Approximating the Performance of Stochastic Distribution Systems

JAMES P. JARVIS / Department of Mathematical Sciences, Clemson University, Clemson, SC 29634-1907; e-mail: jpjrv@clemson.edu

DOUGLAS R. SHIER / Department of Mathematical Sciences, Clemson University, Clemson, SC 29634-1907; e-mail: shierd@clemson.edu

October 1994

Subject classifications: Reliability: multistate systems; Networks/graphs: applications, flows, stochastic

Abstract: A problem encountered in the analysis of telecommunication and other distribution systems is evaluating the performance of the system in meeting user demands with available resources. We consider the case in which user demands and available resources are only known stochastically, and connecting links can operate at various levels. This situation can be modeled as a stochastic network flow problem, in which each edge of the network assumes a finite number of values (corresponding to different capacity levels) with known probabilities. Each state of the network corresponds to a specification of supplies, demands, and link capacities in the given system. For any such state we are interested in whether overall demand can be met using the present supply of resources. If not, we are interested in the maximum demand that can be met using the best allocation of resources. The approach used here to estimate the probability of unmet demand, as well as the average unmet demand, involves generating only "high leverage" states of the system—states having high probability and/or high values of unmet demand. A new method is proposed for generating such states in monotone order, either by probability or unmet demand. Various bounds on performance measures for the system are investigated.

1. Introduction

Telecommunication systems are frequently studied using a graph-theoretic model, in which various sites (nodes) are connected by communication links (edges). Considerable attention has focused on reliability analysis of such systems, typically based on the connectivity of the graph relative to possible link failures. In practice, links degrade rather than simply fail and the system itself achieves various levels of performance, not just working or failed. Consequently, more realistic analyses of distribution and communication systems need to consider congestion, delay, and throughput of the system [2, 16, 20, 25]. Since these systems typically involve the flow of goods or information from one location to another, a network flow model can be useful in conducting such analyses.

As an illustration, packets of information in a telecommunication network need to be routed along capacitated links to satisfy requirements at other locations. In the case of distributed computing, user requests at diverse network nodes can be served by allocating limited resources located throughout the network. In a power distribution system, existing supplies need to be transmitted along capacitated lines of the system in order to satisfy anticipated demands. In all such examples, the flow of supplies to satisfy demands is carried out subject to certain capacity limitations. Deterministic network flow models have numerous additional applications, and a variety of efficient solution algorithms have been developed for this problem [4, 26]. More realistically, the elements of such a flow network should be viewed as *stochastic*, rather than deterministic, since the supplies, demands, and transmission capacities are rarely known with certainty. A stochastic version of the standard network flow problem is the focus of this paper.

One motivating example is that of electric power distribution systems, in which there are generating stations whose output (or supply) is not necessarily fixed, but which can assume a finite number of operating levels with certain known probabilities. Also the demands at each location take on a finite number of levels with known probabilities, and the

transmission lines comprising the distribution system have random capacities. Figure 1.1 shows an example system [5] in which there are seven nodes and nine edges (three edges are bi-directional). Certain characteristics of the network components are described in Table 1.1. Notice that nodes a, b, c, d are classified as both supply and demand nodes; deterministic elements are identified by those with a single capacity level. By adding a new source node s and a sink node t, as well as additional capacitated edges to represent nodal supplies and demands, the power system can be represented by a stochastic network G in which the nodes operate deterministically and only the edges have random capacities. Since node g has deterministic supply and since edges (g, a) and (g, c) have deterministic capacities, the effect of this node has been eliminated in constructing the modified network G, shown in Figure 1.2. The number of distinct capacity levels, if greater than one, is indicated along each edge.

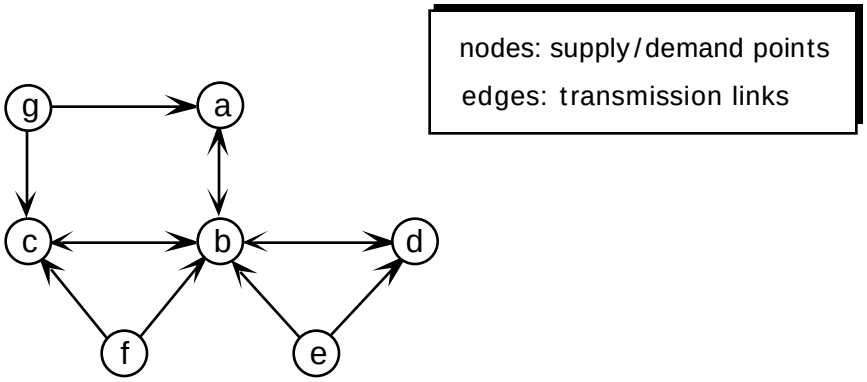

Figure 1.1. Electric power system example

Another motivating example occurs in designing primary and backup databases for users in a local area network. Table 1.2 describes the specific case of seven user sites connected to three databases. For instance, user A can obtain records from either database 1 or database 2, whereas user C can obtain records from any of the three databases. Both user demands and database capacities are stochastic; e.g., user A requires 4, 6, or 8 accesses per time period (with certain probabilities which are given in Table 4.2), while database 1 can

supply 14, 17, or 20 accesses per time period (again with probabilities given in Table 4.2). For simplicity, we assume that the communication links operate deterministically. This problem can again be transformed into a two-terminal stochastic network G by introducing a source node s (joined by an edge to each database) and a sink node t (joined by an edge to each user), as shown in Figure 1.3. Here the source and sink edges operate stochastically (each with three capacity levels), whereas the communication links are assumed to be perfect.

Table 1.1. Characteristics of the power system example

	# Capacity Levels			
Nodes	Supply	Demand	Edges	# Capacity Levels
a	12	1	ab	3
b	11	1	b c	2
c	11	1	bd	2
d	9	1	eb	1
e	6		ed	1
f	5		f b	1
g	1		f c	1
			ga	1
			g c	1

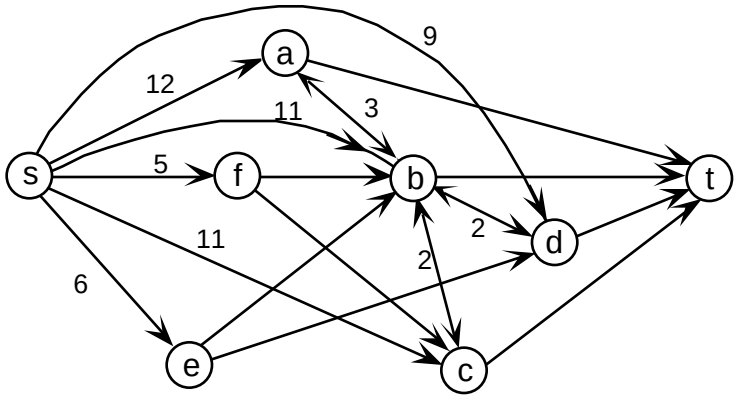

Figure 1.2. Flow network G for power system example

Table 1.2. Characteristics of the database example

Users	Databases Accessed	Demand Levels
A	1, 2	4, 6, 8
B	1, 2	5, 7, 9
C	1, 2, 3	6, 8, 10
D	1, 2, 3	7, 9, 11
E	1, 2, 3	6, 8, 10
F	2, 3	5, 7, 9
G	2, 3	4, 6, 8

Databases	Supply Levels
1	14, 17, 20
2	25, 30, 35
3	14, 17, 20

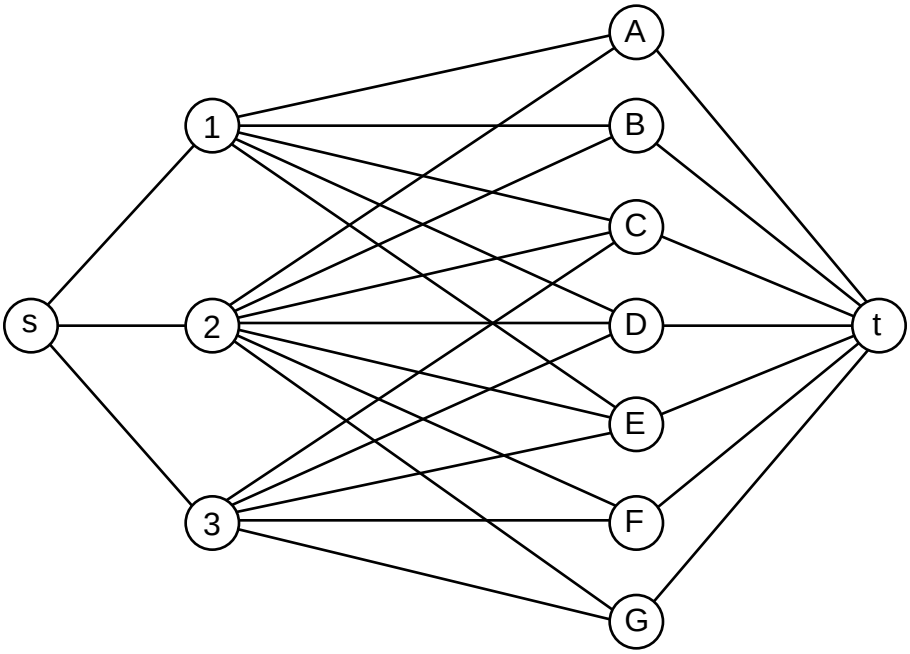

Figure 1.3. Flow network G for database example

In general, for any *realization* **x** of a two-terminal stochastic network (i.e., a specification of capacities for all edges), it is straightforward to determine whether a *feasible* s-t flow exists: that is, a flow from s to t that satisfies the required capacities of all demand edges (leading to node t). If edge capacities are not sufficient to satisfy all these demands, the system has "failed" and there is positive "unsatisfied demand." More formally, given a realization **x** having total demand $\Delta(\mathbf{x})$ required at t, we define the unmet (unsatisfied) demand by $UD(\mathbf{x}) = \Delta(\mathbf{x}) - f^*(s, t; \mathbf{x}) \geq 0$, where $f^*(s, t; \mathbf{x})$ is the maximum flow possible from node s to node t in realization **x**. We wish to calculate certain measures of the *average performance* of this stochastic system: (a) the probability π that positive unmet demand occurs, and (b) the expected amount of unmet demand $\overline{UD}$. The latter is a "performability" measure [1, 18] that captures the joint contribution of reliability and capacity to system effectiveness.

Since it is known that exact computation of the performance measures π and $\overline{UD}$ is difficult (namely, NP-hard), the focus of the present paper is on deriving approximations to

these quantities. Our approach will utilize certain state space approximation methods previously developed in the literature [1, 18] to obtain bounds on system performance. A different approach to improving bounds on performability measures has been recently investigated by Yang and Kubat [28] for networks whose components have two states. Other (static) bounds on the expected value of maximum flows in stochastic networks have been studied by Carey and Hendrickson [7] and by Nagamochi and Ibaraki [19]. Alternative exact approaches to the stochastic network flow problem have been discussed by Bellovin [5], Doulliez and Jamoulle [9], Evans [10], Shogan [23], and Somers [24]. In addition, a variety of simulation approaches have been developed by Fishman and colleagues [3, 11, 12, 13].

Section 2 presents a formal statement of the stochastic network flow problem and discusses how state space generation methods can be applied to this problem. A new state generation rule, particularly appropriate for the stochastic flow problem, is developed in Section 3. Section 4 shows how improved bounds can be derived on the unknown quantities π and $\overline{UD}$. Computational schemes necessary for implementing this approach in an effective way are described in the final section.

2. State Space Generation

Suppose that the given network G has source node s, sink node t, and edges designated 1, 2, …, m. Any edge i assumes a finite number m_i of *modes*, each corresponding to a distinct capacity level for that edge. The modes for edge i are numbered 0, 1, …, $m_i - 1$. Let p_{ij} denote the probability that edge i is found in mode j and let c_{ij} denote the capacity of edge i in mode j. Thus a *state* of the system is represented by the vector $\mathbf{x} = (j_1, j_2, \ldots, j_m)$ where j_i is the mode assumed by edge i. Under the (simplifying) assumption that edges operate independently, the probability of occurrence of state **x** is given by

$$p(\mathbf{x}) = \prod_{i=1}^{m} p_{i,j_i} . \qquad (2.1)$$

Let D denote those edges, all terminating at t, associated with demand requirements. In each state $\mathbf{x} = (j_1, j_2, \ldots, j_m)$ of the system, all edges have fixed capacities. In particular those edges i D have fixed demand requirements. To determine if the total demand into node t in realization **x**

$$\Delta(\mathbf{x}) = \sum_{i \in D} c_{i,j_i}$$

can be satisfied, we calculate the maximum flow f*(s, t; **x**) from node s to node t in G, relative to the capacities specified by **x**. Notice that the edges incident to node t form an s-t cut, so $f^*(s, t; \mathbf{x}) \le \Delta(\mathbf{x})$ certainly holds. If $f^*(s, t; \mathbf{x}) = \Delta(\mathbf{x})$ then all demands can be satisfied and $UD(\mathbf{x}) = 0$. Otherwise, the system incurs an unmet demand of $UD(\mathbf{x}) = \Delta(\mathbf{x}) - f^*(s, t; \mathbf{x}) > 0$. The overall performance of the system can then be measured by the probability π of positive unmet demand and the expected amount of unmet demand $\overline{UD}$:

$$\pi = \Pr\{UD(\mathbf{x}) > 0\} = \sum_{UD(\mathbf{x}) > 0} p(\mathbf{x}), \qquad (2.2)$$

$$\overline{UD} = \sum_{\mathbf{x}} UD(\mathbf{x})\, p(\mathbf{x}). \qquad (2.3)$$

Since the entire state space is exponentially large, consisting of $\prod m_i$ states, determining these quantities by brute force enumeration is impractical. To illustrate, even in the small example of Figure 1.2, the entire state space consists of 4,704,480 states. The small database example described by Table 1.2, consisting of seven users and three databases (with perfect communication links), produces 59,049 states. In each of these states **x** one would need to find the value of UD(**x**) in order to evaluate the performance measures (2.2)–(2.3).

One approach employed to deal with the state space explosion problem has been to use partial state space enumeration [15, 17, 21, 22]. Namely, instead of generating all states of the system, we generate the states one at a time in order of non-increasing *probability*. This

has proved to be an effective method for approximating the performance of certain communication systems, such as in estimating system reliability or average delay. A number of algorithms have been proposed in the literature to carry out the enumeration of "most probable states" for multimode systems [8, 14, 27]. A recent computational study [6] has found that the algorithm of Gaebler and Chen [14] is typically the most effective one in practice. In Section 3 we will describe the main steps of that algorithm as well as a new generation procedure, which (unlike the Gaebler–Chen approach) can be used to produce states in "most important" order.

First, let us suppose that a subset of states $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_r$ has been generated, corresponding (say) to the r most probable states: $p(\mathbf{x}_1) \geq p(\mathbf{x}_2) \geq \cdots \geq p(\mathbf{x}_r)$. Then upper and lower bounds on π and $\overline{UD}$ are easily obtained. To express these bounds, let S(r) denote the set of states $\mathbf{x}_i$, $1 \leq i \leq r$, for which $UD(\mathbf{x}_i) > 0$. We then have

$$\sum_{\mathbf{x} \in S(r)} p(\mathbf{x}) \leq \pi \leq \sum_{\mathbf{x} \in S(r)} p(\mathbf{x}) + [1 - \sum_{i=1}^{r} p(\mathbf{x}_i)], \tag{2.4}$$

$$\sum_{i=1}^{r} UD(\mathbf{x}_i)\, p(\mathbf{x}_i) \leq \overline{UD} \leq \sum_{i=1}^{r} UD(\mathbf{x}_i)\, p(\mathbf{x}_i) + [1 - \sum_{i=1}^{r} p(\mathbf{x}_i)]\, UD_{max}, \tag{2.5}$$

where UD_{max} is the unmet demand when all edges i D are set to their largest capacity and all edges i D are set to their smallest capacity. These bounds will be most useful when the number of generated states is relatively small yet the accumulated probability (or average unmet demand) in these states is close to the unknown π (or $\overline{UD}$). Another way of saying this is that the generated states should be the most significant of the entire state space in terms of the performance measure being estimated.

The example network of Figure 1.2 provides some important insights into the quality of this approximation approach. Figure 2.1 displays the cumulative probability $\sum_{i=1} p(\mathbf{x}_i)$ as a function of the number of states r when the states of this example are generated in most probable order. It is seen that most of the probability of the state space is found in relatively

few of the states; indeed, 99% of the total probability is captured in less than 0.6% of the states. Unfortunately, unmet demand is zero for *all* these generated states (though they account for 99% of the probability), and consequently the bounds (2.4)–(2.5) have mixed utility. The probability of unmet demand π satisfies $0 \le \pi \le 0.01$; however, since $UD_{max} = 1145$, the bounds on $\overline{UD}$ are only $0 \le \overline{UD} \le 11.45$. In other words, the most interesting states (those with positive unmet demand) are not the most probable ones. A similar situation occurs in the database example of Table 1.2. If the most probable states are enumerated, then 95% of the total probability is captured by 9332 states. However, only 9 of these states have positive unmet demand and the bounds $0.0001 \le \overline{UD} \le 0.6001$ produced by (2.5) are not very informative. In this case the exact value is given by $\overline{UD} = 0.0013$.

In these examples, as well as others illustrated in [20], the most important states, and not those which occur most often, should be the focus of the analysis. Using the most probable state methodology of [8, 14, 27] will produce unsatisfactory bounds (2.5) when the most important states (positive unmet demand) correspond to rare events—and this often occurs in practice. On the other hand, crude Monte Carlo sampling can produce unbiased estimates of π and $\overline{UD}$, but huge sample sizes will be required before a sufficient number of interesting states are encountered.

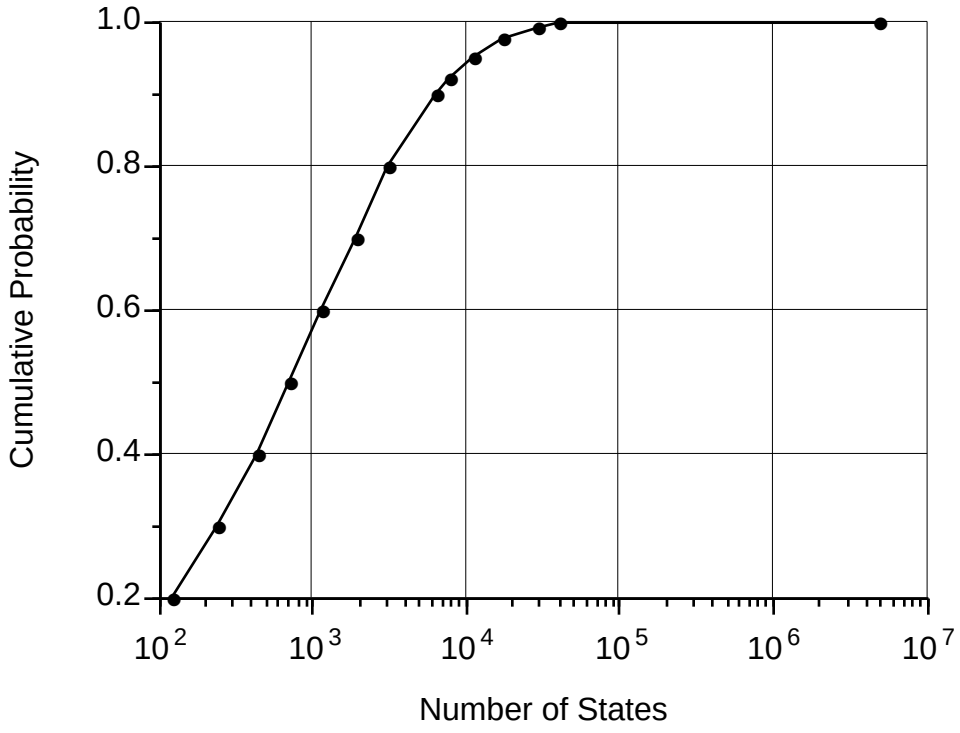

Figure 2.1. Cumulative probability for most probable states

An alternative approach is to generate the states in a different order, one that permits enhanced estimation of $\overline{\text{UD}}$. Specifically, we propose generating the states **x** in order of non-increasing unmet demand UD(**x**). As shown in the next section, this can be done by developing a new procedure for producing the *most important* states of a system. Our procedure has the added flexibility of being able to generate most probable states as well as those having highest unmet demand. Information from these two types of state generation can be combined to yield improved bounds on $\overline{\text{UD}}$, as discussed in Section 4.

3. Algorithms for State Space Generation

Here we discuss monotone generation of the states **x** of a multimode system according to a given criterion function $\phi(\mathbf{x})$. Virtually all the previous literature has concentrated on the case $\phi(\mathbf{x}) = p(\mathbf{x})$, corresponding to monotone generation of most probable states. More generally, this criterion function could be state probability or it might represent delay,

congestion, or unmet demand in the given state. To carry out the generation of states, a list C of *candidate* states is initialized to contain only one state $\mathbf{x}_0$ having maximum value $\phi(\mathbf{x}_0)$ among all states in the state space. At each step a *generated* state is produced by removing from the candidate list C a state $\mathbf{x}$ having maximum value $\phi(\mathbf{x})$. Then all *successor* states of $\mathbf{x}$, defined by certain successor rules, are added to the list C and the process is repeated. The procedure should ensure that each state is generated once and only once by this method, and that the states are in fact generated in monotone order by non-increasing value of $\phi(\mathbf{x})$. This can be assured by successor rules that satisfy the following conditions:

C1. If $\mathbf{y}$ is a successor state of $\mathbf{x}$ then $\phi(\mathbf{y}) \leq \phi(\mathbf{x})$.

C2. If $\mathbf{y} \neq \mathbf{x}_0$ is any state of the system then there is a unique (predecessor) state $\mathbf{x}$ such that $\mathbf{y}$ is a successor of $\mathbf{x}$.

In effect the monotone generation procedure defines a directed tree rooted at node (state) $\mathbf{x}_0$. Successors $\mathbf{y}$ of node $\mathbf{x}$ are placed as immediate descendants ("children") of node $\mathbf{x}$ in the tree. By (C2) there is a unique path from each node $\mathbf{y}$ to the root node $\mathbf{x}_0$, traced by following in turn the predecessors of $\mathbf{y}$ in the tree. Moreover by (C1) the function $\phi(\mathbf{x})$ is guaranteed to be monotone non-decreasing along this path. Of course in practice we will be interested in generating only a small portion of the entire tree. Figure 3.1 illustrates the development of this tree at a typical step of the generation procedure. Certain nodes (denoted by open circles) have already been generated (removed from the candidate list and their successors inserted on the list), while other nodes remain on the candidate list (denoted by darkened circles). All other nodes (descendants in the tree of nodes on the candidate list) remain to be identified.

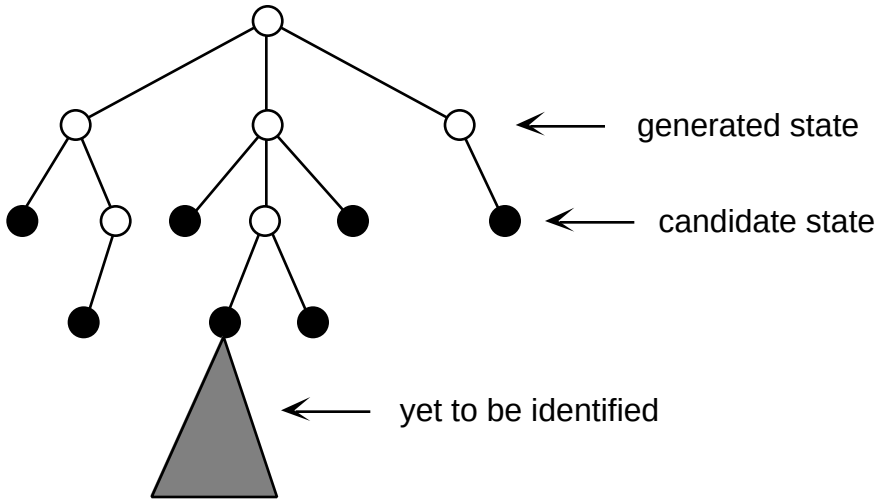

Figure 3.1. Typical state generation tree

A number of authors [8, 14, 27] have proposed rules for generating the *most probable states* of a multimode system in order of non-increasing probability: that is, $\phi(\mathbf{x}) = p(\mathbf{x})$, with $p(\mathbf{x})$ given by (2.1). We focus on the method of Gaebler and Chen [14], which has proved to be an especially effective method in empirical studies [6]. The modes associated with each edge i are first ordered by non-increasing probability: $p_{i,j-1} \geq p_{i,j}$ for all $1 \leq j < m_i$. Also the edges i are ordered by non-increasing ratios of mode 1 to mode 0 probabilities: $p_{i,1} / p_{i,0} \geq p_{i+1,1} / p_{i+1,0}$ for all $1 \leq i < m$. Suppose in the given state $\mathbf{x} = (j_1, j_2, \ldots, j_m)$ that k indicates the largest index such that $j_k > 0$. The state successor rule of Gaebler and Chen produces at most three successor states $\mathbf{y}_0, \mathbf{y}_1, \mathbf{y}_2$ of $\mathbf{x}$ according to:

GC1. if $j_k < m_k - 1$ then $\mathbf{y}_0 = \mathbf{x}$ except that edge k of $\mathbf{y}_0$ has mode $j_k + 1$;

GC2. if $k < m$ then $\mathbf{y}_1 = \mathbf{x}$ except that edge k + 1 of $\mathbf{y}_1$ has mode 1;

GC3. if $k < m$ and $j_k = 1$ (hence $j_{k+1} = 0$) then $\mathbf{y}_2 = \mathbf{x}$ except that edge k of $\mathbf{y}_2$ has mode 0 and edge k + 1 of $\mathbf{y}_2$ has mode 1.

Because of the presumed ordering of modes (by non-increasing probability for each edge), then $\phi(\mathbf{y}_0) \leq \phi(\mathbf{x})$ and $\phi(\mathbf{y}_1) \leq \phi(\mathbf{x})$ hold. In addition, from the presumed ordering of edges (by non-increasing ratios), then $\phi(\mathbf{y}_2) \leq \phi(\mathbf{x})$ holds. Finally, it can be shown that each state

y has a unique predecessor state **x**. Thus (C1)–(C2) are satisfied and this is a valid successor rule for generating most probable states.

We now present a new successor rule for generating states in "most important" order: namely, in order of non-increasing $\phi(\mathbf{x}) = UD(\mathbf{x})$. First the modes for each edge i need to be reordered in the following manner. If i D then the modes of i are ordered by non-increasing capacity levels: $c_{i,j-1} \geq c_{i,j}$ for all $1 \leq j < m_i$. However, if i D then the modes of edge i are ordered by non-decreasing capacity levels: $c_{i,j-1} \leq c_{i,j}$ for all $1 \leq j < m_i$. This ordering insures that increasing the mode of a single edge will not increase the unmet demand. No particular ordering of the edges is required, but empirically it appears effective to order the edges i by non-increasing number of modes: $m_i \geq m_{i+1}$ for all $1 \leq i < m$.

To describe the new state generation rule applied to state $\mathbf{x} = (j_1, j_2, \ldots, j_m)$, suppose again that k is the largest index such that $j_k > 0$. Then the $m - k + 1$ successors $\mathbf{y}_j$ of **x** are given by:

JS1. if $j_k < m_k - 1$ then $\mathbf{y}_0 = \mathbf{x}$ except that edge k of $\mathbf{y}_0$ has mode $j_k + 1$;

JS2. if $k < m$ then for each $1 \leq j \leq m - k$ define $\mathbf{y}_j = \mathbf{x}$ except that edge $k + j$ of $\mathbf{y}_j$ has mode 1.

Since modes are ordered by capacity level, it follows that $\phi(\mathbf{y}_j) \leq \phi(\mathbf{x})$ holds for all $j = 0, 1, \ldots, m - k$. Also any state $\mathbf{y} = (n_1, n_2, \ldots, n_m)$ has a unique predecessor defined by this rule. Namely, if k denotes as before the largest index such that $n_k > 0$, then $\mathbf{x} = (n_1, n_2, \ldots, n_k - 1, \ldots, n_m)$ is the unique predecessor of **y**. Thus (JS1)–(JS2) yield a valid successor rule for generating states in order of non-increasing unmet demand.

It should be emphasized that the Gaebler–Chen rule is *not* valid for monotone state generation by unmet demand, since $\phi(\mathbf{y}_2) \leq \phi(\mathbf{x})$ need not necessarily hold in (GC3) and thus condition (C1) is not satisfied. To see this, notice that in the Gaebler–Chen rule state $\mathbf{y}_2$ corresponds to changing the capacities on *two* different edges (k and k + 1) in the network. Regardless of how the modes for each edge are ordered, unmet demand can either increase

or decrease after both these capacity changes have been made. This justifies the need for our new successor rule (JS1)–(JS2), which does ensure monotone generation of states for $\phi(\mathbf{x}) = \mathrm{UD}(\mathbf{x})$.

Moreover this new rule can be adapted to the generation of states in *most probable order* simply by ordering the modes for each edge by non-increasing probability. It is advantageous to order edges in the JS scheme by *non-decreasing* ratios $p_{i,1}/p_{i,0}$ rather than by non-increasing ratios (as in the Gaebler–Chen procedure). In fact, the JS procedure exhibits significant computational benefits when compared to the GC procedure.

Table 3.1 gives a comparison of the performance of the two procedures when applied to the power system example described in the first section (Figures 1.1 and 1.2). The execution times for both the JS and GC procedures are modest although the JS procedure is generally about 20–25% faster. (Execution times were obtained on a microcomputer equipped with a 40MHz Motorola 68030/68881 processor.) More importantly, the number of candidate states generated is significantly higher for GC than for JS. At a cumulative probability of 0.98, the ratio of GC to JS candidate states is six and increasing. In practical terms, storage requirements rather than execution times are more likely to restrict the problem sizes which can be considered, resulting in a significant advantage for JS over GC.

Table 3.1. Comparative enumeration of high probability states

Cumulative State Probability	Number of States Enumerated	Number of Candidate States		Execution Time (sec)	
		JS	GC	JS	GC
0.101	43	35	65	1	1
0.200	118	71	172	1	1
0.300	237	128	352	1	1
0.400	419	208	639	1	1
0.500	691	302	1060	1	2
0.600	1108	423	1700	2	2
0.700	1791	632	2731	2	2
0.800	3058	974	4601	3	3
0.900	6202	1739	9180	4	5
0.910	6804	1862	10063	4	6
0.920	7518	2067	11114	5	6
0.930	8383	2227	12357	5	7
0.940	9458	2502	13897	6	8
0.950	10832	2845	15870	6	8
0.960	12668	3267	18482	7	10
0.970	15287	3824	22177	9	12
0.980	19472	4635	28020	11	15
0.990	28016	6345	39745	16	21
0.991	29478	6580	41790	17	22
0.992	31166	6961	44073	18	23
0.993	33160	7253	46831	19	25
0.994	35557	7805	†49998	20	27
0.995	38543	8321	—	22	—

† Reached limit of 50,000 candidate states at 0.99396 cumulative probability.

In addition, a series of “random” examples having 10 components and 4 modes per component were constructed. Specifically, for each component i the probabilities for the various modes j were obtained by geometrically scaling successive probabilities by a random factor r_i [1, 5]. Results for such random examples were quite similar, and thus a

single representative example is used for comparative purposes in Figure 3.2. This figure shows the growth in the number of enumerated states with the cumulative probability, as well as the growth in the size of the candidate list for both the GC and JS procedures. In the JS procedure, the candidate list grows rather slowly with the cumulative probability, as contrasted with the GC procedure for which the list grows in step with the number of enumerated states. At 0.927 cumulative probability the GC procedure reached the imposed limit of 50,000 candidate states, whereas the JS procedure had only accumulated 7842 candidate states. Moreover, the JS procedure was able to generate *all* the 4^{10} = 1,048,576 states of this system using at most 23,000 locations to store candidate states; by contrast the GC procedure reached the 50,000 candidate state limit after enumerating only 54,735 of the most probable states.

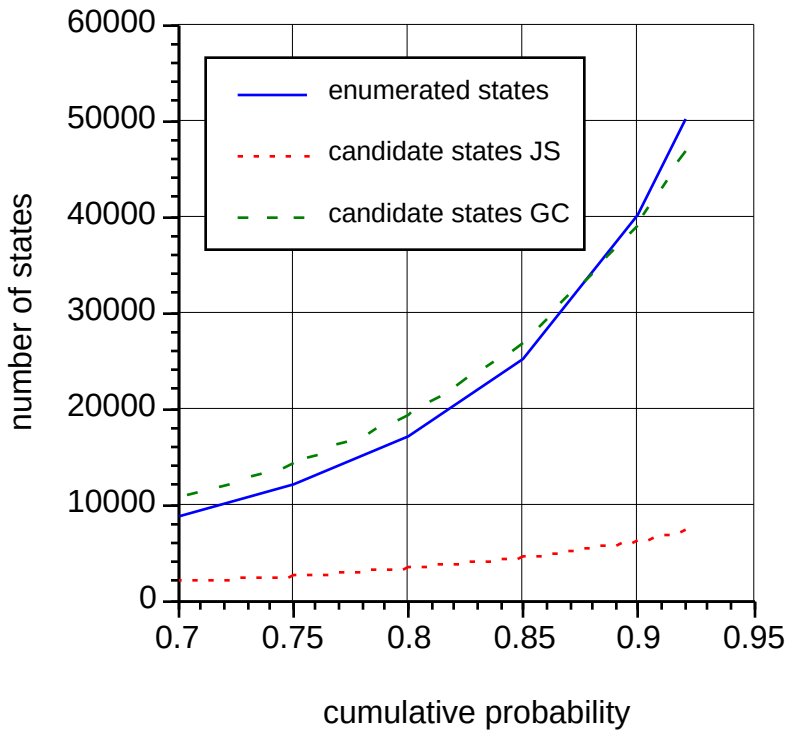

Figure 3.2. Comparison of algorithms on random example

To summarize, the new rule (JS1)–(JS2) given here can be used to generate most important states, in order of non-increasing unmet demand $\phi(\mathbf{x}) = UD(\mathbf{x})$. The Gaebler–Chen algorithm cannot be used to monotonically generate states in order of unmet demand. Moreover, the JS rule can be adapted to produce states in order of non-increasing probability $\phi(\mathbf{x}) = p(\mathbf{x})$. In the empirical tests conducted here, the JS rule appears to be superior to the GC rule when both are used to generate most probable states, notably in space requirements. The next section describes how information from the most probable and most important state generation procedures can be combined to yield better estimates of the desired performance measures (2.2)–(2.3).

4. Improved Bounds on System Performance

As can be seen in (2.3), the value of $\overline{UD}$ can be significantly affected by either high probability states or high unmet demand states. Since both types of states can be generated by the new procedure (JS1)–(JS2), we will investigate in this section how improved bounds on the performability measure $\overline{UD}$ can be derived by combining information from the two state generation procedures.

As an alternative to the bounds on $\overline{UD}$ given by (2.5), consider the state enumeration tree shown in Figure 3.1. There are three types of states represented in the tree: generated states, candidate states, and candidate successor states (yet to be identified). For both the generated and candidate states, unmet demand is calculated as part of the state generation procedure. Unmet demand for the candidate successor states is not known, but for any candidate state $\mathbf{x}$ with unmet demand $UD(\mathbf{x})$, condition (C1) for state generation guarantees that $UD(\mathbf{x})$ is an upper bound on the unmet demand of every successor state of $\mathbf{x}$. In addition, the total probability of all successor states of $\mathbf{x}$ can be easily calculated for the generation rule (JS1)–(JS2).

Let $\mathbf{x} = (j_1, j_2, \ldots, j_m)$ and let k be the largest index with $j_k > 0$. Then PS´(**x**), the total probability of all successors of **x** (including **x**), is given by

$$PS'(\mathbf{x}) = \prod_{i=1}^{k-1} p_{i,j_i} \left[1 - \sum_{r=0}^{j_k - 1} p_{k,r}\right] . \tag{4.1}$$

The above expression for PS´(**x**) is derived by noting that in (JS1)–(JS2) the successor states of **x** are precisely those with any combination of modes for edges k + 1 through m and with modes j_k or higher for edge k. For subsequent analysis, it is convenient to define PS(**x**) = PS´(**x**) – p(**x**), the probability of all successors of **x**, not including **x**.

Using these results, we obtain alternative bounds on $\overline{UD}$:

$$\sum_{\mathbf{x}\ S\ C} UD(\mathbf{x})\, p(\mathbf{x}) \le \overline{UD} \le \sum_{\mathbf{x}\ S\ C} UD(\mathbf{x})\, p(\mathbf{x}) + \sum_{\mathbf{x}\ C} UD(\mathbf{x})\, PS(\mathbf{x}) , \tag{4.2}$$

where S is the set of generated states and C is the set of candidate states. These bounds can be further improved by combining (4.2) with the enumeration of high probability states. This strategy will be illustrated by reference to Figure 4.1, which is a modification of Figure 3.1. The range of the bounds (4.2) depends on the magnitude of the last summation in the upper bound expression, which is associated with the candidate list successors. Rather than simply assigning the upper bound UD(**x**) to all successors of candidate state **x**, we can account explicitly for those high probability states previously enumerated that are successor states of **x**. These high probability states typically have a much lower unmet demand than does their ancestor **x**.

Let SU(**x**) denote the successor states of **x** and let H denote the previously enumerated high probability states. Then the combined bounds on $\overline{UD}$ are given by

$$\sum_{\mathbf{x}\ S\ C\ H} UD(\mathbf{x})\, p(\mathbf{x}) \le \overline{UD}$$

$$\overline{UD} \le \sum_{\mathbf{x}\ S\ C\ H} UD(\mathbf{x})\, p(\mathbf{x}) + \sum_{\mathbf{x}\ C} UD(\mathbf{x}) \left[PS(\mathbf{x}) - \sum_{\mathbf{y}\ SU(\mathbf{x}) \cap H} p(\mathbf{y})\right] . \tag{4.3}$$

As illustrated by Figure 4.1, the bounds (4.2) on $\overline{UD}$ are derived from bounds on the unmet demand of the successor states. These bounds are tightened to (4.3) by explicitly considering the high probability states appearing in each successor set. The efficacy of the various bounds will be illustrated through several examples.

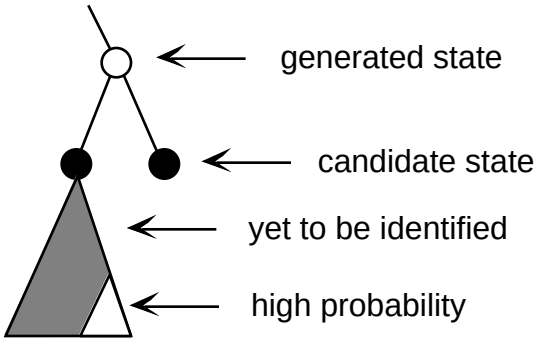

Figure 4.1. Successor states including high probability states

Example 1. Consider the network with 7 nodes and 12 edges shown in Figure 4.2(a). Each edge has either two or three modes, with capacities and probabilities as specified in Table 4.1. In this stochastic network, there are 20,736 different states, and we are interested in the expected value $\bar{f}_{17}$ of the maximum flow from node 1 to node 7. To reformulate this problem in terms of calculation of unmet demand, we add a new sink node 8 and demand edge (7, 8) with deterministic capacity 100, as shown in Figure 4.2(b). Since UD(**x**) = Δ(**x**) – f*(1, 8; **x**) = 100 – f*(1, 7; **x**), then $\overline{UD} = 100 - \bar{f}_{17}$ and so the expected maximum flow from 1 to 7 can be found by first calculating $\overline{UD}$ for the network G of Figure 4.2(b). Using the most probable states alone (with 95% coverage), the bounds (2.5) are $74.42 \le \overline{UD} \le 78.97$. Using the most important states (with UD ≥ 90) alone, the bounds (4.2) are $0.18 \le \overline{UD} \le 85.91$. By combining the most probable and most important state generation procedures, (4.3) produces the improved bounds $76.14 \le \overline{UD} \le 78.65$. A total of 7174 states (out of the 20,736) were enumerated to obtain these latter bounds. This yields the bounds $21.35 \le \bar{f}_{17} \le 23.86$ on the expected maximum flow from 1 to 7.

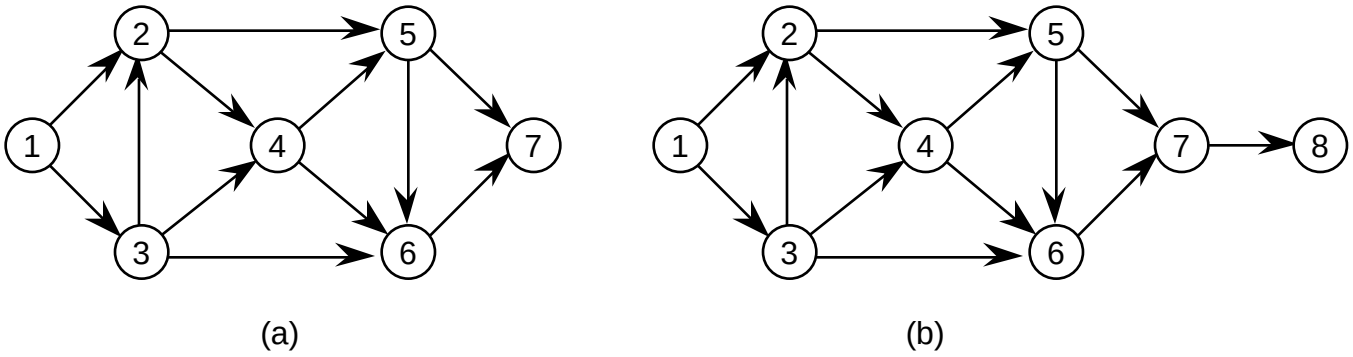

Figure 4.2. A flow network (a) and its transformed network (b)

Table 4.1. Characteristics of the flow network

Edge	Capacities	Probabilities	Edge	Capacities	Probabilities
(1, 2)	8, 12, 15	0.1, 0.8, 0.1	(3, 6)	2, 5	0.1, 0.9
(1, 3)	8, 12, 15	0.1, 0.8, 0.1	(4, 5)	2, 6	0.1, 0.9
(2, 4)	2, 7	0.1, 0.9	(4, 6)	3, 9	0.1, 0.9
(2, 5)	3, 7	0.1, 0.9	(5, 6)	3, 9	0.1, 0.9
(3, 2)	3, 8	0.1, 0.9	(5, 7)	8, 12, 15	0.1, 0.8, 0.1
(3, 4)	2, 8	0.1, 0.9	(6, 7)	8, 12, 15	0.1, 0.8, 0.1

Example 2. We modify the database example given by Table 1.2 and Figure 1.3, by allowing edges (2, A), (2, B), (2, F), (2, G) to be failure-prone: each such edge either works (with probability 0.95) or fails (with probability 0.05). A complete specification of this problem is given in Table 4.2. Edges of Figure 1.3 not listed in this table are perfect and have capacity 100. The system now consists of 944,784 states with $UD_{max} = 12$. Using the combined bounds (4.3) with $p(\mathbf{x}) \geq 0.94$ and $UD(\mathbf{x}) \geq 3$, we obtain the bounds $0.001 \leq \overline{UD}$

≤ 0.061. These bounds require enumerating a total of 57,170 states, representing only 6% of the total number of states.

Table 4.2. Characteristics of the database network

Edge	Capacities	Probabilities	Edge	Capacities	Probabilities
(s, 1)	14, 17, 20	0.1, 0.8, 0.1	(E, t)	6, 8, 10	0.15, 0.7, 0.15
(s, 2)	25, 30, 35	0.05, 0.9, 0.05	(F, t)	5, 7, 9	0.25, 0.5, 0.25
(s, 3)	14, 17, 20	0.1, 0.8, 0.1	(G, t)	4, 6, 8	0.2, 0.6, 0.2
(A, t)	4, 6, 8	0.2, 0.6, 0.2	(2, A)	0, 100	0.05, 0.95
(B, t)	5, 7, 9	0.25, 0.5, 0.25	(2, B)	0, 100	0.05, 0.95
(C, t)	6, 8, 10	0.15, 0.7, 0.15	(2, F)	0, 100	0.05, 0.95
(D, t)	7, 9, 11	0.1, 0.8, 0.1	(2, G)	0, 100	0.05, 0.95

Example 3. We analyze the transportation network with 14 nodes and 30 edges described by Doulliez and Jamoulle [9]. This system has 5 supply nodes, 8 demand nodes, and 2,488,320 states. Using the JS rule with cumulative probability 0.99999 generates 6609 most probable states and 3990 candidate states. In addition, the JS rule produces a set of high unmet demand states with $|S| = 30{,}240$ and $|C| = 9200$. Altogether 50,039 states are thus generated (no duplicate states occurred), representing approximately 2% of the total number of states. The combined high probability/high unmet demand enumerations combine to produce the bounds

$18.26 \leq \overline{UD} \leq 18.27$ and $0.11763 \leq \pi \leq 0.11764$.

Example 4. We consider the system illustrated in Figure 1.2 having 4,704,480 states. Using only high probability states (generated and candidate) with cumulative probability 0.9928 gives the bounds $0.000763 \le \overline{\mathrm{UD}} \le 8.266$. A set of high unmet demand states were also enumerated with |S| = 115,858 and |C| = 9999; as discussed in Section 5, it is the magnitude of |C| which nominally limits the scope of the partial enumeration. Applying (4.2) yields $1.33\times10^{-9} \le \overline{\mathrm{UD}} \le 539.8$. These bounds are so poor because the total probability of the generated and candidate states amounts to only 2.351×10^{-10}; the smallest unmet demand encountered among the generated states is 550 while the smallest unmet demand among the candidate states is a relatively close 450. (Here $\mathrm{UD}_{\max} = 1145$.) However, by incorporating the effect of the most probable states, (4.3) yields $0.000763 \le \overline{\mathrm{UD}} \le 3.894$. In obtaining these results, a total of 34,361 high probability states (28,016 generated, 6345 from the candidate list) totaling 0.9928 probability were examined together with the previously mentioned 115,858 high unmet demand states and 9,999 associated candidate states. Altogether 160,218 states from a total of 4,704,480 were examined, corresponding to 3.4% of the state space. This again shows the efficacy of combined bounds, compared to the use of either most probable or most important states alone, on this very challenging example.

5. Computational Considerations

The basic computational approach taken here derives directly from the new state generation rule (JS1)–(JS2). In enumerating either high probability or high unmet demand states, a balanced k-heap [26] provides an effective means for choosing states with maximum $\phi(\mathbf{x})$ from the candidate list. As a state is placed on the candidate list, a simple but effective flow augmentation via breadth first search [26] is used to determine the maximum flow (and hence unmet demand) for that state.

Although not implemented for the examples given here, maximum flows for successor states can sometimes be determined directly from the maximum flow of the predecessor state. (This is more likely to be useful when the enumeration is by decreasing unmet demand.) An increase in maximum flow through an increase in edge capacity is possible only when the edge belongs to the minimum cut of its predecessor's flow network. Similarly, a decrease in capacity for an edge can produce a change in maximum flow only when the change in capacity exceeds the residual capacity of that edge in the predecessor's flow network. These two conditions can be tested by maintaining a node length array (to identify the minimum cut) and an edge length array (to identify the current maximum flow) for each state in the candidate list whose flow pattern differs from that of its predecessor. In addition, when the first condition holds, the maximum flow for the successor can be found more efficiently by augmenting the known maximum flow of its predecessor.

Since *generated* states are not required for subsequent calculations in state enumeration, these states can be stored off-line or kept by aggregated value (probability, unmet demand). Hence, the limiting computational factor is the size of the *candidate* list. As noted in Section 3, a simple heuristic can effectively limit the growth of the candidate list when enumerating high probability states. For enumeration by decreasing unmet demand, similar behavior is achieved when edges are ordered by non-increasing number of modes. This heuristic tends to produce candidate states with relatively few successor states, which are typically

examined soon after their introduction into the candidate list. In Example 4, this is manifested when over 115,000 high unmet demand states are generated before the size of the candidate list reaches the imposed maximum size of 10,000.

Finally, to compute the bounds given in (4.5), an efficient procedure is needed to determine which enumerated high probability states **y** are included in the candidate list successor states SU(**x**). This is achieved by first encoding each state $\mathbf{x} = (j_1, j_2, \ldots, j_m)$ as an integer I(**x**) where

$$I(\mathbf{x}) = \sum_{k=1}^{m} j_k \prod_{i=k+1}^{m} m_i \,. \tag{5.1}$$

(In the above, any product over the empty set is taken to be 1.) The total number of states is $\prod_{i=1} m_i$ and every state **x** has a unique representation satisfying $0 \le I(\mathbf{x}) \le (\prod_{i=1} m_i) - 1$. For a candidate state $\mathbf{x} = (j_1, j_2, \ldots, j_m)$ with $j_k > 0$ and $j_r = 0$ for $r > k$, an observation similar to that used to derive (4.1) shows that **y** is a successor state of **x** if and only if $I(\mathbf{x}) < I(\mathbf{y}) < I(\mathbf{x}) + (m_k - j_k) \prod_{i=k+1} m_i$. Hence the problem of determining the predecessor **x** C of a state **y** is reduced to (a) ordering the states **x** in the candidate list by their encoded integer I(**x**), and (b) using a binary search to determine the largest value I(**x**) satisfying $I(\mathbf{x}) < I(\mathbf{y})$. If $I(\mathbf{y}) \ge I(\mathbf{x}) + (m_k - j_k) \prod_{i=k+1} m_i$, then the predecessor of **y** has already been generated; otherwise, **x** is the required predecessor.

6. Summary

Determining the performance of systems which can be modeled as stochastic flow networks is a computationally difficult problem which involves the enumeration of an exponentially growing number of states for its complete solution. Bounds on various performance measures can however be obtained by effective partial enumeration of high probability and most important states in such systems. A new technique for both types of enumeration is presented here. Combining the enumeration results for high unmet demand

states with those for high probability states produces bounds on measures of system performance which are superior to those obtained from either approach alone. Several examples are analyzed using this technique and the feasibility of this combined approach is demonstrated. Simply enumerating most probable states would be much less effective in approximating the performance of these examples. Of course, the bounds on unmet demand obtained here can then provide an improved starting point for subsequent Monte Carlo techniques, in order to obtain improved estimates for system performance.

7. References

[1] K. K. Aggarwal, Integration of Reliability and Capacity in Performance Measure of a Telecommunication Network. *IEEE Trans. Rel.* **R-34** (1985) 184–186.

[2] K. K. Aggarwal, Y. C. Chopra, and J. S. Bajwa, Capacity Consideration in Reliability Analysis of Communication Systems. *IEEE Trans. Rel.* **R-31** (1982) 177–181.

[3] C. Alexopoulos and G. S. Fishman, Characterizing Stochastic Flow Networks Using the Monte Carlo Method. *Networks* **21** (1991) 775–798.

[4] M. S. Bazaraa, J. J. Jarvis, and H. D. Sherali, *Linear Programming and Network Flows.* John Wiley, New York (1990).

[5] M. S. Bellovin, Reliability and Shortage Distribution Computations in General Stochastic Transportation Networks. SOL 86-4, Department of Operations Research, Stanford University, January 1986.

[6] E. Bibelnieks, J. P. Jarvis, R. J. Lakin, and D. R. Shier, Algorithms for Approximating the Performance of Multimode Systems. *IEEE INFOCOM 90.* San Francisco, CA, 1990, pp. 741–748.

[7] M. Carey and C. Hendrickson, Bounds on Expected Performance of Networks with Links Subject to Failure. *Networks* **14** (1984) 439–456.

[8] S.-N. Chiou and V. Li, Reliability Analysis of a Communication Network with Multimode Components. *IEEE J. Select. Areas Commun.* **SAC-4** (1986) 1156–1161.

[9] P. Doulliez and E. Jamoulle, Transportation Networks with Random Arc Capacities. *R.A.I.R.O.* **3** (1972) 45–49.

[10] J. R. Evans, Maximum Flows in Probabilistic Graphs — The Discrete Case. *Networks* **6** (1976) 161–183.

[11] G. S. Fishman, The Distribution of Maximum Flow with Applications to Multi-State Reliability Systems. *Operations Research* **35** (1987) 607–618.

[12] G. S. Fishman, Monte Carlo Estimation of the Maximum Flow Distribution in a Network with Discrete Stochastic Arc Capacity Levels. *Naval Res. Logist. Quart.* **36** (1989) 829–849.

[13] G. S. Fishman and T.-Y. D. Shaw, Evaluating Reliability of Stochastic Flow Networks. *Prob. Engin. Infor. Sci.* **3** (1989) 493–509.

[14] R. Gaebler and R. Chen, An Efficient Algorithm for Enumerating States of a System with Multimode Unreliable Components. Technical Report, U.S. Sprint, Overland Park, KS, 1987.

[15] Y. Lam and V. Li, An Improved Algorithm for Performance Analysis of Networks with Unreliable Components. *IEEE Trans. Commun.* **COM-34** (1986) 496–497.

[16] S. H. Lee, Reliability Evaluation of a Flow Network. *IEEE Trans. Rel.* **R-29** (1980) 24–26.

[17] V. Li and J. Silvester, Performance Analysis of Networks with Unreliable Components. *IEEE Trans. Commun.* **COM-32** (1984) 1105–1110.

[18] J. F. Meyer, On Evaluating the Performability of Degradable Computing Systems. *IEEE Trans. Comput.* **29** (1980) 720–731.

[19] H. Nagamochi and T. Ibaraki, Maximum Flows in Probabilistic Networks. *Networks* **21** (1991) 645–666.

[20] B. Sansó and F. Soumis, Communication and Transportation Network Reliability Using Routing Models. *IEEE Trans. Rel.* **40** (1991) 29–38.

[21] D. Shier, A New Algorithm for Performance Analysis of Communication Systems. *IEEE Trans. Communications* **36** (1988) 516–519.

[22] D. Shier, E. Valvo, and R. Jamison, Generating the States of a Binary Stochastic System. *Discrete Appl. Math.* **38** (1992) 489–500.

[23] A. W. Shogan, Modular Decomposition and Reliability Computation in Stochastic Transportation Networks Having Cutnodes. *Networks* **12** (1982) 255–275.

[24] J. E. Somers, Maximum Flow in Networks with a Small Number of Random Arc Capacities. *Networks* **12** (1982) 241–253.

[25] J. D. Spragins, J. C. Sinclair, Y. J. Kang, and H. Jafari, Current Telecommunication Network Reliability Models: A Critical Assessment. *IEEE J. Select. Areas Commun.* **SAC-4** (1986) 1168–1173.

[26] R. E. Tarjan, *Data Structures and Network Algorithms*. SIAM, Philadelphia (1983).

[27] C.-L. Yang and P. Kubat, Efficient Computation of Most Probable States for Communication Networks with Multimode Components. *IEEE Trans. Commun.* **37** (1989) 535–538.

[28] C.-L. Yang and P. Kubat, An Algorithm for Network Reliability Bounds. *ORSA J. Comput.* **2** (1990) 336–345.