Two-Path Subsets: Efficient Counting and Applications to Performability Analysis

by M.O. Ball, J.N. Hagstrom, J.S. Provan

T.R. 96-77

Two-Path Subsets: Efficient Counting and Applications to Performability Analysis [*]

Michael O. Ball

University of Maryland, College Park, MD [†]

Jane N. Hagstrom

University of Illinois, Chicago, IL

J. Scott Provan

University of North Carolina, Chapel Hill, NC

November 8, 1996

[*]University of North Carolina, Department of Operations Research technical report #UNC/OR TR95-5.

[†]The work of this author was supported in part by NSF Grant EEC-9402384

Abstract

The problem of computing performability probabilities in stochastic PERT and flow networks is studied when the network is "minimally designed" to withstand any *two* component failures. Polynomial-time algorithms to compute performability when the network is *planar* — the nonplanar versions being NP-hard — solve related "two-path subset" problems. Given an acyclic graph with weights on the arcs, the algorithms compute the total weight of all subsets of arcs that are contained in (1) two source-sink paths, or (2) two *arc-disjoint* source-sink paths. A polynomial algorithm is given for (1), and for (2) in the case where the graph is a source-sink planar *k-flow graph*, that is, edge-minimal with respect to supporting k units of flow.

AMS Subject Classification 90B25
Keywords: paths, enumeration, performability, PERT, flow, stochastic, planar graph

1 Introduction

Network systems such as communication networks and activity precedence networks are often designed to meet a certain performance criterion with a reasonable reliability. This type of design goal is essentially that of structural engineers who design a bridge to meet anticipated loads and stresses and then add a *margin of safety*. The margin of safety allows for unanticipated environmental stresses or materials failures. Although a desirable goal would be to make the bridge "completely" safe, cost considerations usually prevent attainment of the goal. Thus including a margin of safety usually means over-designing the various specifications for the bridge by a relatively small amount.

In network systems with a performance criterion, the network designer also wants to over-design the network. Its design should allow it to meet the performance criterion, and to be capable of withstanding some level of component failure. In adding this ability to withstand such failures, the designer is including a *margin of reliability*. As in the case of bridge design, cost considerations usually prevent the designer from making this margin large. Steiglitz, Weiner, and Kleitman [7] pursue this aim by defining a survivability criterion which reflects the connectivity of the network and provide a heuristic method of determining a minimum cost design. Monma and Shallcross [3] pursue a similar problem.

In this paper, we pursue two network reliability problems with survivability criteria based on a performance-level criterion: a threshold flow reliability problem, and a threshold project scheduling problem. We will analyze these problems under the assumption that the network has been designed with a small margin of reliability. The two problems are formally defined as follows.

THRESHOLD FLOW PROBLEM

Given: Directed source-sink graph $G = (V, E, s, t)$, with node set V, arc set E, and terminal nodes s and t; probability vector $\boldsymbol{p}$ defined on the arcs of G; flow threshold value f.

Stochastic model: Each arc e operates with probability p_e, in which state it has unit capacity, and fails with probability $1 - p_e$, in which case it can carry no flow.

To find: The probability $RF(G, \boldsymbol{p}, f)$ that the operating arcs of G admit a flow of f or more.

THRESHOLD PROJECT SCHEDULING PROBLEM

Given: Directed acyclic source-sink graph $G = (V, E, s, t)$ with node set V, arc set E, and terminal nodes s and t; vectors $\boldsymbol{a}$ of task times and $\boldsymbol{p}$ of probabilities defined on the arcs of G; project completion threshold time d.

Stochastic model: Each arc e operates with probability p_e, in which state the associated task takes time a_e, and fails with probability $1 - p_e$, in which case the task time is increased by exactly one unit.

To find: The probability $RP(G, \boldsymbol{a}, \boldsymbol{p}, d)$ that the realized task times for the operating and failing arcs admit a project completion time — or equivalently, the length of the longest (s, t) path — of d or less.

These problems were studied in a previous paper by the current authors [1]. Although they show that the problems are NP-hard even with strong restrictions on problem instances, including planarity of G, they show that polynomial algorithms can be constructed in instances where the underlying systems are *r-critical*, that is, minimally over-designed so that they can survive r arc or task completion failures and still maintain the designed capacity or project completion bound. They provide a characterization of an r-critical system in terms of properties of the underlying graph G. For the case when $r = 1$, they provide a polynomial-time algorithm for the threshold flow problem, which can also be applied to the project scheduling problem when G is planar.

In the current paper, we consider 2-critical systems. Provan [6] shows that the 2-critical flow problem is NP-hard. However, we show that, when the 2-critical graph is also (s,t)-planar, both problems can be solved in polynomial time. To do this, we reduce these two problems to those of computing a sum of weighted products over the following two collections:

1. all arc-sets which are contained in the union of two (s,t)-paths;

2. all arc-sets which are contained in the union of two arc-disjoint (s,t)-paths.

We call these arc-sets *two-path subsets* and *two-disjoint-path subsets*, respectively. We then give polynomial-time algorithms for the two-path subset version for acyclic graphs, and the two-disjoint-path subset problem when the graph is also (s,t)-planar and is edge-minimal with respect to supporting k units of flow.

In Section 2 we provide necessary background on acyclic directed graphs, and review the relationship between the two-path enumeration problems above and the computation of threshold reliability. In Section 3, we establish our recursion for the two-path subset enumeration problem. In Section 4, we establish our recursion for the two-disjoint-path subset enumeration problem. In the last section, we survey the results so far on r-critical networks with performance criteria and discuss important directions for future research.

2 Background Material

2.1 Graph Preliminaries

Basic definitions of graph-theoretic terms may be found in the text by Lawler [2]. Throughout this section $G = (V, E, s, t)$ is assumed to be a directed graph with source node s and sink node t. As is usual, each arc $e \in E$ is associated with a pair of vertices in V. We will speak of the arc e as being directed from its *tail* vertex $t(e)$ and into its *head* vertex $h(e)$. Paths in G will always be directed, and for nodes (arcs) x and y an (x, y)-path will be a path whose first node (arc) is x and whose last node (arc) is y. For arcs f_1, f_2, e_1, and e_2, we say that paths P and Q *join* f_1, f_2 to e_1, e_2 if either one is a (f_1, e_1)-path and the other is a (f_2, e_2)-path, or one is a (f_1, e_2)-path and the other is a (f_2, e_1)-path. As an abbreviation, we will refer to P and Q as *disjoint* if they are arc-disjoint.

Definition. Let G be a directed graph. G is called *source-sink planar* if (1) it has a unique node s with indegree 0 and a unique node t with outdegree 0, and (2) G has a plane embedding with s and t on the exterior face.

In the rest of the paper, we will use the shorthand set notation that for arc-set S and arc e, $S + e = S \cup \{e\}$ and $S - e = S \setminus \{e\}$.

It is convenient to think of an acyclic graph G as inducing a partial order on its set of arcs, so that $e \preceq f$ if there is a directed path in G starting at e and ending at f. Arcs e and f are *comparable* if $e \preceq f$ or $f \preceq e$; otherwise they are *incomparable* and we write $e \not\preceq\!\!\succeq f$. With respect to this ordering, we define the (strict) *lower set* (or *lower ideal*) of e to be

$$L(e) \equiv \{f \in E - e | f \preceq e\}.$$

The graph $G(e)$ induced by the arcs of $L(e)$ preserves the partial order on the arcs of G. We will also use the *upper set* (or *upper ideal*) of e,

$$U(e) \equiv \{f \in E - e | e \preceq f\}.$$

We note that for any e, $L(e) \cap U(e) = \emptyset$. We define the *boundary* of $L(e)$ to be $\partial L(e) \equiv \{f \in E \setminus L(e) |\ t(f) \text{ is a vertex of } G(e)\}$.

A property of $\partial L(e)$ that is particularly useful is that it forms a uniformly directed (s,t)-cut of G. A *uniformly directed* (s,t)*-cut* of a directed graph is an (s,t)-cut, in the sense that its removal disconnects s from t, with the additional property that all arcs are directed from the s-side of the cut to the t-side of the cut. (See Provan and Kulkarni [4].) Two distinct arcs belonging to a uniformly directed cut are incomparable with respect to $\preceq$. Furthermore, as long as every arc is on an (s,t)-path, a uniformly directed (s,t)-cut is a minimal (s,t)-cut.

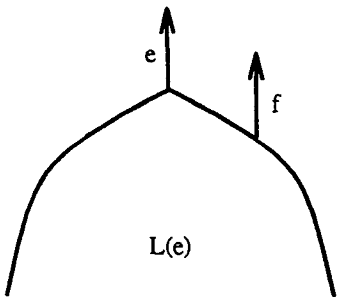

Figure 1: **Schematic of a Lower Set**

Figure 1 provides a schematic drawing of a lower set of an arc e and the boundary of the lower set. The curves can be considered to delineate the lower set of e, with the graph induced by $L(e)$ lying on or below the curves. The arcs belonging to $\partial L(e)$ will be precisely those arcs whose tails lie on the curves and which otherwise are above the curves. Thus e and f belong to $\partial L(e)$.

Uniformly directed (s,t)-cuts have an interesting role in (s,t)-planar graphs. If G is (s,t)-planar, G has an (s,t)-dual graph (see [2], p. 35), obtained by

1. adding arc (s,t) to G (which will not violate planarity);

2. taking the planar dual of G, directing the associated dual arcs by rotating the original arc counterclockwise;

3. removing the arc (s^*,t^*) associated with (s,t).

The resulting directed graph G^D is the (s,t)-dual of G, with dual source and sink nodes s^* and t^*, respectively. It is not difficult to show (and follows immediately from Theorem 8.1 in [2]) that G^D is source-sink planar if G is. Furthermore, a uniformly directed (s,t)-cut of G corresponds to a directed

(s^*, t^*)-path of G^D, and vice-versa. The (s^*, t^*)-paths of G^D induce a partial ordering $\preceq^D$ on its arcs. This leads to a dual ordering in G.

Given $e \not\approx f$ in G, the boundary of the graph induced by $L(e) \cup L(f)$ is also a uniformly directed (s,t)-cut. Thus, if $e \not\approx f$ in G, their dual counterparts e^* and f^* are comparable in G^D since they occur together on an (s^*, t^*)-path. From this we can define e to be to the *right* of f if $e^* \preceq^D f^*$, and e to be to the *left* of f if $f^* \preceq^D e^*$. e is both to the right and to the left of itself; when we want to exclude this case, we will say that e is *strictly* to the left of f. For two paths P and Q, we can define P to be *to the left* (*right*) of Q if there is no arc of $P \setminus Q$ lyin strictly to the right (left) of an arc of Q. In this case we refer to P and Q as a *noncrossing pair*. For two nodes u and v a *leftmost* (*rightmost*) (u, v)-path is a path which lies to the left (right) of every (u, v)-path.

2.2 Relationship Between Critical Threshold Problems and Path Problems

The paper [1] identifies the relationship between the threshold probability problems presented above and certain path problems. Here we give the highlights of that paper as it applies to the problem here. We must define one more concept first.

We define a directed source-sink capacitated graph G with terminals s and t to be a *k-flow graph* if G has a unique maximum flow of k that saturates all arcs of G. When capacities are all one, as in our problem, G is a k-flow graph if and only if G has the following properties:

1. G is acyclic;

2. s has no incoming arcs and exactly k outgoing arcs;

3. t has no outgoing arcs and exactly k incoming arcs;

4. every node v not equal to s or t has the same number of ingoing as outgoing arcs.

Propositions 2.3 and 3.2 in [1] characterize threshold flow and project scheduling systems that are r-critical. The characterizations (with some additional weak conditions) are as follows:

Threshold flow problem: The system is r-critical if and only if G is a k-flow graph, where $k = f + r$.

Threshold project scheduling problem: The system is r-critical if and only if all (s,t)-paths of G have the same length (with respect to the arc task times) of $d - r$.

Notice that the characterization for r-criticality is stated primarily in terms of the graph structure, with the value of r determined by the threshold value f or d. Thus we can talk about r-critical graphs without having to specify r precisely. We will assume henceforth that all instances of the respective problem will be r-critical as indicated above.

Theorems 2.5 and 3.6 in [1] go on to characterize the collection of sets of *failed arcs* that correspond to system operation in an r-critical system.

Threshold flow problem: Let G be an r-critical flow graph with respect to threshold flow f. Then $RF(G, \mathbf{p}, f)$ is equal to the probability that the set S of failed arcs of G is contained in the union of r disjoint (s,t)-paths.

Threshold project scheduling problem: Let G be an r-critical project scheduling graph with respect to threshold completion time d, with G (s,t)-planar, and let G^D be the (s,t)-dual of G, with the same failure probabilities $\boldsymbol{p}$. Then $RP(G,\boldsymbol{a},\boldsymbol{p},d)$ is equal to the probability that the set S of failed arcs in G^D is contained in the union of r (not necessarily disjoint) (s^*,t^*)-paths of G^D.

We will subsequently concentrate on solving these path probability problems. Thus, we will be able to solve the threshold project scheduling problem for (s,t)-planar graphs if we can solve the *r-path problem*, and we will be able to solve the threshold flow problem if we can solve the *r-disjoint-path problem*. The paper [1] gives a polynomial algorithm that solves the 1-path problem (the two versions are identical in this case) on any acyclic graph. As a consequence, when G is 1-critical, we can solve the threshold flow problem in polynomial time. When G is (s,t)-planar, we can also solve the threshold project scheduling problem in polynomial time.

In the current paper, we provide a polynomial algorithm that solves the 2-path problem when G is acyclic. We provide a polynomial algorithm for the 2-disjoint-path problem when G is also source-sink planar and a k-flow graph for some $k \geq 2$. These algorithms are distinctly different for the two types of path problems, but in both cases produce a polynomial-time algorithm for solving the respective threshold problem in the case where G is 2-critical and source-sink planar.

2.3 The Failure-Odds Norm

In the previous section, we cited results showing that for our problems system operability is nicely characterized in terms of allowable sets of failed arcs. In both the current paper and the cited paper, computing the reliability of the system can then be done in terms of a measure defined on the collection of all such allowable sets. We now define that measure.

Let E be a set of elements which we will refer to as the base set. Let $\mathcal{C}$ be a collection of subsets of E. Let each element $e \in E$ have a weight $p_e \neq 0$.

Definition. The *failure-odds norm* is defined as follows:

$$\|\mathcal{C}\| \equiv \sum_{S \in \mathcal{C}} \prod_{e \in S} \frac{1 - p_e}{p_e}$$

We will use some observations about the properties of this norm. We will define $\mathcal{C} \cdot \mathcal{D} \equiv \{S \cup T | S \in \mathcal{C}, T \in \mathcal{D}\}$.

1. $\|\emptyset\| = 0$.

2. $\|\{\emptyset\}\| = 1$.

3. If $\mathcal{C} \cap \mathcal{D} = \emptyset$ then
$$\|\mathcal{C} \cup \mathcal{D}\| = \|\mathcal{C}\| + \|\mathcal{D}\|.$$

4. If $S \cap T = \emptyset$ for all $S \in \mathcal{C}$, $T \in \mathcal{D}$, then
$$\|\mathcal{C} \cdot \mathcal{D}\| = \|\mathcal{C}\| \|\mathcal{D}\|.$$

Note that when $p_e = \frac{1}{2}$ for all $e \in E$, then $\|\mathcal{C}\|$ is the cardinality of $\mathcal{C}$. Because of this observation, we will define an *enumerative expression* to be a collection-valued expression in $\cup$ and $\cdot$ satisfying the disjoint collection and disjoint set conditions of (3) and (4). In [5], a more general version of this in terms of boolean indicators is defined as a *p-normal form expression.*

Reliability computations provide a more general application of these properties. If E is the set of components of a binary system, so that each component has two states, either working or failed, we may define

$$\mathcal{C} \equiv \{S \subseteq E | \text{ the failure of the elements in } S \text{ does not cause the failure of the system } \}.$$

If all elements fail independently and p_e is the probability that element e is working, then the reliability of the system can be factored as $\|\mathcal{C}\| \prod_{e \in E} p_e$. In particular, if we let $\mathcal{C}(G,s,t)$ be the collection of two-path subsets and $\mathcal{D}(G,s,t)$ the collection of two-disjoint-path subsets as defined in Section 1, then from Section 2.2 we get

$$RF(G,\boldsymbol{p},f) = \|\mathcal{D}(G,s,t)\| \prod_{e \in E} p_e$$

and

$$RP(G,\boldsymbol{a},\boldsymbol{p},d) = \|\mathcal{C}(G^D,s^*,t^*)\| \prod_{e \in E} p_e.$$

If we can provide an enumerative expression for $\mathcal{C}$, then the properties of this section may be applied to compute $\|\mathcal{C}\|$ in terms of the norms of smaller collections. We use this approach in the remainder of this paper.

As an application of these ideas, we review the recursive enumeration scheme for $r = 1$ provided in [1].

Let S be a set of failed arcs of the network. If the network is 1-critical, as long as S is contained in a single path, the system continues to operate.

To allow a recursive formulation, define

$$\mathcal{C}^1(e) \equiv \{S \subseteq E| \\ S \text{ is contained in a single path ending at } t(e)\ \}$$

If we create an artificial arc e^t directed out of t, $\mathcal{C}^1(e^t)$ is the collection of all complements of cutsets of the system. Then $\|\mathcal{C}^1(e^t)\| \prod_{e \in E} p_e$ is the reliability of the system. The computation scheme is based on the following recursion.

$$\mathcal{C}^1(e) = \{\emptyset\} \cup \left[\bigcup_{f \in L(e)} \{\{f\}\} \cdot \mathcal{C}^1(f) \right]$$

This enumerative expression for $\mathcal{C}^1(e)$ is readily translated into a recursion in terms of the failure-odds norm. Appropriate ordering of the recursive arguments leads to the polynomial algorithm given in [1] for the 1-critical problems.

In the rest of the paper we will use a similar idea. We will be concerned with collections of sets which are contained in the union of two paths. We will say that a set is *covered* by two paths if it is contained in their union.

3 The Two-Path Problem

In this section we give a polynomial-time algorithm for computing $\|\mathcal{C}(G, s, t)\|$. This algorithm requires only that G be acyclic. The results stated at the end of Section 2.2 indicate that to compute $RP(G, \boldsymbol{a}, \boldsymbol{p}, d)$ it is necessary to assume in addition that G is 2-critical and source-sink planar.

3.1 Recursion

To simplify the presentation of the recursion, we first add additional arcs e_1^s, e_2^s directed into s, e_1^t, e_2^t directed out of t, each with operating probability 1. Now for $e_1, e_2 \in E$, define

$$\mathcal{C}^2(e_1, e_2) \equiv \{S \subseteq E \setminus \{e_1, e_2\} |$$
$$S \text{ is covered by two paths joining } e_1^s, e_2^s \text{ to } e_1, e_2 \}.$$

It follows that

$$\|\mathcal{C}^2(e_1^s, e_2^s)\| = 1.$$

and

$$\|\mathcal{C}^2(e_1^t, e_2^t)\| = \|\mathcal{C}(G, s, t)\|.$$

To produce a recursive formula for general $\|\mathcal{C}^2(e_1, e_2)\|$, we need a more useful characterization of elements in $\mathcal{C}^2(e_1, e_2)$. Let S be a two-path subset. By the definition of $e_1 \not\doteq e_2$, S can be covered by a single path if and only if there do not exist arcs $e_1, e_2 \in S$ such that $e_1 \not\doteq e_2$. If there exists a pair of arcs $e_1, e_2 \in S$ such that $e_1 \not\doteq e_2$, we will say that it is the *highest pair of incomparable arcs in S* if, for any pair $f_1, f_2 \in S$ such that $f_1 \not\doteq f_2$, $\{f_1, f_2\} \subseteq L(e_1) \cup L(e_2) \cup \{e_1, e_2\}$. Since S is contained in the union of two paths, the pair e_1, e_2 is uniquely defined.

If e_1, e_2 is the highest pair of incomparable arcs of S, all arcs of S that are in the upper sets of either of their heads are mutually comparable and comparable to both e_1 and e_2. Thus these arcs belong to $S \cap U(e_1) \cap U(e_2)$ and lie on a single path that can be extended to a path starting at e_1 and also can be extended to a path starting at e_2.

We can compute $||\mathcal{C}^2(e_1^t, e_2^t)||$ recursively if we also have the following collection of sets. For any four arcs e_1, e_2, f_1, f_2 such that $f_1 \in L(e_1)$, $f_2 \in L(e_2)$, define

$$\mathcal{C}^1(e_1, e_2, f_1, f_2) \equiv \{S \subseteq E| \\ S \text{ is covered by a single path } P \subseteq (L(e_1) \cup L(e_2)) \cap (U(f_1) \cap U(f_2)) \ \}.$$

We claim that these two collections can be defined recursively in terms of each other.

Proposition 3.1 *The following equation provides an enumerative expression for* $\mathcal{C}^1(e_1, e_2, f_1, f_2)$.

$$\mathcal{C}^1(e_1, e_2, f_1, f_2) = \{\emptyset\} \cup \\ \bigcup_{g \in (L(e_1) \cup L(e_2)) \cap U(f_1) \cap U(f_2)} \{\{g\}\} \cdot \mathcal{C}^1(g, g, f_1, f_2). \tag{1}$$

Proof: In an acyclic graph, any subset of arcs contained in a path will have a unique highest element. The remaining elements in the path subset will be contained in the lower set of that element. $\{\{g\}\} \cdot \mathcal{C}^1(g, g, f_1, f_2)$ is the collection of all path subsets above f_1 or f_2 with g as their highest element. Each of these collections of path subsets are disjoint and they, together with the empty set, constitute all path subsets contained in $\mathcal{C}^1(e_1, e_2, f_1, f_2)$. ∎

Applying the properties of Section 2.3 yields the following corollary.

Corollary 3.2

$$||\mathcal{C}^1(e_1, e_2, f_1, f_2,)|| = 1 + \\ \sum_{g \in (L(e_1) \cup L(e_2)) \cap U(f_1) \cap U(f_2)} \frac{1 - p_g}{p_g} \cdot ||\mathcal{C}^1(g, g, f_1, f_2)||. \tag{2}$$

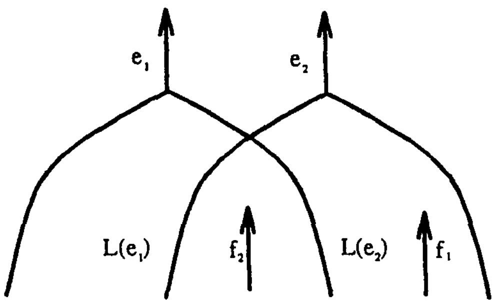

Figure 2: **Schematic of Pair of Arcs in $LL^0(e_1, e_2)$**

The basis of our algorithm for computing $\|\mathcal{C}(G, s, t)\|$ is the next proposition. Define an index set $LL^0(e_1, e_2)$ as follows:

$$\begin{aligned} LL^0(e_1, e_2) \equiv \{ & \{f_1, f_2\} \subseteq E \setminus \{e_1, e_2\} | \\ & f_1 \not\doteq f_2 \text{ and there exist paths joining } f_1, f_2 \text{ to } e_1, e_2 \ \} \end{aligned}$$

A pair of arcs $\{f_1, f_2\}$ belonging to $LL^0(e_1, e_2)$ is illustrated in the schematic in Figure 2.

Proposition 3.3 *The following equation provides an enumerative expression for $\mathcal{C}^2(e_1, e_2)$.*

$$\begin{aligned} \mathcal{C}^2(e_1, e_2) = & \ \mathcal{C}^1(e_1, e_2, e_1^s, e_2^s) \cup \\ & \bigcup_{\{f_1, f_2\} \in LL^0(e_1, e_2)} \mathcal{C}^1(e_1, e_2, f_1, f_2) \cdot \{\{f_1\}\} \cdot \{\{f_2\}\} \cdot \mathcal{C}^2(f_1, f_2). \end{aligned} \tag{3}$$

Proof: In an acyclic graph, any subset of arcs contained in the union of two paths will either be the subset of a single path or will have a unique highest pair of incomparable elements. The first term in the expression includes all relevant arc subsets contained in a single path. The second term contains all other subsets of two paths. The union is taken over all possible highest incomparable arc pairs, $\{f_1, f_2\}$. The term $C^1(e_1, e_2, f_1, f_2)$, within the union, accounts for arcs that can be above the highest incomparable pair, which by necessity would be contained in a single path. ▮

Corollary 3.4

$$\|C^2(e_1, e_2)\| = \|C^1(e_1, e_2, e_1^s, e_2^s)\| + \sum_{\{f_1, f_2\} \in LL^0(e_1, e_2)} \|C^1(e_1, e_2, f_1, f_2)\| \cdot \frac{1 - p_{f_1}}{p_{f_1}} \cdot \frac{1 - p_{f_2}}{p_{f_2}} \cdot \|C^2(f_1, f_2)\| \quad (4)$$

3.2 An Efficient Implementation of the Recursive Equations

In this section Tables 1 and 2 provide algorithms first to compute a look-up table containing $\|C^1\|$ for the necessary arguments and then to compute $\|C^2(e_1^t, e_2^t)\|$. We analyze the computational complexity of these algorithms.

Proposition 3.5 *The Single-Path Algorithm computes a table of values of* $\|C^1\|$ *in* $O(|E|^5)$ *time.*

Proof: The correctness of the algorithm follows from Corollary 3.2. We note that the ordering insures that all quantities on the right side of Equation 2 are computed by the time they are needed.

The **For** loop indexed by f_1, f_2 is of course a nesting of loops where in fact arcs f_1 can be generated by a breadth-first-search as implemented in a precedence numbering algorithm, and given f_1, the arcs f_2 can be generated by a similar breadth-first-search in which arcs above f_1 take priority in

Table 1: **Single-Path Algorithm**

Given: Acyclic graph $G = (V, E, s, t)$ and arc probability vector $\boldsymbol{p}$.

Output: Values of $\|\mathcal{C}^1(e_1, e_2, f_1, f_2)\|$ for arcs satisfying $e_1 \neq e_2$, $f_1 \in L(e_1)$, $f_2 \in L(e_2)$, $f_1 \neq f_2$.

Procedure:

```
Add arcs $e_1^s, e_2^s$ directed into $s$ and arcs $e_1^t, e_2^t$ directed out of $t$. Set
    $\|\mathcal{C}^1(e_1^s, e_2^s, e_1^s, e_2^s)\| = 1$.
For pairs of arcs $f_1 \neq f_2$ do
    for pairs of arcs $e_1 \neq e_2$ or $e_1 = e_2$, $e_1 \in U(f_1), e_2 \in U(f_2)$, in
        nondecreasing order do
        compute $\|\mathcal{C}^1(e_1, e_2, f_1, f_2)\|$ using Equation 2.
Return the table of $\|\mathcal{C}^1(e_1, e_2, f_1, f_2)\|$.
```

Table 2: **Two-Path Algorithm**

Given: Acyclic graph $G = (V, E, s, t)$ and arc probability vector $\boldsymbol{p}$.

Output: $\|\mathcal{C}(G, s, t)\|$.

Procedure:

```
Add arcs $e_1^s, e_2^s$ directed into $s$ and arcs $e_1^t, e_2^t$ directed out of $t$. Set
    $\|\mathcal{C}^2(e_1^s, e_2^s)\| = 1$.
For pairs of arcs $e_1 \neq e_2$ in nondecreasing order do
    compute $\|\mathcal{C}^2(e_1, e_2)\|$ using Equation 4.
Return $\|\mathcal{C}(G, s, t)\| = \|\mathcal{C}^2(e_1^t, e_2^t)\|$.
```

the search over other arcs. Thus the doubly-indexed loop requires $O(|E|^2)$ iterations.

The **for** loop indexed by e_1, e_2 can be handled similarly, requiring $O(|E|^2)$ iterations for each arc pair f_1, f_2. Thus the **compute** step is performed $O(|E|^4)$ times. The summation index arcs for a single application of Equation 2 can be generated in $O(|E|)$ time with breadth-first-search, so that the overall computation time requirements are $O(|E|^5)$. ∎

Proposition 3.6 *Given the output of the Single-Path Algorithm, the Two-Path Algorithm computes $\|\mathcal{C}(G, s, t)\|$ in $O(|E|^4)$ time.*

Proof: As in Proposition 3.5, generating the indexes for the **For** loop in the Two-Path Algorithm requires $O(|E|^2)$ time. Generating the indexes for the summation in Equation 4 can be performed similarly to the way they were generated in the inner **for** loop of the Single-Path Algorithm. Since we can perform the look-up of $||\mathcal{C}^1||$ as we generate our indexes, the Two-Path Algorithm requires $O(|E|^4)$ time. ▮

We combine the two propositions in the following theorem.

Theorem 3.7 *Applying the Single-Path and Two-Path Algorithms to compute* $||\mathcal{C}(G,s,t)||$ *requires* $O(|E|^5)$ *time.*

Corollary 3.8 *If* G *is a source-sink planar acyclic graph, then* $||\mathcal{C}(G,s,t)||$ *can be computed in* $O(|V|^5)$ *time.*

Proof: This is a consequence of $|E| = O(|V|)$ in a planar graph. ▮

A more careful implementation of the same ideas yields a stronger complexity result than that of Theorem 3.7 for the nonplanar case. We state this result and sketch its proof afterwards. We do not present a formal proof since it would require introduction of significant additional notation.

Theorem 3.9 *If* G *is acyclic, then* $||\mathcal{C}(G,s,t)||$ *can be computed in* $O(|V|^4|E|)$ *time.*

We base our arguments on the observation that if $t(e_1) = t(e_1')$, $t(e_2) = t(e_2')$, $h(f_1) = h(f_1')$, $h(f_2) = h(f_2')$, then $\mathcal{C}^2(e_1, e_2) = \mathcal{C}^2(e_1', e_2')$ and $\mathcal{C}^1(e_1, e_2, f_1, f_2,) = \mathcal{C}^1(e_1', e_2', f_1', f_2')$.

Then the Single-Path Algorithm can be written to loop over vertex pairs instead of arc pairs. The pairs of vertices v_1, v_2 that we need are such that either $v_1 = v_2$ or v_1 is incomparable to v_2. Given a vertex v_1, we can compute in $O(|E|)$ time the set of vertices w_1 which are above it and the set of vertices v_2 which are incomparable to it. For each pair v_2, w_1, we can compute in $O(|E|)$ time the set of vertices w_2 which are above v_2 and incomparable to w_1. Thus we can generate acceptable indices for the loops in $O(|V|^2|E|)$ time and create a data structure in which the acceptable vertex quadruples can be read off in correct order in $O(|V|^4)$ time. Using this data structure a version of the Single-Path Algorithm can be written which requires $O(|V|^4|E|)$ time and returns a table based on vertices.

A simpler operation provides acceptable vertex pairs for the **For** loop of a new version of the Two-Path Algorithm, giving the new version a complexity of $O(|V|^2|E|^2)$. This is dominated by the Single-Path Algorithm complexity.

4 Two-Disjoint-Path Problem

In this section we give a polynomial-time algorithm for computing $||\mathcal{D}(G, s, t)||$. This algorithm is more restrictive than the one given in Section 3, since we must assume that G is a source-sink planar k-flow graph. It can then be used to compute $RP(G, \boldsymbol{a}, \boldsymbol{p}, d)$ for instances where G is 2-critical and source-sink planar. The planarity condition is necessitated by the difficulty here of maintaining the existence of two disjoint paths even when a subset of failed arcs can be covered by a single path. This necessity requires stronger conditions relating the recursive entities.

We begin the definitions and write the recursion making no assumptions about the graph other than that it is acyclic. The recursive computation will be shown to be polynomial with the additional assumption that G is an

(s,t)-planar k-flow graph.

4.1 Recursion

To simplify the presentation of the recursion, we first add four artificial arcs to G: e_1^s, e_2^s directed into s, and e_1^t, e_2^t directed out of t, each with operating probability 1. Now for $e_1 \neq e_2$ define

$$\begin{aligned}\mathcal{D}^2(e_1,e_2) \equiv \{S \subseteq E \setminus \{e_1,e_2\} \mid \\ S \text{ is covered by disjoint paths joining } e_1^s, e_2^s \text{ to } e_1, e_2\}.\end{aligned}$$

It follows that

$$\|\mathcal{D}^2(e_1^s,e_2^s)\| = 1.$$

and

$$\|\mathcal{D}^2(e_1^t,e_2^t)\| = \|\mathcal{D}(G,s,t)\|.$$

We extend the definition of $\mathcal{D}^2$ to allow arguments of the form (e,R), where the arcs of $R+e$ are mutually incomparable:

$$\mathcal{D}^2(e,R) \equiv \bigcup_{f\in R} \mathcal{D}^2(e,f)$$

We will actually use these definitions when arguments of $\mathcal{D}^2$ are either a pair of incomparable arcs, or they are a single arc and a subset of the boundary of the arc's lower set.

We next extend the concept of a lower set in order to consider whether or not a pair of arcs can reach another pair via a pair of disjoint paths. Given

$e_1 \neq e_2$, define

$$LL(e_1, e_2) \equiv \{\{f_1, f_2\} \subseteq E \setminus \{e_1, e_2\} | f_1 \neq f_2,$$
$$\text{there exist disjoint paths joining } f_1, f_2 \text{ to } e_1, e_2 \ \}.$$

We extend $LL(e_1, e_2)$ to have as its second argument a set R of arcs, by setting

$$LL(e, R) \equiv \bigcup_{g \in R} LL(e, g).$$

In the previous section, the recursion was able to move from a pair of incomparable arcs to another pair of incomparable arcs. If there were failed arcs below one pair and above the other, we could just ensure that such arcs lay on a single path. The two-disjoint-path case introduces an added complication. The failed arcs between two incomparable pairs must be such that we can maintain two disjoint paths between the incomparable pairs. We must change our perspective, and move from a pair of incomparable arcs to an arc for which we can maintain a disjoint pair and vice versa. We will do this by including as recursive arguments not only pairs of incomparable arcs, but also singleton arcs which are below at least one of a pair of incomparable arcs. In the case of the singleton arc, all its recursive arguments should be restricted to its lower set, for otherwise it can be included in a pair of incomparable arcs. In the case of such a singleton arc, we must ensure the existence of the appropriate disjoint path pair. Given a singleton failed arc f, we can properly identify possible failed arcs in the lower set of f if we properly identify arcs in $\partial L(f)$ which can "carry" a path disjoint from a path containing f.

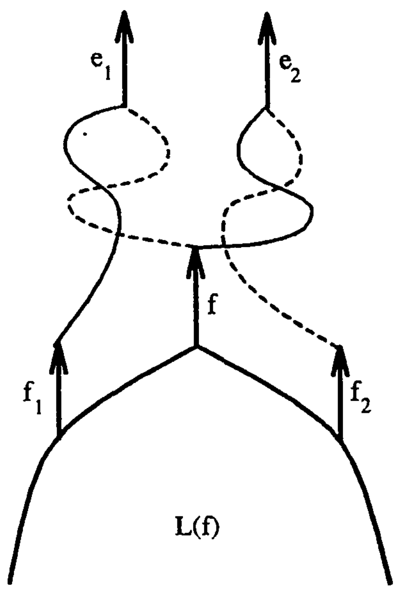

Figure 3: **Reaching Set for Recursion**

The following definition is designed to appropriately handle such a singleton arc f. For $e_1 \neq e_2$, $f \in L(e_1) \cup L(e_2)$, define the *reaching set* associated with f, e_1, e_2 to be

$$W(f, e_1, e_2) \equiv \{g \in \partial L(f) |$$
$$\text{there exist disjoint paths joining } f, g \text{ to } e_1, e_2 \}.$$

In the schematic in Figure 3, both f_1 and f_2 belong to $W(f, e_1, e_2)$.

We extend the definition of W to have as a third argument a set R of arcs:

$$W(f, e, R) \equiv \bigcup_{g \in R} W(f, e, g).$$

In applying these extensions, the set R is always either a single arc incomparable to e or it is a set of arcs belonging to $\partial L(e)$. A key property of pairs

e, R that we consider is that $R + e$ is a set of mutually incomparable arcs. We will use this characterization in the proofs that follow.

The next proposition provides the appropriate recursion for $\mathcal{D}^2$. Define an extension to $L(e)$ by

$$L(e, R) \equiv L(e) \cup \left[\bigcup_{f \in R} L(f) \right].$$

If $R \subset \partial L(e)$ $L(e, R) = L(e)$.

Proposition 4.1 *Let $e \in E$, $R \subseteq E - e$ satisfy $R + e$ is a set of mutually incomparable arcs of G. Then the following equation provides an enumerative expression for $\mathcal{D}^2(e, R)$.*

$$\begin{aligned} \mathcal{D}^2(e, R) = \{\emptyset\} \cup & \\ & \left[\bigcup_{f \in L(e,R)} \{\{f\}\} \cdot \mathcal{D}^2(f, W(f, e, R)) \right] \cup \\ & \left[\bigcup_{\{f_1, f_2\} \in LL(e,R)} \{\{f_1\}\} \cdot \{\{f_2\}\} \cdot \mathcal{D}^2(f_1, f_2) \right]. \end{aligned} \tag{5}$$

Proof: Arc subsets contained in $\mathcal{D}^2(e, R)$ are either (i) the null set, (ii) have a single highest element or (iii) have a unique highest incomparable pair of elements. Cases (i), (ii) and (iii) are respectively included in the first, second and third terms in the expression. ∎

By applying the properties of Section 2.3, we obtain

Corollary 4.2 *Let e and R be as in Proposition 4.1. Then*

$$
\begin{aligned}
\|\mathcal{D}^2(e,R)\| = {} & 1 + \\
& \left[\sum_{f\in L(e,R)} \frac{1-p_f}{p_f}\cdot \|\mathcal{D}^2(f,W(f,e,R))\|\right] + \\
& \left[\sum_{\{f_1,f_2\}\in LL(e,R)} \frac{1-p_{f_1}}{p_{f_1}}\cdot\frac{1-p_{f_2}}{p_{f_2}}\cdot \|\mathcal{D}^2(f_1,f_2)\|\right].
\end{aligned} \tag{6}
$$

4.2 An Efficient Implementation of the Recursive Equations

Corollary 4.2 still does not provide an efficient method for computing $\|\mathcal{D}^2(e_1^t,e_2^t)\|$, since the number of distinct sets R for which we compute Equation 6 may be exponential in the number of arcs of the network. To make this recursion efficient, we need to provide more structure for the sets R and $W(f,e,R)$. In this section, therefore, we make the additional assumption that G is a source-sink planar k-flow graph. Any set R which is needed in Equation 6 is either a singleton arc or is generated as a set $W(f,e,R') \subset \partial L(f)$. If G is an (s,t)-planar k-flow graph, we will establish that each set $W(f,e,R)$ is defined by f and the set's leftmost and rightmost arcs. This reduces the number of possible sets R to a number polynomial in the number of arcs of the network.

As an initial step in developing both the proofs and the computational procedure, we provide a general construction method that produces rightmost and leftmost paths, which is given in Table 3.

Table 3: **Leftmost-Rightmost Path Construction Procedure**

Input: arcs f and e of G with $f \preceq e$.

Output: leftmost (rightmost) path P_l (P_r) whose first arc is f and whose last arc is e.

Procedure:

1. Mark each vertex of G for which there exists a path joining that vertex to $t(e)$.
2. Set P_l (P_r) to be the single arc f.
3. Repeat the following until P_l (P_r) has e as its last arc:
   (a) Let x be the last vertex of P_l (P_r).
   (b) If $x = t(e)$ then add e to P_l (P_r). Otherwise, scan through the arcs directed out of x from right to left. Add to P_l (P_r) the leftmost (rightmost) arc whose head is marked.
4. Return P_l (P_r).

Lemma 4.3 *(1) The Leftmost-Rightmost Path Construction Procedure takes $O(|V|)$ time, and the paths P_l and P_r constructed by the algorithm are the (unique) leftmost and rightmost paths joining f to e, respectively.*
(2) Let e_1 and e_2, f_1 and f_2, be pairs of arcs with e_1 to the left of e_2 and f_1 to the left of f_2. If $\{f_1, f_2\} \in LL^0(e_1, e_2)$, then the leftmost path P_l from f_1 to e_1 is to the left of the rightmost path P_r from f_2 to e_2.
(3) Let e_1 and e_2, f_1 and f_2, be pairs of arcs with e_1 to the left of e_2 and f_1 to the left of f_2. If $\{f_1, f_2\} \in LL(e_1, e_2)$, then the paths P_l and P_r constructed in (2) are disjoint.

Proof: (1) The procedure requires an elementary search from $t(e)$, and another from $h(f)$, each of which take $O(|E|) = O(|V|)$ time (since G is planar). To prove correctness, let P be a path from f to e having an arc strictly to the left of P_l (the other case is symmetric). Let g be the first such arc of P. $t(g)$ is a vertex of P_l. But, since $h(g)$ can reach $t(e)$, g would have been chosen instead of the next arc of P_l after $t(g)$, a contradiction. Thus P_l is the leftmost path from f to e.
(2) Consider the subgraph of G induced by $(L(e_1) \cup L(e_2)) \cap (U(f_1) \cup U(f_2))$. This graph is planar and has $t(e_1), t(e_2), h(f_1), h(f_2)$ on its outer face. We can maintain planarity as we do the following: (i) replace e_1, e_2, directing them both into a new vertex v, (ii) replace f_1, f_2, directing them out of a new vertex w, (iii) add arc f with head at w, (iv) add arc e with tail at v. Any paths joining f_1, f_2 to e_1, e_2 retain their identities.

Apply the Leftmost-Rightmost Path Construction Procedure to produce leftmost and rightmost paths P_l and P_r from f to e. Then in particular, P_l is

to the left of P_r and any pair of paths joining f_1, f_2 to e_1, e_2 must lie between P_l and P_r. It follows that P_l must join f_1 to e_1 and P_r must join f_2 to e_2.
(3) Let Γ_1 and Γ_2 be disjoint paths from f_1, f_2 to e_1, e_2. P_l is to the left of both Γ_1 and Γ_2. P_r is to the right of both of them. Since Γ_1 and Γ_2 have no arcs in common, P_l and P_r can have no arcs in common. ▮

We note that the Leftmost-Rightmost Path Construction Procedure can also be modified to find the following paths:

- A rightmost(leftmost) path which lies to the left(right) or strictly to the left(right) of a given arc e. This can be accommodated by simply removing the arcs to the right(left) or strictly to the right(left) of e, and then running the Leftmost-Rightmost Path Construction Procedure on the remaining subgraph.

- A rightmost(leftmost) path starting in a set of mutually incomparable arcs and/or ending in a set of mutually incomparable arcs. This can be done by creating an adjusted subgraph of G similar to the one constructed in the proof of part (2) of Lemma 4.3. The procedure continues to run in linear time.

We next give a constructive description of $W(f, e, R)$. For two arcs e_l and e_r in $\partial L(e)$, define the *interval* between e_l and e_r with respect to $L(e)$ to be

$$[e_l, e_r]_e \equiv \{ f \in \partial L(e) | \ f \text{ is to the right of } e_l \text{ and to the left of } e_r \}$$

Note that this set is empty if e_l is strictly to the right of e_r. When the graph is planar, the schematics we have been using can be regarded as truly

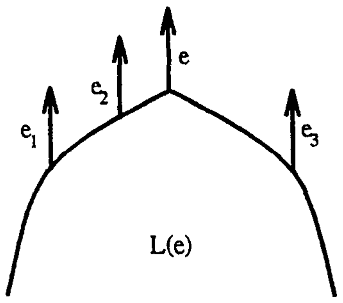

Figure 4: **Interval of a Planar Acyclic Graph**

indicating left-rightness. The schematic in Figure 4 indicates that $[e_1, e_3]_e$ contains e, e_1, e_2, e_3; $[e_1, e_2]_e$ contains only e_1, e_2 from among the arcs shown.

In the application of Equation 6, R is either a subset of $\partial L(e)$ or R consists of a single arc e_0. In the latter case, $R = [e_0, e_0]_{e_0}$. We show that if R is of the form $I - e$ for some interval I, then $W(f, e, R) = I' - f$ for some interval $I' \subseteq \partial L(f)$. To do this we first identify the end arcs f_l and f_r for I' by finding the leftmost and rightmost arcs of $\partial L(f)$ for which $\{f_l, f\}$ and $\{f_r, f\}$ are in $LL(e, R)$. This is done using the procedure given in Table 4.

Figure 5 shows this construction for f_l. Left-bending (right-bending) curves indicate leftmost (rightmost) paths, and e_l' is the unique arc of $\partial L(e)$ that is in P_l^i or Q_l^i not going through e. The Interval Construction Procedure takes $O(|E|)$ time, since it involves eight applications of the Leftmost-Rightmost Path Construction Procedure.

Lemma 4.4 *Let the triple f, e, R defined in the Interval Construction Procedure satisfy $W(f, e, R) \neq \emptyset$. Then both f_l and f_r exist, and further, every $g \in W(f, e, R)$ must lie in $[f_l, f_r]_f$.*

Proof: Let $g \in W(f, e, R)$. Let P and Q be disjoint paths joining g, f to h, e

Table 4: **Interval Construction Procedure**

Given: arc $e \in E$, $R \subseteq E - e$ such that $R + e$ is a set of mutually incomparable arcs of G, and arc $f \preceq e$.

Output: leftmost arc f_l and rightmost arc f_r of $W(f, e, R)$.

Construction of f_l:

Construct the following paths, using the Leftmost-Rightmost Path Construction Procedure.

Q_l^1 is the rightmost path from f to e;

P_l^1 is the leftmost path from $\partial L(f)$ to R that is strictly to the left of Q_l^1;

Q_l^2 is the leftmost path from f to e;

P_l^2 is the leftmost path from $\partial L(f)$ to R that is to the right of e_l and strictly to the right of Q_l^2;

Q_l^3 is the rightmost path from f to R;

P_l^3 is the leftmost path from $\partial L(f)$ to e that is strictly to the left of Q_l^3;

Q_l^4 is the leftmost path from f to R;

P_l^4 is the leftmost path from $\partial L(f)$ to e that is strictly to the right of Q_l^4.

For $i = 1, 2, 3, 4$, if P_l^i and Q_l^i exist then let f_l^i be the arc of $\partial L(f)$ in P_l^i. Set f_l to be the leftmost arc of $\{f_l^1, f_l^2, f_l^3, f_l^4\}$.

Construction of f_r: Same as for f_l, except that the subscript r replaces l everywhere and the roles of "right" and "left" are interchanged.

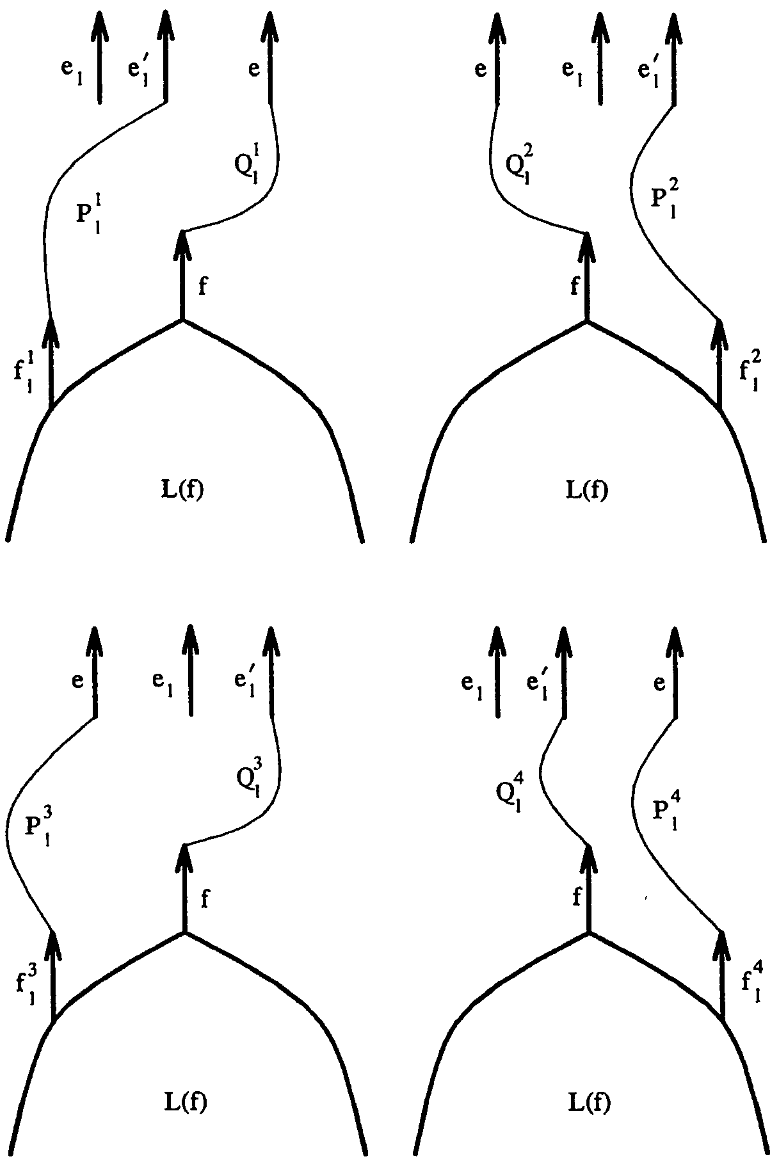

Figure 5: **Construction of** f_l

for some $h \in R$, labeled so that $f \in Q$. We show that f_l exists and that g lies to the right of f_l (the argument for f_r being symmetric). By Lemma 4.3 we can assume without loss of generality that P and Q are noncrossing.

Case 1: Suppose that e is in Q, and that P is (strictly) to the left of Q. Then by construction Q_l^1 exists and lies to the right of Q, and P_l^1 exists and lies to the left of P. Thus f_l^1 exists, and therefore f_l exists and is to the left of g.

The other three cases correspond to the existence of paths P_l^i and Q_l^i for $i = 2, 3, 4$, and the proof proceeds as above. The lemma follows. ∎

Proposition 4.5 *If the set R of Lemma 4.4 satisfies $R = I - e$ for some interval I; then $W(f, e, R) = [f_l, f_r]_f - f$.*

Proof: Since $f \notin W(f, e, R)$, Lemma 4.4 establishes that $W(f, e, R) \subseteq [f_l, f_r]_f - f$. To prove $[f_l, f_r]_f - f \subseteq W(f, e, R)$, let P_l^i, Q_l^i, P_r^j, Q_r^j, where i, j =1, 2, 3, or 4, be the paths defining f_l, f_r as produced by the Interval Construction Procedure. Now choose arc g in $[f_l, f_r]_f - f$. We must show that there exists a pair of disjoint paths joining f, g to e, R.

We can assume by symmetry that g lies to the left of f. Then $f_l = f_l^1$ or $f_l = f_l^3$. Figures 6 and 7 show the eight possible configurations under which f_l and f_r are defined using the Interval Construction Procedure. The arcs labelled e_l' and e_r' are members of R. The cases in which $R = \{e_0\}$ can be considered to be represented by configurations (d), (e), (g), and (h), with $e_0 = e_l' = e_r'$.

We first show that the configurations (a) and (b) given in Figure 6 lead

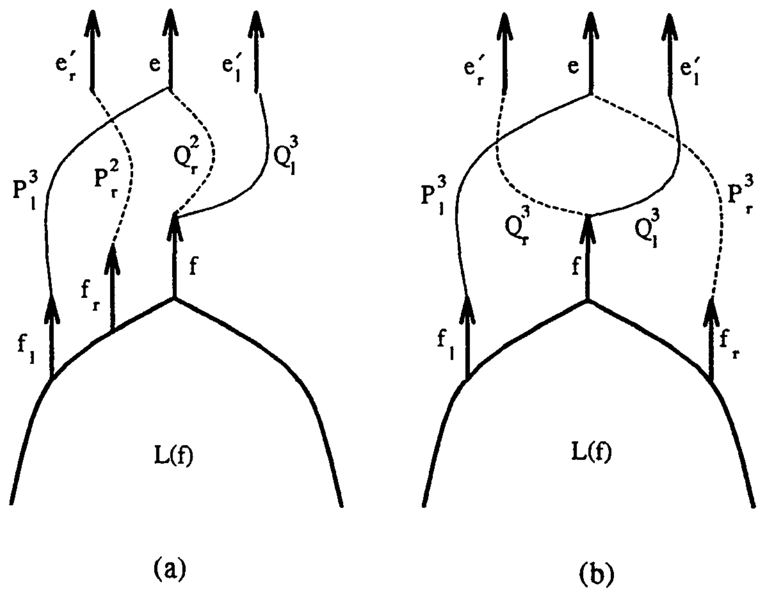

Figure 6: **Paths Produced by Int. Const. Proc.: Cases (a) and (b)**

to the conclusion that $f_l^3 = f_l^1$. In configuration (a) let P be the leftmost path among the union of P_l^3 and P_r^2, and let $Q = Q_r^2$. In configuration (b), let P be the leftmost path among the union of P_l^3 and Q_r^3, and let Q be the leftmost path among the union of P_r^3 and Q_l^3. In both cases it follows that P is strictly to the left of Q. The paths P, Q are as described in Case 1 of the proof of Lemma 4.3. Therefore f_l^1 exists and is to the left of f_l^3. Hence $f_l^3 = f_l = f_l^1$ and we can assume that we have paths in one of the six remaining configurations (c)–(h) given in Figure 7.

The construction method and Lemma 4.3 assure that, in the six configurations we need to consider, the four paths P_l^i, Q_l^i, P_r^j, and Q_r^j do not cross. Consider the leftmost path P_0 from g to t. Referring to Figure 7 we see that P_0 must lie to the left of all of the four paths except P_l^i, and so in particular,

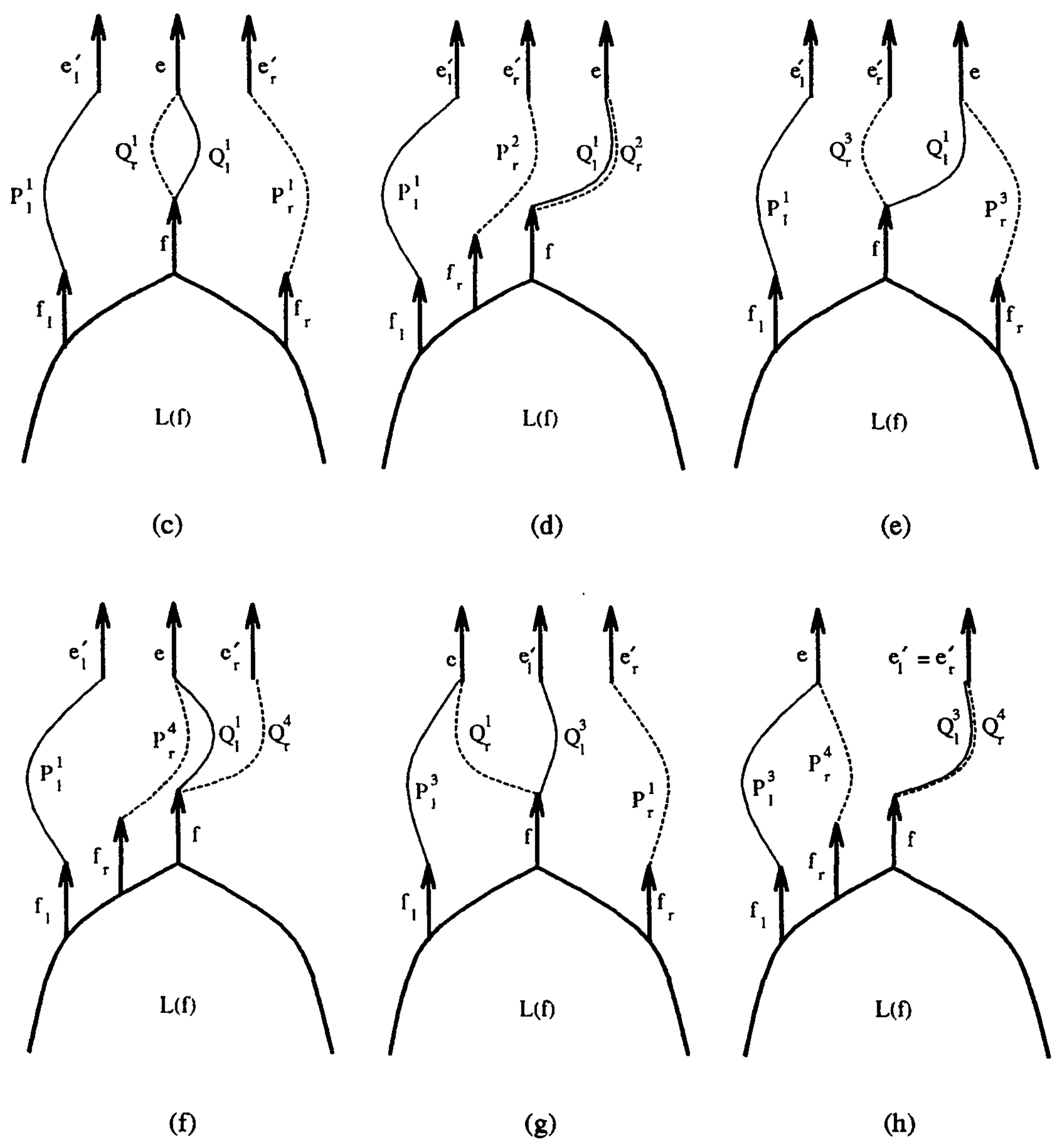

Figure 7: **Paths Produced by Int. Const. Proc.: Cases (c)–(h)**

P_0 has an arc $h \in R + e$. Let P be the portion of P_0 joining g to h. If P is disjoint from Q_l^i then these are disjoint paths joining f, g to e, h, and we are done. We note that for configurations (d) and (h), since P is to the left of P_r^j, which is strictly to the left of $Q_r^j = Q_l^i$, this must be the case.

For the remaining configurations, it appears possible that the paths P, Q_l^i are not disjoint. If they are not disjoint then there is an arc x which is the first arc P and Q_l^i have in common.

For configuration (f), note that x must belong to P_r^4 as well. In this configuration, P is to the left of P_r^4, which in turn is strictly to the left of Q_r^4. Define P' to be formed from the portion of P joining g to x, and the portion of P_r^4 joining x to e. P' and Q_r^4 are disjoint paths joining f, g to e, R.

We now consider configurations (c), (e), and (g). We make the following observations:

- x is an arc of Q_r^j, since P is to the left of Q_r^j, which in turn is to the left of Q_l^i;
- x is not the first arc of either P or Q_l^i, since $g \neq f$;
- there must be at least one additional arc y having the same tail as x, since the arcs of P and Q_l^i immediately before x are different and G is a k-flow graph;
- y must be to the right of x, since otherwise P would have chosen y as its next arc instead of x;
- we can choose y to be to the left of P_r^j, since x is on Q_r^j and therefore is strictly to the left of P_r^j.

For configuration (e), y must be able to reach e, since it lies between Q_l^1 and P_r^3. However, this contradicts the definition of Q_l^1, since it should take the rightmost arc out of $t(x)$ in order to reach e. Similarly, for configuration (g), y must be able to reach R, since it lies between Q_l^3 and P_r^1. This in turn contradicts the definition of Q_l^3. We thus conclude for these configurations that it is impossible for x to exist and P and Q_l^i are disjoint.

Finally, we must exhibit the existence of disjoint paths for configuration (c). Let P' be the *rightmost* path from y to $\partial L(e)$ that lies to the left of P_r^1. Since y lies strictly to the right of Q_l^1, and Q_l^1 is a rightmost path from x to e, then P' cannot subsequently intersect Q_l^1. Let h' be the arc of P' that is in $\partial L(e)$, and note that by construction $h' \in [e_l', e_r']_e$. Let P'' be the path consisting of that portion of P occurring before x followed by P'. P'' and Q_l^1 are disjoint paths joining f, g to e, h', and the proposition follows. ▮

Proposition 4.5 leads to the algorithm in Table 5 for computing $||\mathcal{D}(G, s, t)||$ efficiently.

Theorem 4.6 *The Two-Disjoint-Path Algorithm computes* $||\mathcal{D}(G, s, t)||$ *in* $O(|V|^5)$ *time.*

Proof: The correctness of the algorithm follows from Corollary 4.2 and Proposition 4.5, noting that the order of computation of the $||\mathcal{D}^2(e, R)||$'s insures that the right-hand-side values of Equation 6 are available at the time of computation. For the complexity, note that the **For** loop is indexed by at most 3 arcs, so that there are $|E|^3$ loop iterations. The first sum in Equation 6 requires enumerating $L(e, R)$ and then computing $W(f, e, R)$. Each of these takes $O(|E|)$ time. The second sum requires the computation

Table 5: **Two-Disjoint-Path Algorithm**

```
Given: Source-sink planar 2-critical graph $G = (V, E, s, t)$ and arc probability vector $\boldsymbol{p}$.

Output: $\|\mathcal{D}(G, s, t)\|$.

Procedure:

    Add arcs $e_1^s, e_2^s$ directed into $s$ and arcs $e_1^t, e_2^t$ directed out of $t$. Set
        $\|\mathcal{D}^2(e_1^s, e_2^s)\| = 1$.

    For arcs $e$ in nondecreasing order, and sets $R = \{e_0\}$ for some $e_0 \neq e$
        or $R = [e_l, e_r]_e - e$ for $e_l, e_r \in \partial L(e)$ do

        compute $\|\mathcal{D}^2(e, R)\|$ using Equation 6.

    Return $\|\mathcal{D}(G, s, t)\| = \|\mathcal{D}^2(e_1^t, e_2^t)\|$.
```

of $LL(e, R)$. This can be computed using Lemma 4.3 as follows: For each $f_1 \in L(e)$, find the rightmost (e_1^s, f_1)-path P_1 and the rightmost (f_1, e)-path P_2. Now delete all arcs to the right of $P_1 \cup P_2$ (this includes the arcs on the paths themselves). Now find all arcs f_2 that can reach an arc of R. The set of arcs f_2 found constitute all arcs to the left of f_1 such that the pair $\{f_1, f_2\}$ is an element of $LL(e, R)$. By repeating this for the leftmost paths through f_1 we obtain all arcs f_2 to the right of f_1 having this property. By deleting f_1 after performing this, we guarantee no repeated pairs, and the whole process takes $O(|E|^2)$ time as well.

Finally, since the terms of the sums are already known the computation of Equation 6 takes $O(|E|^2)$ time, for a total time of $O(|E|^5) = O(|V|^5)$ (since G is planar). The theorem follows. ▮

5 Summary and Further Challenges

In this paper, we have provided recursions for enumerating two-path subsets and two-disjoint-path subsets of acyclic graphs. These recursions lead to efficient methods of analyzing the performability of 2-critical network systems. We review the computational results and their application below.

5.1 Computational Complexity

The algorithms provided in this paper and in [1] address the general problems of r-path subset enumeration. The algorithms are efficient methods for either counting the number of these subsets or, more generally, computing a weighted norm such as the failure-odds norm for such sets.

To summarize what is known to date about the path problems discussed in this paper:

- The r-path problem is polynomial when $r = 1$ or 2 and G is acyclic.
- The 1-path problem becomes #P-complete (See [8]) when G is not required to be acyclic.
- When $r \geq 3$ is fixed and G is acyclic, the complexity of the r-path problem is open.
- The r-disjoint-path problem is polynomial when $r = 1$, and when $r = 2$ and G is a source-sink planar k-flow graph.
- The 2-disjoint-path problem becomes #P-complete in a nonplanar graph G even when it is a k-flow graph.
- When $r \geq 3$ is fixed and G is a source-sink planar k-flow graph, the r-disjoint-path problem is open.

The relevant results are found here and in [1], [6].

5.2 Application to Performability Analysis

This paper was motivated by two network performability analysis problems: the evaluation of the reliability of a two-terminal flow network with a capacity criterion, and the evaluation of the reliability of a project network with a project duration criterion. Even with some restrictions we have shown that these problems are NP-hard [1]; therefore we do not expect to find efficient algorithms to handle these evaluations.

However, we observe that such networks are designed to withstand a certain minimal level of failure. For instance, a typical reliability design criterion is that the proposed system/network should be able to withstand the failure of any single component. When higher reliability is desired this criterion is generalized to require that the system be able to withstand the failure of any r components. With this design philosophy in mind, one would expect to find many real systems that are 1- (or more generally r-) critical or "nearly" 1- or r-critical. These arguments led to our study of reliability analysis problems defined on r-critical systems in this paper and our previous paper.

In this paper, we have shown that we can analyze the 2-critical problems of threshold flow and threshold scheduling when the underlying network is source-sink planar. We are able to do this by formulating the problems in terms of the failure-odds norm, and applying the recursive enumerative expressions we have developed.

The natural broad extension of this work would be to study reliability analysis problems on other classes of r-critical systems. We anticipate that the additional structure imposed by this characterization may often lead to polynomial algorithms for seemingly difficult analysis problems. In the network setting, examples that merit attention include computing reliability measures for networks with multi-commodity capacity requirements and other performance criteria. In addition, a similar point of view may lead to addressing performability analysis problems for non-network systems.

References

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

[2] Eugene L. Lawler. *Combinatorial Optimization: Networks and Matroids.* Holt, Rinehart and Winston, New York, 1976.

[3] C. L. Monma and D. F. Shallcross. Methods for designing communications networks with certain two-connected survivability constraints. *Operations Research*, 37:531–541, 1989.

[4] J. S. Provan and V. G. Kulkarni. Exact cuts in networks. *Networks*, 19:281–289, 1989.

[5] J. Scott Provan. Boolean decomposition schemes and the complexity of reliability computations. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 5:213–228, 1991.

[6] J. Scott Provan. The intractability of the 2-critical flow problem in k-flow graphs. Technical Report UNC/OR TR95-3, Department of Operations Research, University of North Carolina, Chapel Hill, 1995.

[7] K. Steiglitz, P. Weiner, and D. J. Kleitman. The design of minimum-cost survivable networks. *IEEE Transactions on Circuit Theory*, CT-16:455–460, 1969.

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