MINIMUM SPANNING TREES IN INFINITE GRAPHS: THEORY AND ALGORITHMS*

CHRISTOPHER THOMAS RYAN†, ROBERT L. SMITH‡, AND MARINA A. EPELMAN§

Abstract. We discuss finding minimum-cost spanning trees (MSTs) on connected graphs with countably many nodes of finite degree. When edge costs are summable and an MST exists (which is not guaranteed in general), we show that an algorithm that finds MSTs on finite subgraphs (called *layers*) converges in objective value to the cost of an MST of the whole graph, as the sizes of the layers grow to infinity. We call this the *layered greedy algorithm* since a greedy algorithm is used to find MSTs on each finite layer. We stress that the overall algorithm is *not* greedy since edges can enter and leave iterate spanning trees as larger layers are considered. However, in the setting where the underlying graph has the *finite cycle* (FC) property (meaning, every edge is contained in at most finitely many cycles) and distinct edge costs, we show that a unique MST T^* exists and the layered greedy algorithm produces iterates that converge to T^* by eventually "locking in" edges after finitely many iterations. Applications to network deployment are discussed.

Key words. minimum spanning trees, infinite graphs, infinite-dimensional optimization

MSC codes. 90C27, 90C35, 90C48

1. Introduction.

The problem of finding minimum-cost spanning trees on finite graphs is a classical combinatorial optimization problem with numerous applications in practice [11, 13, 14, 20]. The problem is used as a subroutine or heuristic for solving other graph optimization problems [3, 9, 25]. To our knowledge, an algorithmic approach to the MST problem on *infinite* graphs has not been systematically pursued, despite there being extensive literature on algorithms for infinite graphs in other contexts (see, for instance, [2, 7, 10, 18]). Several references examine properties of spanning trees in the limit of finite random graphs (see, for instance, [1, 4, 5, 16]), but the focus of these papers is not on the questions of existence and performance of algorithms, topics we emphasize here. The only paper we know of that deliberates on producing an algorithm for finding MSTs in infinite graphs is [15] in the more general context of infinite matroids (we discuss this paper in more detail below).

In a finite graph, an MST always exists and can be found by a greedy algorithm. The MST problem on infinite graphs does not afford such luxuries. As we will show through examples, an MST may not even exist in an infinite graph, and when it does, it may not be reachable by a greedy algorithm.

In response to this, we develop an algorithm to tackle the MST problem (whenever an MST exists) in any connected graph with countably many nodes of finite degree and summable edge costs. This algorithm finds MSTs in a growing sequence of finite subgraphs that, in the limit, converge in cost to that of an MST of the original graph (see Theorem 3.3). We call this result — i.e., convergence of the total cost of the edges of iterate trees to the total cost of the edges of an MST — *convergence in objective value.*

*Submitted to the editors on 30 May 2023.

Funding: This work was funded by the Natural Sciences and Engineering Research Council of Canada Discovery Grant RGPIN-2020-06488.

†University of British Columbia, Sauder School of Business, (chris.ryan@sauder.ubc.ca, https://www.sauder.ubc.ca/people/christopher-thomas-ryan).

‡University of Michigan, Industrial and Operations Engineering, (rlsmith@umich.edu, http://www-personal.umich.edu/~rlsmith/).

§University of Michigan, Industrial and Operations Engineering, (mepelman@umich.edu, http://www-personal.umich.edu/~mepelman/).

The sequence of subgraphs considered by our algorithm are called *layers*, and so we call our algorithm the *layered greedy algorithm* since it applies a greedy algorithm repeatedly in a growing set of layers of the graph. It is important to note that the layered greedy algorithm (as a whole) is not greedy since the MSTs found at each iteration must be computed "from scratch" and do not necessarily extend the previous MSTs from earlier layers in a greedy fashion. We also show, under an assumption akin to discounting of the edge costs in the graph, that finite termination of the infinite algorithm provides "good" solutions with bounded error in finite time.

The fact that the layered greedy algorithm guarantees convergence in objective value on a broad class of infinite graphs is the first important result in our paper. However, it naturally leads to three additional questions.

(Q1) We have convergence in objective value when an MST is known to exist in the original graph. How can we guarantee that an MST exists?

(Q2) Convergence in objective value is a nice feature, but we would also like convergence to an optimal solution. How can we ensure that the finite-sized iterates of the layered greedy algorithm converge to an MST of the original, infinite graph?

(Q3) Since the layered greedy algorithm is not greedy (but only locally greedy within layers), edges may come and go from iterate spanning trees as the algorithm proceeds. What are some sufficient conditions for an edge of the iterates to eventually "lock in" to an edge of the MST after finitely many iterations? Moreover, can these conditions be verified during the execution of the algorithm?

All three questions rely on careful consideration of the topological properties of the graph. Here, by "topological" we refer to questions of closedness, compactness, and convergence in the space of subgraphs of a given graph (and not the common alternative usage of referring to the arrangement of edges and nodes in a graph as its "topology"). Regarding (Q1), topology plays an important role because the existence of optimal solutions is naturally a topological consideration. As in other infinite-dimension optimization problems, we must argue that the feasible region has some notion of compactness and the objective function has some notion of continuity to conclude (using a Weierstrass-type result) that an optimal solution exists.

In (Q2), topology is implicit in the question itself because any notion of convergence requires a specification of topology. At first glance, (Q3) appears unrelated to topology, but the concept of "lock-in" has a topological flavor. Indeed, convergence via lock-in is precisely the notion of convergence in discrete topologies (see, for instance, Section 12 of [17]). This motivates our use of the *product discrete topology*, where convergence corresponds to lock-in edge by edge (here, the product is taken across edges).

With this topology, we can develop weak sufficient conditions for positive answers to the three questions we posed. In particular, we show (in Theorem 5.5) that the following condition:

(C1) *Finite Cycle (FC) property:* every edge of the graph is contained in at most finitely many cycles in the graph

is sufficient to guarantee that an MST always exists, answering (Q1). This follows by showing compactness of the set of spanning trees in the product discrete topology under the FC property (see Lemma 5.4).

If, additionally, the following holds:

(C2) *Distinct Edge Costs:* no two edges have the same cost,

then a *unique* MST always exists (Theorem 6.3). The uniqueness is useful in establishing our answer to (Q2): the sequence of iterates of the layered greedy algorithm converges to the unique MST in the product discrete topology when (C1) and (C2) hold. Finally, because of the nature of convergence in the product discrete topology, (C1) and (C2) give conditions for lock-in of the edges, answering the first part of (Q3). At the end of Section 6, we provide verifiable conditions for discovery of these edges, that have lock-in in agreement with "early" edges of the MST, addressing the second part of the last question.

Related work. We add to the growing literature on algorithms for solving problems on infinite graphs, including recent applications in deep learning (see the review article [26] for a summary of this work). Our work resembles a stream of work for solving network flow problems on infinite graphs [12, 18, 19, 21, 22, 24]. An important distinction between that line of work and what we pursue here is the definition of trees and connectedness. In the network flow literature, trees can be connected through a "node at infinity" that acts as a universal sink for flow generated at supply nodes in the graph. In contrast, our notion of connectivity is more classical, requiring nodes to be connected by a path of finitely many edges. Another important distinction is that network flow problems are typically studied as continuous optimization problems, allowing, for example, duality arguments and generalizations of the max-flow/min-cut theorem to infinite graphs [2]. By contrast, the MST problem is fundamentally discrete.

This paper is also related to the literature on infinite matroids (see, for instance, [8] and references therein). Here, the primary focus is on describing axiom systems for carefully defining the notion of infinite matroid to allow for a convenient matroid duality theory. As far as we know, little attention (other than [15]) has been given to the generality of greedy algorithms in infinite graphs. The workhorse of much of this infinite matroid theory is using Zorn's lemma to show the existence of maximal objects within infinite graphs. By contrast, our theory relies more heavily on topological arguments, including Tychnoff's Theorem, Weierstrass's Theorem, and convergence proofs. Indeed, our layered greedy MST algorithm does not produce a "chain" of nested spanning trees that would be necessary to leverage Zorn-like arguments.

[15] proposes a "greedy" algorithm for finding bases in finitary infinite matroids (corresponding to MSTs in infinite graphs with nodes of finite degree). This algorithm, however, is shown to find an MST using transfinite induction. Infinite graph adaptations of greedy algorithms for finding MSTs in finite graphs may even fail to converge to trees or span the nodes of the graph. A greedy algorithm can be "indefinitely distracted" by an infinite subset of low-cost edges, never getting around to span other parts of the graph. Klee's algorithm avoids this issue by continuing to analyze its execution after an infinite time is exhausted exploring a single subtree of a spanning tree. This is why the algorithm is called transfinite. The arguments in this paper do not use transfinite induction; we analyze the execution of algorithms using standard limiting arguments.

Of course, one can view the graph as a matroid [10] where the independent subsets of the edges of G are its forests. This view would allow to prove some of the results in this paper using matroid theory, but other results, such as solution convergence and early edge detection are established exploiting the special properties we assume about the graphical and cost structures. For this reason, we will use terminology and ideas familiar from studying finite graphs as much as possible, only delving into topics that are peculiar to infinite graphs when necessary and avoiding the even more general language of matroids altogether. We hope that this makes the paper more accessible to readers with little or no exposure to either matroids or infinite graph theory. It is an open direction for future research to examine the implications of our method for general infinite matroids.

Organization of the paper. The paper is organized as follows. In Section 2, we describe the general class of infinite graphs that we consider and define the MST problem in this class. Section 3 presents the layered greedy algorithm and analyzes its convergence in objective value. Section 4 formalizes the finite cycle property (C1). In Section 5, we establish the existence of MSTs under the FC property. In Section 6, we establish convergence of iterates of the layered greedy algorithm to an optimal spanning tree under the FC property and the additional condition (C2) of distinct edge costs. We also explore conditions that allow for discovery of early edges of the infinite MST and its implications for applications. Section 8 concludes the paper.

2. The minimum spanning tree problem.

2.1. Basic definitions. Let $G = (\mathcal{V}, \mathcal{E})$ be an undirected graph with node set $\mathcal{V} = \{1, 2, \dots\}$ and edge set $\mathcal{E}$. Let $c : \mathcal{E} \to \Re$ denote an edge-cost functional for G. We will sometimes use c_{ij} to denote the cost $c(\{i,j\})$ of edge $\{i,j\} \in \mathcal{E}$ when convenient.

The set $I(i)$ denotes the nodes that are adjacent to node i, that is, $I(i) := \{j \in \mathcal{V} \mid \{i,j\} \in \mathcal{E}\}$. The *degree* of node i is the cardinality of $I(i)$. A graph is *locally finite* if every node has finite degree. A *path* in G is a finite sequence of *distinct* nodes $i_1, i_2, \dots, i_n$, where $\{i_k, i_{k+1}\} \in \mathcal{E}$ for $k = 1, \dots, n-1$. A *ray* is an infinite sequence of distinct nodes $i_1, i_2, \dots$, where $\{i_k, i_{k+1}\} \in \mathcal{E}$ for $k = 1, 2, \dots$. Two nodes i and j are *connected* in G if there exists a path starting with node i and ending with node j. The graph G is *connected* if all pairs of nodes i and j in G are connected. We make the following assumption throughout the paper:

Assumption 1. *The graph G is locally finite and connected.* $\triangleleft$

A *cycle* in G is a finite sequence of nodes $i_1, i_2, \dots, i_n, i_1$, where $i_1, i_2, \dots, i_n$ is a path and $\{i_1, i_n\} \in \mathcal{E}$. A *bi-ray* consists of a node i and two distinct rays, that is, rays $(i, i_1, i_2 \dots)$ and $(i, j_1, j_2, \dots)$, where all intermediate nodes i_k and j_ℓ are distinct.

Let H be a subgraph of G and let $\mathcal{V}(H)$ and $\mathcal{E}(H)$ denote the set of nodes and edges in H, respectively. In this paper, we only consider subgraphs with no isolated nodes, that is, for every node $i \in \mathcal{V}(H)$, there exists an edge $\{i,j\} \in \mathcal{E}(H)$ for some node $j \in \mathcal{V}(H)$. In light of this, we will typically refer to a subgraph H simply by its set $\mathcal{E}(H)$ of edges, since the set of nodes is implicit once the edges are defined. The cost function will also be defined on the collection $\mathcal{P}(\mathcal{E})$ of subsets of edges of $\mathcal{E}$ (corresponding to subgraphs), where $c(H) := \sum_{e \in H} c(e)$ for any $H \in \mathcal{P}(\mathcal{E})$.

A *forest* F of G is an acyclic subgraph of G; i.e., a subgraph of G without cycles. A connected forest is a *tree*. If a subgraph of G has node set $\mathcal{V}$, it is said to *span* G. A connected spanning forest is called a *spanning tree*.

One of the nodes in G is called its *root node* r. (The theory developed below is indifferent to which node in G is called the root node.) The first *layer* of nodes, denoted L_1, consists of node r and all nodes that are adjacent to r; that is, $L_1 := \{r\} \cup I(r)$. We define other layers recursively:

$$L_{n+1} := L_n \cup \{i \in I(j) \text{ for some } j \in L_n\}, \ n = 1, 2, \dots,$$

and sometimes refer to $\{r\}$ as layer 0. Since G is locally finite and connected, each layer contains a finite number of nodes, every node is included in some layer, and once a node is in layer L_n, it is in every subsequent layer L_k for $k > n$. Let $G_n := (L_n, \mathcal{E}_n)$ for $n \geq 1$ denote the subgraph of G, where $\mathcal{E}_n := \{\{i, j\} \in \mathcal{E} \mid i, j \in L_n\}$ is the set of edges in the subgraph induced by the set of nodes L_n (we also use the term "layer n" to refer to G_n).

2.2. Formal statement of the minimum spanning tree problem. Recall that the cost $c(T)$ of a spanning tree T of G is the sum of the costs of the edges of T, i.e., $c(T) = \sum_{\{i,j\}\in\mathcal{E}(T)} c_{ij}$. Our problem is to find a minimum-cost spanning tree of G, i.e., solve

$$\text{(P)} \qquad c^\star := \inf\{c(T) \mid T \text{ is a spanning tree of } G\}.$$

We call any optimal solution $T^\star$ of (P) a minimum spanning tree (MST). We say *G possesses an MST* if (P) has an optimal solution (that is, the infimum in (P) is attained).

3. The layered greedy algorithm. We now present the algorithm we analyze in this paper. The algorithm generates a sequence of spanning trees on finite restrictions of the graph. We show that this sequence has nice convergence properties.

Algorithm 3.1 Layered greedy algorithm

1: **Input:** A locally finite and connected graph $G = (\mathcal{V}, \mathcal{E})$ with edge costs.
2: **Initialize:** Set $n \leftarrow 1$ and T to be the empty subgraph of G with empty node set and empty edge set.
3: **while** T is not a spanning tree **do**
4: **Find MST on next layer:** Find an MST T^n on layer G_n using Prim's algorithm (for completeness, we give a description of Prim's algorithm below).
5: Set $T \leftarrow T^n$ and $n \leftarrow n + 1$.

While most of the forthcoming analysis of the layered greedy algorithm is agnostic to the particular method used to find the MSTs on the layers in Step 4, Prim's algorithm is instrumental in the early discovery of edges of an MST on G, which we discuss in subsection 6.2. It is one of the classical greedy algorithms for finding MSTs on *finite* graphs (see [3] for further details). In the usual statement of Prim's algorithm, the starting node that initializes the graph is arbitrary. We want ours to proceed from the root node r.

Algorithm 3.2 Prim's algorithm (for finding an MST on G_n)

1: **Input:** Graph $G_n = (L_n, \mathcal{E}_n)$ with edge costs.
2: **Initialize:** Initialize a tree F to be the root node r.
3: **while** F does not span G_n **do**
4: **Append an edge:** Append to F the minimum-cost edge of $\mathcal{E}_n$ emanating from F (that is, having one node in F and one outside of F), breaking ties arbitrarily.

It is important to note that while Prim's algorithm can be leveraged to find the tree iterates T^n on each of the finite graphs G_n, we may remove as well as add edges as we grow the layers G_n. The next example demonstrates this point.

EXAMPLE 1. *Consider the ladder graph in Figure* 1 *with labeled nodes and edge costs written next to the edges. If node* 1 *is the root node, the nodes in layer* 1 *are*

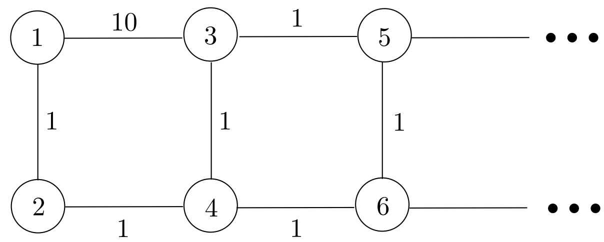

Fig. 1. *Graph for Example 1 illustrating that the layered greedy algorithm is not a greedy algorithm overall.*

nodes 1*,* 2*, and* 3*. The MST of graph* G_1 *consists of edges* $\{1,2\}$ *and* $\{1,3\}$ *for a cost of* 11*. The second layer has node set* $\{1,2,3,4,5\}$*. Now we can avoid the expensive edge* $\{1,3\}$ *to construct the MST of* G_2 *consisting of the edges* $\{1,2\}$*,* $\{2,4\}$*,* $\{3,4\}$*, and* $\{3,5\}$*, for a total cost of* 4*. In other words, the cheapest edges in a given iteration (in this case,* $\{1,3\}$*) may become too expensive by comparison as the subgraph grows, and get dropped in later iterations.* $\triangleleft$

3.1. Some preliminaries. To analyze the performance of the layered greedy algorithm, we need a few preliminaries. First, we start with a classical result in infinite graph theory.

Proposition 3.1 (Proposition 8.1.1 in [10]). *Any locally finite and connected graph (Assumption* 1*) contains a spanning tree.*

Second, we need a mechanism for extending iterates of the layered greedy algorithm, which are not spanning trees of the entire graph G, into spanning trees.

Proposition 3.2. *Suppose* $T^n = (L_n, \mathcal{E}(T^n))$ *is a spanning tree on the connected subgraph corresponding to the* n*-th layer graph* $G_n = (L_n, \mathcal{E}_n)$*. Then there exists a set of edges* $\bar{\mathcal{E}} \subseteq \mathcal{E} \setminus \mathcal{E}_n$ *such that* $(\mathcal{V}, \mathcal{E}(T^n) \cup \bar{\mathcal{E}})$ *is a spanning tree on* G*.*

Proof. Let $\bar{G}$ be the graph obtained by removing from G nodes L_n and all edges incident to them (including both edges $\mathcal{E}_n$ within the n-th layer and the edges connecting nodes in L_n to nodes in $L_{n+1} \setminus L_n$). Each connected component of $\bar{G}$ satisfies Assumption 1, and therefore contains a spanning tree (Proposition 3.1). Moreover, each connected component of $\bar{G}$ contains at least one node that belongs to $L_{n+1} \setminus L_n$ — select one of these nodes in each connected component and select one of the edges connecting it to layer n. Then the union of T^n, the aforementioned spanning trees on the connected components of $\bar{G}$, and the selected edges that connect these connected components to L_n (and thus T^n) is a spanning tree on G. $\square$

Third, we must impose an additional assumption on the cost functional.

Assumption 2. *The edge cost functional* $c : \mathcal{E} \to \Re$ *is such that* $\sum_{e \in \mathcal{E}} |c(e)| < \infty$*.* $\triangleleft$

If we label the costs of the countably many edges in $\mathcal{E}$ by c_ℓ for $\ell = 1, 2, \ldots$, then Assumption 2 becomes $c = (c_1, c_2, \ldots) \in \ell_1$ (where ℓ_1 is the vector space of absolutely summable sequences).

3.2. Convergence in objective value. We are now ready to prove a main result of the paper.

THEOREM 3.3. *Suppose G is a locally finite and connected graph (Assumption 1) whose edge cost functional is absolutely summable (Assumption 2). If G possesses an MST of cost $c^\star$ then the layered greedy algorithm converges in objective value; that is, the sequence T^n of iterates satisfies $c(T^n) \to c^\star$.*

Proof. Let $T^\star$ be an MST of a locally finite connected graph G, and let $T_n^\star$ denote the restriction of $T^\star$ to G_n. By construction, $c(T_n^\star) \to c(T^\star) = c^\star$ as $n \to \infty$. Note that $T_n^\star$ is a forest on G_n, although not necessarily a spanning tree. It can be extended to a spanning tree on G_n with the addition of a finite number of edges (since G_n is a finite graph). Let $\bar{T}_n^\star$ be the cheapest such extension and define

$$\epsilon_n' := c(\bar{T}_n^\star) - c(T_n^\star).$$

Since T^n is an MST on G_n, we have

$$c(T^n) \leq c(\bar{T}_n^\star) = c(T_n^\star) + \epsilon_n'. \tag{3.1}$$

Since $T^\star$ is a spanning tree of G, for every pair of nodes i and j in G, there is a unique finite path P_{ij} connecting them in $T^\star$. Moreover, path P_{ij} must be wholly contained in layer $G_{n_{ij}}$ for $n_{ij} = \max_{k \in P_{ij}} \ell(k)$, where $\ell(k)$ is the number of the smallest layer containing node k. Let

$$m(n) := \max\{m \mid n_{ij} \leq n \text{ for all } i, j \in G_m\}. \tag{3.2}$$

In other words, given n, $m(n)$ is the number of the largest layer such that all pairs of nodes in this layer are connected in $T^\star$ by paths wholly contained in G_n.

Note that none of the edges added to $T_n^\star$ to construct $\bar{T}_n^\star$ are in $G_{m(n)}$, since every pair of nodes in $G_{m(n)}$ is already connected by a path in $T_n^\star$. Hence, $\epsilon_n' = c(\bar{T}_n^\star) - c(T_n^\star) \leq \epsilon_{m(n)}$, where

$$\epsilon_{m(n)} := \sum_{e \in \mathcal{E} \setminus \mathcal{E}_{m(n)}} |c(e)| \tag{3.3}$$

is the sum of the absolute values of costs of edges outside of layer $G_{m(n)}$.

Observe that $m(n) \to \infty$ as $n \to \infty$, which follows from the finiteness of the path P_{ij} between any two nodes i and j and local finiteness and connectedness of G. Hence $\epsilon_{m(n)} \to 0$ as $n \to \infty$ since $c \in \ell_1$ by Assumption 2.

By Proposition 3.2, T^n can be extended to span G. Let S^n denote one such extended spanning tree, with additional edges $\mathcal{E}(S^n) \setminus \mathcal{E}(T^n) \subseteq G \setminus G_n$, and let

$$\Delta_n := c(S^n) - c(T^n).$$

Observe that, by construction, $\Delta_n \to 0$ as $n \to \infty$. Now,

$$c^\star \leq c(S^n) = c(T^n) + \Delta_n \leq c(T_n^\star) + \epsilon_n' + \Delta_n, \tag{3.4}$$

where the first inequality holds since $c^\star$ is the cost of an MST on G and the second inequality holds by (3.1). Since, as $n \to \infty$, $\Delta_n \to 0$, $\epsilon_n' \leq \epsilon_{m(n)} \to 0$, and $c(T_n^\star) \to c(T^\star) = c^\star$, (3.4) implies $c(S^n) \to c^\star$ and $c(T^n) \to c^\star$ as $n \to \infty$, establishing the result. ∎

3.3. Error bound after finite termination. We are also interested in the question of how fast the costs $c(T^n)$ of the iterates T^n approach the optimal value $c^\star$. To provide a partial answer, we need the following additional assumption (which is only made in this subsection and not in the rest of the paper).

ASSUMPTION 3. *The graph $G = (\mathcal{V}, \mathcal{E})$ and the cost function $c : \mathcal{E} \to \Re$ satisfy the following: (i) there exist $\beta \in (0, 1)$ and $\gamma \in (0, +\infty)$ such that for every edge $\{i, j\} \in \mathcal{E}$, $0 \le c_{ij} \le \gamma\beta^{\min\{\ell(i),\ell(j)\}}$, where $\ell(i)$ is the number of the smallest layer containing node i, and (ii) there exists a uniform bound M on the cardinality of node degrees in G, with $M < 1/\beta$.* $\triangleleft$

Under this assumption, we can prove the following.

PROPOSITION 3.4. *Let S^n denote the extensions to spanning trees (via Proposition 3.2) of the iterates T^n produced by the layered greedy algorithm. Under assumptions of Theorem 3.3 and Assumption 3, the errors in cost satisfy the following bound:*

$$0 \le c(S^n) - c(T^\star) \le \frac{M\gamma}{(1-\delta)}(\delta^n + \delta^{m(n)}), \tag{3.5}$$

where $m(n)$ is defined in (3.2) and $\delta = M\beta < 1$.

Proof. This proof refers to several bounds established in the course of the proof of Theorem 3.3. We can bound

$$0 \le c(S^n) - c(T^\star) \le c(T^\star_n) - c(T^\star) + \epsilon'_n + \Delta_n \le \epsilon'_n + \Delta_n, \tag{3.6}$$

where the first inequality follows by optimality of $T^\star$, the second inequality reproduces (3.4), and the last inequality follows because $T^\star_n$ is a subgraph of $T^\star$, and the edge costs are nonnegative by Assumption 3(i).

Recall that, by definition, L_n is the set of all nodes that are at most n edges "away" from the root node r, i.e., for every node in L_n, there exists a path between that node and r that is at most n edges long.

Let $\epsilon_n := \sum_{e\in\mathcal{E}\setminus\mathcal{E}_n} c(e)$ be the sum of the costs of all edges in $\mathcal{E} \setminus \mathcal{E}_n$.[1] From Assumption 3(ii), the number of edges joining layer n to layer $n+1$ is bounded above by M^{n+1}. This follows by induction on the layer number, noticing that the maximum number of nodes in L_n is M times the number of nodes in L_{n-1}. Moreover, the cost of each edge joining layer n to $n+1$ is bounded above by $\gamma\beta^n$, by Assumption 3(i). Combining these observations, we establish

$$\begin{aligned}\Delta_n \le \epsilon_n = \sum_{e\in\mathcal{E}\setminus\mathcal{E}_n} c(e) &\le \sum_{m=n}^{\infty} M^{m+1}\gamma\beta^m \\ &= M\gamma(M\beta)^n \sum_{m=0}^{\infty} (M\beta)^m = M\gamma\delta^n \sum_{m=0}^{\infty} \delta^n = M\gamma(\delta^n/(1-\delta)).\end{aligned}$$

As part of the proof of Theorem 3.3, we showed that ϵ'_n can be bounded above by $\epsilon_{m(n)}$, and so

$$\epsilon'_n \le \epsilon_{m(n)} \le M\gamma(\delta^{m(n)}/(1-\delta)).$$

[1] We introduced similar notation in equation (3.3) in the proof of Theorem 3.3; here, it is no longer necessary to take absolute values since the costs are assumed to be nonnegative.

Substituting these bounds into (3.6), we derive

$$0 \le c(S^n) - c(T^\star) = M\gamma(\delta^n/(1-\delta)) + M\gamma(\delta^{m(n)}/(1-\delta)) = \frac{M\gamma}{(1-\delta)}\left(\delta^n + \delta^{m(n)}\right),$$

as required. ∎

From the proof of Theorem 3.3, we know $m(n) \to \infty$ as $n \to \infty$ and so the error bound in (3.5) converges to 0 as n grows. Of course, there remains the question of assessing the rate at which the sequence $m(n)$ grows with n to further analyze the convergence rate of the algorithm. The growth rate of $m(n)$ depends on the structure of the graph, and different MSTs can give rise to different functions $m(n)$.

Let $L(m)$ be the maximum number of edges over all paths P_{ij} in the tree $T^\star$ connecting nodes i and j in layer G_m. Note that $L(m) < \infty$ since G_m is finite. For $i, j \in G_m$, we have $n_{ij} = \max_{k \in P_{ij}} \ell(k) \le m + L(m)$. Moreover, $\{m \mid n_{ij} \le n \text{ for all } i, j \in G_m\} \supseteq \{m \mid m + L(m) \le n\}$. Hence, $m(n) = \max\{m \mid n_{ij} \le n \text{ for all } i, j \in G_m\} \ge \max\{m \mid m + L(m) \le n\} = \max\{m \mid L(m) \le n - m\}$. Now, note that $L(x)$ is increasing in positive real numbers x so that $\max\{x \mid L(x) \le n - x\}$ is attained at a unique positive real solution $x(n)$ to the equation $L(x) = n - x$. Thus $m(n) = \lfloor x(n) \rfloor$; that is, $m(n)$ is the largest integer less than or equal to $x(n)$. This concrete formula can be used to assess the growth of the bound in (3.5), if one has an understanding of the function $L(m)$ and its connection to the structure of an optimal tree $T^\star$ in specific applications.

REMARK 1. *Since we employ Prim's Algorithm to find an MST in layer G_n, the computational time in iteration n of the layered greedy algorithm is $O(|L_n|^2)$. This, together with* (3.5)*, yields a bound on the computational time to find a spanning tree achieving a cost error within a pre-specified error from optimal.* ◁

4. The finite cycle property.

In Theorem 3.3, we showed that the layered greedy algorithm satisfies convergence in objective value (under Assumptions 1 and 2) whenever the graph possesses an MST. This naturally leads to the question of what graphs possess MSTs (question (Q1) in the introduction). In this section, we describe an elegant sufficient condition (and prove it suffices for existence in the next section).

We say that a graph satisfies the *finite cycle* (FC) property if every edge is contained in at most finitely many cycles of G. The graph in Figure 1 fails the FC property because the edge $\{1, 2\}$ is in infinitely many cycles in the graph. The next example satisfies the FC property.

EXAMPLE 2. *Consider the graph in Figure* 2*. Observe that every edge lies in a unique cycle in the graph, and thus satisfies the FC property.* ◁

We capture the FC property in the following assumption, and refer to this assumption whenever the FC property is invoked later in the paper:

ASSUMPTION 4. *The graph G satisfies the FC property.*

Before moving on to studying the implications of the FC property for the MST problem, we take a brief detour to discuss the simple sufficient condition of absence of bi-rays for a graph to satisfy the FC property. Because the proof of this result will take us off the main path of our development, we put it in an appendix. The reader should be aware, however, that the proof relies on the contents of Section 5.

PROPOSITION 4.1. *If G contains no bi-rays, then G satisfies the FC property.*

Proof. See Appendix A. ∎

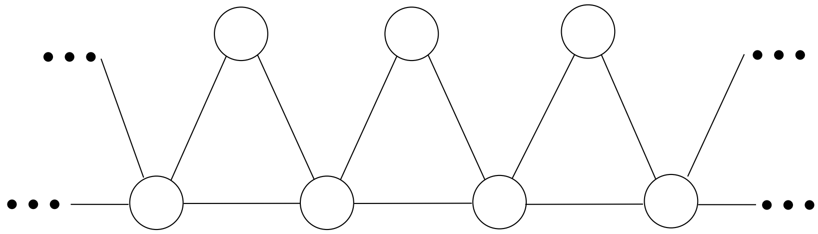

FIG. 2. *A graph where FC holds (see Example 2).*

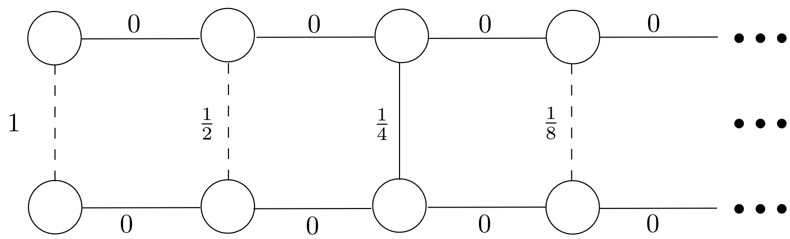

FIG. 3. *A graph with no minimum spanning tree (see Example 3).*

Clearly, the converse of Proposition 4.1 is not true. Consider again the graph in Figure 2. The bottom path connecting all of the "triangle" pieces is a bi-ray, but the graph nonetheless satisfies the FC property.

5. Existence of a minimum spanning tree. Proposition 3.1 shows that a spanning tree always exists, but this does not ensure that an optimal solution to the MST problem (P) exists. Consider the following example.

EXAMPLE 3. *Consider the one-way-infinite ladder graph in Figure 3, with top and bottom rays of 0-cost edges connected by infinitely many rungs with decreasing costs. The most expensive spanning tree has cost 1, consisting of the left-most rung of cost 1 connecting the top and bottom rays. A spanning tree of cost 1/4 is drawn in non-dashed edges in the figure. One can similarly construct spanning trees of cost 1/8, 1/16, etc. Thus, a sequence of spanning trees whose costs converge to 0 can be found in the graph. However, no spanning tree has cost 0 since all edges have nonnegative cost and the 0-cost edges do not form a connected graph. Therefore, a minimum-cost spanning tree does not exist.* ◁

To establish existence, we will use Weierstrass's standard optimization result (see, for instance, Theorem 2.35 in [6]) that minimizing a continuous function over a compact set always yields a minimizer. The challenge here is to develop the appropriate notion of topology to define continuity and compactness.

5.1. The product discrete topology. Our desire to apply Weierstrass's Theorem to (P) motivates the following notion of convergence.[2]

DEFINITION 5.1. *A sequence of subgraphs S^k of graph G converges to a subgraph S in G in the* product discrete topology *if there is a positive integer K_e for each edge $e \in \mathcal{E}$ such that for all $k \geq K_e$, $e \in S^k$ if and only if $e \in S$. We call this the* lock-in

[2]Others use different notions of convergence, mostly based on the fact that they study random graphs and so are interested in probabilistic notions of convergence. See, for instance, [4].

property *of edges of the sequence of subgraphs to the edges of the limiting subgraph.* ◁

We can understand the use of the terminology "product" and "discrete" better in light of the following construction. For each edge $e \in \mathcal{E}$, define a set $B_e := \{0, 1\}$ and endow that set with the discrete metric $d_e(x, y) = 0$ if $x = y$ and 1 if $x \neq y$. That is, $d_e(0, 1) = d_e(1, 0) = 1$ and $d_e(0, 0) = d_e(1, 1) = 0$. Then, clearly, B_e is a metric space under metric d_e. There is a bijection between $\mathcal{P}(\mathcal{E})$ and the product $\prod_{e\in\mathcal{E}} B_e$, where $\mathcal{P}(\mathcal{E})$ is the power set of $\mathcal{E}$. Indeed, any subset H of $\mathcal{E}$ corresponds to an element χ_H of $\prod_{e\in E} B_e$ where $\chi_H(e) = 1$ if $e \in H$ and 0 otherwise (and vice versa). We call χ_H the characteristic function of the subset of edges H.

The product $\prod_{e\in\mathcal{E}} B_e$ can be endowed with the product topology τ of the discrete topologies on B_e for every $e \in \mathcal{E}$. By Theorem 3.36 in [6], the topology τ is metrizable. The significance of this for our purposes is that it suffices to consider subsequences (as opposed to nets) to establish topological properties involving τ. In particular, a set B in $\prod_{e\in\mathcal{E}} B_e$ is closed if every convergent (in τ) sequence χ_k of elements in B has a limit $\chi \in B$. Here, convergence in τ means that for every e, there exists a K_e such that $\chi_k(e) = \chi(e)$ for $k \geq K_e$. Moreover, compactness of a set in B is equivalent to sequential compactness (see Theorem 3.28 in [6]).

Returning to the product discrete topology on $\mathcal{P}(\mathcal{E})$, it can be seen as corresponding to the topology τ on $\prod_{e\in\mathcal{E}} B_e$ under the bijection $H \leftrightarrow \chi_H$. More precisely, a subset H of $\mathcal{P}(\mathcal{E})$ is open in the product discrete topology if and only if the subset $\{\chi_h \mid h \in H\}$ of $\prod_{e\in\mathcal{E}} B_e$ is open in τ. This notion defines a *product discrete topology* on the collection of all subgraphs on G, as defined in Definition 5.1. In particular, if S^k converges to S in the product discrete topology then, for any finite subset of $\mathcal{E}$, the S^k's agree with S on this set of edges for sufficiently large k.

5.2. Cost continuity in the product discrete topology. Having set our topology, we now want to establish the continuity and compactness needed for Weierstrass's Theorem. We start with establishing continuity of the objective function.

LEMMA 5.2. *Suppose the edge cost functional $c : \mathcal{E} \to \Re$ is absolutely summable (Assumption 2). Then $c(\cdot)$ is continuous in the product discrete topology.*

Proof. To establish continuity of $c(\cdot)$, it suffices to show that if a sequence H^k of elements of $\mathcal{P}(\mathcal{E})$ converges to H in the product discrete topology, then $c(H^k) \to c(H)$ in the usual topology on the reals. That is, for an arbitrary $\epsilon > 0$, we want to show that there exists a K_ϵ such that $|c(H^k) - c(H)| < \epsilon$ for all $k \geq K_\epsilon$. Under Assumption 2, there exists a subset E of $\mathcal{E}$ such that $E' = \mathcal{E} \setminus E$ is finite and $\sum_{e\in E} |c(e)| < \epsilon/2$. Since E' is a finite subset of $\mathcal{E}$, there exists a K_ϵ such that H^k agrees with H on all edges in E' for $k \geq K_\epsilon$ by the lock-in property. That is, for all $k \geq K_\epsilon$ we have

$$\begin{aligned}
|c(H^k) - c(H)| &= \left| \sum_{e\in H^k\cap E} c(e) + \sum_{e\in H^k\cap E'} c(e) - \sum_{e\in H\cap E} c(e) - \sum_{e\in H\cap E'} c(e) \right| \\
&= \left| \sum_{e\in H^k\cap E} c(e) - \sum_{e\in H\cap E} c(e) \right| \\
&\leq 2 \sum_{e\in E} |c(e)| < \epsilon.
\end{aligned}$$

This establishes the result. ∎

5.3. Compactness in the product discrete topology. The final ingredient in our existence proof is establishing the compactness of the set of spanning trees. The FC property is crucial to this argument. First, we state a preliminary lemma to establish the compactness of a superset.

LEMMA 5.3. *Let G be a locally finite and connected graph (Assumption* 1*). The space of all subgraphs of G is compact in the product discrete topology τ.*

Proof. Immediate from Tychonoff's theorem (Theorem 2.61 in [6]). ◻

LEMMA 5.4. *Let G be a locally finite and connected graph (Assumption* 1*) that satisfies the FC property (Assumption* 4*). Then, the set of all spanning trees is compact in the product discrete topology.*

Proof. In light of Lemma 5.3, it suffices to show that the set of all spanning trees is closed in the product discrete topology.

Let $S^k, k = 1, 2, \ldots$, be a sequence of spanning trees in G that converges in the product discrete topology to a subgraph S of G. It then suffices to show that S is, itself, a spanning tree. This is achieved in three parts: (i) show S is spanning, (ii) show S is acyclic, and (iii) show S is connected.

To establish (i), observe that if a node i is disconnected from S then each of the edges incident to i can only lie in finitely many of the iterates S^k. Then this means that node i is isolated in S^k for n sufficiently large, a contradiction of the fact that all S^k are connected.

To establish (ii), suppose that S contains a cycle C. Then, since C contains finitely many edges, the lock-in property of convergence in the product discrete topology implies that C is in each S^k for k sufficiently large. This contradicts the fact that each S^k is acyclic.

We now establish (iii). We will show that there is a path from i to j in S for any pair of nodes i and j. By connectedness of each S^k, there are paths P^k connecting i and j in S^k for all k. Consider an arbitrary "reference" path P_{ij} in G connecting i and j. Path P_{ij} contains finitely many edges, and by the FC property, each edge is in at most finitely many cycles in G. Let us collect all these cycles into a finite collection of cycles $\tilde{\mathcal{C}}$, and let $\mathcal{C} := \{C \setminus P_{ij} \mid C \in \tilde{\mathcal{C}}\}$. That is, for every cycle $C \in \tilde{\mathcal{C}}$, the subset of edges of C that are not in the reference path P_{ij} is an element of $\mathcal{C}$. Again by the FC property, $\mathcal{C}$ is a finite collection of subsets of edges in G.

Observe that each P^k arises by taking some edges from P_{ij} and some subsets of edges from $\mathcal{C}$ (in the degenerate cases, P^k can exactly equal P_{ij} or just be composed of subsets of edges taken from $\mathcal{C}$). Thus, there are only finitely many possibilities for the structure of P^n since $\mathcal{C}$ is a finite collection and P_{ij} has finitely many edges. According to the pigeonhole principle, infinitely many of the P^k are thus equal and so a subsequence of them converges in the product discrete topology to a path P that connects i and j. Since we have assumed that the S^k converge to S in the product discrete topology, this implies that P is in S and so i and j are connected in S. This implies that S is connected. ◻

THEOREM 5.5. *Consider the minimum-cost spanning tree problem* (P) *and suppose G is a locally finite and connected graph (Assumption* 1*) with the FC property (Assumption* 4*) and with costs that are absolutely convergent (Assumption* 2*). Then, an MST (i.e., an optimal solution to* (P)*) exists.*

Proof. Note that (i) the objective function of (P) is continuous in the product discrete topology by Lemma 5.2, and (ii) the feasible region is compact in the product discrete topology by Lemma 5.4. The result then follows by Weierstrass's theorem

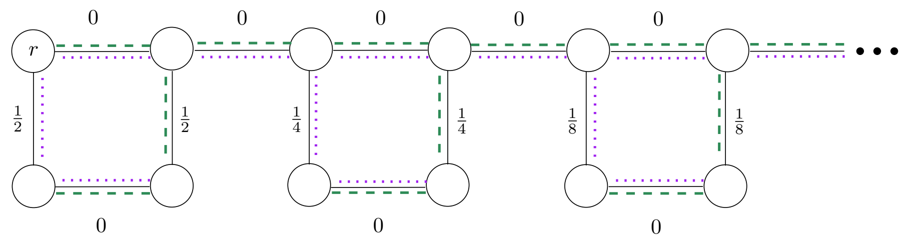

FIG. 4. *Graph for Example 4 illustrating that the layered greedy algorithm fails to find an optimal MST even when one exists.*

(Theorem 2.35 in [6]). ∎

The above result implies that if the graph G has the FC property, then the layered greedy algorithm can be used to find a sequence of trees in G that converges to optimality in objective value (combining Theorems 3.3 and 5.5).

6. Solution convergence.

In the previous section, we showed that if a graph is locally finite, connected, and satisfies the FC property with summable costs (Assumptions 1, 2, and 4) then the layered greedy algorithm always achieves convergence in objective value. However, this does not imply that the iterates of the graph converge to an MST. Consider the following example.

EXAMPLE 4. *Consider the graph in Figure 4, which satisfies Assumptions 1, 2, and 4. If we apply the layered greedy algorithm, there is a tie between the two identical-cost vertical edges within each four-node cycle contained in the layer. Suppose for T^n with n odd, the algorithm chooses the "left" edges (shown as the dotted (purple) edges in Figure 4), and for T^n with n even, the algorithm chooses the "right" edges (shown as the dashed (green) edges in Figure 4). Then the sequence T^n does not converge in the product discrete topology at all, let alone to an MST. Thus, the iterates of the layered greedy algorithm can fail to converge.* ◁

6.1. Solution convergence when there is a unique MST.

One sufficient condition to avoid pathological behavior illustrated in Example 4 is having a unique MST in the graph.

THEOREM 6.1. *Suppose G is a locally finite and connected graph (Assumption 1) that satisfies the FC property (Assumption 4) and whose edge cost functional is absolutely summable (Assumption 2). If G possesses a* unique *MST T^* then the iterates of the layered greedy algorithm converge to T^* in the product discrete topology.*

Proof. Let T^n be the n-th iterate of the layered greedy algorithm. By Proposition 3.2 each iterate can be extended to a spanning tree S^n of G. Suppose, by way of contradiction, that the sequence S^n does not converge to T^* in the product discrete topology. By the compactness of the set of spanning trees (Lemma 5.4), a subsequence S^{n_k}, $k = 1, 2, \ldots$, converges to a spanning tree T' where $T' \neq T^*$. By convergence in objective value (Theorem 3.3) and continuity (Lemma 5.2), we conclude that T' is also an MST. Since T^* is the unique MST, this is a contradiction. ∎

The following simple assumption is sufficient to ensure that a graph has at most one MST:

ASSUMPTION 5. *The graph G has distinct edge costs; that is, for every two distinct edges $\{i, j\}$ and $\{i', j'\}$ we have $c_{ij} \neq c_{i'j'}$.* ◁

To prove uniqueness under Assumption 5, we need the following generalization of a well-known condition in finite graphs (see, for instance, Theorem 13.1 in [3]).

PROPOSITION 6.2 (Cut optimality condition). *If $T^\star$ in an MST of a locally finite and connected (Assumption 1) graph G then for all $\{i, j\} \in T^\star$, $c_{ij} \leq c_{k\ell}$ for any edge $\{k, \ell\}$ crossing the cut formed by deleting edge $\{i, j\}$ from $T^\star$.*

Proof. Suppose the condition is not satisfied for some $\{i, j\} \in T^\star$, and edge $\{k, \ell\}$ with $c_{ij} > c_{k\ell}$ crosses the cut formed by deleting $\{i, j\}$ from $T^\star$. Then, replacing $\{i, j\}$ by $\{k, \ell\}$ in $T^\star$ creates a spanning tree that is cheaper, implying that $T^\star$ is not an MST. □

THEOREM 6.3. *Let G be a locally finite and connected graph (Assumption 1) with distinct arc costs (Assumption 5). If an MST exists for G then this MST is unique.*

Proof. To show uniqueness, suppose S and T are two distinct MSTs (at least one is guaranteed to exist by assumption), and let $\{i, j\} \in S \setminus T$. Furthermore, let $\{k, \ell\} \in T$ be in the cut created in G by removing $\{i, j\}$ from S. Since S and T are both MSTs, they both satisfy the cut optimality condition (Proposition 6.2), i.e., $c_{ij} \leq c_{k\ell}$ and $c_{k\ell} \leq c_{ij}$, implying that $c_{ij} = c_{k\ell}$. This is a contradiction, establishing that $S = T$. □

This result (via Theorem 6.1) shows that when we apply the layered greedy algorithm to a locally finite, connected graph with the FC property and absolutely summable *distinct* edge costs, then the algorithm's iterates converge to an MST, i.e., it provides an affirmative answer to question (Q2). Moreover, for each edge, we get lock-in after finitely many iterations via convergence in the product discrete topology.

6.2. Discovery of early edges of an MST. Of course, we would like a stronger convergence result than Theorem 6.1 in the following sense. Convergence in product discrete topology tells us that every edge *eventually* locks into an edge of an MST of G, but it would be better if we had a verifiable sufficient condition for when an edge has locked in. As we will see, the layered view of the graph and the nature of Prim's algorithm allow us to provide some partial results in this area.

In what follows, we adopt Assumption 5 that the graph has distinct edge costs. By Theorem 13.1 in [3], which is the finite-graph version of Theorem 6.3, this implies that for every n, T^n is the unique MST of the graph G_n and moreover, there will be no tie-breaking in Step 4 of Prim's algorithm.

With this assumption, we can make the following simple, yet powerful, observation. Since in each iteration of the layered greedy algorithm the iterate T^n is constructed via Prim's algorithm, and because Prim's algorithm always starts with the root node and grows the tree T^n from there, the uniqueness in the choice of T^n greatly restricts the possibility of deviation in the "early" edges among the iterates T^n. The next result formalizes this idea.

Let e^n_k be the k-th edge added by Prim's Algorithm applied to G_n initialized with the root node r, where $k = 1, 2, \ldots, |L_n| - 1$. We add a little more interpretation here for clarity. We are executing the layered greedy algorithm and are on its n-th iteration; that is, we are constructing T^n on the graph G_n of layer n. In Step 4 of the layered greedy algorithm, there is a call to Prim's algorithm to construct T^n. The subscript k in e^n_k refers to the k-th iteration of Prim's algorithm *within* Step 4 of the layered greedy algorithm.

Let $k_n^* = \max_{1\le k\le |L_n|-1}\{k \mid e_\ell^n \in \mathcal{E}_{n-1}, \ell = 1,2,\ldots,k\}$, i.e., the last iteration of Prim's algorithm applied to G_n before an edge that is *not* contained in G_{n-1} is selected. Since Prim's algorithm is initialized with the root node, $1 \le k_n^* < |L_n| - 1$ for $n > 1$ (we let $k_1^* = 0$). Furthermore, let

$$F_n^* = \{e_\ell^n,\ \ell = 1,2,\ldots,k_n^*+1\}. \tag{6.1}$$

In other words, F_n^* is the set of edges added by Prim's algorithm applied to G_n up to and including the first edge that connects a node in L_{n-1} and a node in $L_n \setminus L_{n-1}$, namely $e_{k_n^*+1}^n \in \mathcal{E}_n$.

PROPOSITION 6.4. *Suppose G is a locally finite and connected graph (Assumption 1) with distinct edge costs (Assumption 5). Then $F_n^* \subseteq T^m$, for $m \ge n$ and $n = 1,2,\ldots$, where F_n^* is defined in* (6.1).

Proof. Consider an arbitrary $n \ge 1$ and arbitrary $m \ge n$. For $n = 1$, the result is trivially true, since in this case F_n^* will include only the cheapest edge incident to the root node, and this edge will be added as the first iterate of each application of Prim's algorithm. Consider now $n > 1$ and $m \ge n$. We will show that $e_\ell^m = e_\ell^n$ for all $\ell = 1,2,\ldots,k_n^*+1$, which implies that $F_n^* \subseteq T^m$. We will prove this by mathematical induction on ℓ. The claim is clearly true for $\ell = 1$ since the minimum-cost edge emanating from node r is the same for all graphs G_m with $m \ge 1$. Adopt the inductive hypothesis that $e_\ell^m = e_\ell^n$ for all $\ell = 1,2,\ldots,k$ for some $k \le k_n^*$. Then Prim's Algorithm, before its $k+1$-st iteration, has created trees identical to $F_k := \{e_\ell^n,\ \ell = 1,2,\ldots,k\} \subseteq G_{n-1}$ when applied to graphs G_n and G_m for $m \ge n$. Then the $k+1$-st iteration of Prim's algorithm for both graphs finds the same minimum-cost edge e_{k+1}^n out of F_k since all edges emanating from F_k in G_m are in $\mathcal{E}_n$ for all $m \ge n$, thus restoring the inductive hypothesis. ∎

REMARK 2. *The distinct arc costs assumption (Assumption 5) is important to the above result as it ensures that different calls to Prim's algorithm do not need to make tie-breaking decisions and potentially select different edges on earlier layers of the graph.* ◁

If the graph possesses an MST T^*, we can further demonstrate that all edges of F_n^* are guaranteed to be in the set $\mathcal{E}^*$ of edges of T^*.

COROLLARY 6.5. *Suppose G is a locally finite and connected graph (Assumption 1) with distinct edge costs (Assumption 5) and (a unique) MST $T^* = \{\mathcal{V}, \mathcal{E}^*\}$ exists. Then $F_n^* \subseteq \mathcal{E}^*, n = 1,2,\ldots,$ where F_n^* is defined in* (6.1).

Proof. Let $T^*(n)$ be the smallest connected finite subtree of T^* that contains all nodes of layer n, and let $G^*(n)$ be the subgraph of G spanned by $T^*(n)$. It is easy to show (e.g., by contradiction) that $T^*(n)$ is an MST of $G^*(n)$; moreover, it is a unique MST due to Assumption 5. Applying Prim's algorithm to $G^*(n)$ starting with the root node, we will generate F_n^* on the way to generating $T^*(n)$, since $G_n \subseteq G^*(n)$. Hence $F_n^* \subseteq T^*(n) \subseteq \mathcal{E}^*$. ∎

Corollary 6.5 provides a basic sufficient condition for an edge e to lie in an MST under appropriate assumptions: if $e \in F_n^*$ for some n, then e is an edge of an MST. This condition can be readily verified by running Prim's algorithm until it first reaches outside the layer that contains e and checking whether e has been added to T^n by this point. Therefore, we have a partial answer to question (Q3).

It is important to stress that this condition is only sufficient. If an edge e *does not* lie in F_n^* for any n, this does not mean that e is not an edge of any MST. A simple

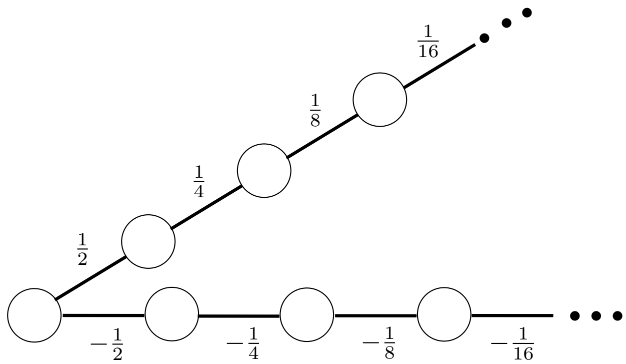

Fig. 5. *A graph with some minimum spanning tree edges that do not satisfy the sufficient condition in Corollary* 6.5 *(see Example* 5*).*

example illustrates this point.

Example 5. *Consider the graph in Figure* 5 *and let the node in the bottom left corner be the root node. Clearly, this graph satisfies the assumptions of Corollary* 6.5*, and its single minimum spanning tree is the whole graph itself. In the* n*-th iteration of the layered greedy algorithm, Prim's algorithm selects every available negative-cost edge before selecting any positive-cost edge. This implies that edge count* K_n^* *is reached* before *a single positive-cost edge is reached. This implies that the positive-cost edges do not lie in* F_n^**, even though they are part of the minimum spanning tree. This implies that the sufficient condition in Corollary* 6.5 *cannot identify the positive-cost edges of this graph as belonging to the minimum spanning tree.* ◁

In the next set of results, we build on Proposition 6.4 and Corollary 6.5 to identify scenarios where we can tell that an entire iterate T^n of the layered greedy algorithm lies in T^*.

Corollary 6.6. *Suppose* G *is a locally finite and connected graph (Assumption* 1*) with distinct edge costs (Assumption* 5*) and (a unique) MST* $T^* = \{\mathcal{V}, \mathcal{E}^*\}$ *exists. Suppose*

$$\min_{e \in \mathcal{K}_{\bar{n}}} c(e) > \max_{e \in \mathcal{E}_{\bar{n}}} c(e) \tag{6.2}$$

for some $\bar{n} > 1$*, where* $\mathcal{K}_{\bar{n}} := \{\{i,j\} : i \in L_{\bar{n}} \text{ and } j \in L_{\bar{n}+1} \setminus L_{\bar{n}}\}$*. Then all edges of layered greedy iterate* $T^{\bar{n}}$ *lie in every subsequent iterate* T^n*,* $n \geq \bar{n}$*, and therefore,* $T^{\bar{n}}$ *is contained in* T^**.*

Proof. Observe that (6.2) ensures that all edges of $T^{\bar{n}}$ lie in $F^*_{\bar{n}+1}$, since this condition implies that, when Prim's algorithm is applied to layer $\bar{n}+1$ and beyond, all nodes *within* layer $\bar{n}$ get spanned before any node outside of this layer is reached. The rest of the argument follows by Proposition 6.4 and Corollary 6.5. ∎

It is straightforward to see that condition (6.2) fails in the graph in Figure 5. The next example provides a case where condition (6.2) holds.

Example 6. *To illustrate condition* (6.2)*, consider the graph in Figure* 6 *that is adapted from Figure 16.7 in [3]. We can see that condition* (6.2) *holds for* $\bar{n} = 2$ *since* $\min\{45, 50, 60\} > \max\{35, 40, 25, 10, 20, 15, 30\}$*. Thus, the layered greedy algorithm locks in the edges of* T^2 *starting with iteration 3. In this case, these edges have costs*

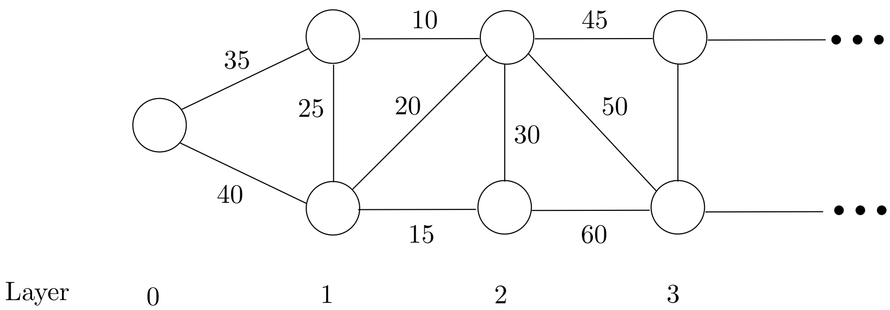

FIG. 6. *An example that satisfies condition* (6.2) *in Corollary* 6.6.

35, 10, 20, and 15, and they are guaranteed to be in T^* independently of the structure and costs of G after layer 3 (aside from ensuring that assumptions of Corollary 6.6 *hold).* ◁

Condition (6.2) can be interpreted as follows: the edges in $\mathcal{K}_{\bar{n}}$ create a "mountain range" or a "ridge" of costs, while all edges within the subgraph $G_{\bar{n}}$ form a cost "valley"; as a result, all the nodes in the valley should be spanned before the MST ventures across the ridge.

Note that in a graph with positive edge costs, this condition cannot hold for all n, or even for an infinite subsequence of n, and satisfy the other assumptions imposed on our graphs. Indeed, for (6.2) to hold on an infinite subsequence n_k, $k = 1, 2, \ldots$, we must have a subsequence of edges with costs that are increasing. But this condition violates Assumption 2, which may be needed to establish existence of an MST, since it requires the sequence of edge costs to converge to 0 for them to be summable.

Luckily, we can provide a modification of condition (6.2) that can hold on a subsequence of layers without contradicting Assumption 2 while providing a workable approach to identifying early edges in T^*. The new condition is discussed in Corollary 6.7 and illustrated in Figure 7.

COROLLARY 6.7. *Suppose G is a locally finite and connected graph (Assumption* 1*) with distinct edge costs (Assumption* 5*), and (a unique) MST $T^* = \{\mathcal{V}, \mathcal{E}^*\}$ exists. Suppose further that there is an increasing sequence n_k, $k = 1, 2, \ldots$, with $n_1 > 1$, that satisfies the following conditions:*

$$(6.3) \qquad \min_{e \in K_{n_1}} c(e) > \max_{e \in \mathcal{E}_{n_1}} c(e), \text{ and } \min_{e \in K_{n_k}} c(e) > \max_{e \in \mathcal{E}(n_{(k-1)}, n_k)} c(e) \text{ for } k > 1,$$

where $\mathcal{E}(n, m) = \mathcal{E}_m \setminus (\mathcal{E}_n \cup \mathcal{K}_n)$ for $n < m$, i.e., it is the set of all edges of G with both endpoints in layer m, but outside layer n (thus extending notation $\mathcal{E}_m = \mathcal{E}(0, m)$). Furthermore, assume that whenever the set $L_{n_k} \setminus L_{n_{(k-1)}}$ contains more than one node, this node set is connected in the graph induced by $\mathcal{E}(n_{(k-1)}, n_k)$. Then, for all $k = 1, 2, \ldots$, all edges of layered greedy iterate T^{n_k} lie in every subsequent iterate T^n, $n \geq n_k$, and therefore, T^{n_k} is contained in T^*.

EXAMPLE 7. *Consider the graph in Figure* 7*. We have $n_1 = 1$, since*

$$\max\left\{\frac{1}{2}, 1\right\} < \min\left\{1 + \frac{1}{2}, 1 + \frac{1}{4}, 1 + \frac{1}{8}\right\},$$

and T^1 consists of the two edges emanating from the root node. It is easy to see that

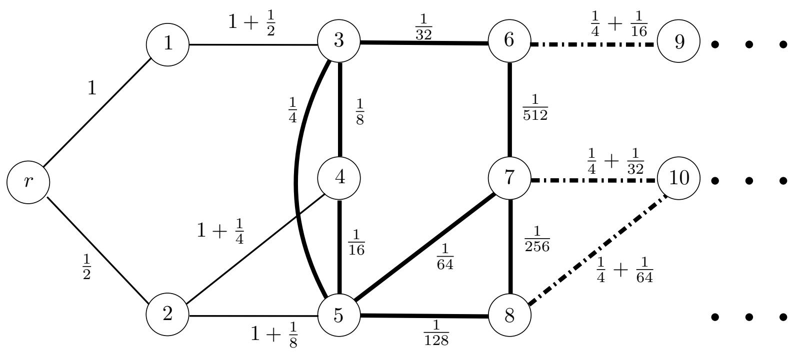

FIG. 7. *Graph for Example 7 illustrating the notation defined in Corollary* 6.7. *Here,* $n_1 = 1$ *and* $n_2 = 3$. *The edges in* K_{n_2} *are dashed. The edges in* $\mathcal{E}(n_1, n_2)$ *are in bold. It is easy to see that* (6.3) *is satisfied for* n_1 *and* n_2.

Prim's algorithm applied to any G_n *with* $n \geq 1$ *in this example will begin by adding these two edges, which are therefore locked in.*

Furthermore, $n_2 = 3$ *satisfies* (6.3)*, since the most expensive of the bold edges has cost* $\frac{1}{4}$*, and the cheapest of the dashed edges has cost* $\frac{1}{4} + \frac{1}{64}$. T^3 *consists of edges with costs* $\frac{1}{2}$, 1, $1 + \frac{1}{8}$, $\frac{1}{128}$, $\frac{1}{256}$, $\frac{1}{512}$, $\frac{1}{32}$, *and* $\frac{1}{16}$ *(listed here in the order they are added by Prim's algorithm). The application of Prim's algorithm to construct* T^4 *will also begin by adding these edges.*

Notice, however, that T^2 *contains the edge with cost* $\frac{1}{8}$*, which is not included in the subsequent iterates, illustrating that the result in Corollary* 6.7 *is only guaranteed to hold on the specified subsequence.*

Proof of Corollary 6.7. We will prove, by induction on k, that $T^{n_k} \subset F^*_{n_k+1}$ for $k = 1, 2, \ldots$. For $k = 1$, condition (6.3) coincides with (6.2), and this conclusion follows by Corollary 6.6. For $k > 1$, let us adopt the inductive hypothesis that $T^{n_{(k-1)}} \subset F^*_{n_{(k-1)}+1}$, and show that $T^{n_k} \subset F^*_{n_k+1}$.

If the set $L_{n_k} \setminus L_{n_{(k-1)}}$ consists of a single node (say, v), the claim is trivially true, since then $n_k = n_{(k-1)} + 1$, T^{n_k} consists of $T^{n_{(k-1)}}$ combined with the cheapest edge connecting $L_{n_{(k-1)}}$ with v, and $F^*_{n_k+1}$ consist of T^{n_k} combined with the cheapest edge connecting v with a node in L_{n_k+1}. We will therefore consider the case when $L_{n_k} \setminus L_{n_{(k-1)}}$ contains multiple nodes.

As before, let e^n_m be the edge added by the m-th iteration of Prim's algorithm applied to G_n. To prove our claim, we need to show that, for $m = 1, \ldots, |L_{n_k}| - 1$,

$$e^{n_k+1}_m = e^{n_k}_m. \tag{6.4}$$

By the inductive hypothesis, (6.4) is true for all $m \leq |L_{n_{(k-1)}}| - 1$ (while both Prim's algorithms are constructing $T^{n_{(k-1)}}$) and for $m = |L_{n_{(k-1)}}|$ (when they both add the cheapest edge from $\mathcal{K}_{n_{(k-1)}}$ to reach $L_{n_{(k-1)}+1}$, thus completing $F^*_{n_{(k-1)}+1}$).

We now construct an induction on ℓ where we suppose (6.4) is true for all $m \leq \ell$, where $|L_{n_{(k-1)}}| \leq \ell < |L_{n_k}| - 1$, and consider the edges each algorithm chooses from in iteration $\ell + 1$. During the first ℓ iterations, the algorithms have spanned, using the same edges, all of $L_{n_{(k-1)}}$ and a strict subset $\mathcal{V}_\ell$ of $L_{n_k} \setminus L_{n_{(k-1)}}$. Let $\mathcal{V}' = (L_{n_k} \setminus L_{n_{(k-1)}}) \setminus \mathcal{V}_\ell$ — these are precisely the nodes of L_{n_k} that have not yet been spanned.

We now prove the inductive step in iteration $\ell + 1$. In that iteration, Prim's algorithm applied to G_{n_k} is comparing the costs of edges in $\mathcal{K}_{n_{(k-1)}}$ incident to nodes in $\mathcal{V}'$ and edges connecting nodes in $\mathcal{V}_\ell$ to nodes in $\mathcal{V}'$, while the algorithm applied to G_{n_k+1} is comparing the costs of all the aforementioned edges as well as any edges in $\mathcal{K}_{n_k}$ incident to nodes in $\mathcal{V}_\ell$. Due to the assumption that node set $L_{n_k} \setminus L_{n_{(k-1)}}$ is connected in $\mathcal{E}(n_{(k-1)}, n_k)$, at least one of the edges from this edge set is considered in the cost comparison by both algorithms and by (6.3), it will be cheaper than any edge in $\mathcal{K}_{n_k}$. Therefore, Prim's algorithm applied to G_{n_k+1} will not choose an edge from $\mathcal{K}_{n_k}$ until all nodes in L_{n_k} have been spanned, i.e., until it constructs the MST T^{n_k}. This establishes (6.4) for $\ell+1$ and completes our induction on ℓ, which in turn closes the outer induction on k.

The rest of the argument follows by Proposition 6.4 and Corollary 6.5. ◻

This last corollary shows that the MST T^* can be constructed by building the smaller finite trees T^{n_k} where later iterations do not add or remove edges from the layer of G spanned by the T^{n_k} uncovered so far.

It is worth noting that assumptions of Corollary 6.7 and Assumption 2 can be met simultaneously in graphs with positive costs. Roughly speaking, condition (6.3) only requires that, occasionally, costs of edges connecting to a new layer form a "ridge," but only relative to the costs of edges in the previous valley. However, the heights of the subsequent ridges K_{n_k} can get smaller as long as the subsequent valleys $\mathcal{E}(n_{(k-1)}, n_k)$ also get more shallow.

Corollaries 6.6 and 6.7 provide additional partial answers to question (Q3).

7. An application: High-speed information channels. In this subsection, we illustrate how the results in this section can be used to solve a minimum spanning tree problem on an infinite graph that arises from an application. The infinite graph models an underlying indefinite but large finite graph whose nodes we expect to ultimately connect via a spanning tree of telecommunication links.

Suppose in particular a telecommunications company is building high-speed information channels (e.g., via laying fiber-optic cables) to connect a large number of locations to a single service provider at minimum cost. The collection of these locations is modeled as countably infinite since the goal is to connect discrete locations over a long but uncertain life of the project. For more discussion of using infinite graphs to study infinite-horizon optimization problems see [21]. The cost of an edge $\{i, j\}$ is the cost of building an information channel that directly connects location i and location j.

We view the layers of the graph as nodes reached by edges over time. The first layer consists of locations that can be connected to the root node (the service provider location) in a certain interval of time, say, 1 year. The second layer consists of locations that can be connected to the root node (via a node in layer 1) in two time periods, say 2 years. Under this time interpretation of layering, it follows that each node has finite degree, since in finite time a location can only be connected to finitely many other locations. This supports Assumption 1. As for Assumption 2, it is natural to assume that future costs are discounted by a discount factor that assures summable costs. These two assumptions then assure that the layered greedy algorithm will find a sequence of spanning-tree iterates that converge in value to optimality.

The nature of the layered greedy algorithm, however, is that the edges in the tree iterates will shift around, as we saw in Example 1. For an application like laying fiber-optic cable, such "shifting around" can lead to very expensive reworking requiring removal of previously added edges. We would prefer to be able to apply a rolling horizon approach to this problem. In particular, we would like to be able

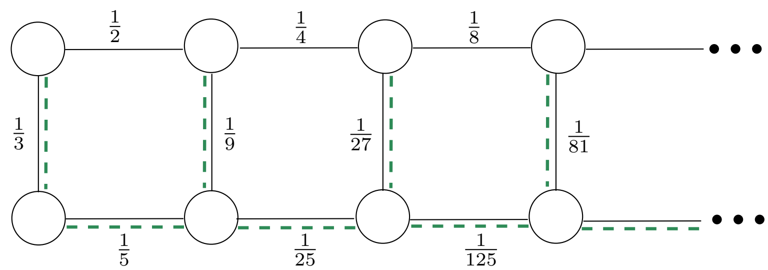

Fig. 8. *A graph with an MST that fails the FC property.*

to finalize our decisions of which potential edges within a few initial layers will and will not be built based on whether they are included in T_{n_1} for some small n_1 (and proceed to lay cable along the chosen edges during the first few years of construction); then finalize the decisions regarding the edges in the next few layers based on T_{n_2}, for some $n_2 > n_1$, etc., without sacrificing optimality of the overall spanning tree that is being constructed.

If we assume more about the underlying graph, we can get stronger convergence results. These assumptions are in fact quite natural in our setting. The condition of distinct edge costs (Assumption 5) is easy to guarantee since it is unlikely that two projects to connect two different pairs of locations have exactly the same costs. The recursive ridges and valleys condition (6.3) is natural in this application, with "valleys" and "mountain ranges" representing either the actual topography of the area or the difference in difficulty and costs of laying cable with and without pre-existing underground conduits. We may assume the costs are summable if we take time discounting into consideration, so even though "far off" mountains may be high, their costs will be sufficiently discounted. Finally, the connectedness assumption of Corollary 6.7 is natural if the population of the valleys is dense enough to allow it to be connected by cheap local infrastructure. Accordingly, we can apply the result of Corollary 6.7 ensuring that we can construct the MST recursively in finite subtrees whose edges become stable at finite intervals (the associated sequence $\{n_k\}$) without edges entering or leaving the MST.

8. Conclusion. In this paper, we gave an algorithm that yields convergence in objective value for a broad class of infinite graphs (locally finite and connected) that works as long as an MST is known to exist (Theorem 3.3). We offer the combination of the FC property on the graph and absolute summability of the costs as a sufficient condition for existence, but acknowledge that these are not necessary conditions. Indeed, consider the graph in Figure 8. It satisfies the properties of absolutely summable and distinct edge costs but fails the FC property. Nonetheless, an MST exists, as indicated in dashed (green) edges. An interesting open question is whether there is a meaningful characterization of when an MST exists in a locally finite and connected graph that is weaker than the FC property, or substantially different from it.

In this paper, we also showed convergence of the layered greedy iterates in the scenario where there exists a unique MST (Theorem 6.1). Unlike in many other optimization problems, where the uniqueness of the optimal solution is hard to verify, this problem has the simple sufficient condition of unique edge costs. We also showed in Example 4 that if there is more than one MST then the iterates of the layered greedy algorithm may fail to converge to an MST. The convergence issue arose because of an "unfortunate" selection of edges of equal cost as the algorithm proceeds. We believe that this "selection" issue could potentially be resolved, using an approach similar in spirit to [23]. We will leave this for future work.

Finally, we explored a verifiable sufficient conditions that allow us to confirm whether an iterate of the layered greedy algorithm has "locked in," i.e., verify that all its edges will be contained in all of the future iterates (and thus the optimal MST if it exists).

Appendix A. Appendix: Proof of Proposition 4.1.

We start with the following preliminary lemma.

LEMMA A.1. *If a locally finite and connected graph G contains no bi-rays, then every pair of rays must have infinitely many nodes in common.*

Proof. Let $(i_1, i_2, \ldots)$ and $(j_1, j_2, \ldots)$ be two rays in the graph, and suppose they have at most finitely many nodes in common. If they have no nodes in common, then a bi-ray is produced by connecting nodes i_1 and j_1. Otherwise, let $k = i_m = j_n$ for some m and n be the last node they share, so that rays $(k, i_{m+1}, i_{m+2}, \ldots)$ and $(k, j_{n+1}, j_{n+2}, \ldots)$ are distinct except for node k. Then the union of these rays is a bi-ray, a contradiction. ∎

LEMMA A.2. *The collection of all paths and rays in a locally finite and connected graph that contains no bi-rays is compact in the product discrete topology.*

Proof. Observe that a subgraph is a path or a ray if and only if it is a connected and acyclic subgraph where each node has degree at most two. (Bi-rays also have these properties, but we are assuming that our graph has no bi-rays.) Let P^k, $k = 1, 2, \ldots,$ be a sequence of paths and rays that converges in the product discrete topology to some subgraph P. We claim that P has no cycles, is connected, and each node in P has degree at most 2, i.e., P is either a path or a ray.

The proof that P is acyclic follows the same logic as claim (ii) in Lemma 5.4 using the lock-in property of convergence.

Next, suppose P has a node of degree 3 or greater. Again, by lock-in, this implies that infinitely many of the P^k also have a node of degree 3 or greater, contradicting the fact they are paths or rays.

Finally, we establish by contradiction that P is connected. Suppose there are two nodes $i, j \in P$ that are not connected in P. Since these two nodes are in P, P contains at least one edge incident to i and at least one edge incident to j. This means that, for sufficiently large k, each P^k contains those edges and thus contains both nodes i and j; we can pass to a subsequence to make this claim for all k. Let P^k_{ij} be the path that connects i and j in P^k.

Let $i^k_1 \in I(i)$ be such that $\{i, i^k_1\} \in P^k_{ij}$. By the pigeonhole principle, one of these edges locks in, so that for some $i_1 \in I(i)$, $\{i, i_1\} \in P^k_{ij}$ for k sufficiently large, and thus $\{i, i_1\} \in P$. Note that $i_1 \neq j$ by our assumption. Let us pass to a subsequence so that $\{i, i_1\} \in P^k_{ij}$ for all k.

We continue following each of the paths P^k_{ij} from i_1 towards j. Consider nodes $i^k_2 \in I(i_1)$ such that $i^k_2 \neq i$ and $\{i_1, i^k_2\} \in P^k_{ij}$. Following the same logic, one of these edges, denoted $\{i_1, i_2\}$, is contained in all paths P^k_{ij} for sufficiently large k, and thus is contained in P. Note that $i_2 \neq j$ and, since P is acyclic, $i_2 \neq i$.

We will repeat the above process iteratively. At each step, we will continue following the paths P^k_{ij} towards j from the most-recently identified node i_m, and adding a node i_{m+1} such that the edge sequence $(\{i, i_1\}, \{i_1, i_2\}, \ldots, \{i_{m-1}, i_m\}, \{i_m, i_{m+1}\})$ is in P^k_{ij} for all (sufficiently large) k, and thus is in P. Since i_{m+1} is different from i_m by construction, and from every other identified node since P^k_{ij} is acyclic, this process will create a ray $R_i = (\{i, i_1\}, \{i_1, i_2\}, \ldots) \subset P$ that does not include node j.

Using the same process starting from j, we can create a ray

$$R_j = (\{j, j_1\}, \{j_1, j_2\}, \ldots) \subset P$$

that does not include node i. Moreover, this ray has no nodes in common with R_i, since otherwise there is a path connecting i and j in P. This, however, contradicts Lemma A.1 in a graph with no bi-rays, thus establishing that P is connected. □

Proof of Proposition 4.1. Suppose G is a locally finite and connected graph with no bi-rays. By way of contradiction, suppose there exists an edge $\{i, j\} \in \mathcal{E}$ that is contained in infinitely many cycles. Deleting the edge from those cycles, we conclude that there are infinitely many distinct paths P^n_{ij}, $n = 1, 2, \ldots$, connecting i and j.

Observe that there must be an infinite subsequence $P^{n_k}_{ij}$, $k = 1, 2, \ldots$, such that $P^{n_{k+1}}_{ij}$ contains strictly more edges than $P^{n_k}_{ij}$, for all k. Suppose otherwise, that there is a maximum number N of edges in all paths between nodes i and j. By local finiteness, there are finitely many potential paths of length N leaving node i. However, we have supposed there are infinitely many paths of length N leaving node i and reaching node j. Hence, such a sequence $P^{n_k}_{ij}$, $k = 1, 2, \ldots$, exists.

Let N_k, $k = 1, 2, \ldots$, denote the increasing sequence of cardinalities of the edge sets of paths $P^{n_k}_{ij}$, and let m_k be the $\lfloor N_k/2 \rfloor$-th node in the path $P^{n_k}_{ij}$. Break each $P^{n_k}_{ij}$ into two subpaths, $P^{n_k}_i$ and $P^{n_k}_j$, where $P^{n_k}_i$ connects node i and node m_k, and $P^{n_k}_j$ connects node j and node m_k; i.e., $P^{n_k}_i$ and $P^{n_k}_j$ have only node m_k in common. Passing to subsequences if necessary and using Lemma A.2, sequences $P^{n_k}_i$ and $P^{n_k}_j$ each have a limit P_i and P_j, respectively, that are either paths or rays. Moreover, by the construction of $P^{n_k}_i$ and $P^{n_k}_j$, they cannot converge to limits with finitely many nodes, and so P_i and P_j must be rays.

Our contradiction comes from the properties of rays P_i and P_j. We argue that P_i and P_j have at most one node in common. Suppose otherwise that P_i and P_j have at least two nodes in common, say, u and v. Then P_i contains a finite path p_i between u and v and P_j contains a finite path p_j between u and v. There are two cases to consider. The first is where p_i and p_j share an edge. In this case, by the lock-in property, $P^{n_k}_i$ and $P^{n_k}_j$ both contain that edge for large enough k, contradicting the fact that $P^{n_k}_i$ and $P^{n_k}_j$ do not have any edges in common by construction.

On the order hand, if p_i and p_j do not share edges, then their union contains a cycle C in $P_i \cup P_j$. Recall that $P^{n_k}_{ij}$ is equal to the union of $P^{n_k}_i$ and $P^{n_k}_j$, and since $P^{n_k}_i$ converges to P_i and $P^{n_k}_j$ converges to P_j, we must have that $P^{n_k}_{ij}$ converges to $P_i \cup P_j$. This implies that infinitely many elements in the sequence $P^{n_k}_{ij}$ contain the cycle C by the lock-in property. This contradicts the fact that each $P^{n_k}_{ij}$ is a path.

This establishes that the rays P_i and P_j intersect in at most one node. On the other hand, since P_i and P_j are rays in a graph with no bi-rays, by Lemma A.1 they must have infinitely many nodes in common. We have arrived at a contradiction, and thus every edge of G is contained in at most finitely many cycles. □

REFERENCES

[1] L. Addario-Berry, N. Broutin, C. Goldschmidt, and G. Miermont, *The scaling limit of the minimum spanning tree of the complete graph*, The Annals of Probability, 45 (2017), pp. 3075–3144.

[2] R. Aharoni, E. Berger, A. Georgakopoulos, A. Perlstein, and P. Sprüssel, *The max-flow min-cut theorem for countable networks*, Journal of Combinatorial Theory Series B, 101 (2011), pp. 1–17.

[3] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows: Theory, Algorithms, and Applications*, Prentice Hall, 1993.

[4] D. Aldous and J. M. Steele, *Asymptotics for Euclidean minimal spanning trees on random points*, Probability Theory and Related Fields, 92 (1992), pp. 247–258.

[5] K. S. Alexander, *Percolation and minimal spanning forests in infinite graphs*, The Annals of Probability, (1995), pp. 87–104.

[6] C. D. Aliprantis and K. C. Border, *Infinite Dimensional Analysis: A Hitchhiker's Guide*, Springer, 3rd ed., 2006.

[7] E. J. Anderson and A. B. Philpott, *A continuous-time network simplex algorithm*, Networks, 19 (1989), pp. 395–425.

[8] H. Bruhn, R. Diestel, M. Kriesell, R. Pendavingh, and P. Wollan, *Axioms for infinite matroids*, Advances in Mathematics, 239 (2013), pp. 18–46.

[9] N. Christofides, *Worst-case analysis of a new heuristic for the travelling salesman problem*, tech. report, Carnegie-Mellon University Technical Report, 1976.

[10] R. Diestel, *Graph Theory*, Springer, 4th ed., 2010.

[11] M. A. Djauhari and S. L. Gan, *Optimality problem of network topology in stock market analysis*, Physica A: Statistical Mechanics and Its Applications, 419 (2015), pp. 108–114.

[12] A. Ghate, *Duality in countably infinite monotropic programs*, SIAM Opt., 27 (2017), pp. 2010–2033.

[13] R. L. Graham and P. Hell, *On the history of the minimum spanning tree problem*, Annals of the History of Computing, 7 (1985), pp. 43–57.

[14] D. Granot and G. Huberman, *Minimum cost spanning tree games*, Mathematical Programming, 21 (1981), pp. 1–18.

[15] V. Klee, *The greedy algorithm for finitary and cofinitary matroids*, in Combinatorics: Proceedings of Symposia in Pure Mathematics, T. S. Motzkin, ed., 1971, pp. 137–152.

[16] R. Lyons, Y. Peres, and O. Schramm, *Minimal spanning forests*, The Annals of Probability, 34 (2006), pp. 1665–1692.

[17] J. Munkres, *Topology*, Prentice Hall, 2000.

[18] S. Nourollahi and A. Ghate, *Duality in convex minimum cost flow problems on infinite networks and hypernetworks*, Networks, 70 (2017), pp. 98–115.

[19] S. Nourollahi and A. Ghate, *Inverse optimization in minimum cost flow problems on countably infinite networks*, Networks, 73 (2019), pp. 292–305.

[20] A. Paul, D. Freund, A. Ferber, D. B. Shmoys, and D. P. Williamson, *Budgeted prize-collecting traveling salesman and minimum spanning tree problems*, Mathematics of Operations Research, 45 (2019), pp. 576–590.

[21] H. E. Romeijn, D. Sharma, and R. L. Smith, *Extreme point characterizations for infinite network flow problems*, Networks, 48 (2006), pp. 209–22.

[22] C. T. Ryan, R. L. Smith, and M. A. Epelman, *A simplex method for uncapacitated pure-supply infinite network flow problems*, SIAM Journal on Optimization, 28 (2018), pp. 2022–2048.

[23] I. E. Schochetman and R. L. Smith, *Convergence of selections with applications in optimization*, Journal of Mathematical Analysis and Applications, 155 (1991), pp. 278–292.

[24] T. C. Sharkey and H. E. Romeijn, *A simplex algorithm for minimum-cost network-flow problems in infinite networks*, Networks, 52 (2008), pp. 14–31.

[25] K. J. Supowit, D. A. Plaisted, and E. M. Reingold, *Heuristics for weighted perfect matching*, in ACM STOC Symposium on Theory of Computing, 1980, pp. 398–419.

[26] S. Zhang, H. Tong, J. Xu, and R. Maciejewski, *Graph convolutional networks: A comprehensive review*, Computational Social Networks, 6 (2019), p. 11.