Compatibility between Three Well-known Broadcast Tree Construction Algorithms and Various Metrics

Chuan Han, Student Member, IEEE, and Yaling Yang, Member, IEEE

Abstract—Broadcast routing is a critical component in the routing design. While there are plenty of routing metrics and broadcast routing schemes in current literature, it remains an unsolved problem as to which metrics are compatible with a specific broadcast routing scheme. In particular, in the wireless broadcast routing context where transmission has an inherent broadcast property, there is a potential danger of incompatible combination of broadcast routing algorithms and metrics. This paper shows that different broadcast routing algorithms have different requirements on the properties of broadcast routing metrics. The metric properties for broadcast routing algorithms in both undirected network topologies and directed network topologies are developed and proved. They are successfully used to verify the compatibility between broadcast routing metrics and broadcast routing algorithms.

Index Terms—Broadcast routing, Routing protocols, Routing metric design.

1 INTRODUCTION

Broadcast routing is a critical component of network administration and management. It delivers and updates various network control information. Without effective broadcast routing, the whole network may degrade into chaos. Effective broadcast routing is especially critical in wireless networks due to the high cost of broadcast, the unreliable wireless channels and the limited available resource in wireless networks. Therefore, designing effective broadcast routing protocols is an important part in routing research and design.

The definition of “effective” broadcast routing, however, varies according to the application scenarios. In an energy-critical sensor network, broadcast routing may need to minimize energy consumption [1], [2], [3] or maximize network lifetime [4], [5], [6]. In an information-sensitive military network, broadcast routing may need to guarantee timely [7] and secure message delivery [8]. In many other application scenarios, multiple performance factors, such as delay, packet loss rate, bandwidth, etc., may need to be jointly considered for effective broadcast routing [9]. All these different performance requirements for broadcast routing are usually reflected in the design of routing metrics [10], [11], which guide the broadcast tree calculation algorithms to prefer one broadcast tree over the others.

While there are a lot of possible ways to design broadcast routing metrics, existing algorithms may not find the optimal broadcast trees (OBTs) for some of these routing metrics. For example, a common metric capturing the total broadcast routing energy consumption [12] is

\[ w(T) = \sum_{i \in N_i(T)} \max_{(i,j) \in E(T)} \varepsilon_{ij}, \]  

where \( N_i(T) \) is the set of transmitting nodes in the broadcast tree \( T \), \( E(T) \) is the set of links in \( T \), and \( \varepsilon_{ij} \) is the minimum energy consumption of transmitting a packet from node \( i \) to node \( j \). While the metric definition in (1) is simple, finding the OBT based on this definition is very challenging. Consider the undirected network topology \( G \) consisting of four nodes \( r, i, j \) and \( k \) in Fig.1 (I), where the source node \( r \) of a broadcast session is marked as the black node, and the number associated with each edge is the \( \varepsilon_{ij} \) in (1). By definition (1), the minimum energy broadcast tree is \( T_1 = \{ri, rj, rk\} \) in Fig.1 (II), whose total energy consumption is 3.5. However, following Prim’s algorithm [13], which is a well-known algorithm for calculating the minimum spanning tree (MST) for undirected graphs, the broadcast tree is \( T_2 = \{ri, ij, jk\} \) in Fig.1 (III), whose total energy consumption is 4.1. Hence, the combination of Prim’s algorithm and the broadcast routing metric (1) fails to find the optimal broadcast routing tree.

The above example shows that a well-known broad-
cast tree construction (BTC) algorithms may not find the OBT in a network. This observation leads to two fundamental questions we seek to answer in this paper. First, since well-known BTC algorithms may not be optimal for all routing metrics, what are the conditions for them to be optimal? Second, since distributed broadcast routing is desired in practical cases and a network generally has more than one OBTs, can these BTC algorithms create inconsistent routing decisions when they are implemented distributedly in a network? These questions are critical for the design of broadcast routing. Unfortunately, while similar questions have been answered for unicast routing [14], [15], [16], [17], the answers to these questions in broadcast routing are not clearly discussed.

In this paper, we fill this critical void. The objective of our work is to provide important compatibility check for the optimality and consistency of several well-known BTC algorithms under various routing metrics. The well-known BTC algorithms discussed in this paper are Prim’s algorithm [13], [18], the generic edge-adding algorithm [13], [18], and Edmonds’ algorithm [19]. Our unique contribution is two-fold. First, using a unique algebra model of broadcast routing, we identify the necessary and sufficient routing metric properties for BTC algorithms to find OBTs. Second, we identify the conditions for BTC algorithms to find consistent and unique broadcast trees and discuss how this affects distributed broadcast routing protocol design.

The notation used in this paper is summarized in Table 1. The remaining part of this paper is organized as follows. In Section 2, the broadcast routing algebra is introduced. Sections 3 and 4 provide the necessary and sufficient metric properties required by broadcast routing for undirected network topologies and directed network topologies, respectively. A distributed broadcast routing protocol and some related properties are discussed in Section 5. The applications of the derived broadcast routing metric properties are presented in Section 6. Finally, Section 7 concludes this paper.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$</td>
<td>Equivalent to $\Sigma$, $\preceq$, $\oplus$, $\ominus$, $w(\cdot)$</td>
</tr>
<tr>
<td>$A^- (n)$</td>
<td>The set of arcs that terminate at node $n$</td>
</tr>
<tr>
<td>$E(G)$</td>
<td>The edge set of graph $G$</td>
</tr>
<tr>
<td>$G = (N(G), E(G))$</td>
<td>The graph whose node set is $N(G)$ and edge set is $E(G)$</td>
</tr>
<tr>
<td>$G_1 \ominus G_2$</td>
<td>Remove the common edges of network topologies $G_1$ and $G_2$ from $G_1$</td>
</tr>
<tr>
<td>$G_1 \oplus G_2$</td>
<td>Merge of two network topologies $G_1$ and $G_2$</td>
</tr>
<tr>
<td>$\delta(G)$</td>
<td>The edge cut of node set $S$</td>
</tr>
<tr>
<td>$\preceq$</td>
<td>Lighter than</td>
</tr>
<tr>
<td>$\preceq^*$</td>
<td>Lighter than or equivalent to</td>
</tr>
<tr>
<td>$\succ$</td>
<td>Heavier than</td>
</tr>
<tr>
<td>$\preceq^*$</td>
<td>The arc emanating from node $i$ and terminating at node $j$</td>
</tr>
<tr>
<td>$d^- (n)$</td>
<td>The indegree of node $n$</td>
</tr>
<tr>
<td>$e_{ij}$</td>
<td>The edge whose two end nodes are $i$ and $j$</td>
</tr>
<tr>
<td>$r$</td>
<td>The root node of graph</td>
</tr>
<tr>
<td>$w(T)$</td>
<td>The weight of network topology $T$</td>
</tr>
<tr>
<td>$</td>
<td>S</td>
</tr>
</tbody>
</table>

2.1 Definition

Definition 1: The broadcast routing algebra is defined as

$$A = (\Sigma, \preceq, \oplus, \ominus, w(\cdot)), \quad (2)$$

where

- $\Sigma$ is the set of signatures describing the characteristics of all the subgraphs of an original graph $G$. These characteristics may include each link’s capacity, energy consumption, etc..
- The symbol $\preceq$ is the preference order operator, where $w(T_1) \preceq w(T_2)$ indicates that topology $T_1$ is better than or equivalent to topology $T_2$ under weight function $w(\cdot)$.
- The symbol $\oplus$ denotes the operator that joins two network topologies. For any edge $e_1 \in E(G_1)$ and node $n_1 \in N(G_1)$ in topology $G_1$, and any edge $e_2 \in E(G_2)$ and node $n_2 \in N(G_2)$ in topology $G_2$, we have $e_1, e_2$ in the edge set $E(G_1 \oplus G_2)$ and $n_1, n_2$ in the node set $N(G_1 \oplus G_2)$.
- The symbol $\ominus$ denotes the operator that removes the common edges of the left and the right operand topologies from the left operand topology. For any edge $e_1 \in E(G_1), e_1 \notin E(G_2)$, and node $n_1 \in N(G_1)$ in topology $G_1$, we have $e_1$ in the edge set $E(G_1 \ominus G_2)$ and $n_1$ in the node set $N(G_1 \ominus G_2)$.
- The symbol $w(\cdot)$ denotes the weight function over the signature of network topologies. With this routing algebra, any routing metric is mathematically represented by the weight function $w(\cdot)$.

We further define $w(a) \prec w(b)$ as $w(a) \preceq w(b)$ and $w(a) \neq w(b)$, and let $w(a) \succ w(b)$ mean $w(b) < w(a)$. Any BTC algorithm essentially builds a subgraph using the $\oplus$ and/or $\ominus$ operator. With the broadcast routing algebra, BTC algorithms can be expressed in an algebraic manner.
3 BROADCAST ROUTING IN UNDIRECTED NETWORK TOPOLOGIES

In our analysis of BTC algorithms’ requirements on routing metrics, there are two different models of the underlying networks: the directed graph model and the undirected graph model. The undirected graph model is appropriate if all the links in the network are bidirectional and the two directions have the same signatures (a.k.a. characteristics). The directed graph model is used to capture more complicated cases where there are asymmetric links. Different BTC algorithms need to be used for undirected and directed network topologies, and these algorithms have different requirements on routing metric design. In this section, we focus on BTC algorithms for undirected network topologies. In the next section, we study BTC algorithms for directed network topologies. Without loss of generality, we formulate the problem of optimal broadcast routing in undirected network topologies as the minimum spanning tree (MST) problem for undirected graphs [20]. In the remainder of this section, we develop and prove the necessary and sufficient metric properties for which Prim’s algorithm and the generic edge-adding algorithm guarantee optimality. Prim’s algorithm is based on subtrees, and the generic edge-adding algorithm is based on subforests. Many well-known MST algorithms, such as Kruskal’s algorithm [13], Boruvka’s algorithm [18], and the GHS algorithm [21], etc., are special cases of the generic edge-adding algorithm.

3.1 Prim’s Algorithm

3.1.1 Algorithm Overview

Prim’s algorithm starts by treating the root node \( r \), which is the broadcast source node, as the initial partial spanning tree \( T_r \). Then it progressively grows the partial spanning tree \( T_r \) by adding the best edge from the edge cut of the current \( T_r \), until \( T_r \) spans the entire graph. The calculation of the best edge \( e^* \) can be generalized based on the binary operation \( \oplus \) and the order relation \( \preceq \) as follows:

\[
e^* = \arg \min_{e \in \partial(T_r)} \{ w(T_r \oplus e) \}, \tag{3}
\]

where \( \partial(T_r) \) is the edge cut of \( T_r \) [22]. Here, the edge cut \( \partial(T_r) \) is the set of edges with one end in \( N(T_r) \) and the other end in \( N(G) - N(T_r) \). Note that in most textbooks, the original Prim’s algorithm is only discussed with linear metrics based on linear link weight aggregation. The generalized form of Prim’s algorithm with minimum weight edge in (3), however, can cover both linear and non-linear metric designs.

3.1.2 Metric Properties and Proof

The required routing metric property for the generalized Prim’s algorithm can be expressed by the following definition.

**Definition 2**: Right \( \oplus \)-isotonicity for trees: A weight function \( w(\cdot) \) of trees is said to be right \( \oplus \)-isotonic for trees

2.2 Properties

The broadcast routing algebra has the following properties:

- \( \Sigma \) is closed under \( \oplus \): \( a \oplus b \in \Sigma \) for any \( a, b \in \Sigma \);
- \( \Sigma \) is closed under \( \ominus \): \( a \ominus b \in \Sigma \) for any \( a, b \in \Sigma \);
- \( \preceq \) is complete: for any \( a, b, c \in \Sigma \), either \( w(a) \preceq w(b) \) or \( w(b) \preceq w(a) \) (or both);
- \( \preceq \) is transitive: for \( a, b, c \in \Sigma \), if \( w(a) \preceq w(b) \), \( w(b) \preceq w(c) \), then \( w(a) \preceq w(c) \);
- \( \oplus \) is idempotent: \( a \oplus a = a \) for any \( a \in \Sigma \);
- \( \oplus \) is commutative: \( a \oplus b = b \oplus a \) for any \( a, b \in \Sigma \);
- \( \oplus \) is associative: \( (a \oplus b) \oplus c = a \oplus (b \oplus c) \) for any \( a, b, c \in \Sigma \);
- \( \ominus \) is non-commutative: \( a \ominus b \neq b \ominus a \) for some \( a, b \in \Sigma \);
- \( \ominus \) is only left-associative but non-associative: \( a \ominus (b \ominus c) = (a \ominus b) \ominus c \) for any \( a, b, c \in \Sigma \), but \( (a \ominus b) \ominus c \neq a \ominus (b \ominus c) \) for some \( a, b, c \in \Sigma \).
over a graph $G$ if for any tree $T \subset G$ generated by Prim’s algorithm, we have
\[ w(T \oplus e) \leq w(T \oplus e') \Rightarrow w(T \oplus e \oplus F) \leq w(T \oplus e' \oplus F), \quad (4) \]
for any edge $e, e' \in \partial(T)$ and forest $F$ satisfying $E(F) \cap \left(E(T) \cup \partial(T)\right) = \emptyset$ such that both $T \oplus e \oplus F$ and $T \oplus e' \oplus F$ are still trees.

Essentially, the right $\oplus$-isotonicity for trees specifies that the preference order $\preceq$ between the two trees that are expanded from a tree generated by Prim’s algorithm remains unchanged, when they are further expanded into larger trees through the same set of additional edges.

**Theorem 1:** Given any connected and undirected network topology $G$ whose root node is $r$, Prim’s algorithm produces the MST, if and only if the broadcast routing metric $w(\cdot)$ is right $\oplus$-isotonic for trees.

**Proof:** 
**Sufficient condition:** Let $T_p$ be the Prim tree that is generated by Prim’s algorithm. Denote $e_1, \ldots, e_{|N(G)|-1}$ as the order in which Prim’s algorithm selects edges, where $|N(G)|$ is the total number of nodes in graph $G$. We next prove that if the broadcast routing metric satisfies the property in (4), Prim’s algorithm produces a MST. This can be proved by contradiction.

Suppose $T_p$ is not a MST. Following the order $e_1, \ldots, e_{|N(G)|-1}$, compare each edge in $E(T_p)$ with edges of a MST. Denote the MST that shares the largest number of consecutive common edges with $T_p$, as $T^*$ and the first edge that $T_p$ differs with $T^*$ as $e_i$. By Prim’s algorithm, the edge set $\{e_1, e_2, \ldots, e_{i-1}\}$ is a tree. Denote this tree as $T$ as shown in Fig.3 (I). Tree $T$ is a subgraph for both $T_p$ and $T^*$. Consider adding edge $e_i$ to $T^*$ as shown in Fig.3 (II). Then, there must exist a cycle containing $e_i$ and within the cycle there exists an edge $f \in \partial(T)$, $f \in T^*$, $f \neq e_i$. By Prim’s algorithm, it follows $w(T \oplus e_i) \leq w(T \oplus f)$. Substituting $f$ with $e_i$, the resulting subgraph $T^{**} = T^* \oplus e_i \oplus f$ in Fig.3 (III) is still a spanning tree. Since $w(T \oplus e_i) \leq w(T \oplus f)$, by the property in (4), it follows that $w(T^{**}) = T^* \oplus e_i \oplus f = w(T \oplus e_i \oplus f \oplus T \oplus e_i \oplus f) \leq w(T \oplus f \oplus T \oplus e_i \oplus f) = w(T^* \oplus f \oplus T \oplus e_i) = w(T^*)$. Since $T^*$ is a MST, $T^{**}$ is also a MST. The fact that $T^{**}$ has one more common edge $e_i$ with $T_p$ than $T^*$ contradicts the definition of $T^*$. Hence, $T_p$ is also a MST.

**Necessary condition:** We need to prove that if Prim’s algorithm produces the MST on any network topology, then the metric satisfies the property in (4). This can be proved by showing that its contrapositive is correct, i.e., if a metric does not satisfy the property in (4), then there is at least one network topology for which the algorithm does not guarantee optimality.

Consider the network topology $G$ in Fig.4 (I). Suppose $w(r \oplus e_1) \leq w(r \oplus e_2)$ and the metric does not satisfy the property in (4). Note that $r \oplus e_1 \oplus e_3$ and $r \oplus e_2 \oplus e_3$ are trees, as shown in Fig.4 (II) and (III), respectively. Since for the given metric the property in (4) does not hold, it is possible that $w(r \oplus e_1 \oplus e_3) > w(r \oplus e_2 \oplus e_3)$. Meanwhile, the tree produced by Prim’s algorithm is $T_p = r \oplus e_1 \oplus e_3$ which is not the MST. Hence, for the given metric, Prim’s algorithm does not produce the MST for this particular network topology. Therefore, for Prim’s algorithm to guarantee optimality on any network topology, the metric must satisfy the property in (4).

Note that the property in (4) is different from the consistency property in [23]. The consistency property in [23] is only the sufficient but not necessary condition to guarantee that Prim’s algorithm produces the MST, while the property in (4) is both sufficient and necessary. This can be shown by the tree depth example in Section 6.1.1.

### 3.2 Generic Edge-adding Algorithm
There is a generic edge-adding algorithm on undirected graphs [13, 18]. It covers typical forest-based MST algorithms, e.g., Kruskal’s algorithm, Boruvka’s algorithm, and the GHS algorithm, etc. These algorithms only differ in their edge-adding orders.

#### 3.2.1 Algorithm Overview
The generic edge-adding algorithm starts by initializing a forest $F$ as $F = F_0 = (N(G), \emptyset)$, where $\emptyset$ is the empty set. In each of the following steps, it picks one edge $e^*$ to add to $F$ until $F$ is a spanning tree of $G$. The edge $e^*$ is determined by first finding a node set $S$ such that $F$ does not have any edge that belongs to the edge cut of $S$. Here, the edge cut of $S$ is the set of edges in $G$ with one end in $S$ and the other end in $N(G) - S$. The edge $e^*$ is then chosen from the edge cut of $S$ [22], denoted as $\partial(S)$, as follows:

\[
e^* = \arg\min_{e \in \partial(S)} \{w(F \oplus e)\}, \quad (5)\]

where $\partial(S) \cap E(F) = \emptyset$ and $S \subset N(G)$.
The edge-adding process can be illustrated by an example. Consider the undirected graph $G$ in Fig. 5 (I) and the initial forest $F_0 = (N(G), \emptyset)$. In the first step shown in Fig. 5 (II), $e_1$ is chosen from $\partial(S_1)$, i.e., the edge cut of the node set $S_1$. This results in a forest $F = F_0 \oplus e_1$. Note that in the next step, the node set $S_2$ shown in Fig. 5 (III) cannot be chosen since $\partial(S_2) \cap E(F_0 \oplus e_1) = \{e_1\} \neq \emptyset$. Instead, one can choose the node set $S_2$ and pick edge $e_2$ within $\partial(S_2)$ as shown in Fig. 5 (IV). This process continues until the resulting forest is a spanning tree of $G$.

3.2.2 Metric Properties and Proof

The required routing metric property for the generic edge-adding algorithm can be expressed by the following definition.

**Definition 3:** Right $\oplus$-isotonicity for forests: A weight function $w(\cdot)$ of forests is said to be right $\oplus$-isotonic for forests over a graph $G$ if for any forest $F \subseteq G$ generated by the generic edge-adding algorithm, given any node set $S \subseteq N(G)$ satisfying $\partial(S) \cap E(F) = \emptyset$, we have

$$w(F \oplus e) \leq w(F \oplus e') \quad \text{and} \quad w(F \oplus e \oplus F') \leq w(F \oplus e' \oplus F'), \quad (6)$$

for any edge $e$, $e' \in \partial(S)$ and forest $F' \subseteq G$ such that $F \oplus e \oplus F'$ and $F \oplus e' \oplus F'$ are forests.

Essentially, the right $\oplus$-isotonicity for forests specifies that the preference order $\preceq$ between the two forests that are expanded from a forest generated by the generic edge-adding algorithm remains unchanged, when they are further expanded into larger forests through adding a common set of edges.

**Theorem 2:** Given any connected and undirected network topology $G$ whose root node is $r$, the generic edge-adding algorithm produces the MST, if and only if the broadcast routing metric $w(\cdot)$ is right $\oplus$-isotonic for forests.

**Proof:** Sufficient condition: The sufficient condition proof is similar to that of Theorem 1. It is based on contradiction. The only difference is that the consecutive common edge set forms a forest rather than a tree. To save space, we omit the proof details in this paper. The full proof can be found in our technical report [24].

Necessary condition: Similar to the necessary condition proof of Theorem 1, we prove the necessity by showing that its contrapositive is correct, i.e., if there is at least one metric that does not satisfy the property in (6), there is at least one network topology for which the spanning tree generated by the generic edge-adding algorithm is not a MST.

Suppose, for a network topology, $T_a$ is the spanning tree generated by the generic edge-adding algorithm, as shown in Fig. 6 (I). By the generic edge-adding algorithm, it follows that $w(F \oplus e_1) \preceq w(F \oplus e_2)$, $e_1, e_2 \in \partial(S)$, $\partial(S) \cap E(F) = \emptyset$, where $F$ is the partially generated forest before adding edge $e_1$ and $S \subseteq N(G)$ is the selected node set before adding edge $e_1$. Let $F' = T_a \oplus e_1 \oplus F$ as shown in Fig. 6 (II). Since for the given metric the property in (6) is not guaranteed, it is possible that $w(T_a) = w(F \oplus e_1 \oplus F') \succ w(F \oplus e_2 \oplus F')$ as shown in Fig. 6 (III). That is $T_a$ cannot be the MST of the original network topology. Hence, if $T_a$ is a MST, then the metric satisfies the property in (6).

Note that the above proof can be applied to any specific order of adding edges. Therefore, Theorem 2 also holds for typical edge-adding algorithms, e.g., Kruskal’s algorithm, Boruvka’s algorithm and the GHS algorithm.

4 Broadcast Routing in Directed Network Topologies

In a directed network topology, the link from node $n_1$ to node $n_2$ may have different characteristics compared to the link from node $n_2$ to node $n_1$. The problem of optimal broadcast routing in directed network topologies can be formulated as the minimum weight spanning $r$-arborescence problem for directed graphs. We first define the $r$-arborescence. For more details about arborescence, readers are referred to [18].

**Definition 4:** A $r$-arborescence is a directed graph in which node $r$ is called the root node, and there exists exactly only one directed path from the root node $r$ to any other non-root node.

Next, we define the spanning $r$-arborescence and the partition subgraph of a directed graph as follows.

**Definition 5:** Given a connected and directed graph $D$ whose root node is $r$, a spanning $r$-arborescence is a directed subgraph of $D$ such that there exists exactly one directed path from the root node $r$ to any other non-root node.

**Definition 6:** Given a directed graph $D$ whose root node is $r$, a partition subgraph is a subgraph $D'$ of $D$ such that the indegree of the root node $r$ is equal to 0 and the indegrees of other nodes are less than or equal to 1, i.e., $d^-(r) = 0, d^-(n) \leq 1, \forall n \in N(D'), n \neq r$. 

![Fig. 5. The generic edge-adding algorithm example](image-url)
The most well-known algorithm for solving the minimum weight spanning r-arborescence problem is Edmonds’ algorithm [19]. In this section, for directed network topologies, the necessary and sufficient metric property of the generalized Edmonds’ algorithm is developed and proved.

4.1 Edmonds’ Algorithm

4.1.1 Overview of Edmonds’ Algorithm

The core idea of Edmonds’ algorithm is summarized as follows. Given a directed graph $D$, whose root node is $r$, remove all the inbound arcs of root node $r$, and denote the resultant subgraph as $D_0$. Edmonds’ algorithm proceeds to build the minimum $r$-arborescence spanning $D_0$ following a three-phase procedure.

Initialisation: Send $D_0$ to Phase I as Phase I’s input.

Phase I: Given the input graph $D_i$ for the $i$th iteration of Phase I, create a directed graph $T_i$ that includes all nodes in $N(D_i)$ and no arcs between nodes. Each node in $T_i$, hence, is a separate arborescence. For each non-root node $n$ whose indegree is 0, a new arc $a^* \in E(D_0)$ is selected to be included in $T_i$. The new arc $a^*$ is selected as follows:

$$a^* = \arg \min_{a \in A(n)} \{w(a)\},$$

where $A(n) = \{a | a \in A^- (n), n \in N(D_i), n \neq r, d^- (n) = 0\}, A^- (n)$ is the set of arcs that terminate at node $n$, and $d^- (n)$ is the indegree of node $n$ in $T_i$.

The above arc adding process ends when $d^- (n) = 1$ for any $n \in N(T_i), n \neq r$, i.e., the indegree of any non-root node in $T_i$ is 1. If there is no circuit in $T_i$ after the arc adding process, then go to phase II with $T_i$ as its input. If there are circuits in $T_i$, then go to phase II and let $T_i$ and $T_{i+1}$ be its input.

Phase II: Given the input graph $T_i$ and $D_i$, pick a circuit $C_i$ in $T_i$ to eliminate as follows. Find a pair of arcs $(a_{e',c}^*, a_{nc}^*)$ that can be used to break $C_i$ by replacing $a_{e',c}^*$ with $a_{nc}^*$, where $a_{e',c}^* \in C_i, a_{nc}^* \in D_i$ and $a_{nc}^* \notin T_i$. Both $a_{e',c}^*$ and $a_{nc}^*$ are inbound arcs to node $c \in N(C_i)$. Arrows $a_{e',c}^*$ and $a_{nc}^*$ are selected based on the following equation:

$$(a_{e',c}^*, a_{nc}^*) = \arg \min_{a_{e',c} \in T_i, a_{nc} \in T_i} \{w(C_i \oplus a_{e',c} \oplus a_{nc})\},$$

where $a_{e',c}$ is any arc in $C_i, C_i \oplus a_{e',c}$ is the path generated by deleting arc $a_{e',c}$ from $C_i$, and $a_{nc}$ is an inbound arc to node $c$. Denote $P_i^*$ as the path generated by breaking the circuit $C_i$, i.e., $P_i^* = C_i \oplus a_{e',c} \oplus a_{nc}$.

This circuit breaking operation can be illustrated by an example. Consider the circuit $C_i$ in Fig. 7 (I), where the circuit is the solid part of the graph. The circuit $C_i$ is broken by replacing $a_{e',c}$ by $a_{nc}$ as shown in Fig. 7 (II), where the resultant arborescence is the solid part of the graph. The optimal circuit-breaking scheme is shown in Fig. 7 (III), where the resultant arborescence $P_i^*$ is the solid-line part of the graph.

4.1.2 Metric Properties and Proof

The required routing metric property for Edmonds’ algorithm can be expressed by the following definition.

**Definition 7:** Right $\oplus$-isotonicity for partition subgraphs: A weight function $w(\cdot)$ of partition subgraphs is said to be right $\oplus$-isotonic for partition subgraphs over a directed graph $D$ if for any partition subgraphs $D_1, D_2$ of $D$, we have

$$w(D_1) \preceq w(D_2) \Rightarrow w(D_1 \oplus D_3) \preceq w(D_2 \oplus D_3),$$

where $D_3$ is also a partition subgraph of $D$ such that $D_1 \oplus D_3, D_2 \oplus D_3$ are still partition subgraphs of $D$.

Essentially, the right $\oplus$-isotonicity for partition subgraphs specifies that the preference order $\preceq$ between two partition subgraphs $D_1$ and $D_2$ remains unchanged, when we expand them into larger partition subgraphs through adding a common set of arcs.

**Theorem 3:** Given any connected and directed network topology $D$ whose root node is $r$, Edmonds’ algorithm produces the minimum weight $r$-arborescence spanning $D$, if and only if the broadcast routing metric is right $\oplus$-isotonic for partition subgraphs.

**Proof:** Sufficient condition: Consider a connected and directed network topology $D$ whose root node is $r$. We are going to prove that if the broadcast routing metric satisfies the property in (9), Edmonds’ algorithm produces the minimum weight $r$-arborescence spanning $D$.

With $(a_{e',c}^*, a_{nc}^*)$ identified, shrink the circuit $C_i$ in $D_i$ to a pseudo-node $n_i$. Replace $N(C_i)$ in $D_i$ by $n_i$. Consider the terminating nodes of the outbound arcs going out of nodes in $C_i$. Their signatures remain unchanged. Set the inbound arc of $n_i$ as $a_{nc}^*$ and modify its signature to be the signature of $P_i^*$, i.e., the signature of the path used to replace the original circuit. Denote this newly generated graph through the shrinking operation as $D_{i+1}$. Go to the beginning of Phase I and let $D_{i+1}$ be Phase I’s input.

Phase III: Given the input $T_i$ after $l$ iterations of phase I and II, expand the pseudo-node $n_i$ to $P_i$ in the reverse order ($i = l, l-1, \ldots, 1$) of their generation sequence in phase II. The resulting subgraph $T_e$ is the minimum weight spanning $r$-arborescence of $D_0$.

Note that in most textbooks, Edmonds’ algorithm is only discussed with linear metric based on linear link weight aggregation. The algorithm discussed in this section is extended to cover both linear and nonlinear link metric aggregation.
Let the generated subgraph be the subgraph consisting of the inbound arcs of \(N\). Assume \(T_e \neq T^*\). There must exist some nodes \(n_j, j = 1, \ldots, m\) whose inbound arc \(a_j\) in \(T_e\) is different from the inbound arc \(b_j\) in \(T^*\) as shown in Fig. 8. By Edmonds’ algorithm, \(w(a_j) \leq w(b_j)\). Starting from \(T^*\), replace \(b_1\) in \(T^*\) by the corresponding \(a_1\) in \(T_e\). Denote the generated subgraph as \(T^*_1\). By the property in (9), it follows that
\[
\begin{align*}
&w(T^*_1) = w(T^* \oplus b_1 \oplus a_1) \leq w(T^* \oplus b_1 \oplus b_1) = w(T^*).
\end{align*}
\]
Similarly, replacing \(b_2\) by \(a_2\) in \(T^*_1\), we get \(T^*_2\) that satisfies \(w(T^*_2) \leq w(T^*_1)\). By continuing this arc replacing process, we get a series of graphs \(T^*_1, T^*_2, \ldots, T^*_m = T_e\) that satisfy \(w(T^*_j) \leq w(T^*_j), j = 1, \ldots, m - 1\). Hence, it follows that \(w(T^*_m) \leq w(T^*_j)\). Since \(T^*_m\) is a minimum weight spanning \(r\)-arborescence, \(w(T^*_m) = w(T^*)\). \(T^*_m\) is also a minimum weight spanning \(r\)-arborescence.

Denote \(T^*\) as the subgraph generated after expanding the pseudo node \(n_i\) to \(P^*\) in phase III. We next prove that \(T^*_i\) is the minimum weight spanning \(r\)-arborescence of \(D_i\) through induction hypothesis. First, for the base case, note that \(T_i\) is the minimum weight spanning \(r\)-arborescence of \(D_i\) and our hypothesis is satisfied. Then, assume that \(T^*_{i+1}\) is the minimum weight spanning \(r\)-arborescence of \(D_{i+1}\). We next show that \(T^*_{i+1}\) is the minimum weight spanning \(r\)-arborescence of \(D_{i+1}\).

Let \(T^*_{i+1}\) be an arbitrary spanning \(r\)-arborescence of \(D_{i+1}\). By the fact that \(T^*_{i+1}\) is a spanning \(r\)-arborescence, there must exist an arc \(a_{nc}\) that emanates from some node \(n \in \mathcal{N}(D_i) \setminus \mathcal{N}(C_i)\) and terminates at some node \(c \in \mathcal{N}(C_i)\). Let \(P^*_{ic} = C_i \oplus a_{ic} \oplus a_{nc}\) be the subgraph generated by replacing arc \(a_{ic}\) by arc \(a_{nc}\), where \(a_{cic} \in C_i\). Denote the subgraph consisting of the inbound arcs of \(N(C_i)\) in \(T^*_{i+1}\) as \(F_i\). If \(P^*_{ic} = F_i\), then \(w(P^*_{ic}) = w(F_i)\). If \(P^*_{ic} \neq F_i\), there must exist some nodes in \(N(C_i)\) whose inbound arc \(a_j\) in \(F_i\) is different from the inbound arc \(b_j\) in \(P^*_{ic}\), \(j = 1, 2, \ldots, t\). By Edmonds’ algorithm, \(w(b_j) \leq w(a_j)\). Starting from \(F_i\), replace \(a_1\) in \(F_i\) by the corresponding \(b_1\) in \(P^*_{ic}\). Let the generated subgraph be \(F^*_{i+1}\). By the property in (9), it follows that
\[
\begin{align*}
&w(F^*_{i+1}) = w(F_i \oplus a_1 \oplus b_1) \leq w(F_i \oplus a_1 \oplus a_1) = w(F_i). \quad (10)
\end{align*}
\]
Similarly, replacing \(a_2\) by \(b_2\) in \(F^*_{i+1}\), we get \(F^*_{i+2}\) that satisfies \(w(F^*_{i+2}) \leq w(F^*_{i+1})\). By continuing this arc replacing

5 Distributed Broadcast Routing

Theorems 1, 2, and 3 provide the necessary and sufficient metric properties for which Prim’s algorithm, the generic
edge-adding algorithm, and Edmonds’ algorithm guarantee optimality, respectively. It is important to note that although all these BTC algorithms require knowledge of global topology, optimal broadcast routing protocols based on these algorithms do not need to be centralized.

In this section, we discuss the distributed implementation of broadcast routing, and how its complexity is related to the underlying uniqueness problem of broadcast trees. Here, the uniqueness problem refers to whether the broadcast trees for different source nodes are the same. We find that this uniqueness problem is related to a property called root independence.

5.1 A Schematic Protocol

In the following, we outline a simple example of distributed broadcast routing protocols. Similar to link-state routing protocols such as OSPF [25], every node periodically advertises its local connectivity to the entire network. In this way, every node can learn the global topology. For each possible broadcast source node, a node runs one of the BTC algorithms to compute the broadcast tree rooted at the source node. In the broadcast tree, the node identifies its children and stores them as the outgoing links in its routing table. When each non-source node receives a broadcast routing packet, it checks the source address of the packet and forwards the packet to the outgoing links according to its routing table. Since each node has the same view of the topology and runs the same BTC algorithm, the forwarding information stored in each node’s routing table is consistent with the same unique broadcast tree and hence routing loops are avoided.

In general, the OBT for one root node is different from the OBT for another root node so that per source computation for its broadcast tree and per source routing entry is needed at each router of the network to support broadcast routing. The computation overhead and routing table size at the routers can be huge, if there are many broadcast source nodes. Therefore, we are interested in developing metric properties for which the computed broadcast trees for different roots have the same topology. That is the MST or minimum weight spanning arborescence rooted at one node is the same as the MST or minimum weight spanning arborescence rooted at another node. When this is true, each node, including the source node and non-source node, only computes and maintains one broadcast tree in their routing tables for all source nodes. This saves a lot of computation time and also reduces the routing table size, when the network scale is large and there are multiple broadcast source nodes.

5.2 Uniqueness Properties for Optimal Broadcast Trees

In this section, we show that to ensure the same broadcast tree for different sources, the routing metric must have a property called root independence.

Definition 8: A metric \( w(\cdot) \) is root independent, if \( w(F_1) = w(F_2) \), where \( F_1 \) and \( F_2 \) are subgraphs that have different root nodes and satisfy \( N(F_1) = N(F_2), E(F_1) = E(F_2) \).

A simple example of a root independent metric is the summation of all the link weights in an undirected tree. Given an undirected tree, the metric weight remains unchanged regardless of the root node. An example of a root dependent metric is the tree depth metric that measures path length of the longest shortest path from the root node to any leaf node in a tree. Given a tree, in general, the tree depth changes when the root node changes.

Theorem 4: Given any connected and undirected network topology \( G \), the MST is the same for different roots, if and only if the metric definition is independent of root node.

Proof: Sufficient condition: Let \( T_1^* \) be one MST of the connected and undirected network topology \( G \) whose root node is \( r_1 \). It follows that \( w(T_1^*) \leq w(T_1) \), where \( T_1 \) is any spanning tree rooted at \( r_1 \). Let \( T_2^* \) be the spanning tree generated by changing the root node of \( T_1^* \) from \( r_1 \) to \( r_2 \). Note that any spanning tree rooted at \( r_1 \) can also be a spanning tree rooted at \( r_2 \), and vice versa. Since the metric definition is independent of root node, for any \( r_1 \)-rooted spanning tree \( T_1 \) of \( G \), we have a \( r_2 \)-rooted spanning tree \( T_2 \) of \( G \) such that \( N(T_1) = N(T_2), E(T_1) = E(T_2), w(T_1) = w(T_2) \). It follows that \( w(T_2^*) = w(T_2^*) \leq w(T_1) = w(T_2) \), i.e., \( T_2^* \) is a MST of \( G \). Therefore, the MST is independent of the root node.

Necessary condition: Given any connected and undirected network topology with a root node, we need to prove that if the MST is the same for all sources then the metric definition is independent of the root node. This can be proved by showing that its contrapositive is correct, i.e., for a metric whose weight computation is related to the identity of the root node, there exists at least one topology, where the MST is dependent on the root node.

Suppose, for a network topology \( G \) whose root node is \( r_1 \), \( T_1^* \) is a MST rooted at \( r_1 \). It follows that \( w(T_1^*) \leq w(T_1) \), where \( T_1 \) is any spanning tree rooted at \( r_1 \). Change the root node \( r_1 \) of \( T_1^* \) and \( T_1 \) to \( r_2 \), and let the generated spanning trees as \( T_2^* \) and \( T_2 \), respectively. Since the metric definition is dependent on the root node, it is possible \( w(T_1^*) \neq w(T_2^*) \) and \( w(T_1) \neq w(T_2) \). Hence, it is possible that \( w(T_2^*) > w(T_2) \). Therefore, the MST is dependent on the root node.

Theorem 5: Given any connected and directed network topology \( D \) whose root node is \( r \), the minimum weight spanning arborescence is dependent on the root node \( r \).

Proof: In directed network topologies, the definition of minimum weight spanning arborescence is dependent on the root node \( r \). For two nodes \( n_1 \) and \( n_2 \), the weight of the link from \( n_1 \) to \( n_2 \) is generally different from the weight of the link from \( n_2 \) to \( n_1 \). Moreover, the link between \( n_1 \) and \( n_2 \) may be unidirectional rather than...
bidirectional. For some node, there may not exist a spanning arborescence rooted at this node. Therefore, in the general case, the root-independence property does not hold for the minimum weight spanning arborescence.

6 Case Study Based on Compatibility Analysis

In this section, we illustrate by examples how to apply our analytical results in Sections 3 and 4 to judge the compatibility between broadcast routing metrics and BTC algorithms. The case study results are summarized in Table 2, where Y means the algorithm is compatible with the metric, N means the algorithm is not compatible with the metric, and NA means the algorithm is not applicable with the metric.

6.1 Case 1: Minimum Tree Depth Broadcast Routing

Consider the case where the objective is to minimize the message delivery delay in a possibly partitioned network by minimizing the depth of broadcast trees in each of the topology components. The definition of the routing metric for such a purpose is

$$w(F) = \max_{T \subseteq F} \text{depth}(T,i),$$

(13)

where $F$ is a graph composed of multiple components, $T$ is a component of $F$, and $\text{depth}(T,i)$ is the depth of component $T$ with broadcast source node $i$. The depth of a component $T$ is the maximum number of edges along the shortest simple path from the root node to any other nodes in $T$. Mathematically, it means that

$$\text{depth}(T,i) = \max_{j \in N(T), j \neq i} SP(i,j),$$

(14)

where $SP(i,j)$ is the length of the shortest path from node $i$ to node $j$.

6.1.1 Compatibility with Prim’s Algorithm

To facilitate the case study, we first prove the following lemma.

**Lemma 1:** The metric in (13) satisfies the right $\oplus$-isotonicity property for trees in (4).

**Proof:** Let $T$ be the tree generated by Prim’s algorithm, and $T_1 = T \oplus e, T_2 = T \oplus e'$. Note that there is only one path from a non-root node to the root node in a tree. Suppose the longest path to the root node in $T$ is $i, i \geq 0$ hops long. Since $T$ is generated by Prim’s algorithm, new edges $e$ and $e'$ can only be appended to nodes that are $i$ hops to the root node in $T$, or be appended to nodes that are $i - 1$ hops to the root node in $T$ if there exist nodes which are $i - 1$ hops to the root node and $e$ or $e'$ can be appended to. The lemma can be proved in two parts.

First, we consider the case $w(T \oplus e) = w(T \oplus e')$. In this case, either both $e$ and $e'$ are appended to nodes that are $i$ hops to the root node in $T$, or both $e$ and $e'$ are appended to nodes that are $i - 1$ hops to the root node in $T$. For both subcases, by the fact that $w(T \oplus e) = w(T \oplus e')$, we have $w(T \oplus e \oplus F) = w(T \oplus e' \oplus F)$.

Second, we consider the case $w(T \oplus e) < w(T \oplus e')$. In this case, $e$ must be appended to a node that is $i - 1$ hops to the root node in $T$ and $e'$ must be appended to a node that is $i$ hops to the root node in $T$. There are two cases to consider. In the first case, $e'$ is on the longest path to the root node in $T \oplus e' \oplus F$, and we further have two subcases to consider. In the first subcase, edges $e$ and $e'$ share the same end node in the node set $N(G) - N(T)$ as shown in Fig.10 (I), where $F = \{F_1 \oplus F_2 \oplus F_3\}$. For this subcase, subtrees of $F$ can be appended to the shared end node of $e$ and $e'$, nodes that are $i$ hops to the root node in $T$, or nodes that are $i - 1$ hops to the root node in $T$. By the fact that $w(T \oplus e) < w(T \oplus e')$, we have $w(T \oplus e \oplus F) \leq w(T \oplus e' \oplus F)$. In the second subcase, edges $e$ and $e'$ have different end nodes in the node set $N(G) - N(T)$ as shown in Fig.10 (II), where $F = \{F_2 \oplus F_3\}$. In this subcase, during the $T_1 \oplus F$ and $T_2 \oplus F$ operation, there exist no subtrees in $F$ which can be appended to either $e$ or $e'$. Otherwise, $T \oplus e \oplus F$ and $T \oplus e' \oplus F$ are forests rather than trees. It follows that subtrees of $F$ can only be appended to nodes that are $i$ hops to the root node in $T$ or nodes that are $i - 1$ hops to the root node in $T$. By the fact that $w(T \oplus e) < w(T \oplus e')$, we still have $w(T \oplus e \oplus F) \leq w(T \oplus e' \oplus F)$. In the second case, $e'$ is not on the longest path to the root node in $T \oplus e' \oplus F$, and we have $w(T \oplus e \oplus F) = w(T \oplus e' \oplus F)$. In summary, for the case $w(T \oplus e) < w(T \oplus e')$, we have $w(T \oplus e \oplus F) \leq w(T \oplus e' \oplus F)$.

By Lemma 1 and Theorem 1, Prim’s algorithm can find the minimum depth broadcast tree.

Next, we show that the consistency property in [23] is only the sufficient but not necessary property for Prim’s algorithm to find the MST. Consider the tree depth metric in (13), and the example topology in Fig.11 (I). Notice that $w(T \oplus e_1) = w(T \oplus e_2) = 2$ but $w(T \oplus e_1 \oplus e_3) = 3$.
Consider the network topology shown in Fig. 12 (I). Let the generic edge-adding algorithm be verified by noticing that one possible tree generated by finding the minimum depth broadcast tree. This can be verified by observing that one possible tree generated by Edmonds’ algorithm is not satisfies the property in (9). Therefore, by Theorem 3, Edmonds’ algorithm cannot guarantee producing the minimum depth broadcast arborescence. This can be verified by noticing that one possible Edmonds’ algorithm output is $F \oplus a_5 \oplus a_1 \oplus a_3$, which is not optimal.

**6.2 Case 2: Most Reliable Widest Bandwidth Broadcast Routing**

Consider a most reliable widest bandwidth metric whose design is based on the following two observations. First, to maximize the capacity of links over the broadcast tree $T$, it is desirable to design a metric maximizing the bandwidth of the tree. The bandwidth of the tree is defined as the minimum bandwidth among all the links of the tree. Second, to ensure successful delivery of a broadcast message, a proper broadcast routing metric should minimize the probability that the broadcast message is lost by some nodes in the broadcast tree $T$. This probability can be calculated as

$$1 - \prod_{i \in E(T)} (1 - p_i) \approx \sum_{i \in E(T)} p_i,$$

where $p_i$ is the estimated packet loss rate for a link in the broadcast tree, and the approximation is based on the assumption that each link’s packet loss rate is small enough. This is valid for most scenarios. Hence, it is reasonable to consider the following lexicographic metric

$$(b, p),$$

where $b$ is the bandwidth of the tree, $p$ is the packet loss rate of the tree. We have $w(b_1, p_1) \leq w(b_2, p_2)$ if either $b_1 > b_2$ or $b_1 = b_2, p_1 \leq p_2$. The objective of the most reliable widest bandwidth broadcast routing is to find the lowest packet loss rate spanning tree within all widest bandwidth spanning trees.

**6.2.1 Compatibility with Prim’s Algorithm**

The metric in (16) does not satisfy the required property in (4). Consider the network topology in Fig. 14 (I). Let $w(e_1) = (10, 0.03), w(e_2) = (8, 0.01), w(e_3) = (5, 0.05)$. Note that $w(F \oplus a_5) = 1 < w(F \oplus a_4) = 2$ as shown in Fig. 13 (II) and (III), but $w(F \oplus a_5 \oplus a_1 \oplus a_3) = 3 > w(F \oplus a_4 \oplus a_1 \oplus a_3) = 2$ as shown in Fig. 13 (IV) and (V). Therefore, the depth metric does not satisfy the property in (9). Therefore, by Theorem 3, Edmonds’ algorithm cannot guarantee producing the minimum depth broadcast arborescence. This can be verified by noticing that one possible Edmonds’ algorithm output is $F \oplus a_5 \oplus a_1 \oplus a_3$, which is not optimal.

**6.2.2 Compatibility with the Generic Edge-adding Algorithm**

The depth metric definition in (13) does not satisfy the property in (6). Consider the network topology $D$ as shown in Fig. 13 (I). Let $F = (N(D),\{a_2\})$. Note that $w(F \oplus a_5) = 1 < w(F \oplus a_4) = 2$ as shown in Fig. 13 (II) and (III), but $w(F \oplus a_5 \oplus e_1 \oplus e_3) = 3 > w(F \oplus e_4 \oplus e_1 \oplus e_3) = 2$ as shown in Fig. 13 (IV) and (V). Therefore, it does not satisfy the property in (6). Therefore, by Theorem 2, the generic edge-adding algorithm cannot guarantee finding the minimum depth broadcast tree. This can be verified by noticing that one possible tree generated by the generic edge-adding algorithm is $F \oplus e_5 \oplus e_1 \oplus e_3$, which is not optimal.

**6.1.3 Compatibility with Edmonds’ Algorithm**

The depth metric in (13) does not satisfy the property in (9). Consider the network topology $D$ as shown in Fig. 13 (I). Let $F = (N(D),\{a_2\})$. Note that $w(F \oplus a_5) = 1 < w(F \oplus a_4) = 2$ as shown in Fig. 13 (II) and (III), but $w(F \oplus a_5 \oplus a_1 \oplus a_3) = 3 > w(F \oplus a_4 \oplus a_1 \oplus a_3) = 2$ as shown in Fig. 13 (IV) and (V). Therefore, the depth metric does not satisfy the property in (9). Therefore, by Theorem 3, Edmonds’ algorithm cannot guarantee producing the minimum depth broadcast arborescence. This can be verified by noticing that one possible Edmonds’ algorithm output is $F \oplus a_5 \oplus a_1 \oplus a_3$, which is not optimal.

**TABLE 2**

Summary of the case study

<table>
<thead>
<tr>
<th>Metrics</th>
<th>Prim’s</th>
<th>Edge-adding</th>
<th>Edmonds</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum tree depth broadcast routing</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Most reliable widest bandwidth routing</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Minimum energy broadcast routing</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Lexicographically optimal routing</td>
<td>NA*</td>
<td>NA</td>
<td>Y</td>
</tr>
<tr>
<td>Maximum network lifetime broadcast</td>
<td>NA*</td>
<td>NA</td>
<td>Y</td>
</tr>
</tbody>
</table>

*The modified Prim’s algorithm (termed as DMST) in [4] is compatible with the network lifetime metric.
6.2.2 Compatibility with the Generic Edge-adding Algorithm

The metric in (16) does not have the required property in (6). Consider the network topology \( G \) shown in Fig.15 (I). Let \( w(e_1) = (10, 0.03), w(e_2) = (8, 0.01), w(e_3) = (5, 0.05) \). Note that \( w(F_0 \oplus e_1) = (10, 0.03) \prec w(F_0 \oplus e_2) = (8, 0.01) \) but \( w(F_0 \oplus e_1 \oplus e_3) = (5, 0.08) \prec w(F_0 \oplus e_2 \oplus e_3) = (5, 0.06) \), where \( F_0 = (N(G), \emptyset) \), and \( F_0 \oplus e_1, F_0 \oplus e_2, F_0 \oplus e_1 \oplus e_3, F_0 \oplus e_2 \oplus e_3 \) are shown in Fig.15 (II), (III), (IV), (V), respectively. Therefore, the most reliable widest bandwidth metric does not satisfy the property in (6). By Theorem 2, the generic edge-adding algorithm cannot guarantee producing the MST. This can be easily shown by noticing that one possible algorithm output is \( F_0 \oplus e_1 \oplus e_3 \), which is not optimal.

6.2.3 Compatibility with Edmonds’ Algorithm

The metric in (16) does not have the required property in (9). Consider the network topology \( D \) shown in Fig.16 (I). Let \( w(a_1) = (10, 0.03), w(a_2) = (8, 0.01), w(a_3) = (5, 0.05) \). Note that \( w(a_1) \prec w(a_2), w(a_1 \oplus a_3) = (5, 0.08) \prec w(a_2 \oplus a_3) = (5, 0.06) \), where \( a_1 \oplus a_3, a_2 \oplus a_3 \) are shown in Fig.16 (II), (III), respectively. Therefore, the most reliable widest bandwidth metric does not satisfy the property in (9). By Theorem 3, Edmonds’ algorithm does not guarantee optimality. This can be shown by noticing that one possible algorithm output \( a_1 \oplus a_3 \) is not the minimum weight spanning arborescence.

6.3 Case 3: Minimum Energy Broadcast Routing

Let us consider the minimum energy broadcast routing, whose metric is defined in (1).

6.3.1 Compatibility with Prim’s Algorithm

The metric in (1) does not satisfy the property in (4). Consider the network topology \( G \) shown in Fig.17 (I). The numbers associated with the edges are the weights of these edges. Let the partial spanning tree be \( T_r = \{ (r, i), (e_{r_1}) \} \) as shown by the bold part of Fig.17 (I). Note that \( w(T_r \oplus e_{ij}) = 2 \prec w(T_r \oplus e_{rj}) = 2.5 \) as shown in Fig.17 (II) and (III), but \( w(T_r \oplus e_{ij} \oplus e_{rj}) = 4.5 \succ w(T_r \oplus e_{rj} \oplus e_{r_k}) = 3.5 \) as shown in Fig.17 (VII) and (VIII). Hence, the metric in (1) does not satisfy the property in (4). Therefore, by Theorem 1, Prim’s algorithm cannot guarantee producing the minimum energy broadcast routing tree based on the metric in (1). This can be verified by noticing that \( w(T_p) = 4.1 \succ w(T^*) = 3.5 \), where \( T_p \) is the Prim tree generated by Prim’s algorithm as shown in Fig.17 (IX) and \( T^* \) is the minimum energy broadcast tree as shown in Fig.17 (VIII).

6.3.2 Compatibility with the Generic Edge-adding Algorithm

The metric in (1) does not satisfy the property in (6). Again, consider the network topology \( G \) shown in Fig.17 (I). Suppose the partial spanning forest generated by the generic edge-adding algorithm is \( F = (N(G), \{ e_{r_1} \}) \) as shown in Fig.17 (IV). Note that \( w(F \oplus e_{ij}) = 2 \prec w(F \oplus e_{rj}) = 2.5 \) as shown in Fig.17 (V) and (VI), but \( w(F \oplus e_{ij} \oplus e_{rj}) = 4.5 \succ w(F \oplus e_{rj} \oplus e_{r_k}) = 3.5 \) as shown in Fig.17 (VII) and (VIII). Hence, the metric does not satisfy the property in (6). Therefore, by Theorem 2, the generic edge-adding algorithm cannot guarantee producing the minimum weight broadcast routing tree based on the metric in (1). This can be verified by noticing that, for the generic edge-adding tree \( T_a \) and the minimum weight broadcast routing tree \( T^* \), \( w(T_a) = 4.1 \succ w(T^*) = 3.5 \), as shown in Fig.17 (IX) and (VIII).

6.3.3 Compatibility with Edmonds’ Algorithm

The metric in (1) does not satisfy the required property in (9). Consider the connected and directed network topology \( D \) whose root node is \( r \) as shown in Fig.18 (I). The numbers associated with arcs are the weights of these arcs. Let \( F \) be a partial spanning arborescence.

Fig. 14. The Prim algorithm example for the most reliable widest bandwidth broadcast routing

Fig. 15. The generic edge-adding algorithm example for the most reliable widest bandwidth broadcast routing

Fig. 16. The Edmonds algorithm example for the most reliable widest bandwidth broadcast routing
There exists a number \( a \) such that \( \sum_{l=1}^{m} a_l \geq \sum_{l=1}^{m} b_l \). Mathematically, the metric is expressed as:

\[
\sum_{l=1}^{m} a_l \mid \forall l \leq 1 \leq i \leq m - 1, a_i \leq b_i.
\]

Consider a broadcast routing metric finding the lexicographically OBT in terms of packet loss rate on each link. The metric is a sorted list of link packet loss rates of the considered network topology. Elements of the list are sorted in a decreasing order. Mathematically, the metric can be expressed as:

\[
w(a) = (a_1, a_2, ..., a_m), \quad w(F \oplus a_j) = 4.5 \succ w(F \oplus a_{r_j}) = 3.5 \text{ as shown in Fig.18 (III) and (IV). Hence, the minimum energy broadcast routing metric in (1) does not satisfy the property in (9). Therefore, Edmonds’ algorithm does not produce the minimum weight broadcast routing arborescence based on the metric in (1). This conclusion can be verified by noticing that } w(T_e) = 4.1 \succ w(T^*) = 3.5, \text{ where } T_e \text{ is the Edmonds arborescence generated by Edmonds’ algorithm and } T^* \text{ is the minimum energy broadcast routing arborescence.}
\]

**6.4 Case 4: Lexicographically Optimal Broadcast Routing based on Packet Loss Rate**

Consider a broadcast routing metric finding the lexicographically OBT in terms of packet loss rate on each link. The metric is a sorted list of link packet loss rates of the considered network topology. Elements of the list are sorted in a decreasing order. Mathematically, the metric can be expressed as:

\[
w(a) = (a_1, a_2, ..., a_m), \quad w(F \oplus a_j) = 4.5 \succ w(F \oplus a_{r_j}) = 3.5 \text{ as shown in Fig.18 (III) and (IV). Hence, the minimum energy broadcast routing metric in (1) does not satisfy the property in (9). Therefore, Edmonds’ algorithm does not produce the minimum weight broadcast routing arborescence based on the metric in (1). This conclusion can be verified by noticing that } w(T_e) = 4.1 \succ w(T^*) = 3.5, \text{ where } T_e \text{ is the Edmonds arborescence generated by Edmonds’ algorithm and } T^* \text{ is the minimum energy broadcast routing arborescence.}
\]

**Lemma 2:** The lexicographical optimality metric satisfies the properties in (4), (6) and (9).

**Proof:** We first prove that the lexicographical optimality metric satisfies the properties in (4) and (6). Let \( p \) and \( q \) be network topologies consisting of \( m \) and \( n \) edges, respectively, and \( e \) and \( e' \) be two edges. Given that \( w(p \oplus e) \preceq w(p \oplus e') \), we need to prove that \( w(p \oplus e \oplus q) \preceq w(p \oplus e' \oplus q) \). By the metric definition, we have \( w(p \oplus e \oplus q) = (q_1, q_2, ..., q_n) \). Let \( a = p \oplus e, b = p \oplus e' \), and \( c = p \oplus e \oplus q, d = p \oplus e' \oplus q \). When \( w(p \oplus e) \preceq w(p \oplus e') \), by the metric definition, we have \( w(p \oplus e \oplus q) = w(p \oplus e' \oplus q) \). Therefore, the lexicographical optimality metric satisfies the properties in (4) and (6).

We next prove that the lexicographical optimality metric satisfies the property in (9). Let \( p \) and \( q \) be two network topologies consisting of \( m \) arcs, and \( u \) be a network topology consisting of \( n \) arcs. Given that \( w(p) \preceq w(q) \), we need to prove that \( w(p \oplus u) \preceq w(q \oplus u) \). This can be shown by a similar proof technique as the proof for properties (4) and (6). To save space, we omit the proof details. For a complete proof, readers are referred to our technical report [24]. Therefore, the lexicographical...
optimality metric satisfies the property in (9).

6.4.1 Compatibility with Prim’s Algorithm

By Lemma 2 and Theorem 1, Prim’s algorithm can find the OBT.

6.4.2 Compatibility with the Generic Edge-adding Algorithm

By Lemma 2 and Theorem 2, the generic edge-adding algorithm can find the OBT.

6.4.3 Compatibility with Edmonds’ Algorithm

By Lemma 2 and Theorem 3, Edmonds’ algorithm can find the OBT.

6.5 Case 5: Maximum Network Lifetime Broadcast Routing

Consider the broadcast routing metric which maximizes the network lifetime [4]. It is defined as

\[
\min_{i \in N(T)} \frac{\varepsilon_i}{\max_{(i,j) \in E(T)} P_{ij}},
\]

(17)

where \( N(T) \) is the node set of the broadcast tree \( T \), \( \varepsilon_i \) is the residual energy of node \( i \), \( E(T) \) is the link set of the broadcast tree \( T \), and \( P_{ij} \) is the minimum power consumption of transmission from node \( i \) to node \( j \). The broadcast routing metric definition given in (17) is essentially the minimum lifetime of all the links in the broadcast tree \( T \). Also note the fact that the underlying network topology is undirected when the residual energy of each node is equal, and the underlying network topology is directed when the residual energy of each node is different [4].

6.5.1 Compatibility with Prim’s Algorithm

Note that the property definition in Theorem 1 is based on undirected network topologies. Therefore, when the residual energy of each node is different, the metric in (17) is not compatible with the property definition in Theorem 1. When the residual energy of each node is equal, the network can be modeled as an undirected graph and hence Theorem 1 can be used to check the applicability of Prim’s algorithm. In this case, it is clear that the minimization operation of the metric satisfies the property in (4). Therefore, we have the following conclusions.

- If the residual energy of each node is different, Prim’s algorithm cannot find the maximum network lifetime broadcast tree based on the metric in (17).
- If the residual energy of each node is equal, Prim’s algorithm produces the maximum network lifetime broadcast tree based on the metric in (17).

6.5.2 Compatibility with the Generic Edge-adding Algorithm

Since the property definition in Theorem 2 is based on undirected network topologies, the metric in (17) is not compatible with the property definition in Theorem 2 when the residual energy of each node is different. When the residual energy of each node is equal, we further check if the metric in (17) satisfies the property in (6). Clearly, the minimization operation of the metric satisfies the property in (6). Therefore, we have the following conclusions.

- If the residual energy of each node is different, the generic edge-adding algorithm cannot find the maximum network lifetime broadcast tree based on the metric in (17).
- If the residual energy of each node is equal, the generic edge-adding algorithm produces the maximum network lifetime broadcast tree based on the metric in (17).

6.5.3 Compatibility with Edmonds’ Algorithm

Since the property definition in Theorem 3 is based on directed network topologies and undirected topologies can be transformed into directed network topologies, the metric in (17) is compatible with the property definition in Theorem 3. Next, we check if the metric in (17) satisfies the property in (9). Clearly, the minimization operation of the maximum network lifetime metric satisfies the property in (9). Therefore, we have the following conclusions.

- If the residual energy of each node is different, Edmonds’ algorithm produces the maximum network lifetime broadcast arborescence based on the metric in (17).
- If the residual energy of each node is equal, Edmonds’ algorithm also produces the maximum network lifetime broadcast arborescence based on the metric in (17).

It is important to note that besides Edmonds’ algorithm there are other algorithms which can also find the OBT for the network lifetime metric (17). In fact, in [4], a new algorithm, termed as DMST, has been shown to be able to find the OBT for (17). Currently, the exact necessary and sufficient condition for DMST to compute OBTs for any given metric is still an open problem and is part of our future work.

7 Conclusion

In this paper, the potential incompatibility between broadcast routing metrics and BTC algorithms is identified. Using our broadcast routing algebra model, we develop and prove the necessary and sufficient properties of broadcast routing metrics which guarantee optimal broadcast routing in both undirected network topologies and directed network topologies. Further, we study the uniqueness properties required by distributed routing
implementation. These properties are used to identify compatibility between broadcast routing metrics and broadcast routing algorithms.

ACKNOWLEDGMENTS

This work was supported in part by the US National Science Foundation under grant CNS-0831865 and the Institute for Critical Technology and Applied Science (ICTAS) of Virginia Tech.

REFERENCES


Chuan Han Chuan Han is currently a Ph.D. student at the Electrical and Computer Engineering Department at Virginia Polytechnic Institute and State University (Virginia Tech). He received his B.S. degree in communications engineering from University of Electronic Science and Technology of China (UESTC), Chengdu, China, in 2004. and M.S. degree from the National Key Laboratory of Communications (NCL) of UESTC in 2007. His research is in wireless communications, networking and cognitive radio.

Yaling Yang Yaling Yang is currently an assistant professor at the Electrical and Computer Engineering Department at Virginia Tech. She received her Ph.D. degree of Computer Science from the University of Illinois at Urbana-Champaign. Her research interests include resource management and QoS in wireless networks, network routing, cognitive networks and congestion control. For a list of publications and more detailed information, please visit: http://www.ece.vt.edu/yyang8.