STABLE ROUTING AND UNIQUE-MAX COLORING ON TREES

NICOLAI HAHNLE†, LAURA SANITÀ‡, AND RICO ZENKLUSEN§

Abstract. Some of the routing protocols used in telecommunication networks route traffic on a shortest path tree according to configurable integral link weights. One crucial issue for network operators is finding a weight function that ensures a stable routing: when some link fails, traffic whose path does not use that link should not be rerouted. In this paper we improve on several previously best results for finding small stable weights. As a conceptual contribution, we draw a connection between the stable weights problem and the seemingly unrelated unique-max coloring problem. In unique-max coloring, one is given a set of points and a family of subsets of those points called regions. The task is to assign to each region a color represented as an integer such that, for every point, one region containing it has a color strictly larger than the color of any other region containing this point. In our setting, points and regions become edges and paths of the shortest path tree, respectively, and based on this connection, we provide stable weight functions with a maximum weight of $O(n \log n)$ in the case of single link failure, where $n$ is the number of vertices in the network. Furthermore, if the root of the shortest path tree is known, we present an algorithm for determining stable weights bounded by $4n$, which is optimal up to constant factors. For the case of an arbitrary number of failures, we show how stable weights bounded by $3n$ can be obtained. All the results improve on the previously best known bounds.

Key words. routing protocols, unique-max coloring

AMS subject classifications. 68M10, 90C27, 68M12

DOI. 10.1137/100817565

1. Introduction. An important issue in the design of communication networks is the capability of a network to remain operational whenever some components, like links or nodes, fail. After a failure occurs in an operational network, there is a set of demands to be rerouted. A key property of the rerouting strategy, called the stability property by Grandoni et al. [6], requires that traffic demands that are not affected by the failure are not rerouted [3, 15]. Stability reduces transmission delays caused by the rerouting process and is therefore important for a good level of quality of service [5, 10].

Enforcing the desired routing in telecommunication networks is nontrivial, since in practice, routing is ruled by fixed routing protocols. Some popular routing protocols belong to the spanning tree protocol (STP) family. From an algorithmic point of view, the STP computes a shortest path tree rooted in one node according to link weights settled by network operators. Each demand is then routed on the corresponding unique path determined by this tree. Observe that this path is not necessarily a shortest path between its endpoints in the underlying network. To ensure stability,
Fig. 1. Labels on the edges indicate their weight, and shortest path trees rooted at vertex $a$ are indicated by thick edges. The weight function in (a) is not $1$-stable, since if the link $\{a,d\}$ fails and $a$ is chosen as the root node, traffic between $c$ and $d$ will be rerouted as shown in (b), even though it was not directly affected by the failure. The weight function in (c) is $1$-stable: whatever is the root node, and whatever is the failing edge, the new shortest path tree contains all nonfailed edges of the previous shortest path tree.

Network operators must find a suitable *weight function* for the links of the network. A desirable property of a weight function is that the failure of a single link, or more generally even up to some number $k$ of links, leads to a new shortest path tree that contains all nonfailed links of the shortest path tree before failure(s). Hence rerouting is minimized. Such weight functions are called *stable*, or more precisely $k$-stable if up to $k$ link failures are considered. For technological reasons, there can be an upper bound on the weights of the links, so it is important to find weight functions where the maximum weight is small.

For each link in the network, there is a set of unused backup links that can potentially be used as a replacement when the link fails. Which of these backup links is going to be used depends on the weight assignment, so one can think of these weights roughly as priorities. A backup link with low weight has a high priority of being used in case of a link failure. Intuitively, to ensure stability we could require that when a link fails, there is exactly one backup link that clearly has the highest priority among all potential backup links. The problem is therefore to assign priorities to backup links *consistently*, such that for every link in the network there is one clear winner among its potential backup links. However, as we noticed before, we also need to keep low the maximum assigned weight, and these two objectives are conflicting, as highlighted by a sample case in Figure 1. We present procedures for various settings to obtain such a both consistent and frugal priority assignment. A main conceptual contribution of this paper is based on the insight that such a consistent priority assignment can sometimes be obtained by using unique-max colorings.

Unique-max and conflict-free coloring are very closely related problems first introduced by Even et al. [4], motivated by frequency assignment problems in cellular networks. The unique-max coloring problem is defined as follows. Given a set of points $S$ and a family $R$ of subsets of $S$ (called the regions), the problem is to assign to each region an integer, called the region’s color, such that each point $p$ is contained
in one region whose color is strictly larger than all colors of the other regions containing \( p \). The objective is to use as few colors as possible. Unlike the more typical geometric settings, we consider unique-max colorings for the case where \( S \) is the set of links of a given tree \( T \), and \( R \) is a set of paths on \( T \).

The weaker property of conflict-free coloring, which we will define later, has been studied extensively in geometric settings. Results on unique-max colorings have often been obtained as well, though sometimes this has not been stated very explicitly. Emphasizing the attention on the stronger unique-max property, interesting new applications of the theory of conflict-free colorings, such as the one presented in this paper, may be found.

### 1.1. Detailed description of the problems.

**Stability under the STP.** We consider a telecommunication network represented as an undirected connected graph \( G = (V, E) \), where \( V \) is a set of \( n \) vertices and \( E \) is a set of edges representing the nodes and links of the network. There are traffic demands between each pair of vertices in \( V \). Network operators can set integer edge weights \( w : E \to \mathbb{N} \geq 1 \). Given such a weight function, the STP chooses one vertex \( r \in V \) as a root and computes a shortest path tree \( T = (V, E(T)) \) rooted at \( r \). Traffic demand between \( u \) and \( v \) in \( V \) will then be routed along the unique \( u-v \) path in \( T \). In general, this path need not be a shortest path from \( u \) to \( v \) in \( G \) with respect to \( w \).

Suppose now that a subset of edges \( F \subset E \) of \( G \) fails. The STP will recompute a new shortest path tree \( T' \) in \( G' = (V, E \setminus F) \) with respect to the same root \( r \) and reroute demands whose path has changed in \( T' \) compared to \( T \). In case \( G' \) is not connected anymore, the protocol acts independently in each connected component. In this case, we let \( T' \) be the union of the trees obtained in each component.

Clearly, any demand that was previously routed over a path containing some edge in \( F \) will be rerouted (unless the endpoints of the path are now in different components, in which case the demand can no longer be satisfied). Our goal is to find a weight function such that traffic demands that are not affected by the failure, i.e., demands whose corresponding paths do not contain any edge of \( F \), are not rerouted. This is equivalent to the requirement that all edges in \( E(T) \setminus F \) belong to \( E(T') \). A formal definition [6] is (see Figure 1 for an example) as follows.

**Definition 1.** \((k\text{-stable weights})\). Let \( G = (V, E) \) be a connected graph, and let \( T = (V, E(T)) \) be a spanning tree in \( G \). A function \( w : E \to \mathbb{N} \geq 1 \) is a \( k \)-stable weight function for \( T \) if, for all \( F \subset E \), \(|F| \leq k \) and for all vertices \( v \in V \) every shortest path tree rooted at \( v \) in \( G \setminus F = (V, E \setminus F) \) with respect to \( w \) contains all edges \( e \in E(T) \setminus F \) that are reachable from \( v \) in \( G \setminus F \).

In particular, by setting \( F = \emptyset \), the above definition implies that, for any \( v \in V \), \( T \) must be the unique shortest path tree in \( G \) with respect to \( w \) and rooted at \( v \). Note that the tree \( T \) is assumed to be given, so that network administrators can arbitrarily choose the initial routing. We would like to emphasize that we allow only assigning weights \( \geq 1 \) (in particular, not zero) since this is a restriction of the STP protocol (see the STP standard IEEE 802.1D at [18]). However, even if we allow assigning weights of value zero, this would not change our results significantly. One can observe that this would improve our bounds by a factor of \( n \), but it would also decrease the lower bound by the same factor.

As noted by Grandoni et al. [6], given any tree \( T \), obtaining a weight function that is \( k \)-stable for any \( k \) is rather easy if we are allowed to set arbitrary large weights: just assign a weight of 1 to the edges of \( T \), and a weight \( 2^k \cdot n \) for all other edges \( e_i \in E \setminus E(T) \). The maximum assigned weight in this case is \( O(n2^{|E(T)|}) \). Since assigned
weights cannot be arbitrarily large in practice (see some ranges in [19]), it is important
to find \(k\)-stable weight functions with a small maximum assigned weight. In particular,
the smaller the weights are that we need to assign, the larger the size can be of the
networks (or the number of simultaneous failures) that can be efficiently handled by
network operators.

**Unique-max coloring.** In this paper, we consider only the setting where the “points”
are edges of a tree \(T\) and the “regions” are paths in \(T\).

**Definition 2 (unique-max and conflict-free coloring).** Let \(T\) be a tree, and let
\(\mathcal{P} = \{P_1, \ldots, P_t\}\) be a family of paths \(P_j \subseteq E(T)\). Let \(c : \mathcal{P} \rightarrow [q]\)
be a coloring of the paths. We say that \(c\) is unique-max if for all \(e \in \bigcup_{j=1}^t P_j\) the maximum color
\(i = \max\{c(P_j) \mid P_j \ni e\}\) is such that exactly one path \(P_j \in \mathcal{P}\) with \(e \in P_j\) has color \(i\).
We say that \(c\) is conflict-free if for all \(e \in \bigcup_{j=1}^t P_j\) there is a color \(i \in [q]\) such that
exactly one path \(P_j \ni e\) has color \(i\).

Such a unique-max (or conflict-free) coloring is said to be \(k\)-fault tolerant if it
remains unique-max (or conflict-free) when up to \(k\) paths are removed.

### 1.2. Related works.

The problem of finding a weight function for the network links so that routing protocols will compute a particular desired routing is well studied
in the telecommunication world. A number of papers (e.g., [7, 16, 17]) focus on the
problem of computing a weight function in order to minimize congestion, in networks
ruled by famous protocols like OSPF\(^2\) and IS-IS.\(^3\) The latter protocols route each \(u-v\)
demand on the shortest path from \(u\) to \(v\), differently from protocols like STP, RSTP,
and MSTP, which, as we already discussed, route demands on a tree topology. De Sousa and Soares [15] and Iovanna et al. [8] address the problem of setting weights in
order to achieve stability in networks using MSTP and STP, under the assumption
that no more than one single link fails at a time. This assumption is quite common,
since, as an example, in IP backbone networks roughly 70\% of the unplanned fail-
ures involve no more than one link [9]. In particular, the maximum weight assigned
in [8] is \(O(n^4)\). Grandoni et al. [6] consider the possibility of any number \(k\) of fail-
ures: they give randomized and deterministic algorithms where the maximum weight is \(O((2kn)^{k+1} n^2)\). This quantity is polynomially bounded when \(k = O(1)\), which is
the case in most edge failure situations. However, if we consider also node failure,
the situation becomes different. In fact, a node failure can be treated as the simul-
taneous failure of all incident edges. However, in this case the number \(k\) could be
superconstant, even for the failure of a single node. For arbitrary (not bounded) \(k\),
the maximum weight given in [6] is \(2^{O(n \log n)}\).

Conflict-free and unique-max coloring problems were introduced by Even et al. [4].
They considered many types of regions arising in geometry (e.g., disks, axis-parallel
rectangles, centrally symmetric convex regions). Furthermore, they showed how to
find conflict-free colorings with \(O(\log m)\) colors, where \(m\) is the number of regions,
and proved that the bound is worst-case optimal. Since then, unique-max coloring
problems have received a lot of attention in the literature (see, e.g., [1, 2, 11, 13, 14]),
particularly in the discrete geometry community, and have been generalized to several
different settings such as online algorithms (see, e.g., [2]) and fault-tolerance (see, e.g.,
[1]). In particular, Abam, de Berg, and Poon [1] consider the \(k\)-fault tolerant conflict-
free coloring problem, where the elements are points in the plane and the regions
are disks. They find conflict-free colorings with \(O(k \log m)\) colors, and show that

\(^2\)RFC 1131—OSPF protocol.

\(^3\)ISO/IEC 10589 : 2002—IS-IS protocol.
the bound is the best possible. Bar-Noy et al. [2] define the very general notion of a $q$-degenerate hypergraph and show that for many classes of geometric conflict-free coloring problems, the corresponding hypergraph is $q$-degenerate. Furthermore, they give a randomized online algorithm for computing a conflict-free coloring of $q$-degenerate hypergraphs using $O(q \log m)$ colors with high probability. Note that, although not explicitly stated in their paper, their algorithm actually computes a unique-max coloring.

1.3. Our results. A main conceptual contribution of this paper is a connection between stable weights and unique-max coloring based on consistent assignment of priorities, as sketched in the introduction and elaborated in section 2. This provides an interesting fresh application of the theory of conflict-free/unique-max colorings, which is considerably different than the geometric settings considered so far in this context.

The concrete contributions are twofold. On one hand, we provide stable weight functions that improve some of the best known upper bounds for the maximum weight. On the other hand, we give some new results on conflict-free and unique-max colorings in the setting defined by paths on trees.

We start in section 2 with the results on stable routing. We give an algorithm which computes a $k$-stable weight function for arbitrary (not bounded) $k$ such that the maximum weight is $3^n$, computable in time $O(|E|)$. This improves on the algorithms of Grandoni et al. [6]: namely, one algorithm leading to weights bounded by $2^{O(n \log n)}$, whose running time is not polynomially bounded in $n$, and a further algorithm which leads to weights bounded by $2^{O(n^3)}$, whose running time is $O(|E|)$. We then focus on the case $k = 1$ (i.e., single link failure), and present a 1-stable weight function with maximum weight $O(n \log n)$ using our connection to unique-max coloring, computable in time $O(\log |E| \cdot |E| \cdot n)$. Our bound on the maximum weight improves on the previous bound of $O(n^3)$ [8], although the algorithm in [8] has running time $O(|E|)$. Furthermore, for the case where the root of the shortest path tree chosen by the STP is known in advance, we provide an algorithm whose maximum assigned weight is $4n$, i.e., linear in the number of nodes, in time $O(|E|)$. This is asymptotically best possible and may be of particular interest in practice.

In section 3, we provide our results for unique-max colorings. We prove that we can efficiently unique-max color any set of $m$ paths with $O(\log m)$ colors. In fact, we generalize the result to $k$-fault tolerant unique-max colorings of paths, providing a simple algorithm that uses $O(k \log m)$ colors. We also show that such bounds are asymptotically the best possible.

2. Stable weight functions. We start this section with a simple weight function with maximum weight bounded by $3^n$, which is $k$-stable for all $k$ and is computable in time $O(|E|)$.

**Theorem 1.** Let $G = (V, E)$ be a connected graph, where $V = \{v_0, \ldots, v_{n-1}\}$, and let $T$ be a spanning tree in $G$. Define a weight function $w : E \rightarrow \mathbb{N}$ as follows:

$$w(e) = \begin{cases} 1 & \text{if } e \in E(T), \\ n \cdot (3^i + 3^j) & \text{if } e = \{v_i, v_j\} \notin E(T). \end{cases}$$

Then, for all $k \in \mathbb{N}$, $w$ is a $k$-stable weight function for $T$ and $\max_{e \in E} w(e) \leq 3^n n$.

**Proof.** The upper bound on the weights is clear, so let us prove that $w$ is stable. First of all, the unique shortest path in $G$ between any two vertices is the path of length at most $n - 1$ given by $T$. That means $T$ is the shortest path tree on $G$. 

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.
rooted at any vertex in $V$. Now let $F \subset E$ be an arbitrary set of failed edges. In the following, we always work in the graph $G \setminus F$. Fix a vertex $r \in V$, and let $e = \{u, v\} \in E(T) \setminus F$ be an edge that is reachable from $r$ in $G \setminus F$. We need to show that $e$ must be contained in all shortest path trees rooted at $r$. We refer to the edges $B = E \setminus E(T)$ as backup edges.

Let $P_u$ be a shortest path from $r$ to $u$, and let $P_v$ be a shortest path from $r$ to $v$. As $P_u$ and $P_v$ are simple paths, they touch each vertex at most twice and contain at most $n - 1$ edges in total. This implies that for every vertex $z \in V$, $w(P_u)$ unambiguously encodes how many backup edges incident to $z$ are contained in $P_u$, and similarly for $P_v$. To see this, divide $w(P_u)$ by $n$, throw away any fractional parts, and read the result as a ternary number. Each base 3 digit of this number corresponds to one vertex $z \in V$.

Now assume, for the purpose of obtaining a contradiction, that neither $P_u$ nor $P_v$ contains $e$. Walk backwards from $u$ and $v$, respectively, on both paths until using a backup edge. We must encounter a backup edge $g \in B$ on at least one of the paths before the two paths meet, for otherwise those last segments of the paths together with $e$ would form a cycle in $T$; see Figure 2. Without loss of generality, we encounter the edge $g$ on the path $P_u$. Let $z \in V$ be the endpoint of $g$ that is closer to $u$ on the path $P_u$, i.e., $P_u$ contains exactly one backup edge incident to $z$, while $P_v$ contains no backup edge incident to $z$. From this we deduce

$$|w(P_u \setminus E(T)) - w(P_v \setminus E(T))| \geq n.$$ 

Now $P_u$ never visits $v$ and $P_v$ never visits $u$, for we could otherwise find a shortcut using $e$, contradicting our choice of $P_u$ and $P_v$ as shortest paths. Thus, both paths visit at most $n - 1$ vertices and contain at most $n - 2$ edges of $T$ each. Using the definition of $w$, this implies $|w(P_u) - w(P_v)| \geq 2$. But now we have a contradiction: in the case $w(P_u) \geq w(P_v) + 2$, the path $P_v \cup \{e\}$ is a shorter path from $r$ to $u$ than $P_u$. The case $w(P_v) \geq w(P_u) + 2$ is similar.

To conclude, whenever we have shortest paths from $r$ to both $u$ and $v$, one of those paths must contain $e$, and therefore $e$ is contained in all shortest path trees rooted at $r$. \[ \square \]

From now on, we consider the case where $k = 1$ is fixed. We provide a simple algorithm for finding a 1-stable weight function $w$ such that the maximum weight is $O(n \log n)$.

The idea behind this result is the following. Whenever an edge $e \in E(T)$ fails, the tree $T$ splits into two components. We would like to the weight function $w$ force the STP to select a unique edge $f \notin E(T)$, which replaces $e$ in connecting the two components. We will determine the replacement for each $e \in E(T)$ by solving one suitable unique-max coloring problem. In particular, we use the following theorem, which will be proved in section 3.
Theorem 2. There exists a deterministic algorithm that, for any unique-max coloring instance \((E(T), \mathcal{P})\), computes a unique-max coloring using \(O(\log |\mathcal{P}|)\) colors in time bounded by \(O(\log |\mathcal{P}| \cdot |\mathcal{P}| \cdot n)\).

Using the above result, we derive the following theorem, whose proof exploits a close connection between 1-stable weight functions and unique-max coloring of paths on trees.

Theorem 3. There exists a deterministic algorithm with runtime bounded by \(O(\log |E| \cdot |E| \cdot n)\) that, given a connected graph \(G = (V, E)\) and a spanning tree \(T\) in \(G\), computes a 1-stable weight function \(w\) for \(T\) such that \(\max_{e \in E} w(e) = O(n \cdot \log n)\).

Proof. As before, we refer to the edges \(B = E \setminus E(T)\) as backup edges. Each backup edge \(e \in B\) induces a unique path \(P_e\) between its endpoints in \(T\). Let \(\mathcal{P} = \{P_e \mid e \in B\}\) be the collection of these paths, and compute a unique-max coloring of \((E(T), \mathcal{P})\) according to Theorem 2. Let \(q\) be the number of colors used. Assign weights to edges by

\[
w(e) = \begin{cases} 
1 & \text{if } e \in E(T), \\
i \cdot n & \text{if } e \in B \text{ and the corresponding path } P_e \text{ has color } q - i + 1.
\end{cases}
\]

We claim that \(w\) is 1-stable. In the original graph \(G\), the unique shortest path between any two vertices is the path of length at most \(n - 1\) given by \(T\), so \(T\) is the shortest path tree on \(G\) for any root vertex \(r \in V\).

Suppose now that there is an edge \(f \in E(T)\) that fails. When we remove \(f \in E(T)\), \(T\) splits into two connected components. The unique shortest path between any two vertices in the same connected component is still given by \(E(T) \setminus \{f\}\). Now let \(u, v\) be two vertices from different connected components of \(E(T) \setminus \{f\}\). Every path between \(u\) and \(v\) in \(G \setminus \{f\}\) must use an edge \(e \in B\) whose corresponding path \(P_e\) contains \(f\). We can connect \(u\) and \(v\) using the unique-max colored edge / path among those paths and additional edges of \(E(T) \setminus \{f\}\) in a unique way, giving a unique minimal path connecting \(u\) and \(v\) of cost at most \(i \cdot n + (n - 2)\). Any other \(u-v\) path in \(G \setminus \{f\}\) has cost at least \((i + 1) \cdot n > i \cdot n + (n - 2)\). It follows that, if \(T'\) is the shortest path tree in \(G \setminus \{f\}\) (rooted at any \(r \in V\)), we have \(E(T) \setminus \{f\} \subseteq E(T')\).

We conclude that \(w\) is a 1-stable weight function. Since \(|\mathcal{P}| = |E \setminus E(T)| = O(n^2)\), the number of colors used according to Theorem 2 is \(O(\log n)\), and therefore \(\max_{e \in E} w(e) = O(n \cdot \log n)\).

2.1. Stable weights of linear size for fixed root. The STP selects the root node as follows: each node of the network has a unique fixed ID, and the root is chosen as the node with the lowest ID (see, e.g., [15]). This implies that, whenever a set of edges \(F\) fails, the root node will not change if \(G \setminus F\) is still connected. However, particularly in the case of multiple edge failures that disconnect \(G \setminus F\), the relevant root nodes in each component could change in ways that are difficult to trace. This is why Definition 1 requires that every shortest path tree rooted at any node in \(G \setminus F\) contain the nonfailing edges. With this definition, stability is ensured independently of the IDs of the nodes.

On the other hand, in the case of single link failure, the situation is simpler. In fact, when one link fails, the network will always stay connected unless the failing edge is a bridge. We now show that, assuming that the initial root node is known, there is an efficient algorithm to compute a 1-stable weight function with respect to only that root, where the maximum weight is linear in the number of nodes.

We remark that the linear bound is asymptotically the best possible. To see this, consider a ring network \(G\) that is a cycle with \(n\) nodes. Clearly, there is only one
edge $e = \{u, v\}$ that is not used in a given spanning tree $T$, and indeed $T$ is simply the $u - v$ path $P_{uw}$ in $G$ composed by $n - 1$ edges. Assume that the root is $u$. Then, by definition, any 1-stable weight function is such that $w(e) \geq \sum_{f \in P_{uw}} w(f) + 1 \geq n$, where the last inequality follows since any edge must have a weight $\geq 1$.

We now formally define admissible weights for the 1-stable weight problem with fixed root. In the following definition, we assume that the network is 2-edge-connected. We will discuss the case of a network with bridges later.

**Definition 3 (root-stable weights).** Let $G = (V, E)$ be a 2-edge-connected graph with a fixed root $r \in V$, and let $T$ be a spanning tree in $G$. A function $w : E \rightarrow \mathbb{N}_{\geq 1}$ is a root-stable weight function for $T$ with respect to the root $r$ if the following conditions are satisfied:

(i) $T$ is the unique shortest path tree rooted at $r$ with respect to the weights $w$.

(ii) For each edge $e \in E(T)$ there is a unique shortest path tree $T_e$ with respect to the weights $w$ and with root $r$ in the graph $G \setminus \{e\} = (V, E \setminus \{e\})$. It satisfies $E(T) \setminus \{e\} \subseteq E(T_e)$.

Analogously to the 1-stable weight problem, the root-stable weight problem asks to find a root-stable weight where the maximum assigned weight is as small as possible, given a graph $G$ with root $r$ and spanning tree $T$. A quadruple $(G, r, T, w)$ which is a feasible solution to this problem is called a root-stable network. In the following, we continue to use the notation $T_e$ for the unique shortest path tree in $G \setminus \{e\}$.

Notice that the second condition of root-stable weights does not imply the first one. Consider a graph consisting of a cycle. In this case, there are weight functions with respect to which the shortest path tree rooted at $r$ is not unique; however, once a single edge is removed, there is only one tree left, which is trivially unique. The main result of this section is an algorithm that solves the root-stable weight problem using labels of only linear size.

**Theorem 4.** There is an $O(|E|)$-time algorithm that, given a 2-edge-connected graph $G = (V, E)$ with root $r \in V$ and a spanning tree $T$ of $G$, computes weights $w : E \rightarrow [3n]$ such that $(G, r, T, w)$ is a root-stable network.

In the following, when dealing with a graph $G = (V, E)$ with root $r$ and weights $w$, we denote by $d_T(v)$, for $v \in V$ and $U \subseteq E$, the distance of a shortest path between $r$ and $v$ in the subgraph $(V, U)$, where the lengths are induced by $w$.

We will prove Theorem 4 by presenting an iterative algorithm with the required properties. The algorithm itself is simple to state; the main difficulty lies in its analysis. We will show that at each iteration of the algorithm to be presented, we satisfy the following set of properties, which we call $m$-boundedness, where $m$ depends on the iteration of the algorithm.

**Definition 4 (property ($m$-bounded)).** A root-stable network $(G, r, T, w)$ is called $m$-bounded for some $m \in \mathbb{N}$ if the following properties are satisfied:

(i) $w(e) \leq m$ for all $e \in E$.

(ii) $d_T(v) + d_{T_e}(v) \leq m$ for all $v \in V, e \in E(T)$.

The key idea in designing our algorithm is that an $m$-bounded network $(G = (V, E), r, T, w)$ can be extended to a larger $m'$-bounded network $(G' = (V', E'), r, T', w')$ by adding nodes and edges induced by what we call an ear (see Figure 3). More precisely, the extension is defined as follows: the added nodes $V' \setminus V$ are covered by a simple path (named ear) $U \subseteq E' \setminus E$ with endpoints $a, b \in V$, where we allow $a = b$. The added edges are $U \cup U'$, where $U$ is the set of edges of the ear, and $U'$ contains edges between $V$ and $V' \setminus V$ and edges between vertices in $V' \setminus V$ that are not contained in $U$ (possibly, $U' = \emptyset$).
The next theorem is the key to making this extension work.

**Theorem 5.** Let \((G, r, T, w)\) with \(G = (V, E)\) be an \(m\)-bounded network for some \(m \geq 2n - 2\), and let \(G' = (V', E')\) be an extension of \(G\) with a corresponding ear \(U \subseteq E'\) with endpoints \(a, b \in V\). Let \(f \in U\). Then \((G', r, T', w')\) is \(m'\)-bounded for \(m' = m + 2(|V'| - n) + 1\), where \(E(T') = E(T) \cup (U - f)\), and \(w' : E' \to \mathbb{N}\) is an extension of \(w\) defined by

\[
w'(e) = \begin{cases} w(e) & \text{if } e \in E, \\ 1 & \text{if } e \in U - f, \\ m - d_T(a) - d_T(b) + |V'| - n + 1 & \text{if } e = f, \\ m' & \text{otherwise.} \end{cases}
\]

Notice that the ear used in Theorem 5 contains only edges of \(T\) except for precisely one edge, the edge \(f\). Before proving Theorem 5, we introduce some notation and a result that provides an easy way to test whether a given spanning tree is a shortest path spanning tree. To simplify notation, we use + and − for the addition and deletion of single elements from a set; for example, \(S - i + j\) denotes \((S \setminus \{i\}) \cup \{j\}\).

The following proposition gives an easy-to-test condition which guarantees that some spanning tree is a (unique) shortest path tree in a graph. The proposition follows directly from classical results on potentials and characterizes a shortest path tree by the nonpresence of shortcuts (for more information, see [12]).

**Proposition 1.** Let \(G = (V, E)\) be an undirected graph with root \(r \in V\) and edge weights \(w : E \to \mathbb{R}_{\geq 0}\), and let \(T\) be a spanning tree in \(G\). Then \(T\) is the unique shortest path tree in \(G\) rooted at \(r\) if and only if \(|d_T(u) - d_T(v)| < w(\{u, v\})\) for all \(\{u, v\} \in E \setminus E(T)\).

**Proof of Theorem 5.** We first show that \((G', r, T', w')\) is root-stable, and in a second step we prove the additional properties of \(m'\)-boundedness.

**Root-stability.** To check that \((G', r, T', w')\) is root-stable, we show the second condition of a root-stable tree; i.e., for every \(e \in E(T')\), \(T'_e\) is the unique shortest path tree rooted at \(r\) in \((V', E' - e)\). The condition that \(T'\) is the unique shortest path tree rooted at \(r\) in \(G'\) follows by similar arguments. We distinguish two cases: \(e \in E(T)\) and \(e \in E(T') \setminus E(T)\).

**Case 1:** \(e \in E(T)\). Let \(T_e\) be the unique shortest path tree rooted at \(r\) in \((V, E - e)\). We show that \(\overline{T}\) defined by \(E(\overline{T}) = E(T_e) \cup (E(T') \setminus E(T))\) is the unique shortest
path tree $T'_e$ in $(V', E' - e)$. By Proposition 1, this is equivalent to

\[(1) \quad |d_{T'}(u) - d_{T'}(v)| < w'(\{u, v\}) \quad \forall \{u, v\} \in E' \setminus (E(T) + e).\]

We distinguish three cases: \{u, v\} $\in E \setminus (E(T) + e)$, \{u, v\} $= f$, and \{u, v\} $\in E' \setminus (E(T) \cup E + f)$.

Case 1.a: \{u, v\} $\in E \setminus (E(T) + e)$. Notice that since $T_e$ is a subtree of $T$, we have $d_{T_e}(v) = d_{T'}(v)$ for all $v \in V$. Hence, by stability of $(G, r, T, w)$, condition (1) holds for all edges \{u, v\} $\in E \setminus (E(T) + e)$.

Case 1.b: \{u, v\} $= f$. We have

\[(2) \quad |d_{T'}(u) - d_{T'}(v)| \leq |d_{T'}(a) - d_{T'}(b)| + |V'| - n = |d_{T_e}(a) - d_{T_e}(b)| + |V'| - n,
\]

since the two paths in $T$ from $r$ to $u$ and $r$ to $v$ go first from $r$ to $a$ and from $r$ to $b$, respectively. Let us assume without loss of generality that $d_{T_e}(a) \geq d_{T_e}(b)$. Then using the fact that $d_{T}(a) + d_{T_e}(a) \leq m$ by $m$-boundedness, and $d_{T}(b) \leq d_{T_e}(b)$ by the fact that $T$ is the unique shortest path tree in $G$, we get

\[|d_{T_e}(a) - d_{T_e}(b)| = d_{T_e}(a) - d_{T_e}(b) \leq m - d_{T}(a) - d_{T}(b).\]

Combined with the previous equation and the definition of $w'(f)$, this gives us $|d_{T'}(u) - d_{T'}(v)| < w'(f)$, as desired.

Case 1.c: \{u, v\} $\in E' \setminus (E(T) \cup E + f)$. Since $w'(\{u, v\}) = m'$ by definition, it suffices to see that $d_{T'}(w) < m'$ for all $w \in V'$. For $w \in V$ we have $d_{T'}(w) = d_{T}(w) \leq m - m'$ by $m$-boundedness. For $w \in V' \setminus V$ we have

\[d_{T'}(w) \leq \max\{d_{T_e}(a), d_{T_e}(b)\} + |V'| - n \leq m + |V'| - n < m'.\]

Case 2: $e \in E(T') \setminus E(T)$. Let $T'$ be the tree defined by $E(T') = E(T) \cup (U + f - e)$. Again, we will show that $T'$ is the unique shortest path tree rooted at $r$ in $(V', E' - e)$ by verifying condition (1). Condition (1) clearly holds for all edges \{u, w\} $\in E \setminus E(T)$, since $T$ is the unique shortest path tree rooted at $r$ in $G$. For all other edges \{u, w\} $\in E' \setminus (E \cup E(T))$, we have $w'(\{u, w\}) = m'$. Notice that $d_{T'}(v) < m'$ for all $v \in V'$ because of the following. Since $T$ is a subtree of $T'$, we have $d_{T'}(v) = d_{T}(v) < m < m'$ for all $v \in V$. Furthermore, for $v \in V' \setminus V$, the path from $r$ to $v$ in $T'$ goes through either $a$ or $b$ and uses at most all edges in $U - e$. Hence, we get

\[d_{T'}(v) \leq \max\{d_{T_e}(a), d_{T_e}(b)\} + |V'| - n - 1 + w'(f) \leq m + 2(|V'| - n) < m'.\]

Therefore, $d_{T'}(v) < m'$ for all $v \in V'$, and thus $|d_{T'}(u) - d_{T'}(w)| < m' = w'(\{u, w\})$, as desired.

$m'$-boundedness. By the definition of the weights $w'$, the first condition of $m'$-boundedness clearly holds for $(G', T', r, w')$. For the second condition we fix an edge $e \in E(T')$ to be removed, and we distinguish two cases: $v \in V$ and $v \in V' \setminus V$.

Case 1: $v \in V$. Since $E(T) \subseteq E(T')$ and $E(T_e) \subseteq E(T'_e)$, we have $d_{T'}(v) = d_{T}(v)$ and $d_{T}(v) = d_{T_e}(v)$. By the $m$-boundedness of $(G, T, r, w)$ we obtain $d_{T}(v) + d_{T_e}(v) \leq m$, and therefore $d_{T'}(v) + d_{T_e}(v) \leq m < m'$.

Case 2: $v \in V' \setminus V$. We assume without loss of generality that the path from $r$ to $v$ in $T'$ goes through the vertex $a$. The other case, where the path goes through $b$, is identical. We distinguish $e \in E(T') \cap E = E(T)$ and $e \in E(T') \setminus E$. 

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.
Case 2.a: \( e \in E(T) \). In this case we have \( d_{T'}(v) \leq d_T(a) + |V'| - n \). Furthermore, \( d_{T'}(v) \leq d_T(a) + |V'| - n \), and hence

\[
d_{T'}(v) + d_{T'}(v) \leq d_T(a) + d_{T'}(a) + 2(|V'|-n) \leq m + 2(|V'|-n) < m',
\]

where the second inequality follows by the \( m \)-boundedness of \((G,T,r,w)\).

Case 2.b: \( e \in E(T') \setminus E \). We assume that \( e \) lies on the path from \( r \) to \( v \) in \( T' \), as otherwise the statement follows trivially since then \( d_{T'}(v) = d_{T'}(v) < |V'| < m'/2 \). Let \( P_{rv} \) be the path from \( r \) to \( v \) in \( T' \); hence, \( w'(P_{rv}) = d_{T'}(v) \). \( P_{rv} \) can be partitioned into the path \( P_{ra} \) from \( r \) to \( a \) in \( T' \) and the path \( P_{av} \) from \( a \) to \( v \) in \( T' \). Similarly let \( Q_{rv} \) be the path from \( r \) to \( v \) in \( T'_e \). The path \( Q_{rv} \) can be partitioned into the path \( Q_{rb} \) from \( r \) to \( b \) in \( T'_e \) and \( Q_{bv} \) from \( b \) to \( v \) in \( T'_e \). Hence,

\[
d_{T'}(v) + d_{T'}(v) = w'(P_{ra}) + w'(P_{av}) + w'(Q_{rb}) + w'(Q_{bv}).
\]

Since \( e \in E(T') \setminus E \), we have \( P_{ra} = d_T(a) \) and \( Q_{rb} = d_T(b) \). Furthermore, \( P_{av} \cup Q_{bv} = U \). Since \( U \) contains \(|V'| - n + 1\) edges, all of them except \( f \) having a weight of one, we have \( w'(U) = |V'| - n + w'(f) \). Therefore,

\[
d_{T'}(v) + d_{T'}(v) = d_T(a) + d_T(b) + |V'| - n + w'(f) = m + 2(|V'| - n) + 1 = m'.
\]

Using Theorem 5, we can design an algorithm for obtaining root-stable weights of linear size by starting with a graph consisting of the root, and extending it successively by adding ears as follows (recall that we assume \( G \) to be 2-edge-connected).

RS-WEIGHTS\( (G, r, T) \).
1. Let \( H = (F, W) \) be the graph consisting only of the root \( r \).
2. \( m \leftarrow 0 \)
3. \( \text{while } F \neq V \)
4. \( \quad \text{do} \)
5. \( \quad \quad \text{Find an ear } U \subseteq E \setminus W \text{ with respect to } H \text{ such that } U \setminus E(T) \text{ consists of a single edge } f. \)
6. \( \quad \quad F \leftarrow F \cup A, \text{ where } A \subseteq V \setminus F \text{ are the new vertices covered by } U. \)
7. \( \quad \quad W \leftarrow \{\{u, v\} \in E \mid u, v \in F\}. \)
8. \( \quad \quad \text{Use Theorem 5 with the ear } U \text{ and edge } f \text{ to assign weights } w \text{ to } W. \)
9. \( \quad \quad m \leftarrow m + 2(|V'| - n) + 1. \)

We first observe that if we can always find an ear as claimed in line 4 of the algorithm, then the returned spanning tree is root-stable, and all weights are bounded by \( 3n \), as stated in Theorem 4. The loop invariant that \((H, r, (F, E(T) \cap W), w)\) is \( m \)-bounded at the beginning of every iteration is easily verified using Theorem 5. In particular, \((F, E(T) \cap W)\) is a spanning tree of \( H \). By Theorem 5, the weights \( w \) computed by RS-WEIGHTS are root-stable. Furthermore, since the while-loop has at most \( n \) iterations, the final weights \( w \) are \( m \)-bounded for \( m \leq 3n \).

Thus, to prove Theorem 4, it remains to show the following.

Lemma 1. An ear as described in line 4 of algorithm RS-WEIGHTS can always be found, and the algorithm can be implemented to run in \( O(|E|) \) time.

Proof. We describe the procedure for finding an ear \( U \subseteq E \setminus W \) as requested in line 4 of the algorithm. This will show at the same time that such an ear always exists.

At the beginning of the algorithm, we orient all edges in \( T \) towards the root \( r \). Furthermore, at each step of the algorithm we keep track of the set \( D = \delta(F) \setminus E(T) \). Notice that keeping track of \( D \) can easily be done in \( O(|E|) \) time: whenever a new ear is added to \( H \), we look at all edges adjacent to the newly added vertices and update
their membership in $D$; this way, each edge is considered exactly twice. To find a new ear we choose an arbitrary edge $\{u, v\} \in D$, with $u \in F$, $v \notin F$. Notice that as long as $F \neq V$, the set $D$ cannot be empty since the graph is 2-edge-connected. Let $P \subseteq E$ be the path starting at $v$ and going along the unique directed path in $T$ until the first vertex in $F$ is reached. This is indeed well-defined since $F$ contains the root $r$ and $T$ is a spanning tree directed towards $r$. The next ear is chosen to be $U = P \cup \{\{u, v\}\}$, which clearly satisfies $|U \setminus E(T)| = |\{\{u, v\}\}| = 1$ as required. Consider the time needed to find all ears. Notice that during the construction of the ears, we follow each arc of $T$ precisely once. Hence, the total time needed to perform line 4 of the algorithm is bounded by $O(n)$. Updating $F$, $W$, and $m$ in the loop can clearly be done in a total time bounded by $O(|E|)$. The last remaining step is the assignment of the weights in line 7. This can also clearly be performed in $O(|E|)$ total time, since the only nontrivial terms used in the computation of the weights are $d_T(a)$ and $d_T(b)$, where $a, b$ are the two endpoints of the added ear. However, since $d_T(v)$ for $v \in V$ simply measures the distance of $v$ to the root (in number of edges), all values $d_T(v)$ for $v \in V$ can be computed in $O(n)$ time at the beginning of the algorithm by a simple traversal algorithm starting at the root.

We close this section by discussing what happens in the more general case when the network contains bridges (i.e., edges whose removal disconnects the graph). In that case, we claim that the problem can be solved independently in each 2-edge-connected component, by using the algorithm RS-Weights with a little modification, and then merged. Specifically, the root for each 2-edge-connected component will be the node adjacent to the bridge whose removal disconnects the component from the real root node. In addition, when applying the algorithm to each component, we initialize $m$ by setting $m \leftarrow n$ instead of $m \leftarrow 0$. In this way, all the edges not in the tree $T$ will have a weight $\geq n$. This ensures that when a bridge fails and the root changes for a component $C$, the initial tree $T$ restricted to $C$ is still the unique shortest path tree in $C$ with respect to any root node, and therefore stability is ensured also in case of a failing bridge. Note that this is the only case where the root node may change. Putting all together, we arrive at the following result.

**Theorem 6.** There is an $O(|E|)$-time algorithm that, given any graph $G = (V, E)$ with fixed and known root $r \in V$, and a spanning tree $T$ of $G$, computes weights $w : E \to \{0, 1\}$ such that the routing computed by the STP is stable in case of single link failure.

### 3. \(k\)-fault tolerant unique-max colorings.

The aim of this section is proving Theorem 2. In fact, we prove the following more general theorem, which is equivalent to Theorem 2 for $k = 0$.

**Theorem 7.** There exists a deterministic algorithm that, given an instance $(E(T), \mathcal{P})$ of the unique-max coloring problem and an integer $k \geq 0$, computes a $k$-fault tolerant unique-max coloring of $(E(T), \mathcal{P})$ with $O((k + 1) \log |\mathcal{P}|)$ colors in time bounded by $O((k + 1) \cdot \log |\mathcal{P}| \cdot |\mathcal{P}| \cdot \max\{n, k\})$.

We use the approach via admissible sets; see, e.g., Sariel and Smorodinsky [13] and Abam, de Berg, and Poon [1]. Given any set $\mathcal{P}$ of paths and an edge $e$, we define as a shorthand the set $\mathcal{P}_e := \{P \in \mathcal{P} \mid P \ni e\}$ of paths that contain $e$.

**Definition 5.** A subset $A \subseteq \mathcal{P}$ is called a $k$-admissible set of paths if for every $e \in E(T)$ one of the following holds: (i) $|A_e| \in \{0, 1\}$; (ii) $|(\mathcal{P} \setminus A)_e| \geq k + 1$.

Our algorithm is a simple iterative procedure that at each iteration computes an admissible set $A$, assigns a new color to all paths in $A$, and removes them from $\mathcal{P}$.
UM-COLOR($E(T), \mathcal{P}, k$).

1. $j \leftarrow 1$
2. while $\mathcal{P}$ is not empty
   3. do Compute a $k$-admissible set $A \subseteq \mathcal{P}$.
   4. Color all paths in $A$ using color $j$ and remove them from $\mathcal{P}$.
   5. $j \leftarrow j + 1$.

We are now going to prove that (i) the coloring output of the algorithm is a $k$-fault tolerant unique-max coloring; (ii) there is an efficient way to compute admissible sets such that the number of iterations of the algorithm (i.e., the number of used colors) is $O((k + 1) \log |\mathcal{P}|)$. Part (i) is implied by the next lemma, whose proof is implicitly given by the analogous result for disks in [1]. We give the details here to get a more complete and self-contained presentation.

**Lemma 2.** When UM-COLOR halts, it has computed a $k$-fault tolerant unique-max coloring.

*Proof.* Let $e \in E(T)$, and consider the state of the algorithm after the completion of some iteration of the while-loop. Hence, only some subset of the paths $\mathcal{P}$ are colored so far. Consider all currently colored paths that contain $e$. We are interested in the largest number $k$ such that for each of the largest $k$ colors of paths containing $e$ there is only one path with that color that contains $e$. We denote by $\mu(e)$ the largest value of $k$ for which this holds. In particular, we have that at the end of the algorithm a coloring is $k$-fault tolerant unique-max if and only if $\mu(e) \geq \min\{|\mathcal{P}_e|, k + 1\}$ for all edges $e \in E(T)$.

We claim that the invariant 

$$\mu(e) + |\mathcal{P}_e| \geq \min\{|\mathcal{P}_e^0|, k + 1\} \quad \text{for all } e \in E(T)$$

holds at the beginning and the end of each iteration of the while loop. Here, $\mu(e)$ and $|\mathcal{P}_e|$ refer to the current state of the coloring and the currently remaining uncolored paths, respectively, while $|\mathcal{P}_e^0|$ is the size of the set of all paths containing $e$, including both colored and uncolored paths.

Initially, the invariant is trivially true. To argue that the invariant is maintained in each iteration, we distinguish the two cases of Definition 5 for each edge $e \in E(T)$.

1. When $|A_e| = 0$, neither $\mu(e)$ nor $|\mathcal{P}_e|$ changes, so the invariant stays true. Otherwise if $|A_e| = 1$, the current $j$ becomes the new largest color on $e$. It is unique, and so $\mu(e)$ increases by one, while $|\mathcal{P}_e|$ is decreased by one.

2. When $|P_e \setminus A_e| \geq k + 1$, then clearly even after removing the paths in $A$ we have $\mu(e) + |P_e| \geq |\mathcal{P}_e| \geq k + 1 \geq \min\{|\mathcal{P}_e^0|, k + 1\}$; that is, the invariant holds.

The algorithm stops when $\mathcal{P}$ is empty, so the invariant then implies that $\mu(e) \geq \min\{|\mathcal{P}_e^0|, k + 1\}$, which in turn implies that we obtained a $k$-fault tolerant unique-max coloring.

The crucial part is how to find admissible sets that are big enough in each iteration. Using the next lemmas, we will show that one can efficiently find admissible sets of size $|\mathcal{P}|/(2k + 3)$. We would like to mention that for simplicity we did not try to heavily optimize the running time of the following procedure to find admissible sets.

**Lemma 3.** Let $T$ be a tree, let $\mathcal{P}$ be a system of paths in $T$, and let $k \geq 0$. There exists a coloring of $\mathcal{P}$ using $(2k + 3)$ colors such that every edge $e \in E(T)$ is covered by at least $\min\{|\mathcal{P}_e|, k + 2\}$ colors. Furthermore, such a coloring can be computed in time $O(|\mathcal{P}| \cdot \max\{n, k\})$.

*Proof.* To avoid confusion, we remark that the coloring we are looking for in this lemma is an auxiliary coloring which is not required to be conflict-free. We assign
colors greedily. Throughout our algorithm, we maintain the set \( \mathcal{P}' \) of paths that have been colored so far, and a satisfied subtree \( S \) with \( E(S) \subseteq E(T) \); i.e., every edge \( e \in E(S) \) is covered by at least \( \min\{|\mathcal{P}_e|, k+2\} \) colors that have been assigned so far. Denote by \( N \subseteq E(T) \setminus E(S) \) the set of neighbor edges, i.e., edges that share exactly one vertex with \( S \). Initially, \( E(S) \) is empty; for this case, we designate an arbitrary root vertex \( r \) as part of the tree and define \( N \) to be its set \( \delta(r) \) of incident edges. We maintain the following invariants: (i) \( S \) is a satisfied subtree with \( E(S) \subseteq E(T) \), (ii) every colored path \( P \in \mathcal{P}' \) has at least one vertex in common with \( S \), and (iii) for all \( e \in N \) the set \( \mathcal{P}_e' \) of colored paths leaving \( S \) through \( e \) is of size at most \( k+1 \), with no two paths in the set having the same color.

The last two invariants together imply that the colored subtrees hanging off edges \( e \in N \) have a simple structure: they arise from the union of at most \( k+1 \) paths with pairwise different colors that—when restricted to the subtree—start at the endpoint of \( e \).

The algorithm is as follows. In each iteration, we choose an arbitrary edge \( e \in N \). Since \( e \) is not satisfied yet, we can choose an uncolored path \( P \) through \( e \). Consider which colors we can assign to \( P \) without violating any invariants. The intersection of \( P \) with \( S \) does not limit the choice of colors, since the edges of \( S \) are already satisfied. Outside of \( S \), \( P \) intersects at most two colored subtrees hanging off \( e \) and another edge \( e' \in N \). As seen above, at most \( 2k+2 \) colors appear in those subtrees in total; we assign the remaining color to \( P \). Finally, we grow the tree \( S \) of satisfied edges if possible, to ensure that the first part of the last invariant is maintained. Note that there may be satisfied edges \( f \in E(T) \) which cannot be added to \( S \) because there are unsatisfied edges between \( f \) and \( S \).

This entire procedure is repeated until \( E(S) = E(T) \). Any remaining uncolored paths can be colored arbitrarily. Since all edges are satisfied in the end, we obtain a coloring of the desired form.

It remains to discuss the running time. Given an edge \( e \in N \), we have to find an uncolored path \( P \) containing \( e \). One simple way to do this is to save together with every edge \( f \in E(T) \) a doubly linked list of all uncolored paths going through \( f \). Setting up these lists takes \( O(n|\mathcal{P}|) \) time. Whenever a path \( P \) is colored, we remove the path from all lists corresponding to edges on the path. This can be done in \( O(|\mathcal{P}|) \) time which is bounded by \( O(n) \), by storing for each path pointers to the occurrences of the corresponding path in the doubly linked lists. Since we color \( |\mathcal{P}| \) paths in total, the total time for updating the doubly linked lists is bounded by \( O(|\mathcal{P}|n) \). Given an uncolored path \( P \) that goes through \( e \), we can find the other edge \( e' \in P \cap N \) (i.e., \( e' \neq e \)) simply by checking all edges on \( P \) in \( O(n) \) time. Checking which of the \( 2k+3 \) colors is missing at \( e \) and \( e' \) can be done in \( O(k) \) time by simply maintaining a Boolean array with \( 2k+3 \) entries for each edge, which tracks by which colors the corresponding edge is already covered. Furthermore, for each edge \( f \in E(T) \) we can save the number of different colors used so far. This allows us to determine in constant time whether \( f \) is satisfied. Hence, after each newly colored path \( P \) we can check in \( O(n) \) time for all edges on \( P \) whether they are satisfied and update \( S \) accordingly. Similarly, we update after each iteration the set \( N \) in \( O(n) \) time. Hence, the total running time of the algorithm is indeed bounded by \( O(|\mathcal{P}| \max\{n, k\}) \).

**Lemma 4.** Every color class \( A \) of the coloring of Lemma 3 is a \( k \)-admissible set of paths.

**Proof.** Let \( e \in E(T) \). If \( |\mathcal{P}_e| \geq k+2 \), then \( e \) is covered by \( k+2 \) colors. This means that \( P \setminus A \) contains \( k+1 \) colors covering \( e \), and thus condition (ii) of Definition 5...
holds. Otherwise, \( e \) is covered by exactly \( |P_\varepsilon| \) different colors, which means that colors of the paths covering \( e \) are pairwise different. Therefore, \( |A_\varepsilon| \leq 1 \), so condition (i) of Definition 5 is satisfied.

Theorem 7 now follows as a rather direct consequence.

Proof of Theorem 7. To obtain a \( k \)-fault tolerant unique-max coloring we rely on algorithm \( \text{UM-Color} \). In each iteration of \( \text{UM-Color} \), we compute an auxiliary coloring according to the algorithm of Lemma 3. We then choose the largest color class in this auxiliary coloring as the admissible set \( A \), giving us an admissible set of size at least \( |P|/(2k+3) \). A simple calculation shows that the number of iterations of \( \text{UM-Color} \) is then bounded by \( O((k+1) \log |P|) \). The bottleneck of the algorithm is the application of Lemma 3 in each iteration, which leads to the claimed running time of \( O((k+1) \cdot \log |P| \cdot |P| \cdot \max(n,k)) \).

We also have a matching lower bound on the number of colors needed up to a constant factor.

Theorem 8. There is a family of instances \((E(T), P)\), where every \( k \)-fault tolerant conflict-free coloring uses at least \( \Omega((k+1) \log |P|) \) colors.

Proof. We construct a family of instances \((E(T), P)\), where every \( k \)-fault tolerant conflict-free coloring uses at least \( \Omega((k+1) \log |P|) \) colors. Let \( T^{(q)} \) denote a complete binary tree of height \( q \), with an additional leaf \( v \) hanging off the root by edge \( e \); see Figure 4. Let \( P^{(q)} \) be the system of \((k+1) \cdot 2^q\) paths in \( T^{(q)} \) formed by considering each leaf of the tree and connecting it to \( v \) by \( k+1 \) paths. (Here, we admit including the same path \( k+1 \) times, but this can be trivially avoided by splitting vertex \( v \) into \( k+1 \) copies.)

Note that \( \log |P| = q + \log(k+1) = O(q) \), assuming \( 2^q > k \).

We now claim the following.

Claim. Every \( k \)-fault tolerant conflict-free coloring of \((E(T^{(q)}), P^{(q)})\) has at least \( \frac{(k+1)^2}{2} \cdot (q+1) \) colors.

The case \( q = 0 \) is clear. We proceed by induction. Consider a \( k \)-fault tolerant conflict-free coloring \( c \) of \((E(T^{(q)}), P^{(q)})\). Since every path contains the edge \( e \), there are \( k+1 \) paths in \( P \) which have a unique color \( i_1, i_2, \ldots, i_{k+1} \). At least half of these paths connect \( v \) to leaves in one of the two subtrees “below” the root \( r \). The other subtree, together with the root, forms a tree \( T^{(q-1)} \), and \( c \) induces a \( k \)-fault tolerant conflict-free coloring \( c' \) of \((E(T^{(q-1)}), P^{(q-1)})\) in this subtree. Half of the colors \( i_1, i_2, \ldots, i_{k+1} \) are not used in \( c' \), and by induction hypothesis, \( c' \) has at least \( \frac{(k+1)^2}{2} \cdot q \) colors. Therefore, \( c \) has at least \( \frac{(k+1)^2}{2} \cdot (q+1) \) colors.

4. Conclusions and final remarks. In this paper we studied the problem of designing \( k \)-stable weight functions for networks ruled by routing protocols like the STP, while minimizing the maximum assigned weight.

For \( k = 1 \) we gave a bound of \( O(n \log n) \) on the maximum weight, by reducing this problem to an instance of the unique-max coloring problem defined by a set of
paths on a tree. We do not know of any simpler way to prove the same result without relying on this new connection. This gives a new application for the theory of unique-max colorings, which so far has been studied mainly in purely geometric settings. Unfortunately, we are not able to exploit this connection to improve the best known bounds for the case $k > 1$. In fact, using Theorem 7, one can design weight functions bounded by $O(kn \log n)$ that are stable whenever one edge of the tree and additionally (up to) $k - 1$ backup edges simultaneously fail. The difficult part is dealing with the simultaneous failure of more than one edge of the initial tree. Note that, while our results on unique-max coloring are asymptotically tight, we do not know whether the corresponding results for stable weight functions are tight as well. In particular, the best lower bound on the maximum weight we can prove is $\Omega(n)$.

For the case of a fixed root, we showed how to obtain a 1-stable weight function with an $O(n)$ bound on the maximum weight, matching the previously mentioned lower bound. However, as in the previous case, the extension to the case $k > 1$ seems to be nontrivial.

Furthermore, we improved the previous best known upper bound on the maximum weight when $k$ is unbounded. We remark that this case is interesting since it captures the node failure case. In fact, a node failure can be modeled as the simultaneous failure of all its incident edges. Hence, even the failure of a single node may therefore correspond to the failure of a superconstant number of edges. On the other hand, also in this case our bound $O(3^n n)$ is far from being tight, since we do not have any nontrivial lower bound better than $\Omega(n)$. This is one of the main questions that are left open.

Acknowledgments. We would like to thank the anonymous referees, whose comments and suggestions helped to considerably improve this paper.

REFERENCES


