -f_{ij}$. Go to Step 2.
Step 2: (Termination) If excess(i) = excess$(j) = 0$, then output S_2^r, M_2 and stop. Else go to Step 3.
Step 3: (Iteration)

Step 3a: If there exists a path P_1 from node i to node j in the network S_2^r then, augment a flow of $f_1 = \min(\text{excess}(i), \min_{(l,m)\in P_1} w_{lm})$ on path P_1. Update S_2^r, $M_2 \leftarrow M_2 + f_1$, excess$(i) \leftarrow$ excess$(i) - f_1$, excess$(j) \leftarrow$ excess$(j) + f_1$; go to Step 2.

Step 3b: Else if excess$(i) > 0$ then, search for a path P_2 from node i to node s in network S_2^r and augment a flow of $f_2 = \min(\text{excess}(i), \min_{(l,m)\in P_2} w_{lm})$ on this path. Update S_2^r, excess$(i) \leftarrow$ excess$(i) - f_2$. If excess$(j) < 0$ then, search for a path P_3 from node t to node j in network S_2^r and augment a flow of $f_3 = \min(-\text{excess}(j), \min_{(l,m)\in P_3} w_{lm})$ on this path. Update S_2^r, excess$(j) \leftarrow$ excess$(j) + f_3$. Go to Step 2.

Theorem 2. *Given the residual network S_1^r of state S_1, the arc a, and the maximum s–t flow M_1 through state S_1, DELETE-ARC correctly determines the maximum s–t flow M_2 through state S_2.*

Proof. Note that $M_2 \leq M_1$.

If the termination condition in Step 2 is satisfied immediately after Step 1, then $f_{ij} = 0$, $M_1 = M_2$ and the output from DELETE-ARC is trivially correct.

Let us now consider the situations where termination occurs in Step 2, reached after executing Step 3. If the algorithm terminates when Step 2 is reached immediately after executing Step 3a, then the whole of the flow along arc (i, j) could be re-routed through the network and $M_1 = M_2$. Obviously, in this case DELETE-ARC outputs a correct solution. If however, the algorithm terminates when Step 2 is reached immediately after executing Step 3b, then $M_2 < M_1$. Assume, by contradiction, that the flow M_2 output by DELETE-ARC in such a situation is not the maximum flow through state S_2. Then there must be at least one augmenting path from s to t in the residual network S_2^r output by the algorithm. Such an augmenting path can exist in one of the following four ways.

1. The augmenting path does not include any (reverse) arc generated during the re-routing process in Step 3 of DELETE-ARC. But this implies that all the arcs in the augmenting path existed in S_1^r, which in turn implies that M_1 was not the maximum flow through S_1. Therefore, such an augmenting path can not exist.

2. The augmenting path, say AP, includes (reverse) arcs generated during the re-routing of flow from i to s in Step 3b. Let $AP = s, \ldots, m, n, \ldots, t$ and let (m, n) be one of the arcs generated during

the re-routing of flow along $P = i, \ldots, n, m, \ldots, s$. At the stage when such a path is generated in the algorithm, excess$(j) \leq 0$. If excess$(j) < 0$ there exists at least one path from t to j say, $A = t, \ldots, j$, which implies that a path $Q = i, \ldots, n, \ldots, t, \ldots, j$ existed in the network before re-routing the flow along path P. Such a path would have been identified during Step 3a of the algorithm and Step 3b would not be reached. If excess$(j) = 0$ then Q would have existed before the augmentation of the last t to j path and Step 3b would not be reached. Therefore such an augmenting path can not exist.

3. The augmenting path, say AP, includes (reverse) arcs generated during the re-routing of flow from t to j in Step 3b. Let $AP = s, \ldots, m, n, \ldots, t$ and let (m, n) be one of the arcs generated during the re-routing of flow along $P = t, \ldots, n, m, \ldots, j$. At the stage when such a path is generated in the algorithm, excess$(i) \geq 0$. If excess$(i) > 0$, then there exists at least one i to s path, $i, \ldots, s$, which implies that a path $Q = i, \ldots, s, \ldots, m, \ldots, j$ existed in the network before re-routing the flow along path P. Such a path would have been identified in Step 3a of the algorithm and Step 3b would not be reached. If excess$(i) = 0$, then Q would have existed before the augmentation of the last i to s path and Step 3b would not be reached. Therefore such an augmenting path can not exist.

4. The augmenting path includes (reverse) arcs, but only those which are not a part of any re-routing of i to s flows or t to j flows in Step 3b. This would imply that a path of the form $i \ldots s \ldots t \ldots j$ exists in the residual network S_2^T. If such a path exists and excess$(i) > 0$, then DELETE-ARC would have utilized this in Step 3a. If such a path exists and excess$(i) = 0$, such an augmenting path would imply that $M_2 > M_1$. Hence such an augmenting path can not exist.

Since none of the four alternatives is possible, we conclude that DELETE-ARC outputs the correct maximum flow through state S_2. ■

Remark 3. *A single execution of ADD-ARC can add multiple arcs to a network state, but a single execution of DELETE-ARC can remove exactly one arc from a network.*

3.3 Generating the sequence of evaluation of states through GENERATE-SEQUENCE

To estimate the expected maximum flow using Monte Carlo simulation, a subset of all the states of the network is generated randomly, and the maximum flow in each of the states in the subset is computed. The expected maximum flow in the reliable network is estimated by computing the average of the maximum flow values thus obtained. The primary idea behind warm start is to use the residual network obtained from the computation of maximum flow in one state to compute the maximum flow in another state.

In order to use the warm start evaluation strategy, we have to generate the sequence in which we consider network states for maximum flow computation, and for each state being considered, we have to determine whether to cold start that state, or to warm start it from some other state for which the maximum flow has already been computed. Later in this section, we will model this decision problem as a minimum directed spanning forest problem.

Note that as a consequence of Remark 3, if we have to warm start state $S_2 = \{V, A_2, s, t\}$ from a state $S_1 = \{V, A_1, s, t\}$ of a reliable network $\{V, A, s, t\}$, we have to use DELETE-ARC $|A_1 \setminus A_2|$ times,

and ADD-ARC at most once (only if $|A_2 \setminus A_1| > 0$). Given a value of $|A_2 \setminus A_1|$, a smaller value of $|A_1 \setminus A_2|$ is likely to reduce the time taken to warm start S_2 from S_1. Similarly, given a value of $|A_1 \setminus A_2|$, a smaller value of $|A_2 \setminus A_1|$ is likely to result in shorter warm start time. As a result we see that a larger number of common arcs between two states is likely to increase the time saving when we warm start one from the other. If the values of $|A_1 \setminus A_2|$ and $|A_2 \setminus A_1|$ are large however, it is more efficient to compute the maximum flows in the two states independently.

We define the following measure of distance between two states of a reliable network for the purpose of deciding whether to warm start one state from the other.

Definition (distance)**.** *The distance $d(S_i, S_j)$ from state $S_i = \{V, A_i, s, t\}$ to state $S_j = \{V, A_j, s, t\}$ of a reliable network is given by*

$$d(S_i, S_j) = |A_i \setminus A_j| + |A_j \setminus A_i|.$$

We can now describe how to convert the problem of generating the sequence and manner of evaluating network states into a minimum spanning forest problem. Given a set $\mathcal{S} = \{S_i\}$ of states of a reliable network and a threshold τ, we first create a digraph in the following manner. We define a node in the digraph for every state in $\mathcal{S}$. For each pair (S_i, S_j), $S_i, S_j \in \mathcal{S}$, we define an arc from the node corresponding to S_i to the node corresponding to S_j iff $d(S_i, S_j) \leq \tau$. The weight associated with the arc if it is present is $d(S_i, S_j)$. Notice that the underlying graph of this digraph may not be connected. We next find a minimum directed spanning forest in the digraph thus generated. To generate the evaluation strategy, we consider each of the trees in the forest in arbitrary sequence. For each tree in the minimum spanning forest, we cold start the root at first. For each arc in the tree for which the maximum flow in the state corresponding to the tail has already been computed, we warm start the state corresponding to the head of the arc from the state corresponding to its tail.

While this warm start evaluation strategy reduces the total number of flow augmentations required by the cold start evaluation strategy, preliminary computational experiments reveal that the time required by any algorithm to compute the minimum spanning forest is significant and negates any time advantage gained by using the warm start evaluation strategy. Hence we propose the simplistic heuristic algorithm GENERATE-SEQUENCE to obtain the sequence in which each state would be considered in the warm start evaluation strategy and the manner in which the maximum flow in each state would be obtained. This heuristic first generates a pre-specified number k of states, labeled N_1 through N_k, of the reliable network which may not be in $\mathcal{S}$, but which have a large number of arcs in common with some of the states in $\mathcal{S}$ (see Steps 1 through 4 in Algorithm GENERATE-SEQUENCE). Each of these states have up to a pre-specified number L_u of functional arcs. It then connects each state $S_i \in \mathcal{S}$ to that state N_j, $1 \leq j \leq k$, which has the maximum number of arcs in common with it and satisfies $d(N_j, S_i) \leq \tau$ (see Steps 5 through 7). It then prescribes that all states in $\mathcal{S}$ that are not connected to any of the states N_1 through N_k and the states N_1 through N_k themselves be cold started, and each of the remaining states in $\mathcal{S}$ be warm started from the residual network of the state N_j that it is connected to (see Steps 8 through 10).

4 Computational Experience

We compared the performance of the warm start evaluation strategy (WSES) to that of the cold start evaluation strategy (CSES) conventionally used in the literature. We used the Generic Augmenting Path algorithm (see Ahuja et al., 1993) for determining maximum flows. We compared the performance

Algorithm 3 GENERATE-SEQUENCE: Heuristic to Generate the Sequence in which States are Evaluated

Input: A set of states $\mathcal{S}$ of a reliable network, parameters k, L_u, and τ.
Output: Sequence and manner of computing maximum flows in the states in $\mathcal{S}$.
Steps:

Step 1: (N_i initialization) For each arc a generate $p_a = \frac{|\mathcal{S}_a|}{|\mathcal{S}|}$ where $\mathcal{S}_a = \{S \in \mathcal{S} : a \in A_S\}$. Set $i \leftarrow 1$. Go to Step 2.
Step 2: (N_i termination) If $i > k$ go to Step 4, else go to Step 3.
Step 3: (N_i generation) Set $A_i \leftarrow \emptyset$. Consider an arc $a \in A$ and generate a random number r. If $r \leq p_a$, set $A_i \leftarrow A_i \cup \{a\}$. Repeat for different arcs until $|A_i| = L_u$ or until all arcs have been considered. Go to Step 4.
Step 4: (Update) For each arc $a \in A_i$, set $p_a \leftarrow \min\{p_a, 1 - p_a\}$. Set $N_i \leftarrow (V, A_i, s, t)$, $i \leftarrow i + 1$, and go to Step 2.

Step 5: (Forest initialization) Set $F \leftarrow \emptyset$, and $f \leftarrow 1$. Go to Step 6.
Step 6: (Forest termination) If $f > |\mathcal{S}|$ go to Step 8, else go to Step 7.
Step 7: (Forest generation) For $S_f \in \mathcal{S}$, find j such that $d(N_j, S_f) = \min_{1 \leq i \leq k}\{d(N_i, S_f)\}$. If $d(N_j, S_f) \leq \tau$ then set $F \leftarrow F \cup (N_j, S_f)$. Set $f \leftarrow f + 1$, and go to Step 5.

Step 8: (Strategy initialization) Evaluate the maximum flows in all states of $\mathcal{S}$ that are not connected from any N_i through arcs in F in random order, and evaluate them using cold start. Set $j \leftarrow 1$. Go to Step 9.
Step 9: (Strategy termination) If $j > k$, stop. Else go to Step 10.
Step 10: (Strategy generation) Evaluate the maximum flow through N_j using cold start. For each arc of the form $(N_j, S_f) \in F$, evaluate the maximum flow in S_f through warm start, using the residual network from state N_j. Set $j \leftarrow j + 1$, and go to Step 9.

of the two strategies using two ratios:

$$\text{time ratio } R_T = \frac{\text{computational time using the WSES}}{\text{computational time using the CSES}},$$

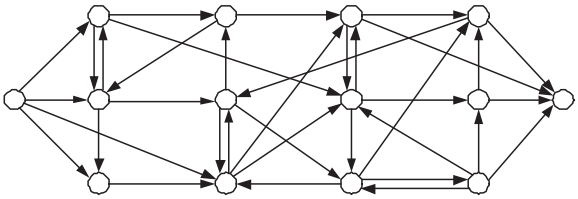
$$\text{and augmentation ratio } R_A = \frac{\text{number of flow augmentations using the WSES}}{\text{number of flow augmentations using the CSES}}.$$

In this section we report the results of our comparison. In Section 4.1 we describe how we generated the test problems on which we compared the two strategies. In Section 4.2 we present the results of our experiments and follow it up in Section 4.3 with a discussion on some interesting observations from our computational experience.

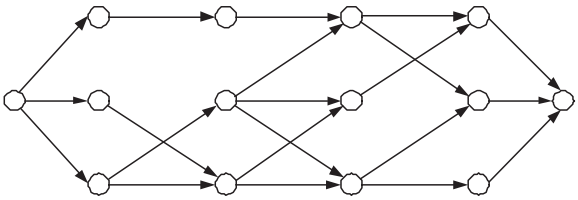
4.1 Test Bed Generation

Since no standard data set is available for reliable network flow problems, we generated our own data sets to test the two strategies. We generated three types of problem instances, completely random network instances, layered network instances, and grid network instances; see Figure 1 for examples

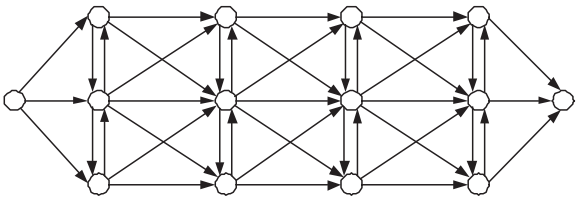
of these three types of networks. The reason to consider three types of instances was an observation made in Ahuja et al. (1997) that maximum flow problems on completely random networks are generally easy to solve and randomly generated layered and grid networks are generally harder to solve. We next describe the manner in which we generated these three types of networks.



A random network



A layered network with $L = 4$ and $W = 3$



A grid network with $L = 4$ and $W = 3$

Figure 1: Types of networks in the test bed

Completely random networks: To generate a completely random network on n nodes with around m arcs, we scatter n points randomly on a square of 100×100 units and label them 1 through n. Each node i is connected to a random number d_i of its nearest neighbors, where d_i ranges between $[1, \lceil 2m/n \rceil]$. To ensure that the network contains at least one s–t path, we ensure that arcs $(i,\ i+1)$ exist in the network, $1 \leq i < n$. For arcs $(s, j),\ j \in V \setminus s$ and $(i, t),\ i \in V \setminus t$, we assign capacities randomly in the range $[50000, 100000]$. For the remaining arcs we assign capacities randomly in the range $[500, 10000]$. All arcs are assigned reliability randomly in the range [0.8, 1.0]. For our experiments, we generated 20 completely random network instances with $n = 50$, and $m = 500$, 20 more with $n = 75$, and $m = 1350$, and another 20 with $n = 100$ and $m = 2400$.

Layered networks: Layered networks are characterized by three parameters, the width (W) of the network, the length (L) of the network, and the average outdegree (d) of all nodes in the network except the source and terminal nodes. The number of nodes in such a network is $n = LW + 2$. The nodes are arranged in $L + 2$ layers, where the first layer consists of only the source node and the last layer consists of only the terminal node. Each of the remaining layers contains W nodes. The source node is

connected to all the nodes in the second layer, and all the nodes in the last but one layer are connected to the terminal node. Each node in layer i, $2 \leq i \leq L$ is connected to a random set of r nodes in layer $i+1$, where r is a random number drawn from the interval $[1, 2d-1]$. For arcs $(s, j),\ j \in V \setminus s$ and $(i, t),\ i \in V \setminus t$, we assign capacities randomly in the range $[50000, 100000]$. For the remaining arcs we assign capacities randomly in the range $[500, 10000]$. All arcs are assigned reliability randomly in the range $[0.8, 1.0]$. According to Ahuja et al. (1997) results for layered networks with $L = 2W$ can be generalized to other layered networks with different L/W ratios, so we chose $L = 2W$ for the networks we generated. For our experiments we generated 20 network instances with $W = 8$, $L = 16$, and $d = 4$, 20 more with $W = 12$, $L = 24$, and $d = 6$, and another 20 with $W = 14$, $L = 28$ and $d = 7$.

Grid networks: Grid networks are characterized by two parameters, the width (W) of the network and the length (L) of the network. The number of nodes in such a network is $n = LW + 2$. The nodes are arranged in $L + 2$ layers, where the first layer consists of only the source node and the last layer consists of only the terminal node. Each of the remaining layers contains W nodes. The source node is connected to all the nodes in the second layer, and all the nodes in the last but one layer are connected to the destination node. The nodes in each of the layers from layer 2 to layer $L + 1$ are ordered using numbers from 1 through W. Node i in layer j is connected to nodes $i - 1$ and $i + 1$ in layer j, and to nodes $i - 1$, i, and $i + 1$ in layer $j + 1$ if such nodes exist. For arcs $(s, j),\ j \in V \setminus s$ and $(i, t),\ i \in V \setminus t$, we assign capacities randomly in the range $[50000, 100000]$. For the remaining arcs we assign capacities randomly in the range $[500, 10000]$. All arcs are assigned reliability randomly in the range $[0.8, 1.0]$. Again according to Ahuja et al. (1997) results for grid networks with $L = 2W$ can be generalized to other grid networks with different L/W ratios, so we chose $L = 2W$ for the networks we generated. For our experiments we generated 20 network instances with $W = 8$ and $L = 16$, 20 more with $W = 12$ and $L = 24$, and another 20 with $W = 16$ and $L = 32$.

4.2 Computational Results

We compared the performance of the warm start evaluation strategy and the cold start evaluation strategy on the three types of networks described in Section 4.1. For this comparison we programmed both the strategies in C using similar data structures, compiled them with the `gcc` compiler, and executed them on a computer with a 1.73 GHz Pentium M processor with 768MB RAM running Linux. The problem sizes in each of the network types were chosen such that the numbers of arcs in the three types of networks were similar. Thus, for each type of network considered, problem instances had either around 500 arcs, or around 1350 arcs, or around 2400 arcs. As mentioned earlier, for each problem size we generated 20 problem instances. For each problem instance, we estimated the expected maximum flow five times. For each estimation, we chose a new random sample of 10000 states. Therefore the results that we state in this section are averages of 100 estimations. Through preliminary experiments we found that we obtain high quality results when parameters k, τ, and L_u required for Algorithm GENERATE-SEQUENCE are set to 5, $|A|$, and $|A|$ respectively. Therefore, we carried out our experiments using these parameter values.

Table 1 presents a summary of the results of our experiments on completely random networks. The first column of the table shows the number of nodes in the network, and the second column shows the number of arcs in the network averaged over 20 randomly generated instances for the same number of nodes. Columns 3 and 5 report the average values of R_T and R_A over the 20 instances, and columns 4 and 6 report the standard deviations of the R_T and R_A values. So, for example, in estimating the expected maximum flow through 20 randomly generated reliable networks with 75 nodes and with on

average 1343.25 arcs, our warm start evaluation strategy required 31.13% of the time taken by the cold start evaluation strategy on average, with a standard deviation of 7.04%. In this process, our warm start evaluation strategy executed, on average, 47.36% of the number of flow augmentations executed by the conventional cold start evaluation strategy with a standard deviation of 4.09%. Figure 2 presents the frequency distributions of R_T and R_A values for completely random networks of different sizes.

Table 1: Performance of warm start strategy for completely random networks

no. of nodes	mean no. of arcs	mean of R_T values	s.d. of R_T values	mean of R_A values	s.d. of R_A values
50	537.40	0.3573	0.0875	0.4725	0.0676
75	1343.25	0.3113	0.0704	0.4736	0.0409
100	2408.20	0.2941	0.0772	0.4736	0.0671

Tables 2 and 3 present the results of our experiments on randomly generated layered networks and grid networks respectively while Figures 3 and 4 show the corresponding frequency distribution of R_T and R_A values.

Table 2: Performance of warm start strategy for random layered networks

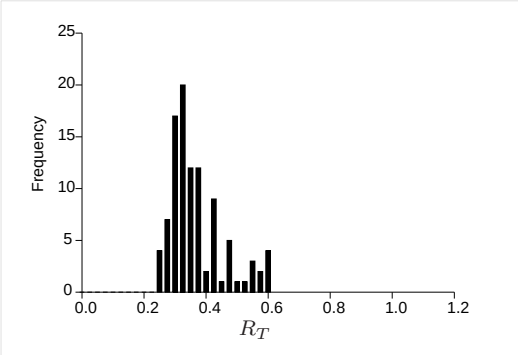
W	L	d	no. of nodes	mean no. of arcs	mean of R_T values	s.d. of R_T values	mean of R_A values	s.d. of R_A values
8	16	4	130	498.35	0.5511	0.1112	0.9596	0.1118
12	24	6	290	1409.85	0.3818	0.0577	0.9008	0.0637
14	28	7	394	2567.10	0.2793	0.0459	0.7698	0.0578

Table 3: Performance of warm start strategy for random grid networks

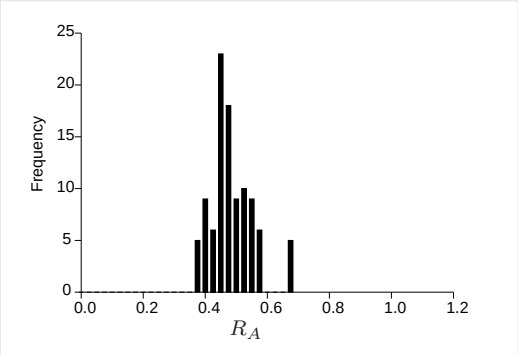
W	L	no. of nodes	no. of arcs	mean of R_T values	s.d. of R_T values	mean of R_A values	s.d. of R_A values
8	16	130	570	0.5083	0.0877	0.9487	0.0914
12	24	290	1334	0.4924	0.0912	1.0146	0.0823
16	32	514	2418	0.4781	0.0888	1.0562	0.1071

4.3 Discussion

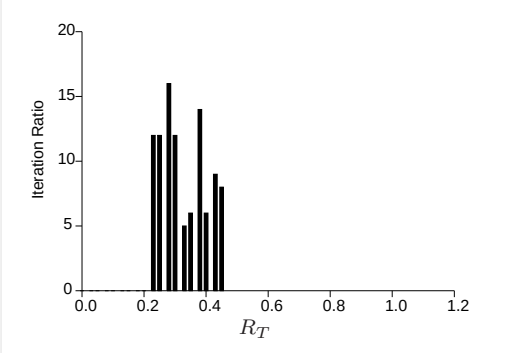
From Tables 1 through 3, we see that the warm start evaluation strategy outperforms the conventional cold start evaluation strategy for all the network types considered in our experiments. Given the standard deviation values of the ratios, it is clear that this improvement is statistically significant. We also see that the domination of the warm start evaluation strategy over the cold start evaluation



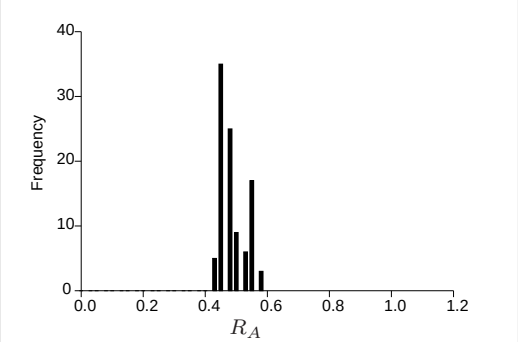
distribution of R_T for 50 node networks



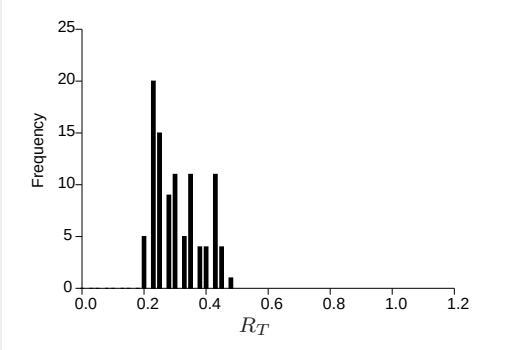
distribution of R_A for 50 node networks



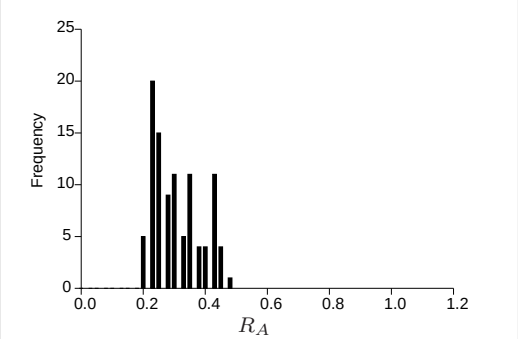
distribution of R_T for 75 node networks



distribution of R_A for 75 node networks



distribution of R_T for 100 node networks

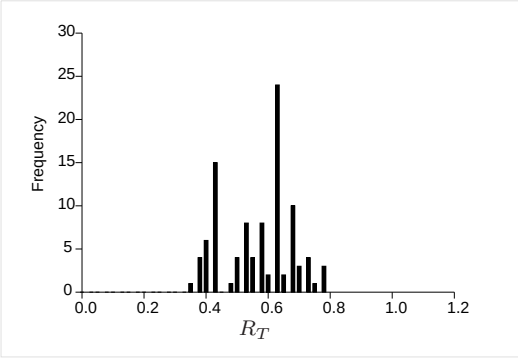


distribution of R_A for 100 node networks

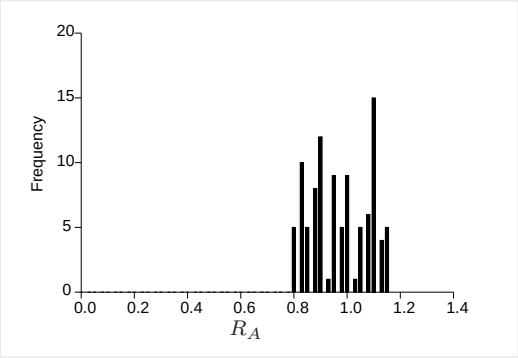
Figure 2: Frequency distributions of Time Ratio (R_T) and Augmentation Ratio (R_A) values for completely random networks

strategy gets even more pronounced as problem sizes increase, thus making the warm start evaluation strategy a more attractive choice for larger problems.

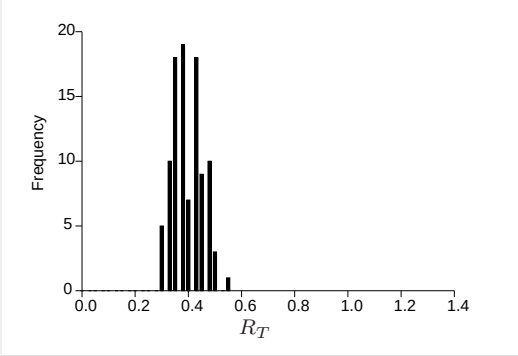
From these tables, we also see that the averages of R_A values are consistently higher than the averages of R_T values for all the problem sizes and types considered in our experiments. From our



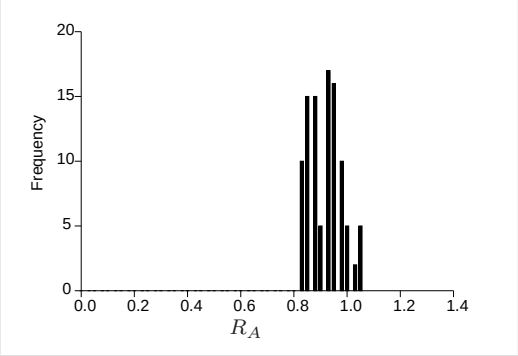
distribution of R_T for 130 node networks



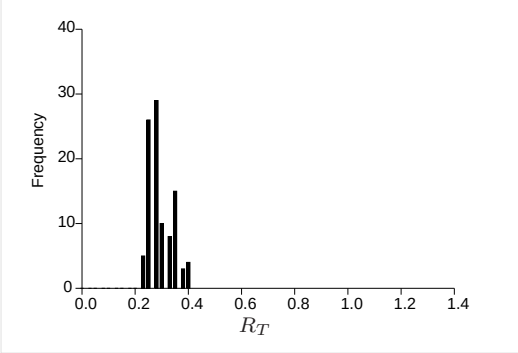
distribution of R_A for 130 node networks



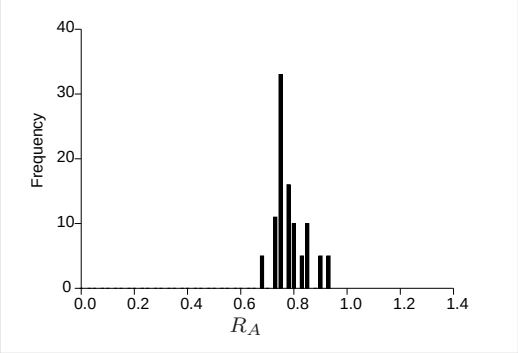
distribution of R_T for 290 node networks



distribution of R_A for 290 node networks



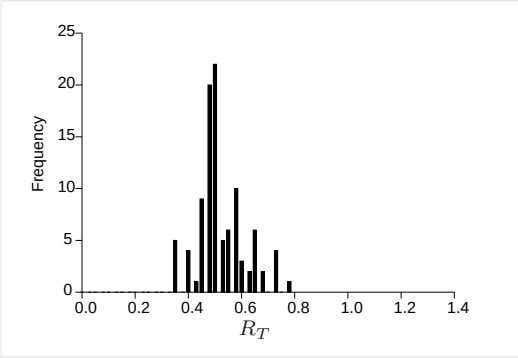
distribution of R_T for 394 node networks



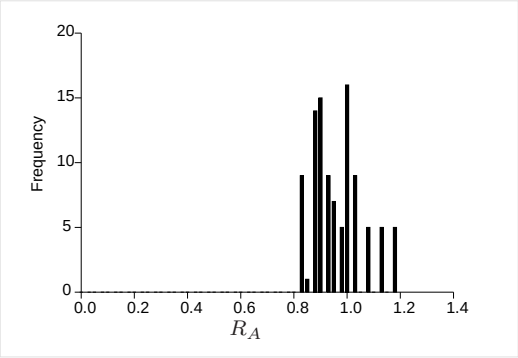
distribution of R_A for 394 node networks

Figure 3: Frequency distributions of Time Ratio (R_T) and Augmentation Ratio (R_A) values for random layered networks

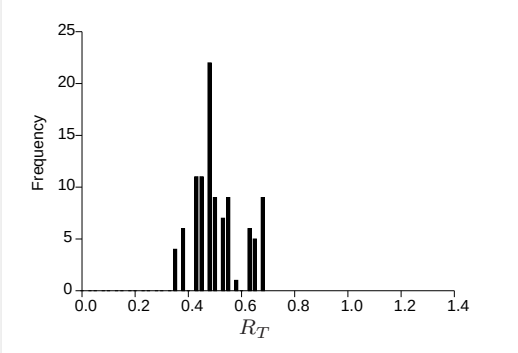
experiments, we found that the augmenting paths found during the execution of the DELETE-ARC algorithm are, on average, significantly shorter than the augmenting paths found during cold start. Therefore, the time required for searching for a path and augmenting flow along it is considerably lower for the DELETE-ARC algorithm than while using cold start. This observation does not hold for the



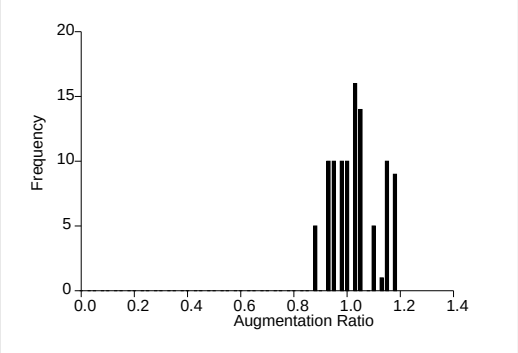
distribution of R_T for 130 node networks



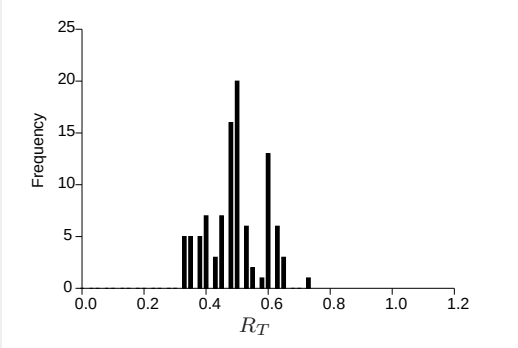
distribution of R_A for 130 node networks



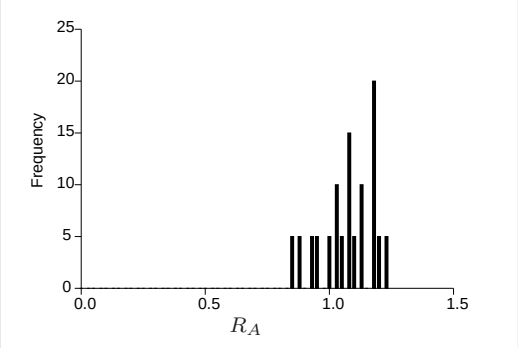
distribution of R_T for 290 node networks



distribution of R_A for 290 node networks



distribution of R_T for 394 node networks



distribution of R_A for 394 node networks

Figure 4: Frequency distributions of Time Ratio (R_T) and Augmentation Ratio (R_A) values for random grid networks

ADD-ARC algorithm of the warm start evaluation strategy. Therefore, in the warm start evaluation strategy, if the number of arcs that are being deleted is significantly higher than the number of arcs being added, the execution time per flow augmentation is less than that for the cold start evaluation strategy. The savings in the computational time due to warm start evaluation strategy consist of two

parts, savings due to reduction in the number of flow augmentations, and savings due to reduction in the computational time per flow augmentation. Only the first part contributes to the savings in the number of flow augmentations. Hence the R_T values are seen to be smaller than the R_A values.

Note that the R_T/R_A ratios are smaller for layered and grid networks than those for completely random networks. This implies that the time saving per augmentation is more for layered and grid networks. This is because the length of an augmenting path for layered and grid networks under the conventional strategy is bounded below by the $L+1$, where L is the length of the network as described in Section 4.1. Therefore every augmenting path in the conventional strategy involves at least $L+1$ arcs. For the warm start evaluation strategy this lower bound is essentially 2 when L_u is equal to the number of arcs in the network. Therefore the savings in computational time per augmentation are larger for such networks compared to the completely random network.

From other experiments which we do not report here, we see that both R_T and R_A values increase when the arc reliabilities are lowered. This is explained when we consider the nature of randomly generated network states of reliable networks. When the arc reliabilities are high, randomly generated network states have a large number of functional arcs. So, given two randomly generated network states the number of arcs that are functional in one of the states and not functional in the other is relatively small. When the arc reliabilities are closer to 0.5, then each state has fewer arcs functional, and hence the number of arcs that are functional in one of the states and not functional in the other is larger. Empirically, we have tested this on a network with 75 nodes and 1300 arcs. When the arc reliabilities are drawn from the interval $[0.8, 1.0]$, the average number of arcs functional in one of a pair of network states and not in the other is 463.57, while for the same network, if the arc reliabilities are drawn from the interval $[0.6, 0.8]$ the average number increases to 1040.97. Therefore, if the arc reliabilities in a reliable network are lowered, then to use the residual network of one state to compute the maximum flow in another, one must make a larger number of calls to DELETE-ARC and the number of arcs added in ADD-ARC also increases. As a result the time required for the warm start evaluation strategy increases and the savings in computational times and flow augmentations both reduce. When the reliabilities of arcs in the reliable network reduce further to values significantly lower than 0.5, the number of arcs that are functional in one of the states and not functional in the other again become relatively small, but under these conditions, cold start is quite efficient compared to warm start, since the number of s–t paths in a network state of such networks is small.

5 Summary

In this paper we presented a new evaluation strategy for estimating the expected maximum flow through a reliable network. Our strategy, called the warm start evaluation strategy, tries to minimize the repetition of computational effort while estimating the expected maximum flow through a reliable network by using the information obtained during the evaluation of one network state to evaluate other state(s). We compared the performance of our strategy with that of the conventionally used strategy and found that our strategy reduces the time required by as much as 70.59% on average, compared to the time required by the conventional strategy. Our strategy reduces the number of flow augmentations by as much as 52.64% on average. This reduction in evaluation time is likely to be beneficial while designing reliable networks in which one needs to carry out a large number of such expected maximum flow estimations. This implies that using our strategy, a network designer can estimate the expected maximum flow through the network using a larger sample in a given amount of time, thus getting estimates with narrower confidence interval. We have also reported that the dominance of the warm start

evaluation strategy over the cold start evaluation strategy reduces when the reliabilities of individual arcs reduce. However in actual applications on telecommunication networks, arc reliabilities are generally high; for example, fiber optic connections have reliabilities greater than 0.95. Therefore, in designing such networks, the warm start evaluation strategy outlined here can still result in considerable amount of time savings.

To the best of our knowledge such a strategy has not been reported in the literature. The only work that comes close to our work is a recent paper by Kashyap et al. (2006). It appears that the paper makes use of an algorithm similar to ADD-ARC, although it does not mention it explicitly. It only deals with a special case of a layered network with $L = 2$.

The work reported in this paper can be extended in a few ways. First, our Algorithm GENERATE-SEQUENCE does not generate an optimal evaluation strategy, simply because such a strategy is too expensive to generate. An interesting extension would be to evaluate alternate evaluation strategy generation mechanisms which have a better tradeoff between execution times and evaluation strategy quality. Secondly, one can examine whether strategies similar to ours can be used when the reliabilities of arcs in a network have lower values. Finally, one can look at other network flow estimation problems on reliable networks and develop similar strategies.

References

R.K. Ahuja, T.L. Magnanti, and J.B. Orlin. *Network Flows: Theory, Algorithms and Applications*. Prentice Hall, 1993.

R.K. Ahuja, M. Kodialam, A.K. Mishra, and J.B. Orlin. Computational investigation of maximum flow algorithm. *European Journal of Operational Research*, 97:509–542, 1997.

M.O. Ball, J.N. Hagstrom, and J.S. Provan. Threshold reliability of networks with small failure sets. *Networks*, 25:101–115, 1995.

M.O. Ball, T.L. Magnanti, C.L. Monma, and G.L. Nemhauser, editors. *Handbooks in Operations Research and Management Science*, volume 7, Network Models. Elsevier Science B.V., The Netherlands, 1996.

A.L. Buchsbaum and M. Mihail. Monte Carlo and markov chain techniques for network reliability and sampling. *Networks*, 25:117–130, 1995.

M. Carey and C. Hendrickson. Bounds on expected performance of networks with links subject to failure. *Networks*, 14(3):439–456, 1984.

C.J. Colbourn and D.D. Harms. Evaluating performability: most probable states and bounds. *Telecommunication Systems*, 2(1):275–300, 1993.

J.R. Evans. Maximum flow in probabilistic graphs - the discrete case. *Networks*, 6:161–183, 1976.

G.S. Fishman. A comparison of four Monte Carlo methods for estimating the probability of s-t connectedness. *IEEE Transactions on Reliability*, R-35(2):145–155, 1986.

S. Kashyap, S. Khuller, Y.-C. Wan, and L. Golubchik. Fast reconfiguration of data placement in parallel disks. In *Workshop on Algorithm Engineering and Experiments*, 2006.

S.H. Lee. Reliability evaluation of a flow network. *IEEE Transactions on Reliability*, R-29:24–26, 1980.

V.O.K. Li and J.A. Silvester. Performance analysis of networks with unreliable components. *IEEE Transactions on Communications*, COM-32(10):1105–1110, 1984.

P.B. Mirchandani. Shortest distance and reliability of probabilistic networks. *Computers and operations research*, 3:347–355, 1976.

L.D. Nel and C.J. Colbourn. Combining Monte Carlo estimates and bounds for network reliability. *Networks*, 20(3):277–298, 1990.

J.S. Provan and M.O. Ball. The complexity of counting cuts and of computing the probability that a graph is connected. *SIAM Journal on Computing*, 12(4):777–788, 1983.

A. Satyanarayan. A unified formula for the analysis of some network reliability problems. *IEEE Transactions on Reliability*, R-31:23–32, 1982.

J.E. Somers. Maximum flow in a network with a small number of random arc capacities. *Networks*, 12: 242–253, 1982.

L.G. Valiant. The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8:410–421, 1979.