MARA: Maximum Alternative Routing Algorithm

Yasuhiro Ohara
Japan Advanced Institute of Science and Technology
Ishikawa, Japan 923–1292
Email: yasu@jaist.ac.jp

Shinji Imahori
University of Tokyo
Tokyo, Japan 113–8656
Email: imahori@mist.i.u-tokyo.ac.jp

Rodney Van Meter
Keio University
Kanagawa, Japan 252–8520
Email: rdv@sfc.wide.ad.jp

***Abstract*—In hop-by-hop networks, provision of multipath routes for all nodes can improve fault tolerance and performance. In this paper we study the multipath route calculation by constructing a directed acyclic graph (DAG) which includes all edges in the network. We define new DAG construction problems with the objectives of 1) maximizing the minimum connectivity, 2) maximizing the minimum max-flow, and 3) maximizing the minimum max-flow as an extension of shortest path routing. A family of new algorithms called Maximum Alternative Routing Algorithms (MARAs) is described, proven formally to solve the problems optimally, and contrasted with existing multipath algorithms. MARAs are evaluated for the number of paths, the length of paths, the computational complexity, and the computation time, using simulations based on several real Internet Autonomous System (AS) network topologies. We show that MARAs run in sub-second times on moderate-speed processors and achieve a significant increase in the number of paths compared to existing multipath routing algorithms. These results should help further the process of deploying multipath routing in real-world networks.**

I. Introduction

The most essential task of Internet routing systems is to calculate routes that do not include routing loops and are consistent on all routers. The Internet is designed as a hop-by-hop network so that routers in the network need not maintain per-session communication state, improving network scalability in performance, in number of nodes, and in number of sessions. Since all routers in the Internet forward packets autonomously, they must independently make consistent decisions about the path to a destination in order to avoid routing loops. Configuring consistent loop-free routes for a destination is synonymous with constructing a Directed Acyclic Graph (DAG).

To a first approximation, the routing in the Internet is single-path routing. Its goal is to find the shortest path by assigning each edge a cost and making the routing decision so that, for each source and destination pair of nodes, the sum of the edge costs in the path is the minimum. This principle can be seen in all commonly used routing protocols, including RIP [1], OSPF [2], IS-IS [3], and BGP [4].

Compared to the single-path routing commonly used in the Internet, enabling the use of multiple paths simultaneously, called *multipath routing*, can improve availability and transport capacity. However, in the current shortest-path routing, each router usually selects a single shortest path to a destination. Thus, the routes to the destination form a sink tree to the destination (called a Shortest Path Tree, or SPT), where multiple paths in the network can rarely be used. Although the existing IP routing architecture allows multipath routes where the path from the current node to the destination branches to multiple nexthops, multipaths in the shortest path routing require the costs to the destination on these paths to be equal, which rarely happens. The rare multipath routes in the shortest path routing are called Equal Cost Multi Paths (ECMPs) [2]. Note that both SPT and its ECMP extension are still a kind of DAG[1].

In order to be considered as a viable option for the evolution of the Internet, multipath routing must be theoretically sound, as well as practically implementable. Prior work has shown that multipath-aware end nodes are able to actively participate in failure avoidance [5], [6], [7], but unmodified nodes also gain benefits from the increase in connectedness and aggregate throughput. In this paper, we focus on advancing the theory of multipath routing and demonstrate via implementation of our algorithms that multipath route calculation is indeed feasible. Thus, multipath is advancing toward a complete, practical system, and this paper checks off important points in the list of work items.

We propose a family of novel multipath route calculation algorithms, called Maximum Alternative Routing Algorithms (MARAs). MARAs construct a DAG that includes all edges in the network graph structure, in order to provide a significant number of alternative paths among all nodes to a destination. This is the first study of such DAGs for Internet routing, to the best of our knowledge. We introduce three new graph problems, *all-to-one maximum connectivity routing problem*, *all-to-one max-flow routing problem*, and *all-to-one maximum shortest path alternatives problem*, to describe the objectives of the algorithms. The new routing algorithms that can solve the problems optimally, called MARA-MC, MARA-MMMF, and MARA-SPE, are developed. All of them are essentially the applications of an existing algorithm, called Maximum Adjacency Ordering (MA ordering [8], [9]) algorithm, to Internet routing.

The main contributions of this paper are: (a) the introduction of the new graph routing problems; (b) the proposal of new multipath routing algorithms that apply MA ordering algo-

[1]An ECMP extension of SPT is no longer precisely a tree. In this paper we still refer to the ECMP extension as SPT, for ease of explanation.

rithm to the Internet routing; (c) the findings of the optimality proofs; and (d) the analysis of the algorithms on several realistic ISP topologies.

The rest of this paper is organized as follows. Concepts and related work for multipath routing are presented in Section II. Section III models the routing problem and reviews the MA ordering algorithm. Sections IV-A, IV-B, and IV-C describe the problem, the algorithm, and the optimality proof for the objectives of improving the connectivity, the transport capacity, and the transport capacity with the shortest path routing compatibility, respectively. Section V evaluates the algorithms with the analysis of the results (i.e., calculated number of routes and path length), the computational complexities, and the computation time. Section VI gives the concluding remarks.

II. Multipath Routing

Multipath routing can contribute to the availability of the communication network. Although it is commonly believed that Internet routing systems avoid failures, in the real world, complex problems that cannot be detected by routing systems occur, such as hardware malfunctions and software bugs in routers [10], [11]. In theory, the rich connectivity of the Internet should exhibit high reliability. In practice, Internet routing protocols reduce the richer graph to a non-redundant tree in most cases. The non-redundant tree is vulnerable to failures, which degrade communication performance and connectivity until the route recomputation completes. For example, there exist problems in the forwarding plane of routers, which may remain undetected for extended periods and may ultimately require manual intervention to correct.

Along with a path switching mechanism that does not depend on the failure detection of a particular routing system, advance multipath route calculation can contribute to the availability of a communication system. Examples of such a path switching mechanism are Deflection [5] and Drouting [6], [7], where the end host can request a change of the communication path by changing the packet tag embedded in IP packets.

Multipath routing may also contribute to the transport capacity of a network. In the case of shortest-path routing, the maximum available bandwidth between a source and a destination is limited to that of the shortest path. The transport capacity of the network is unreasonably constrained, despite the possible availability of alternative paths with additional bandwidth. Multipath routing can alleviate this problem by splitting traffic to multipaths to balance the network load. Although MPLS [12] may be considered to be a multipath routing mechanism in this sense, this paper also considers networks that do not employ MPLS.

Maximum flow algorithms such as Dinits' algorithm [13] consider the max-flow for a source-destination node pair in the graph, while the all-to-one max-flow routing problem considers the max-flow among all nodes to a destination.

Some multipath route calculation algorithms have been proposed in the past. Existing multipath routing methods are based on, and are extensions of, shortest path routing. Hence they require the administratively-established routing cost. MPDA [14] is a link state routing algorithm which distributes only partial topology information. MDVA [15] is a distance vector routing algorithm that uses diffusing computation [16]. MPATH [17] is another distance vector routing algorithm that distributes predecessor node information of paths. MPDA, MDVA and MPATH calculate multipath routes that are loop-free, using the Loop Free Invariant (LFI) condition on the routing metrics.

FIR [18] computes per network interface routing tables by executing the Shortest Path First (SPF) calculations separately for each of its neighbors in order to route around the failure. In [5], Yang and Wetherall proposed Deflection, which extends the LFI condition by utilizing the identity of the previous hop to produce an increased number of nexthops. FIR and Deflection are multipath routing methods created with the goal of failure avoidance. Because they provide backtracking paths that transit the same node twice, they require complex path switching mechanisms, such as changing a packet's nexthop based on the previous hop or the incoming network interface, on a per-packet basis.

MARAs, as proposed in this paper, do not consider the administrative routing costs normally attached to links. Our method provides simple hop-by-hop multipath routes without backtrackings, and thus the modification required in the path switching mechanism is rather simple. Furthermore, in contrast to the LFI, which sometimes fails to utilize all the edges in the network for routing depending on the routing metric setting, MARAs always calculate the routes using all edges.

III. Prerequisites

A. *General routing problem*

An undirected graph $G = (V, E)$ and a destination node $t \in V$ are given, as the graph structure of the network on which the routes are to be calculated. The input graph is assumed to be connected and may have multiple edges between a pair of nodes, i.e., multigraph. In the following we utilize notations n and m for $|V|$ and $|E|$, respectively. For the destination, it is required to calculate routing directions on edges to indicate routes for the hop-by-hop routing from every node to the destination on the graph. To avoid routing loops, the resulting directed graph should not have any directed cycles; i.e., the goal is to find a directed acyclic graph (DAG) on the input graph. The DAG is sometimes referred to as a *routing graph* in this paper.

A number of DAGs is allowed for a hop-by-hop network to be used as a routing graph. Note that a DAG signifies only routing directions for a single destination node. Thus multiple DAGs must be calculated to produce routes for all source-destination pairs in the network.

This paper discusses undirected graphs as input for ease of explanation. The algorithms in this paper support directed graphs; that is, for a given directed graph and a destination node, a DAG that achieves the particular objective can be found, without any modification required to the presented algorithms.

Instead of deciding the direction of each edge independently, a permutation of nodes is determined by our algorithms (i.e., nodes are labeled from 1 to n). A permutation of nodes sets the direction of each edge from the higher-labeled node to the lower-labeled node. It is known that constructing a DAG on an undirected graph is equivalent to deciding a topological order of nodes [19].

In the next section, we review the MA ordering proposed by Nagamochi and Ibaraki [8] that produces a permutation of nodes. MA ordering is utilized and extended by our algorithms, described in Sections IV-A2, IV-B2, and IV-C2.

B. MA Ordering

With $G = (V, E)$ as an input, an ordering $v_1, v_2, \ldots, v_n$ of nodes is called an MA ordering if an arbitrary node s is chosen as a starting point v_1, and after choosing the first i nodes as $v_1, v_2, \ldots, v_i$, the $(i+1)$-th node v_{i+1} is chosen from the remaining nodes v that have the largest number of edges between v and $\{v_1, \ldots, v_i\}$. It is known that MA ordering is useful for various problems on graphs, such as identifying a minimum cut between two nodes and solving the edge-connectivity augmentation problem [9]. An algorithm to compute an MA ordering is given in Algorithm 1, where $d(v, S)$ denotes the number of edges between a node v and a set of nodes S (i.e., the number of edges from v to one of the nodes in S).

Algorithm 1 MA ordering algorithm

```
1: procedure MA ORDERING($G = (V, E), s \in V$)
2:     $v_1 \leftarrow s,\ S = \{s\},\ T = V \setminus \{s\}$
3:     $i \leftarrow 2$
4:     while $i \leq |V|$ do
5:         choose a node $v \in T$ with the largest $d(v, S)$
6:         $v_i \leftarrow v,\ S = S \cup \{v\},\ T = T \setminus \{v\}$
7:         $i \leftarrow i + 1$
8:     end while
9:     output ordering $(v_1, v_2, \ldots, v_n)$ of nodes
10: end procedure
```

For a capacitated, undirected graph $G = (V, E, cap)$, an ordering similar to the MA ordering is also defined. In this case, instead of choosing a node $v \in T$ with the largest $d(v, S)$ in Line 5, a node v with the largest $\sum_{w \in S} cap(v, w)$ is chosen.

An MA ordering can be obtained in $O(m + n)$ time for the case without link capacity using a custom data structure, and in $O(m + n \log n)$ time for the case with link capacity using Fibonacci heap [20], as shown by Nagamochi and Ibaraki [8].

IV. MARAs

Examples of the multipath routes calculated by MARAs are illustrated in Figure 1. The graph labeled NETWORK denotes the network topology of AS1221. On the topology, the routes destined to a node (upper-right in the figure) calculated by Dijkstra, LFI, and two variants of MARAs (MARA-MC and MARA-SPE) are shown. Notice that the result of Dijkstra is almost non-redundant and nearly a tree, and LFI lacks some links present in the original graph (e.g., the upper left one).

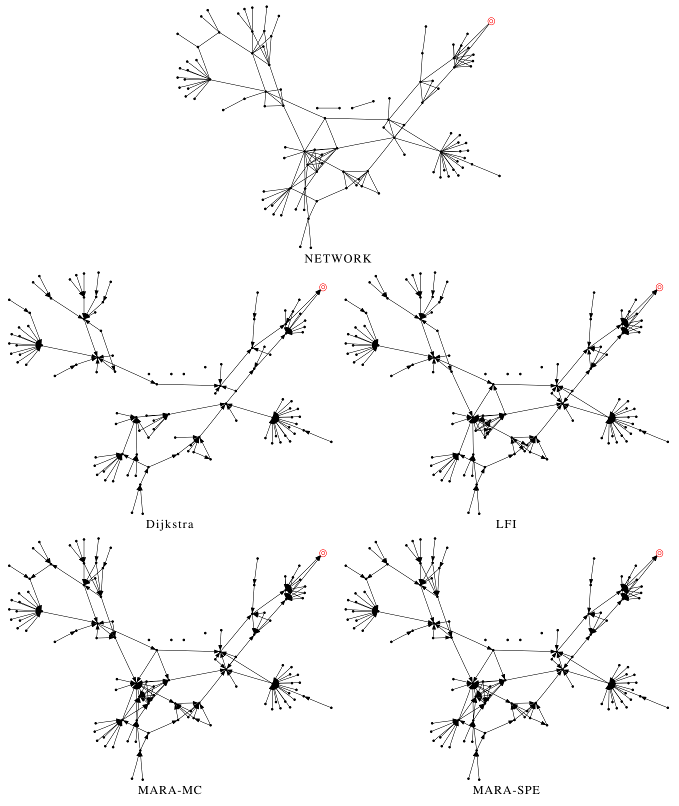

Fig. 1. Illustration of routes for a destination computed by Dijkstra, LFI, MARA-MC and MARA-SPE.

LFI and MARA-SPE are supersets of Dijkstra but in different ways, and MARA-MC has a totally different orientation from the others. Only MARAs employ all of the links in the network for the destination.

A. Improving connectivity

1) All-to-one Maximum Connectivity Routing Problem: The first objective is to calculate a DAG with robustness. The robustness of reaching from all nodes to a destination in the network is determined by the minimum connectivity in the routing graph. By maximizing the minimum connectivity among all nodes to the destination node, an ideal routing that achieves the highest robustness for the destination can be obtained. We define the problem of deciding the directions of edges in order to maximize the minimum connectivity among all nodes to a destination as the *all-to-one maximum connectivity routing problem*. By calculating such routing for every destination, the whole routing that achieves the most robustness for all source-destination pairs can be obtained.

A formal description of the all-to-one maximum connectivity routing problem is as follows. An undirected graph $G = (V, E)$ and the destination $t \in V$ are given as inputs. A possible solution (a routing graph, DAG) is expressed by an orientation of edges (a set of directions of all edges). For each orientation p of the edges, $k_p(v, t)$ denotes the edge connectivity from v to t in the DAG determined by

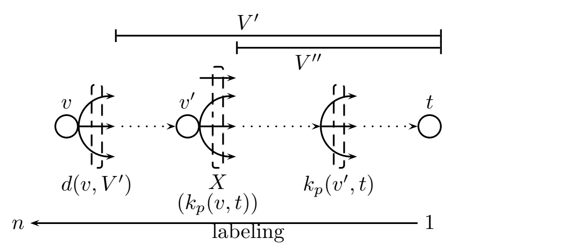

Fig. 2. The relation between v, V', v', V'', X and t in Lemma 4.1

the orientation p. ($k_p(t,t) = +\infty$ is assumed for every orientation p.) The all-to-one maximum connectivity routing problem is to find an orientation of edges in the given graph that maximizes the minimum connectivity among all nodes to the destination t under the condition that the resulting directed graph is acyclic. The problem is formulated as follows.

$$\begin{array}{ll} \text{Maximize} & \min_{v \in V} k_p(v,t) \\ \text{subject to} & p \text{ is an acyclic orientation.} \end{array}$$

2) MARA-MC: We propose an algorithm, MARA-MC (Maximum Connectivity), that solves the all-to-one maximum connectivity routing problem optimally. MARA-MC is very simple: An MA ordering for an undirected graph $G = (V, E)$ and an destination node $t \in V$ is computed using the destination t as the initial node s in Algorithm 1. Then the direction of each edge is set from the higher-labeled node to the lower-labeled node. The direction on an edge is interpreted as a route to the destination, with the head of the directed edge being the nexthop of the route. The MA ordering calculates routes from all nodes to the destination t. In order to determine routes among all source-destination node pairs in the network, the MA ordering algorithm must be executed for each destination in the network separately, i.e., n times. MARA-MC runs in $O(m + n)$ time for one destination, hence $O(mn + n^2)$ time for all source-destination pairs. A node must calculate routes for all source-destination pairs to find the routes from the node to all destinations in the network.

Below, we give an optimality proof for MARA-MC. Let v be the bottleneck node, which is defined as the lowest-labeled node with minimum $k_p(v,t)$ in an edge orientation p. First, in the following lemma, it is shown that the minimum cut between the bottleneck node v and the destination t is always the neighboring cut of v.

Lemma 4.1: Let v be the lowest-labeled node with the minimum $k_p(v,t)$. Then, $k_p(v,t) = d(v, V')$ holds, where V' is the set of nodes which have lower-labels than v.

Proof: By using a relationship between the cut and the connectivity on a graph, $k_p(u,t) \leq d(u, U)$ holds for every node $u \neq t$, where U is a set of nodes having lower-labels than u. In other words, the connectivity from u to t, i.e., $k_p(u,t)$, cannot exceed the number of edges in its neighboring cut, $d(u, U)$.

The equality in Lemma 4.1 is proven by contradiction. Suppose that $k_p(v,t) < d(v, V')$ holds. The max-flow min-cut theorem on a directed graph [21] implies that there exists a directed cut X (i.e., a partition of nodes) whose size is equal to the connectivity from v to t, i.e., $k_p(v,t)$. Figure 2 shows the cut X in the center, separated from v by the assumption. v' is assumed to be the lowest-labeled node such that v and v' belong to the same subset of nodes partitioned by X, and thus is neighboring to X. Then, the following inequalities hold:

$$k_p(v,t) \geq d(v', V'') \geq k_p(v', t),$$

where V'' is the set of nodes which have lower-labels than v' (V'' is also shown in Figure 2). This contradicts the assumption that v is the lowest-labeled node with the minimum connectivity to t. ■

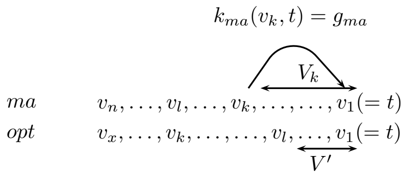
$k_{ma}(v_k, t) = g_{ma}$
V_k
ma
$v_n, \ldots, v_l, \ldots, v_k, \ldots, \ldots, v_1 (= t)$
opt
$v_x, \ldots, v_k, \ldots, \ldots, v_l, \ldots, v_1 (= t)$
V'

Fig. 3. The optimality proof of MARA-MC

Theorem 4.2: MARA-MC solves the all-to-one maximum connectivity routing problem optimally.

Proof: Suppose that the MA ordering algorithm starting from the destination node t produced a node-ordering such that the label i is given to the node v_i for every node in V (the destination node has label 1, $t = v_1$). Call the orientation of edges from the higher-labeled node to the lower-labeled node in the node ordering "ma," and let g_{ma} be the objective value for this orientation, i.e., the minimum connectivity among all nodes to the destination t. $V_i = \{v_1, v_2, \ldots, v_{i-1}\}$ denotes the set of nodes which have smaller labels than v_i. By the definition of $d(v, V')$, $d(v, A) \leq d(v, B)$ if $A \subseteq B$ holds.

Let v_k be the lowest-labeled node whose connectivity to the destination node t in "ma," $k_{ma}(v_k, t)$, is the smallest. By using Lemma 4.1, the following equalities hold:

$$g_{ma} = k_{ma}(v_k, t) = d(v_k, V_k).$$

From a property of the MA ordering algorithm (in Line 5 of Algorithm 1), $d(v_i, V_k) \leq d(v_k, V_k)$ holds for $i = k+1, k+2, \ldots, n$, since v_k is chosen earlier than v_i.

Assume that there exists the other optimal ordering, whose orientation is denoted by "*opt*," that is better than the MA ordering (i.e., $g_{opt} > g_{ma}$ holds). Figure 3 illustrates the assumption, using the same node labels as ma also for *opt*. Let v_l be the node with the lowest label in *opt* among a set of nodes $\{v_k, v_{k+1}, \ldots, v_n\}$. In other words, v_l is the node which is to the left of v_k in "ma", but is the rightmost node in "*opt*" among the nodes $\{v_k, v_{k+1}, \ldots, v_n\}$. Thus, assuming V' be the set of nodes having smaller labels than v_l in *opt*, $V' \subseteq V_k$ holds in Figure 3. Then, the following equalities and inequalities hold:

$$\begin{aligned} g_{opt} &\leq k_{opt}(v_l, t) \leq d(v_l, V') \leq d(v_l, V_k) \\ &\leq d(v_k, V_k) = g_{ma}. \end{aligned}$$

This contradicts the assumption that there exists a better orientation opt with $g_{opt} > g_{ma}$. ■

B. Improving transport capacity

1) All-to-one Max-flow Routing Problem: The next objective is to support the highest possible traffic load. A typical routing algorithm uses only the shortest path to support traffic for each source-destination pair. By also using roundabout paths, it becomes possible to support a traffic load that exceeds the capacity of the shortest path. The total traffic which can be supported from a node to the destination is determined by the max-flow of the routing graph. Since the minimum of max-flows among all nodes to the destination node (called the bottleneck) limits the supported range of the traffic demands, it is important to increase the minimum max-flow. By maximizing the minimum max-flow among all nodes to the destination, an ideal routing that achieves the best support for the excess (over capacity) traffic to the destination can be obtained. We call this maximization problem the *all-to-one max-flow routing problem*. By calculating such routing for every destination, the routing that achieves the best support for all source-destination node pairs can be obtained.

A formal description of the all-to-one max-flow routing problem is as follows. An undirected graph with link capacity $G = (V, E, cap)$ is given, where $cap(v, w)$ is the capacity of the edge (v, w) ($cap(v, w) > 0$ if $(v, w) \in E$, otherwise 0). A node $t \in V$ is also given as the destination.

A solution (a routing graph, DAG) is expressed by an orientation of edges. For each orientation p of edges, $f_p(v, t)$ denotes the amount of max-flow from v to t in the DAG determined by the orientation p. ($f_p(t, t) = +\infty$ is assumed for every orientation p.) The all-to-one max-flow routing problem is to find an orientation of edges in the given graph that maximizes the minimum max-flow among all nodes to the destination t under the condition that the resulting directed graph is acyclic. The problem is formulated as follows.

$$\begin{array}{ll} \text{Maximize} & \min\limits_{v \in V} f_p(v, t) \\ \text{subject to} & p \text{ is an acyclic orientation.} \end{array}$$

Note that, if all link capacities in the input graph are identical, the all-to-one max-flow routing problem is synonymous with the all-to-one maximum connectivity routing problem, and the optimal solution would be identical.

2) MARA-MMMF: The algorithm that solves the all-to-one max-flow routing problem optimally is called MARA-MMMF (Maximizing the Minimum Max-Flow). MARA-MMMF computes an MA ordering for an undirected graph with link capacity $G = (V, E, cap)$ using $\sum_{w \in S} cap(v, w)$ instead of $d(v, S)$ to consider the max-flow rather than the connectivity. The difference between MARA-MC and MARA-MMMF is only the input (the graph contains link capacity) and the preference on the node (larger max-flow is preferred over larger connectivity). MARA-MMMF runs in $O(m + n \log n)$ time for one destination node, and in $O(mn + n^2 \log n)$ time for all node pairs.

The optimality proof for MARA-MMMF is very similar to that of MARA-MC. The optimality of MARA-MMMF can be obtained by transforming the proof of MARA-MC where $d(v, S)$ and $k_p(v, t)$ are substituted by $\sum_{w \in S} cap(v, w)$ and $f_p(v, t)$, respectively.

C. Improving transport capacity with shortest-path compatibility

1) All-to-one Maximum Shortest Path Alternatives problem: Most existing Internet routers employ shortest-path routing. When creating a new routing graph, it may be preferable to construct a DAG such that it includes the SPT. If a new routing is consistent and is a superset of the shortest path routing, new routers and shortest path routers can coexist in the network simultaneously.

The objective of this section is to calculate a DAG that is consistent and is a superset of the SPT, with the better support for the excess traffic. We define the problem of deciding the directions of edges to maximize the minimum max-flow among all nodes to a destination under the constraint of including SPT, as the *all-to-one maximum shortest path alternatives problem*. It is also possible to consider augmentation of connectivity, if $d(v, S)$ is used instead of $\sum_{w \in S} cap(v, w)$ below.

A formal description of the all-to-one maximum shortest path alternatives problem is as follows. An undirected graph with link costs and capacity $G = (V, E, cost, cap)$ is given, where $cost(v, w) > 0$ gives the administrative cost of the edge (v, w) if $(v, w) \in E$, otherwise $+\infty$ (the path with cost $+\infty$ is considered as unreachable). A node $t \in V$ is also given as the destination.

The all-to-one maximum shortest path alternatives problem is to find an orientation of edges in the given graph that maximizes the minimum max-flow among all nodes to the destination t under the condition that the resulting DAG (denoted by DAG_p) is a superset of SPT. The problem is formulated as follows.

$$\begin{array}{ll} \text{Maximize} & \min\limits_{v \in V} f_p(v, t) \\ \text{subject to} & \text{DAG}_p \supseteq \text{SPT}. \end{array}$$

2) MARA-SPE: In this section, we propose an algorithm called MARA-SPE (Shortest Path Extension) that solves the all-to-one maximum shortest path alternatives problem. MARA-SPE consists of two stages: (1) an SPT on the graph $G = (V, E, cost, cap)$ is computed using the destination node t as the root, such that the SPT sinks to the root[2], and (2) a DAG is calculated so that all edges have identical directions to the SPT, if the edge is included in the SPT.

In the first stage, MARA-SPE computes a SPT using Dijkstra's algorithm. This is done in $O(m + n \log n)$ time. In the second stage, MARA-SPE computes an ordering of n nodes via an algorithm similar to the MA ordering algorithm. This is formally given in Algorithm 2, where T denotes the set of nodes such that within the SPT, all the ancestors of any node $v \in T$ belong to S.

[2]This is the opposite direction from the typical SPT.

Algorithm 2 MARA-SPE

```
 1: procedure MARA-SPE($G = (V, E, cost, cap), t \in V$)
 2:     $v_1 \leftarrow t,\ S = \{t\}$
 3:     $i \leftarrow 2$
 4:     compute $U = \{e \in E \mid U$ is the SPT$\}$
 5:     while $i \leq |V|$ do
 6:         compute $T = \{v \in V \setminus S \mid \forall (v, w) \in U,\ w \in S\}$
 7:         choose a node $v \in T$ with the largest $\sum_{w \in S} cap(v, w)$
 8:         $v_i \leftarrow v,\ S = S \cup \{v\}$
 9:         $i \leftarrow i + 1$
10:     end while
11:     output ordering $(v_1, v_2, \ldots, v_n)$ of nodes
12: end procedure
```

TABLE I
THE SAMPLE NETWORK GRAPHS USED IN THIS PAPER (COLLECTED BY ROCKETFUEL). THE #NODES AND #LINKS INCLUDE A FEW DISCONNECTED COMPONENTS.

Name	#nodes	#links	description
AS1221	108	153	Telstra
AS1239	315	972	Sprint
AS1755	87	166	Ebone
AS3257	161	328	Tiscali
AS3967	79	147	Exodus
AS6461	141	374	Abovenet

The total computation time to update T (Line 6) is $O(m + n \log n)$. Choosing the node $v \in T$ with the largest $\sum_{w \in S} cap(v, w)$ (Line 7) runs in $O(\log n)$ time with a Fibonacci heap, and the n iterations (Line 5) give a total execution cost of $O(n \log n)$. Thus, MARA-SPE runs in $O(m + n \log n)$ time for one destination node, and in $O(mn + n^2 \log n)$ time for all node pairs.

The optimality proof for MARA-SPE is as follows. First, MARA-SPE always outputs a permutation of n nodes; that is, the size of node set T is at least 1 for any $i \leq |V|$ in Line 7, since the SPT must not have any directed cycles. It is also clear that the resulting DAG includes all the edges in the SPT with those correct directions (this property comes from Line 6). The optimality on the objective value can be obtained in a fashion similar to that of MARA-MC: Let v_k be the bottleneck node with the lowest label in the orientation of MARA-SPE (denoted by *spe*), and v_l be the node with the smallest label in *opt* among a set of nodes $\{v_k, v_{k+1}, \ldots, v_n\}$. Because of the ordering of the *opt*, all the ancestors of v_l in SPT must be in the set V', and thus also in V_k, since $V' \subseteq V_k$. From the ordering of *spe*, $\sum_{w \in V_k} cap(v_l, w) \leq \sum_{w \in V_k} cap(v_k, w)$ must hold. This contradicts the assumption that the bottleneck with the lowest label is the v_k.

V. EVALUATION

In this section MARAs are evaluated for the number and the length of paths, the computational complexity, and the computation time. The input graphs are drawn from the Rocketfuel project [22], and summarized in Table I. Since Rocketfuel does not provide the bandwidth of the links, and since we do not know any reasonable model for synthetic bandwidth for the links, MARA-MMMF cannot be evaluated in fair comparison with others, and is omitted from the simulation below.

A. Number and Length of Paths

MARA-MC and MARA-SPE are evaluated by comparing the number and the length of the computed paths to that of Dijkstra and LFI, on AS1221, AS1755, AS3257, and AS3967. The paths are calculated by enumerating all of the paths on the calculated DAG, for each source-destination pair. Since the enumeration of paths takes time exponential in the number of edges, the evaluation of two larger ISPs, AS1239 and AS6461, are omitted here.

MARA-MC and MARA-SPE exhibited a significant improvement. The average and standard deviation of the number, the average length (in nodes), and the maximum length of paths among source-destination pairs are shown in Table II. The table shows that on most topologies, LFI adds only a small number of multipaths to existing Dijkstra, MARA-SPE outperforms LFI for the number of additional multipaths even under the strict restriction to retain the shortest paths, and MARA-MC succeeds in calculating a huge number of paths. In AS3967, LFI seems to outperform MARA-MC and MARA-SPE in the average number of paths. This is because LFI computed a large number of paths to a small set of source-destination pairs (more than 1000 paths for 1.31% of the source-destination pairs), and leads to the highest mean average number of paths (this is sometimes meaningless in terms of network robustness; see the distribution and its discussion below). The average and maximum length of paths naturally tend to become larger when more multipaths are calculated. The use of multipaths implicitly presumes the use of longer roundabout paths.

The CCDF of the number, the average length, and the maximum length of paths are shown in Figure 4. The figure shows that MARA-MC consistently calculates a large number of multipaths to a large fraction of source-destination pairs, MARA-SPE calculates a moderate number, yet outperforms LFI. The number of multipaths found by LFI is either worse or equivalent to that of MARA-SPE, depending on the topology and the administrative routing cost setting. For example, in AS1221, MARA-MC calculates more than 15 multipaths for 52.65% of source-destination pairs, MARA-SPE for 34.86%, while LFI does for only 2.17%. Dijkstra, which calculates only ECMPs, finds few for most source-destination pairs. Figure 4 also shows that MARAs exhibit longer path length. For example, in AS1221, the fraction of source-destination pairs that has the average path length more than 8 nodes is 14.89% for Dijkstra, and 16.5% for LFI, while it is 39.85% for MARA-SPE, and 56.13% for MARA-MC.

As mentioned earlier, the average number of paths by LFI in AS3967 outperformed MARA-MC (Table II). However, examining the distribution of number of paths in AS3967, we see that MARAs still exhibit preferable characteristics. In the low range, MARAs provide the same number of paths to a *larger* fraction of source-destination pairs than LFI (although for MARA-SPE the difference is very small). For example, LFI provides more than 15 paths to 38.43%

TABLE II
AVERAGE AND STANDARD DEVIATION OF THE NUMBER, THE AVERAGE LENGTH, AND THE MAXIMUM LENGTH OF PATHS IS DENOTED AS AVG (±STD). THE AVERAGE LENGTH INCREASES WITH MULTIPATH ALGORITHMS DUE TO THE ACTIVE USE OF SECONDARY, LONGER PATHS.

Name	algorithm	#paths		avg path len		max path len	
	Dijkstra	1.37	(± 0.62)	5.79	(± 1.78)	5.93	(± 1.87)
AS1221	LFI	3.37	(± 3.91)	6.03	(± 1.85)	6.35	(± 2.01)
	MARA-SPE	16.18	(± 22.63)	6.96	(± 2.37)	8.27	(± 3.18)
	MARA-MC	26.27	(± 31.47)	8.24	(± 2.99)	10.60	(± 4.30)
	Dijkstra	2.07	(± 1.69)	6.08	(± 2.13)	6.23	(± 2.22)
AS1755	LFI	16.59	(± 44.47)	6.58	(± 2.21)	7.31	(± 2.66)
	MARA-SPE	48.04	(± 111.18)	7.56	(± 2.85)	9.32	(± 3.97)
	MARA-MC	872.53	(± 2335.69)	11.71	(± 4.53)	16.44	(± 6.79)
	Dijkstra	2.14	(± 1.97)	6.82	(± 2.74)	7.17	(± 2.95)
AS3257	LFI	607.56	(± 2999.15)	7.98	(± 3.09)	9.66	(± 4.33)
	MARA-SPE	3743.34	(± 14590.81)	10.36	(± 4.26)	14.75	(± 6.95)
	MARA-MC	10176.30	(± 37125.45)	12.40	(± 4.54)	18.45	(± 7.50)
	Dijkstra	1.89	(± 1.69)	5.88	(± 2.16)	6.08	(± 2.37)
AS3967	LFI	75.98	(± 232.45)	7.02	(± 3.01)	8.36	(± 4.06)
	MARA-SPE	32.52	(± 76.92)	7.33	(± 2.84)	9.30	(± 4.18)
	MARA-MC	42.08	(± 66.94)	8.62	(± 3.31)	10.95	(± 4.64)

TABLE III
COMPUTATIONAL COMPLEXITIES OF ROUTING ALGORITHMS. l DENOTES THE NUMBER OF NEIGHBORING NODES FOR A NODE, m THE NUMBER OF EDGES IN THE GRAPH, AND n THE NUMBER OF NODES.

Algorithm	At each node	Overall
Bellman-Ford	$O(mn)$	$O(mn^2)$
Dijkstra	$O(m + n \log n)$	$O(mn + n^2 \log n)$
LFI & Deflect-1	$O(lm + ln \log n)$	$O(mn + n^2 \log n)$
MARA-MC	$O(mn + n^2)$	$O(mn + n^2)$
MARA-MMMF	$O(mn + n^2 \log n)$	$O(mn + n^2 \log n)$
MARA-SPE	$O(mn + n^2 \log n)$	$O(mn + n^2 \log n)$

of source-destination pairs, while MARA-SPE provides 15 paths to 41.5%, and MARA-MC does to 53.03%. For failure avoidance, providing a small number of multipaths to a larger part of the network is preferred over providing a large number of multipaths to a small part of the network.

B. Computational Complexity

Table III gives the computational complexities for several algorithms. The middle column is the complexity for the algorithm at *each* node (router) in the network; the right column is the *network-wide total* computation necessary for all nodes to calculate the routes for all source-destination pairs. l denotes the number of neighboring nodes for a node.

Bellman-Ford [23] is the routing algorithm utilized in Distance Vector routing algorithms such as RIP. For each node, it calculates n routes in at most m steps, and globally it calculates this n times.

The complexity of Dijkstra, used in OSPF and IS-IS, is similar to that of MA Ordering, in that it is $O(m + n \log n)$ by appropriate use of Fibonacci heap. Since Dijkstra calculates all routes from a source node to all destinations, globally it must be calculated n times (once for each source node).

LFI, which is equivalent to Deflection rule-1 (Deflect-1), is used in MPDA, MDVA, and MPATH. For each node, LFI and Deflect-1 require the Dijkstra calculation for each neighboring node, and comparisons of the neighbor's cost with the node's cost for each destination, which is $O(ln)$. The complexity of LFI and Deflect-1 is globally n times Dijkstra plus m cost comparisons for each destination ($O(mn)$).

The bottom three lines of the table show the complexity of our MARA algorithms. The calculation is the same at every node or router; it is, in theory, possible to share results of the calculation among multiple nodes if desired.

Of course, the $O(\cdot)$ notation, by definition, hides a constant factor and transient terms in the execution time. Those terms can be of critical importance when considering whether to deploy a system in the real world. The next subsection addresses these concerns.

C. Computation Time

The CDF of the computation time of 100 executions of Dijkstra, LFI, MARA-MC, and MARA-SPE, for a single node (not overall), are shown in Figure 5. The algorithms are implemented in C and executed on a FreeBSD virtual machine on a VMWare ESX Server® with 2-way Dual-Core AMD Opteron® 2.4GHz CPU and 16 GB memory. The algorithms use the more practical priority queue, i.e., a basic heap [24], rather than a Fibonacci heap, hence the complexities differ somewhat from those shown in Section V-B. Our implementation of Dijkstra is $O((m+n) \log n)$, LFI $O(l(m+n) \log n)$, and MARA-MC and MARA-SPE $O(n(m + n) \log n)$. The time is measured by reading the processor's hardware timer using `rdtsc` from a user-mode software application, hence the operating system's process scheduling may be negatively affecting the variability of the results.

Carefully implemented, Dijkstra and LFI exhibited very fast computation in all topologies. Since theoretically all execution of MARAs at different nodes are identical, the variation in computation time is believed to be the result of process scheduling. The price for computing a significant number of multipath routes in MARA-MC and MARA-SPE is the exhibited significant increase in execution time; however, the computation times shown are for real-world network topologies and demonstrate the feasibility of actual deployment in a real network. The average and standard deviation of the

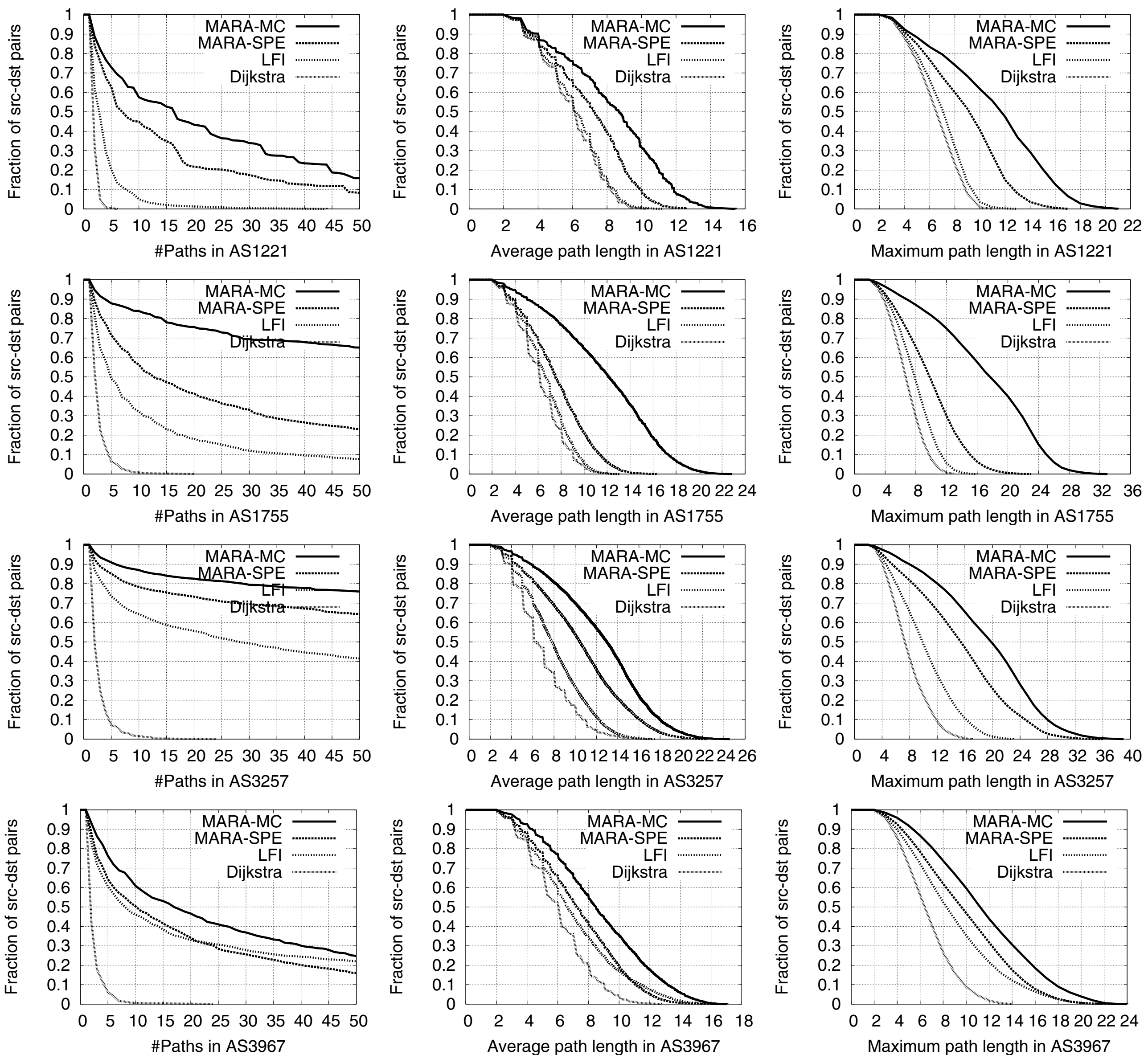

Fig. 4. Comparison on distributions on the number, the average, and the maximum length of paths.

computation time are summarized in Table IV. Figure 5 shows the CDF of the computation time.

VI. Conclusion

Migrating the Internet from essentially single-path routing to multipath routing can potentially improve the fault resilience of the network, raise the aggregate bandwidth available between two nodes, and increase the utilization of otherwise idle resources. One step toward the realization of this goal is the development of algorithms for finding the multipath routes on the network graph.

We have developed a family of three optimal algorithms, which we call MARA-MC, MARA-MMMF, and MARA-SPE. This family is called MARAs, *Maximum Alternative Routing Algorithms*. All three algorithms calculate a DAG that includes all of the edges in the graph. These routing algorithms compute many multipaths, which may be used in the Internet to improve failure avoidance. We evaluated MARAs on the number and the length of paths and on the algorithm's computational time, using topologies inferred from real, large networks. The results showed that MARA-MC calculate a significant number of multipaths; for the several autonomous systems (ASes) we evaluated, the lowest average number of paths found was more than twenty-five. The computation time on a modern processor is sub-one second, verifying its feasibility in practice.

References

[1] G. Malkin, "RIP Version 2," IETF, RFC 2453, Nov. 1998.
[2] J. Moy, "OSPF Version 2," IETF, RFC 2328, Apr. 1998.

TABLE IV
AVERAGE AND STANDARD DEVIATION (AVG (± STD)) OF THE COMPUTATION TIME IN MICRO SECONDS (μsec).

algorithm	AS1221		AS1239		AS1755	
Dijkstra	119.41	(± 100.66)	635.34	(± 272.23)	148.60	(± 129.42)
LFI	468.18	(± 426.14)	4,546.10	(± 4,584.48)	656.71	(± 361.14)
MARA-MC	10,459.72	(± 2,651.98)	163,202.06	(± 34,714.66)	9,732.54	(± 2,339.74)
MARA-SPE	22,534.48	(± 5,481.52)	372,349.32	(± 84,535.17)	22,196.77	(± 4,486.81)
algorithm	**AS3257**		**AS3967**		**AS6461**	
Dijkstra	270.71	(± 180.51)	134.70	(± 114.31)	282.43	(± 155.58)
LFI	1,301.60	(± 1,170.92)	632.77	(± 338.44)	1,767.83	(± 1,070.39)
MARA-MC	30,802.33	(± 12,719.84)	8,822.24	(± 1,667.75)	32,992.62	(± 7,831.13)
MARA-SPE	76,288.48	(± 28,535.07)	19,193.98	(± 2,815.41)	70,674.97	(± 14,926.35)

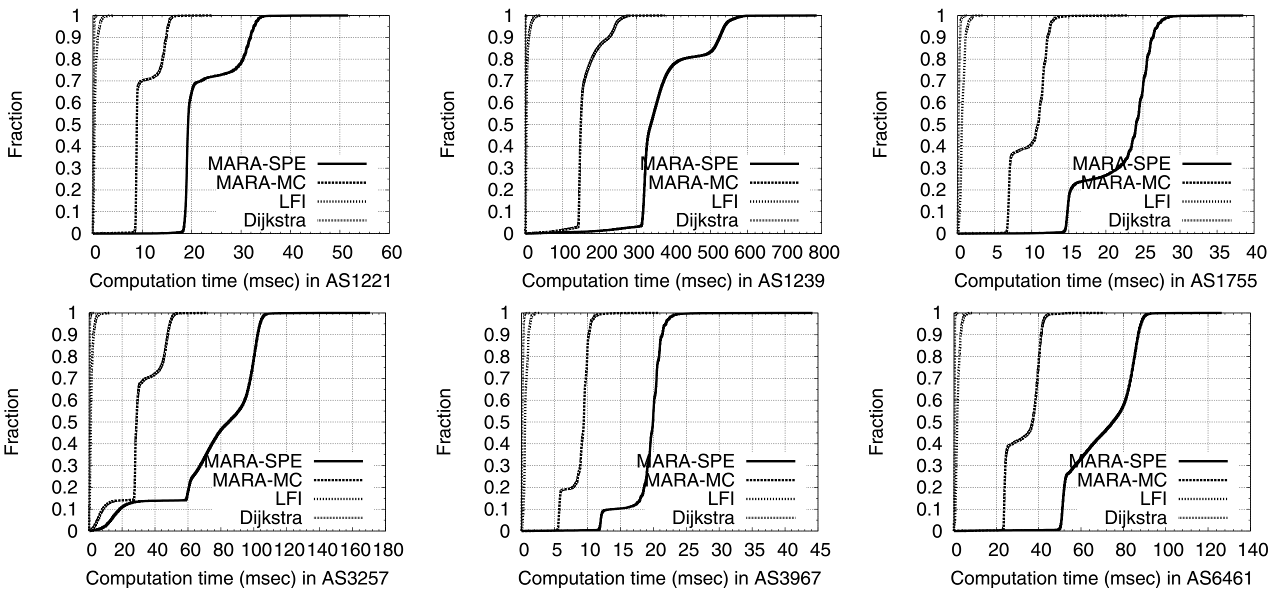

Fig. 5. Distribution of computation time of algorithms in milliseconds ($msec$).

[3] *Intermediate system to Intermediate system routeing information exchange protocol for use in conjunction with the Protocol for providing the Connectionless-mode Network Service (ISO 8473)*, ISO/IEC 10589:2001.

[4] Y. Rekhter, T. Li, and S. Hares, "A border gateway protocol 4 (BGP-4)," IETF, RFC 4271, Jan. 2006.

[5] X. Yang and D. Wetherall, "Source selectable path diversity via routing deflections," in *ACM SIGCOMM*, 2006, pp. 159–170.

[6] Y. Ohara, H. Kusumoto, O. Nakamura, and J. Murai, "Drouting architecture: Improvement of failure avoidance capability using multipath routing." *IEICE Transactions*, vol. 91-B, no. 5, pp. 1403–1415, 2008.

[7] Y. Ohara, "Routing Architecture for the Dependable Internet," Ph.D. dissertation, Keio University, 2008.

[8] H. Nagamochi and T. Ibaraki, "Computing edge-connectivity in multigraphs and capacitated graphs," *SIAM J. Discret. Math.*, vol. 5, no. 1, pp. 54–66, 1992.

[9] ——, "Graph connectivity and its augmentation: applications of MA orderings," *Discrete Appl. Math.*, vol. 123, no. 1-3, pp. 447–472, 2002.

[10] E. C. Rosen, "Vulnerabilities of network control protocols: an example," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 11, no. 3, pp. 10–16, 1981.

[11] T. Nolle, "What we should learn from the AT&T outage," *Network World*, vol. 15, p. 73, May 4 1998.

[12] E. Rosen, A. Viswanathan, and R. Callon, "Multiprotocol Label Switching Architecture," IETF, RFC 3031, Jan. 2001.

[13] E. A. Dinits, "Algorithm for solution of a problem of maximum flow in a network with power estimation," *Soviet Mathematics-Doklady*, vol. 11, no. 5, pp. 1277–1280, 1970.

[14] S. Vutukury and J. J. Garcia-Luna-Aceves, "A simple approximation to minimum-delay routing," in *ACM SIGCOMM*, 1999, pp. 227–238.

[15] ——, "MDVA: A distance-vector multipath routing protocol." in *IEEE INFOCOM*, 2001, pp. 557–564.

[16] E. W. Dijkstra and C. S. Scholten, "Termination detection for diffusing computations." *Inf. Process. Lett.*, vol. 11, no. 1, pp. 1–4, 1980.

[17] S. Vutukury and J. J. Garcia-Luna-Aceves, "An algorithm for multipath computation using distance-vectors with predecessor information," in *IEEE ICCCN*, 1999, pp. 534–539.

[18] S. Lee, Y. Yu, S. Nelakuditi, Z.-L. Zhang, and C.-N. Chuah, "Proactive vs reactive approaches to failure resilient routing." in *IEEE INFOCOM*, 2004, pp. 176–186.

[19] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network flows: theory, algorithms, and applications*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1993.

[20] M. L. Fredman and R. E. Tarjan, "Fibonacci heaps and their uses in improved network optimization algorithms," *J. ACM*, vol. 34, no. 3, pp. 596–615, 1987.

[21] G. B. Dantzig and D. R. Fulkerson, "On the max flow min cut theorem of networks," The RAND Corporation, Santa Monica, California, Tech. Rep., 1955, paper P-826.

[22] N. Spring, R. Mahajan, D. Wetherall, and T. Anderson, "Measuring ISP topologies with rocketfuel," *IEEE/ACM Trans. Netw.*, vol. 12, no. 1, pp. 2–16, 2004.

[23] C. Huitema, *Routing in the Internet*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1995.

[24] R. Sedgewick, *Algorithms in C*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2002.