Minimum Cost Paths Subject to Minimum Vulnerability for Reliable Communications

Bing Yang[†], Mei Yang[‡], Jianping Wang[⋆], and S.Q. Zheng[∗]

[†] Cisco Systems, 2200 East President George Bush Highway, Richardson, TX 75082, USA

[‡] Department of Electrical and Computer Engineering, University of Nevada, Las Vegas, NV 89154, USA

[⋆] Department of Computer Science, Georgia Southern University, Statesboro, GA 30460, USA

[∗] Department of Computer Science, University of Texas at Dallas, Richardson, TX 75083-0688, USA

E-mail: [†]binyang@cisco.com, [†]meiyang@egr.unlv.edu, [⋆]jpwang@georgiasouthern.edu, [∗]sizheng@utdallas.edu

***Abstract*— In real networks, disjoint paths are needed for providing protection against single link/node failure. When disjoint paths cannot be found, an alternative solution is to find paths with the minimum shared links/nodes. In the literature, there is little work addressing this problem. In this paper, defining vulnerability as the number of times a link/node is shared among different paths, we consider the problems of finding k paths with minimum edge/node vulnerability. We study three problems and propose polynomial algorithms to solve these problems. The work presented in this paper can be directly applied to a number of network applications such as reliable unicast/multicast communications, reliable client-server communications, protection for the dual-homing architecture, etc.**

***Index Terms*— Networks, graph, algorithm, protection, reliability, survivability, disjoint paths, shortest path, complexity, network flow, minimum cost network flow.**

I. Introduction

A reliable telecommunication network is designed in such a way that multiple paths exist between every pair of nodes. To protect the network against single link/node failure, multiple disjoint paths are needed between a pair of source and destination nodes. The paths may be node-disjoint or edge-disjoint, and the network may be directed or undirected. Thus, the problem of finding disjoint paths have four versions. For a given pair of nodes, finding k ($k > 1$) disjoint paths, though desirable, may not always be possible in practical network applications for at least two reasons. First, if the network is too sparse, such paths may not physically exist. Second, if some links are overly saturated, additional traffic on these links may be prohibited so that two disjoint paths without using these prohibited links do not exist.

When k disjoint paths do not exist, alternatively, k paths from the source to the destination with minimum shared links/nodes should be found. We adopt the concept of *vulnerability* introduced in [12] and redefine it as the number of times a link/node is shared among k paths. k paths with minimum vulnerability can provide *partial protection* [18].

Finding k paths with minimum vulnerability is useful for many applications, including reliable unicast/multicast communications and client-server communications. Consider a multicast session consisting of a source S and four destination nodes $D_1, \cdots, D_4$. The classical connection for connecting S to its destinations is a multicast tree, as one shown in Fig. 1 with edge vulnerability (which will be defined shortly) 4. Suppose a subnetwork for this multicast session has the structure of edge vulnerability 2 as shown in Fig. 2. In case of failures of links $A - C$ and $B - E$, the number of affected destinations of the structure of Fig. 2 is smaller than the number of affected destinations of the structure of Fig. 1.

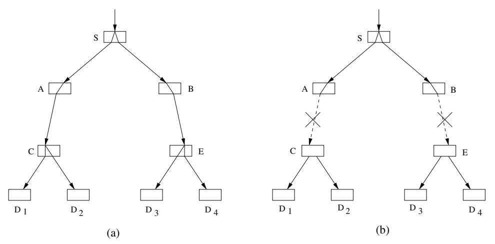

Fig. 1. Multiple paths for multicasting with tree structure. The paths from s to $\{D_1, D_2, D_3, D_4\}$ are $S-A-C-D_1$, $S-A-C-D_2$, $S-B-E-D_3$, $S-B-E-D_4$. (a) Paths when there is no link failure. (b) Paths when there are 2 link failures, leading to all destinations disconnected.

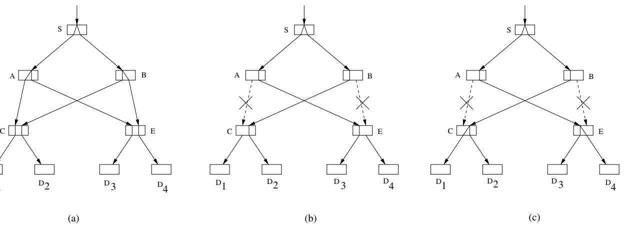

Fig. 2. Multiple paths for multicasting. The paths from s to $\{D_1, D_2, D_3, D_4\}$ are $S-A-C-D_1$, $S-B-C-D_2$, $S-A-E-D_3$, $S-B-E-D_4$. (a) Paths when there is no link failure. (b) Paths when there are 2 link failures, resulting 2 destinations disconnected. (c) If reconfiguration at switches/routers is allowed, all destinations remain connected.

The problem of finding k paths with minimum vulnerability can also be applied to network protection under the dual homing architecture. A network consist of core routers and edge routers. Host machines are connected to edge routers through access links. In a dual-homing architecture [3], [7], [8], [9], [12], a host can be connected to two edge routers so that traffic between the source and destination hosts is protected. The paths between pairs of source and destination hosts with minimum vulnerability are useful for assigning each host a pair of edge routers to best protect the host.

In this paper, we use the notion of edge vulnerability and node vulnerability to characterize the degree of edge sharing and node sharing among different paths. Larger edge/node vulnerability implies more edge/node sharing among a set of paths. A set of paths are edge-disjoint if the edge vulnerability of the paths is 0, and they are node-disjoint if their node vulnerability is 0 (in this case, their edge vulnerability is also 0). We study three problems of finding k paths with minimum edge or node vulnerability, namely, 1-to-1, 1-to-k, and single source all pairs of destinations. We have a dual objective function, namely minimizing total edge cost subject to minimum edge and node vulnerability. We show that all the problems considered can be solved in polynomial time and our algorithms always output such paths with minimum total cost.

The rest of the paper is organized as follows. Section II reviews existing work of disjoint paths. Section III presents the algorithms for solving the minimum edge/node vulnerability k paths problem between a pair of nodes. Section IV presents the algorithm for finding the minimum vulnerability paths from one source to k destinations. Section V presents the algorithm for finding the minimum vulnerability 2 paths from one source to all-pairs of destinations. Section VI concludes the paper.

II. Related Work

Assume a network is modeled as a weighted graph $G = (V, E)$, where V is the set of nodes, E is the set of links connecting nodes, and each edge is associated with a nonnegative cost. In the literature, various problems of finding optimized disjoint paths between two nodes $s, t \in G$ have been investigated.

Ford and Fulkson proposed a polynomial-time algorithm for finding two paths with minimum total cost (named the *MIN-SUM 2-Path Problem*) based on minimum cost network flow model [2]. Suurballe and Tarjan provided different treatment, and presented algorithms that are more efficient [10], [11]. Li *et al.* proved that all four versions of the problem of finding two disjoint paths such that the cost of the longer path is minimized (named the *Min-Max 2-Path Problem*) are strongly NP-complete [5]. They also considered a generalized Min-Sum problem (referred as the *G-Min-Sum k-Path Problem*) assuming that each edge is associated with k different costs. The objective of this problem is to find k disjoint paths such that the total cost of the paths is minimized, where the j^{th} edge-cost is associated with the j^{th} path. They showed that all four versions of the G-Min-Sum k-path problem are strongly NP-complete even for $k = 2$ [6].

In [14], the problem of finding two disjoint paths such that the cost of the shorter path is minimized (named the *Min-Min 2-path problem*) is discussed. All four versions of the Min-Min 2-path problem are shown to be strongly NP-complete and there does not exist approximation ratio unless $P = NP$. In [15], [16], a generalized weighted 2-path problem called the *α-MIN-SUM 2-path problem* is investigated. The objective of the problem is to find two disjoint paths P_1 and P_2 from s to t such that $l(P_1) + \alpha \cdot l(P_2)$ is minimized ($l(P_i)$ is defined as total cost of edges on path P_i). It is shown that several versions of the problem are NP-complete. In [17], the *MinSum-MinMin 2-path problem* is considered. The objective of this problem is to find two disjoint paths P_1 and P_2 from s to t satisfying: 1) $l(P_1) + l(P_2)$ is minimum and 2) $\min\{l(P_1), l(P_2)$ is minimum among all pairs of (P_1, P_2) satisfying 1). It is proved that this problem is NP-Complete on directed graphs for both edge-disjoint and node-disjoint cases, and there does not exist a constant approximation ratio unless $P = NP$.

In [4], the problem of finding k edge-disjoint paths of minimum total cost between a pair of nodes subject to minimum node sharing is studied and shown to be polynomial-time solvable with the same complexity as the minimum cost network flow (MCNF) problem. This problem is a special case of the minimum vulnerability 1-to-1 k-path problem investigated in this paper.

III. k Paths with Minimum Edge/Node Vulnerability between a Pair of Nodes

A. k Paths with Minimum Edge Vulnerability

We first consider the problem of finding k paths with minimum edge vulnerability, which provides partial protection against single link failure. In all our discussions, we consider a directed graph since an undirected graph can be converted to a directed graph simply by replacing each edge by two directed edges.

Let $G = (V, E)$ be a directed graph with non-negative cost $l(e)$ or $l(u, v)$ defined for edge $e = (u, v)$. Further, we assume that there are no parallel edges between any two nodes in G. Given a source s and destination t, let $P = \{P_1, P_2, \cdots, P_k\}$ be a set of paths in a graph $G = (V, E)$ from s to t. We define

$$\delta(e, P_i) = \begin{cases} 1, & e \in P_i \\ 0, & e \notin P_i \end{cases}, \quad \delta(e, P) = \sum_{i=1}^{k} \delta(e, P_i).$$

Then $\delta(e, P)$ represents the number of times that e appears in P. We define

$$\beta(P) = \sum_{e \in P} (\delta(e, P) - 1).$$

$\beta(P)$ is named *edge vulnerability* of the paths in P. Clearly, if $\beta(P) = 0$, then all paths in P are edge-disjoint.

We define $l(P) = \sum_{i=1}^{k} l(P_i)$, $L = \sum_{e \in E} l(e)$, and $M = kL + 1$. The objective of the problem of finding k paths with minimum edge vulnerability is to find a set of k paths $P = \{P_1, P_2, \cdots, P_k\}$ from s to t in G with: 1) $\min \beta(P)$; 2) $\min\{l(P) | P \text{ satisfying } 1)\}$.

Our first algorithm, named *PATHFINDER I*, consists of three major steps, 1) construct a network flow model $G' = (V', E')$ from G, 2) apply the minimum cost flow algorithms on G' to find a flow of value k, 3) obtain k paths from the flow. We list the pseudo-code of the algorithm as follows, where $l(e'), c(e'), f(e')$ represent the cost, capacity and flow value of an edge $e' \in E'$, and $|f| = \sum_{v' \in V'} f(s, v')$ represents the value of flow f.

```
Algorithm PATHFINDER I($G, s, t, k$):
begin
    //Step 1: Construct a flow network $G' = (V', E')$ from $G$:
    for each $e = (u, v) \in E$ do add a new parallel edge $\overline{e} = \overline{(u, v)}$.
    let $\overline{E} = \{\overline{(u, v)} | (u, v) \in E\}$.
    let $V' = V$ and $E' = E \cup \overline{E}$,
    for each $e \in E$ do assign capacity $c(e) = 1$,
    for each $\overline{e} \in \overline{E}$ do assign capacity $c(\overline{e}) = k - 1$,
    for each $e \in E$ do assign cost as its cost in $G$, $l(e)$,
    for each $\overline{e} \in \overline{E}$ do assign cost $l(\overline{e}) = M + l(e)$.
    //Step 2: Compute a flow on $G'$:
    get a minimum cost flow $f$ with $|f| = k$,
    by running an MCNF algorithm on $G'$.
    //Step 3: Obtain $k$ paths $P = \{P_1, P_2, \cdots, P_k\}$ of $G$:
    for each $e' \in E'$ do
        if $f(e') = 0$ then remove $e'$ from $G'$.
    end-for
    for $i = 1$ to $k$ do
        find a shortest path $P_i'$ from $s$ to $t$ in $G'$,
        obtain $P_i$ by replacing each $\overline{(u, v)} \in P_i'$ with $(u, v)$.
        for each $e' \in P_i'$ do
            $f(e') = f(e') - 1$,
            if $f(e') = 0$ then
                remove $e'$ from $G'$.
        end-for
    end-for
end
```

In G', we have $|V'| = |V|$ and $|E'| = 2|E|$. The MCNF algorithm will run properly on G' though there exist parallel edges between any pairs of nodes [1]. Obviously, the complexity of this algorithm is the same as the complexity of the MCNF algorithm it uses. Since M is a much bigger number than k, the *Successive Shortest Path Algorithm* [1] is an efficient algorithm for Step 2, which takes $O(k \cdot (|E| + |V| \log |V|))$ time.

Following we will prove the correctness of above algorithm. We have following 3 claims:

1) For any network flow f in G' with flow value $|f| = k$, the *Step 3* in algorithm PATHFINDER I retrieves k paths from s to t of G.
2) For any k paths $P = \{P_1, \cdots, P_k\}$ from s to t of G, we can create a flow f in G' with value $|f| = k$, by following steps (called *Flow Creation* procedure):
   - Set $f(e') = 0$ for every $e' \in E'$.
   - For $i = 1$ to k do
     - For each edge (u, v) in P_i: if $f(u, v) = 0$, set $f(u, v) = 1$; if $f(u, v) = 1$, set $f\overline{(u, v)} = f\overline{(u, v)} + 1$.
3) The above 2 procedures are reversible each other. That is, the result flow of applying *Flow Creation* procedure on paths that are retrieved from flow f by procedure *Step 3* is f; the result paths of applying *Step 3* procedure on flow which is created by procedure *Flow Creation* from k paths P are the same k paths P. We call f the *corresponding flow* of P, and call P the *corresponding paths* of f.

We use $c(f) = \sum_{e' \in E'} l(e') \cdot f(e')$ to denote the total cost of flow f.

Lemma 1: For a k paths $P = \{P_1, \cdots, P_k\}$ of G and its corresponding network flow f in G', we have:

$$\left\{ \begin{array}{lcl} c(f) & = & l(P) + \beta(P) \cdot M \\ \beta(P) & = & \lfloor c(f)/M \rfloor \\ l(P) & = & c(f) - \beta(P) \cdot M \end{array} \right. \tag{1}$$

Proof: As $\delta(e, p) - 1$ is exactly the flow value on $\overline{e} = \overline{(u, v)}$, we have:

$$\begin{aligned} c(f) &= \sum_{e \in P} l(e) + \sum_{e \in P} (\delta(e, P) - 1) \cdot l(\overline{e}) \\ &= \sum_{e \in P} l(e) + \sum_{e \in P} (\delta(e, P) - 1) \cdot (M + l(e)) \\ &= \sum_{e \in P} \delta(e, P) \cdot l(e) + \sum_{e \in P} (\delta(e, P) - 1) \cdot M \\ &= l(P) + \beta(P) \cdot M \end{aligned}$$

Since $l(P) = \sum_{i=1}^{k} l(P_i) \leq k \cdot L < M$, we have:

$$\beta(P) = \lfloor c(f)/M \rfloor, \;\; l(P) = c(f) - \beta(P) \cdot M.$$

■

Theorem 1: For a weighted directed graph $G = (V, E)$ with source s and destination t, algorithm PATHFINDER I computes a k-path solution $P = \{P_1, P_2, \cdots, P_k\}$ such that $\beta(P)$ is minimized and $l(P)$ is minimized among all possible sets of k paths in G with minimum $\beta(P)$.

Proof: Suppose for the sake of contradiction the claim is not true, then there exists a different set of k paths from s to t, $P' = \{P_1', P_2', \cdots, P_k'\}$ such that one of the following conditions holds:

1) $\beta(P') < \beta(P)$;
2) $\beta(P') = \beta(P)$, and $l(P') < l(P)$.

Let f and f' be the corresponding flows for P and P'. From lemma 1, we have $c(f) = (l(P) + \beta(P) \cdot M)$ and $c(f') = (l(P') + \beta(P') \cdot M)$. Hence:

$$c(f) - c(f') = (\beta(P) - \beta(P')) \cdot M + (l(P) - l(P')).$$

Case 1: $\beta(P') < \beta(P)$.
Then $\beta(P) - \beta(P') \geq 1$. Thus:

$$\begin{aligned} c(f) - c(f') &\geq M + l(P) - l(P') \\ &\geq M - l(P') \\ &\geq M - \sum_{i=1}^{k} l(P_i') \\ &\geq M - k \cdot L \\ &= 1. \end{aligned}$$

This contradicts the assumption that f is a minimum cost flow.

Case 2: $\beta(P') = \beta(P)$, and $l(P') < l(P)$.
Then, we have

$$c(f) - c(f') = l(P) - l(P') > 0.$$

This contradicts the assumption that f is a minimum cost flow.

■

Note that, if $\beta(P) = 0$, all k paths in P are edge-disjoint, and they are also a solution for the MinSum k-path problem.

B. k Paths with Minimum Node Vulnerability Subject to Minimum Edge Vulnerability

We next consider the problem of finding k paths with minimum node vulnerability, which can provide partial protection against single node failure. For $P = \{P_1, P_2, \cdots P_k\}$ we define

$$\eta(v, P_i) = \begin{cases} 1, & v \in P_i \\ 0, & v \notin P_i \end{cases}, \quad \eta(v, P) = \sum_{i=1}^{k} \eta(v, P_i).$$

Then $\eta(v, P)$ represents the number of times that node v is shared among paths in P. We define

$$\gamma(P) = \sum_{v \in P, v \neq s,t} (\eta(v, P) - 1).$$

$\gamma(P)$ is named the *node vulnerability* of P. Clearly, if $\gamma(P) = 0$, then all paths in P are node-disjoint.

The objective of the problem of finding shortest k paths with minimum node vulnerability in addition to minimum edge vulnerability is to find a set of k paths $P = \{P_1, P_2, \cdots, P_k\}$ from s to t in G with: 1) $\min \beta(P)$; 2) $\min\{\gamma(P) | P$ satisfying 1)$\}$; 3) $\min\{l(P) | P$ satisfying 1) and 2)$\}$. We define $M' = ((|V| - 2)(k-1) + 1) \cdot M$.

To solve this problem, we present algorithm *PATHFINDER II* as follows.

```
Algorithm PATHFINDER II($G, s, t, k$):
begin
    //Step 1: Construct a flow network $G'' = (V'', E'')$ from $G$:
    construct $G' = (V', E')$ from $G$ as shown in Step 1 of PATHFINDER I.
    for each $e \in E$ do assign cost as its cost in $G$, $l(e)$,
    for each $e' \in \overline{E}$ do set its cost $l(e') = M' + l(e)$.
    for each $v' \in V' - \{s, t\}$ do
        replace $v'$ with two nodes $v''_1, v''_2$ and add
        two edges $(v''_1, v''_2)$ and $\overline{(v''_1, v''_2)}$ as shown in Fig. 3,
        assign $(v''_1, v''_2)$ capacity $c(v''_1, v''_2) = 1$,
        assign $\overline{(v''_1, v''_2)}$ capacity $c\overline{(v''_1, v''_2)} = k - 1$,
        assign $(v''_1, v''_2)$ cost $l(v''_1, v''_2) = 0$,
        assign $\overline{(v''_1, v''_2)}$ cost $l\overline{(v''_1, v''_2)} = M$.
    end-for
    let $\hat{V} = \{v''_1, v''_2 \mid v' \in V' - \{s, t\}\}$.
    let $\hat{E} = \{(v''_1, v''_2), \overline{(v''_1, v''_2)} \mid v' \in V' - \{s, t\}\}$.
    let $V'' = \hat{V} \cup \{s, t\}, E'' = E \cup \overline{E} \cup \hat{E}$.
    //Step 2: Compute a flow on $G''$.
    get a minimum cost flow $f$ with $|f| = k$,
    by running an MCNF algorithm on $G''$.
    //Step 3: Obtain $k$ paths $P = \{P_1, P_2, \cdots, P_k\}$ of $G$:
    for each $e'' \in E''$ do
        if $f(e'') = 0$ then remove $e''$ from $G''$.
    end-for
    for $i = 1$ to $k$ do
        find a shortest path $P''_i$ from $s$ to $t$ in $G''$,
        obtain $P_i$ by replacing each $(v''_1, v''_2)$ or $\overline{(v''_1, v''_2)}$ in $P''_i \cap \hat{E}$ with
        node $v \in V$, replacing each $\overline{(u, v)} \in P''_i \cap \overline{E}$ with $(u, v)$ and
        for each $e'' \in P''_i$ do
            $f(e'') = f(e'') - 1$,
            if $f(e'') = 0$ then remove $e''$ from $G''$.
        end-for
    end-for
end
```

In G'', we have $|V''| = 2 \cdot |V|$, and $|E''| = |E'| + 2|V| = 2 \cdot (|E| + |V|)$. Obviously, the complexity of this algorithm is the same as the complexity of the MCNF algorithm it uses. Again, if we use the *Successive Shortest Path Algorithm*, the complexity is still $O(k \cdot (|E| + |V| + |V| \log |V|)) = O(k \cdot (|E| + |V| \log |V|))$.

See figure 4 for an example of graph transformation from G to G'' with $k = 3$.

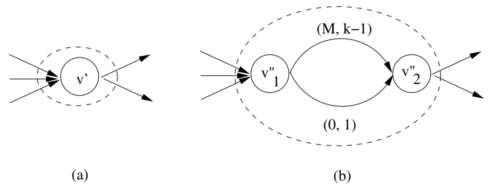

Fig. 3. Node splitting for G''. (a) The original node. (b) The new node.

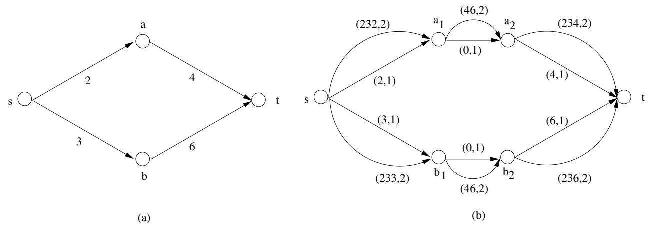

Fig. 4. (a) Before *Step 1*: graph $G = (V, E)$. (b) After *Step 1*: flow network $G'' = (V'', E'')$. $M = (2+3+4+6) \cdot 3 + 1 = 46$, $M' = (2 \cdot 2 + 1) \cdot M = 230$. Symbol (x, y) represents the (cost, capacity) pair for each edge.

Similarly, we have following claims:

1) For any network flow f in G'' with flow value $|f| = k$, the *Step 3* in algorithm PATHFINDER II retrieves k paths from s to t of G.
2) For any k paths $P = \{P_1, \cdots, P_k\}$ from s to t of G, we can create a flow f in G'' with value $|f| = k$, by following steps (called *Flow Creation* procedure):
   - Set $f(e'') = 0$ for every $e'' \in E''$.
   - For $i = 1$ to k do:
     - For each node $v \neq s, t$ in P_i: if $f(v''_1, v''_2) = 0$, set $f(v''_1, v''_2) = 1$; if $f(v''_1, v''_2) = 1$, set $f\overline{(u, v)} = f\overline{(u, v)} + 1$.
     - For each edge (u, v) in P_i: if $f(u''_2, v''_1) = 0$, set $f(u''_2, v''_1) = 1$; if $f(u''_2, v''_1) = 1$, set $f\overline{(u''_2, v''_1)} = f\overline{(u''_2, v''_1)} + 1$.
3) The above 2 procedures are reversible each other. We call f the *corresponding flow* of P, and call P the *corresponding paths* of f.

Lemma 2: For a k paths $P = \{P_1, \cdots, P_k\}$ of G and its corresponding network flow f in G'', we have:

$$\begin{cases} c(f) &= \beta(P) \cdot M' + \gamma(P) \cdot M + l(P) \\ \beta(P) &= \lfloor c(f)/M' \rfloor \\ \gamma(P) &= \lfloor (c(f) - \beta(P) \cdot M')/M \rfloor \\ l(P) &= c(f) - \beta(P) \cdot M' - \gamma(P) \cdot M \end{cases} \qquad (2)$$

Proof: Note that $\eta(v, P) - 1$ is exactly the flow value on $\overline{(v''_1, v''_2)}$. Similarly, $\delta(e, P) - 1$ is exactly the flow value on $\overline{(u''_2, v''_1)}$. Thus we have:

$$\begin{aligned}
c(f) &= \sum_{v \in P, v \neq s,t} (\eta(v,P)-1) \cdot M + \\
&\quad \sum_{e \in P} l(e) + \sum_{e=(u,v) \in P} (\delta(e,P)-1) \cdot l(\overline{(u_2'', v_1'')}) \\
&= \gamma(P) \cdot M + \\
&\quad \sum_{e \in P} l(e) + \sum_{e \in P} (\delta(e,P)-1) \cdot (M' + l(e)) \\
&= \gamma(P) \cdot M + \\
&\quad \sum_{e \in P} \delta(e,P) \cdot l(e) + \sum_{e \in P} (\delta(e,P)-1) \cdot M' \\
&= \beta(P) \cdot M' + \gamma(P) \cdot M + l(P).
\end{aligned}$$

Since $l(P) < M$ (from the proof of lemma 1), and $\gamma(P) = \sum_{v \in P, v \neq s,t} (\eta(v,P)-1) \leq (|V|-2)(k-1)$, we have $M' > \gamma(P) \cdot M + l(P)$. Thus,

$$\begin{cases} \beta(P) &= \lfloor c(f)/M' \rfloor \\ \gamma(P) &= \lfloor (c(f) - \beta(P) \cdot M')/M \rfloor \\ l(P) &= c(f) - \beta(P) \cdot M' - \gamma(P) \cdot M \end{cases}$$

■

Theorem 2: For a weighted directed graph $G = (V, E)$ with source s and destination t, algorithm PATHFINDER II computes a k-path solution $P^* = \{P_1, P_2, \cdots, P_k\}$ such that P^* satisfies the following properties:

(1) $\beta(P^*) = \beta^* = \min\{\beta(P) | P$ is a k-path solution$\}$;

(2) $\gamma(P^*) = \min\{\gamma(P) | P$ is a k-path solution such that $\beta(P) = \beta^*\}$;

(3) $l(P^*) = \min\{l(P) | P$ is a k-path solution such that $\gamma(P) = \gamma(P^*)$ and $\beta(P) = \beta^*\}$.

Proof: Suppose for the sake of contradiction the claim is not true, then there exists a different set of k paths from s to t of G, $P' = \{P'_1, P'_2, \cdots, P'_k\}$ such that one of following conditions holds:

1) $\beta(P') < \beta(P^*)$;
2) $\beta(P') = \beta(P^*)$, and $\gamma(P') < \gamma(P^*)$;
3) $\beta(P') = \beta(P^*), \gamma(P') = \gamma(P^*)$, and $l(P') < l(P^*)$.

Let f and f' be the corresponding flows for P^* and P'. From lemma 2, we have $c(f) = \beta(P^*) \cdot M' + \gamma(P^*) \cdot M + l(P^*)$, $c(f') = \beta(P') \cdot M' + \gamma(P') \cdot M + l(P')$. Hence:

$$\begin{aligned}
c(f) - c(f') &= (\beta(P^*) - \beta(P')) \cdot M' + \\
&\quad (\gamma(P^*) - \gamma(P')) \cdot M + \\
&\quad (l(P^*) - l(P')).
\end{aligned}$$

Case 1: $\beta(P') < \beta(P^*)$.
Then $\beta(P^*) - \beta(P') \geq 1$, and

$$\begin{aligned}
c(f) - c(f') &\geq M' - (\gamma(P') \cdot M + l(P')) \\
&\geq M' - ((|V|-2)(k-1) \cdot M + \sum_{i=1}^{k} l(P'_i)) \\
&\geq M' - ((|V|-2)(k-1) \cdot M + k \cdot L) \\
&= M' - ((|V|-2)(k-1) \cdot M + (M-1)) \\
&= M' - ((|V|-2)(k-1)+1) \cdot M + 1 \\
&= 1.
\end{aligned}$$

This contradicts the assumption that f is a minimum cost flow.

Case 2: $\beta(P') = \beta(P^*)$, *and* $\gamma(P') < \gamma(P^*)$.
Then $\gamma(P^*) - \gamma(P') \geq 1$. Thus,

$$\begin{aligned}
c(f) - c(f') &\geq M + l(P^*) - l(P') \\
&\geq M - l(P') \\
&\geq M - k \cdot L \\
&= 1
\end{aligned}$$

This contradicts the assumption that f is a minimum cost flow.

Case 3: $\beta(P') = \beta(P^*), \gamma(P') = \gamma(P^*)$, $l(P') < l(P^*)$.
Then,

$$c(f) - c(f') = l(P^*) - l(P') > 0.$$

This contradicts the assumption that f is a minimum cost flow.

■

Note that, if $\beta(P^*) = 0$, then all paths in P^* are edge-disjoint. If $\gamma(P^*) = 0$, then $\beta(P^*) = 0$ and all paths in P^* are node-disjoint (and, of course, edge-disjoint).

IV. Minimum Vulnerability k Paths from One Source to k Destinations

We then consider the problem of finding a set of k paths $P = \{P_1, P_2, \cdots, P_k\}$ from s to a set of destinations $T = \{t_1, \cdots, t_k\}$ in a graph $G = (V, E)$ with minimum vulnerability. The definitions of L, M, and M' are the same as in the last section. We first present our algorithm, named *PATHFINDER III*, for finding k paths with minimum edge vulnerability from s to T. Similar to the previous two algorithms, PATHFINDER III also consists of three major steps:

```
Algorithm PATHFINDER III($G, s, T, k$):
begin
    //Step 1: Construct a flow network $G' = (V', E')$ from $G$:
    for each $e = (u, v) \in E$ do add a new parallel edge $\overline{e} = \overline{(u, v)}$.
    let $\overline{E} = \{\overline{(u, v)} | (u, v) \in E\}$.
    add a new node $t$ and set $V' = V + \{t\}$.
    for $i = 1$ to $t$ do add edge $(t_i, t)$.
    let $\tilde{E} = \{(t_i, t) | \forall t_i \in T\}$.
    let $E' = E \cup \overline{E} \cup \tilde{E}$.
    for each $e \in E \cup \tilde{E}$ do assign capacity $c(e) = 1$,
    for each $\overline{e} \in \overline{E}$ do assign capacity $c(\overline{e}) = k - 1$,
    for each $e \in E$ do assign cost as its cost in $G$, $l(e)$,
    for each $\tilde{e} \in \tilde{E}$ do assign cost $l(\tilde{e}) = 0$,
    for each $\overline{e} \in \overline{E}$ do assign cost $l(\overline{e}) = M + l(e)$.
    //Step 2: Compute a flow on $G'$:
    get a minimum cost flow $f$ with $|f| = k$,
    by running an MCNF algorithm on $G'$.
    //Step 3: Obtain $k$ paths $P = \{P_1, P_2, \cdots, P_k\}$ of $G$:
    for each $e' \in E'$ do
        if $f(e') = 0$ then remove $e'$ from $G'$.
    end-for
    for $i = 1$ to $k$ do
        find a shortest path $P'_i$ from $s$ to $t$ in $G'$,
        obtain $P_i$ by replacing each $\overline{(u, v)} \in P'_i$ with $(u, v)$,
        and removing edges in $\tilde{E}$.
        for each $e' \in P'_i$ do
            $f(e') = f(e') - 1$,
            if $f(e') = 0$ then
                remove $e'$ from $G'$.
        end-for
    end-for
end
```

In G', we have $|V'| = |V| + 1$, $|E'| = 2|E| + k$. Again, the complexity of this algorithm is the same as the complexity of the MCNF algorithm it uses. If we use the *Successive Shortest Path Algorithm*, the complexity is still $O(k\cdot(|E|+|V|\log|V|))$. Then we have the following result.

Theorem 3: For a weighted directed graph $G=(V,E)$, algorithm PATHFINDER III computes a k-path solution $P_T=\{P_1,P_2,\cdots,P_k\}$ from s to $T=\{t_1,\cdots,t_k\}$ such that $\beta(P)$ is minimized and $l(P)$ is minimized among all possible sets of k paths in G with minimum $\beta(P)$.

The proof is almost exactly the same as the proof of theorem 1.

Applying the node splitting technique used in algorithm PATHFINDER II, we can modify PATHFINDER III to find shortest k paths with minimum node vulnerability $\gamma(P_T)$ subject to minimum edge vulnerability $\beta(P_T)$.

V. Minimum-Vulnerability Paths from Single Source to All-Pairs of Destinations

In this section, we consider the problem of finding 2 paths with minimum edge vulnerability for all pairs of nodes $\{t_1,t_2\}\subseteq V-\{s\}$ in a given graph $G=(V,E)$. We define

$$\begin{aligned}\beta'(\{t_1,t_2\}) &= \min\{\beta(\{P_1,P_2\})|P_1,P_2 \text{ are paths}\\ &\qquad \text{from } s \text{ to } t_1 \text{ and } t_2, \text{respectively}\}.\end{aligned}$$

Algorithm PATHFINDER III presented in the previous section can be used to compute $\beta'(\{t_1,t_2\})$ for two nodes $t_1,t_2\neq s$. Instead of applying PATHFINDER III to all pairs of nodes, we can use the *Min-Sum Single-Source All-Destination-Pairs Shortest Disjoint Two Paths* algorithms proposed in [13] to solve this problem more efficiently.

Firstly, we transform G into G' using Step 1 of PATHFINDER I. Then we apply Algorithm I in [13] to get the shortest distance matrix $\mathcal{M}=(d_{uv},p_{uv},q_{uv})$, where d_{uv} represents the total cost of the shortest edge-disjoint path pair from s to u,v, and p_{uv},q_{uv} are used to backtrack the preceding nodes in the two disjoint paths. The matrix $\mathcal{M}$ can be computed in $\Theta(|V|^2)$ time and $\Theta(|V|^2)$ space using the algorithm of [13]. Thus, for each pair $u,v\in V-\{s\}$, we have the following claims:

1) $\beta'(\{u,v\})=\lfloor d_{uv}/M\rfloor$.
2) For $P'=(P'_1,P'_2)$ obtained from p_{uv},q_{uv} (which takes $\Theta(L(n))$ time, where $L(n)$ is the number of nodes in P'_1,P'_2), $\beta(P')=\beta'(\{u,v\})$. Let $P=\underline{(P_1,P_2)}$ be the paths obtained by replacing each edge $\overline{(x,y)}$ in $\overline{E}$ with edge (x,y). Then (P_1,P_2) are two paths from s to u,v in G. We claim that $l(P_1)+l(P_2)$ is the shortest among all possible two paths from s to u,v with vulnerability $\beta'(\{u,v\})$.

We omit the proof here. Note the total complexity of finding $\beta'(\{t_1,t_2\})$ for every pairs $t_1,t_2\in V-\{s\}$ is only $O(|V|^2)$.

Similarly, the same method can be extended to solve the problem of finding shortest paths from s to all pairs of nodes in directed graphs subject to minimum edge vulnerability $\beta(P)$ and minimum node vulnerability $\gamma(P)$ using the node splitting technique in PATHFINDER II.

VI. Concluding Remarks

To protect a network against single link/node failure, disjoint paths satisfying specific connection requirements are needed. However, in a real network, disjoint paths may not always exist. A viable approach is to find paths with minimum number of shared links or nodes. In this paper, we characterized the degree of edge sharing and node sharing by the notion of edge vulnerability and node vulnerability. We provided a complete study of three variant problems of finding k paths with minimum edge/node vulnerability and showed that all these problems are polynomially solvable by reducing them to the minimum cost network flow problem. The algorithms presented in this paper are very useful for many network applications including reliable unicast/multicast communications, reliable client-server communications, and protection under the dual homing architecture. Future work includes finding k paths with minimum vulnerability satisfying load balancing requirements.

References

[1] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin, *Network Flows*, Prentice-Hall, 1993.

[2] L.R. Ford and D.R. Fulkerson, *Flows in Networks*, Princeton, 1962.

[3] C. Lee and S. Koh, “A Design of the Minimum Cost Ring-Chain Network with Dual-Homing Survivability: A Tabu Search Approach,” *Computers Operation Research*, vol. 24, no. 9, 1997, pp. 883-897.

[4] S.-W. Lee and C.-S. Wu,“A K-best Paths Algorithm for Highly Reliable Communication Networks,” *IEICE Trans. Commun.*, vol. E82-B, no. 4, 1999, pp. 586-590.

[5] C.L. Li, S.T. McCormick, and D. Simchi-Levi, “The Complexity of Finding Two Disjoint Paths with Min-Max Objective Function,” *Discrete Appl. Math.*, vol. 26, no. 1, 1990, pp. 105-115.

[6] C.L. Li, S.T. McCormick, and D. Simchi-Levi, “Finding Disjoint Paths with Different Path-Costs: Complexity and Algorithms,” *Networks*, vol. 22, 1992, pp. 653-667.

[7] C.H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization - Algorithms and Complexity*, Dover, 1998.

[8] A. Proestaki and M. Sinclair, “Design and Dimensioning of Dual-Homing Hierarchical Multi-ring Networks,” *IEE Proc. Commun.*, vol. 147, no. 2, 2000, pp. 96-104.

[9] J. Shi and J. Fonseka, “Analysis and Design of Survivable Telecommunication Networks,” *IEE Proc. Commun.*, vol. 144, no. 5, 1997, pp. 322-330.

[10] J. W. Suurballe, “Disjoint Paths in a Network,” *Networks*, vol. 4, 1974, pp. 125-145.

[11] J. W. Suurballe and R. E. Tarjan, “A Quick Method for Finding Shortest Pairs of Disjoint Paths,” *Networks*, vol. 14, 1984, pp. 325-336.

[12] J. Wang, M. Yang, X. Qi, and R. Cook, “Dual-homing multicast protection,” *Proc. IEEE GLOBECOM*, 2004, pp. 1123-1127.

[13] B. Yang and S.Q. Zheng, “Finding Min-Sum Disjoint Shortest Paths from a Single Source to All Pairs of Destinations,” *Technical Report, Dept. of Computer Science, Univ. of Texas at Dallas*, Apr., 2005.

[14] B. Yang, S.Q. Zheng, and S. Katukam, “Finding Two Disjoint Paths in a Network with Min-Min Objective Function,” *Proc. the 15th IASTED Int'l Conf. PDCS*, 2003, pp. 75-80.

[15] B. Yang, S.Q. Zheng, and E. Lu, “Finding Two Disjoint Paths in a Network with Normalized α^+-Min-Sum Objective Function,” *Technical Report, Dept. of Computer Science, Univ. of Texas at Dallas*, Apr. 2005.

[16] B. Yang, S.Q. Zheng, and E. Lu, “Finding Two Disjoint Paths in a Network with Normalized α^--Min-Sum Objective Function,” *Technical Report, Dept. of Computer Science, Univ. of Texas at Dallas*, Apr. 2005.

[17] B. Yang, S.Q. Zheng, and E. Lu, “Finding Two Disjoint Paths in a Network with MinSum-MinMin Objective Function,” *Technical Report, Dept. of Computer Science, Univ. of Texas at Dallas*, Apr., 2005.

[18] M. Yang, J. Wang, X. Qi, and Y. Jiang, “On Finding the Best Partial Multicast Protection Tree under Dual-Homing Architecture,” to appear in *Proc. IEEE HPSR*, 2005, Hong Kong, May 2005.