
MOST, AND LEAST, COMPACT SPANNING TREES OF A GRAPH

A PREPRINT

Gyan Ranjan*
Netlabs.Ai & Ericsson Inc.
California, USA
gyan.ranjan@netlabs.ai

Nishant Saurabh
Dept. of Information & Computing Sciences
Utrecht University, The Netherlands
n.saurabh@uu.nl

Amit Ashutosh*
Cisco Systems Inc.
California, USA
amashuto@cisco.com

June 14, 2022

ABSTRACT

We introduce the concept of *Most, and Least, Compact Spanning Trees* - denoted respectively by $T^*(G)$ and $T^\#(G)$ - of a **simple, connected, undirected and unweighted** graph $G(V, E, W)$. For a spanning tree $T(G) \in \mathcal{T}(G)$ to be considered $T^*(G)$, where $\mathcal{T}(G)$ represents the set of all the spanning trees of the graph G , it must have the *least sum of inter-vertex pair shortest path distances* from amongst the members of the set $\mathcal{T}(G)$. Similarly, for it to be considered $T^\#(G)$, it must have the *highest sum of inter-vertex pair shortest path distances*. In this work, we present an iteratively greedy *rank-and-regress* method that produces at least one $T^*(G)$ or $T^\#(G)$ by eliminating one extremal edge per iteration. The rank function for performing the elimination is based on the elements of the matrix of *relative forest accessibilities* of a graph and the related *forest distance* [22]. We provide empirical evidence in support of our methodology using some standard graph families; and discuss potentials for computational efficiencies, along with relevant trade-offs, to enable the extraction of $T^*(G)$ and $T^\#(G)$ within reasonable time limits on standard platforms.

Keywords Spanning Sub-Graph · Minimum Spanning Tree · Prim’s Algorithm · Kruskal’s Algorithm · Forest Matrix · Forest Distance · Compact Spanning Tree

1 Introduction

Compact sub-structures (sub-graphs) of graphs find use in many scientific disciplines: from resource placement [43] to routing [30, 31], structural analysis to data compression [45, 46], information coding and machine learning [15, 19], to name but a few. One of the most popular and widely studied compact sub-structures of a graph are the spanning trees; and, in particular, for weighted graphs - and digraphs (directed graphs) - the *Minimum Spanning Trees* (MST) [48]. For a spanning tree to be considered an MST, it must have the least sum of edge weights from amongst all the spanning trees of the graph. Several well known algorithms exist for extracting MSTs of a graph; the Prim’s and Kruskal’s algorithms being the two most widely used [48]. Both algorithms follow a *greedy* selection approach towards constructing the MST; induced by a rank order defined over the edge weights. Alas, when a graph G is unweighted - i.e. all edges have a unit weight - every spanning tree of G is an MST, with a cumulative weight of $n - 1$, n being the number of nodes/vertices in the graph. This renders the utility of MST as a compact sub-structure rather meaningless.

Another class of compact spanning sub-graphs is the so-called *shortest path tree* [1, 3, 5], which seems to have been used a lot in the communication networks area [4, 2]. A shortest path tree is a spanning sub-graph which is constructed by choosing a vertex as root, and ensuring the least possible shortest path distance between it and the rest of the vertices in the graph. The optimization of path lengths is with respect to the root vertex and the trees are not necessarily the most compact spanning acyclic sub-graphs of the graph.

*The authors are currently employed at Ericsson, USA; and Cisco Systems, USA, respectively. However, this research has been conducted entirely in personal capacity based on individual interests. Ericsson and Cisco, neither bear responsibility, nor claim any credit for it.

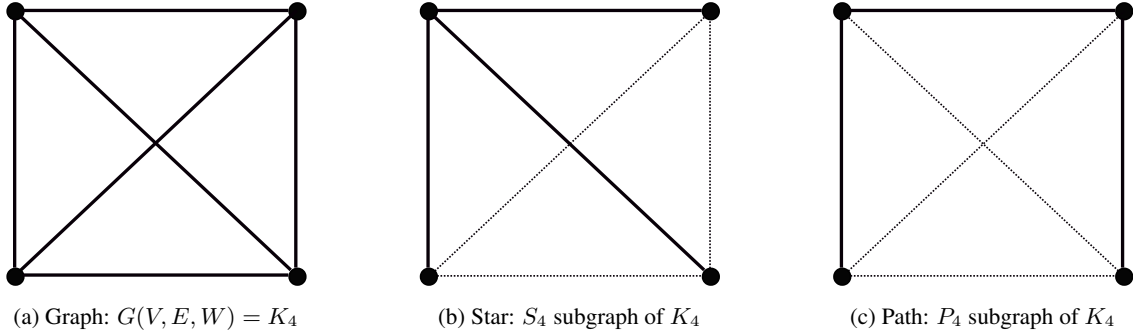


Figure 1: (a) The complete graph K_4 of order $n = 4$, (b) an S_4 : one of four MCSTs of K_4 , (c) a P_4 : one of twelve LCSTs of K_4 . Solid lines in (b) and (c) represent the edges of S_4 and P_4 respectively, while the dotted lines represent edges from K_4 that are not retained in the respective trees.

In this work, we introduce and study a new class of spanning acyclic sub-graphs: the *Most* and the *Least Compact Spanning Trees*. For brevity, denoted henceforth as MCST ($T^*(G)$) and LCST ($T^\#(G)$), respectively. For a spanning tree $T(G) \in \mathcal{T}(G)$ to be considered $T^*(G)$ - where $\mathcal{T}(G)$ represents the set of all the spanning trees of the graph G - it must have the *least sum of inter-node shortest path distances* from amongst the members of the set $\mathcal{T}(G)$. On the other hand, for it to be considered $T^\#(G)$, it must have the *highest sum of inter-node pair shortest path distances*. Clearly, neither $T^*(G)$ nor $T^\#(G)$ are necessarily unique i.e. there can be more than one of each in a given graph G . For example, consider a graph G in which at least one vertex, say $v_i \in V(G)$, has a degree $d(v_i) = |V(G)| - 1 = (n - 1)$. Clearly, the star graph S_n rooted at v_i represents one of the MCST of G . More than one such vertices/nodes may exist in G . Similarly, if the path graph (P_n) is a sub-graph of G , then it is an LCST of G . Once again, more than one paths of length $(n - 1)$ may exist in G (e.g. in a graph with a fundamental cycle C_n of order n).

The MCST and LCST algorithms use the inter-node-pair *Forest Distance* [22], limited to node pairs connected by an edge, and eliminate one edge per iteration until a spanning tree results in the end i.e. $(n - 1)$ edges remain. No *bridge* edge - an edge that when deleted renders the remaining graph disconnected - ever gets deleted during the process. This is done by using the *effective resistance distance* which - when measured across the end-points of a bridge edge, is always 1 (for an unweighted graph). The *iteratively greedy* nature of our algorithm is a result of the fact that the rank order of edges needs to be recomputed in each iteration, unlike the universal edge weight based rank order used by the Prim's and Kruskal's algorithms for extracting MSTs [48].

The rest of this work is organized into the following sections: we start by formally stating the problem, along with all the notations and paraphernalia in §2. In §3, we cover the *matrix of relative forest accessibilities* of a graph, and the associated *forest distance*, followed by our *iteratively greedy rank-and-regress* algorithm based on the elements corresponding to the set of edges in the graph. Next, in §4, we present thorough empirical evaluations using several standard graph families, and establish that the iteratively greedy rank-and-regress method indeed produces the MCST ($T^*(G)$) and LSCT ($T^\#(G)$) respectively, for the two extremal choices. This is followed up, in §5, with a discussion on the computational complexity and various ways in which the costs can be optimized. We review related work literature in §6. And finally, in §7, we conclude with a discussion of future work and potential research directions.

2 Problem Formulation

In this section, we formally pose the problem of extracting an MCST ($T^*(G)$) and an LCST ($T^\#(G)$). But first a few notations and preliminaries are in order.

2.1 Notations and Preliminaries

2.1.1 Topological Primitives

We denote by $G(V, E, W)$ a **simple, connected, undirected and unweighted graph**; where the set of vertices/nodes of G are represented by $V(G)$ and the set of edges/links by $E(G)$. The number of vertices $|V(G)| = n$ is also called the *order* of the graph G and the number of edges $|E(G)| = m$ is related to the *volume* of the graph (defined in the next paragraph)².

²In the rest of this work we use the terms vertices and nodes interchangeably; the same goes for edges and links.

We denote by $v_i \in V(G)$ the i^{th} vertex of G (the ordinality is arbitrary); and an edge connecting v_i and v_j by e_{ij} . As the graph is unweighted, all edges have a unit weight, i.e. $w_{ij} = 1$ denotes the weight of edge e_{ij} . The quantity $d(v_i)$ denotes the degree of the vertex v_i and represents the count of vertices that v_i is directly connected to through an edge. Note that $\text{Vol}(G) = \sum_{i=1}^n d(v_i) = 2m$; is the aforementioned *volume*. Finally, $\mathcal{T}(G)$ represents the set of all the spanning trees of G .

2.1.2 Algebraic Paraphernalia

Next, we introduce all the algebraic functions that help us design the metrics required in our algorithms. By $A \in \mathbb{R}^{n \times n}$, we denote the adjacency matrix of $G(V, E, W)$ [12], such that:

$$\begin{aligned} A_{ij} &= 1 \quad \text{if } e_{ij} \in E(G), \\ &= 0 \quad \text{Otherwise.} \end{aligned}$$

Clearly, A is a square symmetric matrix. The *combinatorial Laplacian* of the graph $G(V, E, W)$, is defined as:

$$L = D - A \quad (1)$$

where $D = [d_{ii}] = d(v_i)$, is a diagonal matrix with the node degrees on the diagonal; and 0 elsewhere.

L is a square, symmetric, doubly-centered (all rows and columns sum up to 0) and positive semi-definite matrix with n non-negative eigen values [36, 37, 38]. Its eigen vectors form the orthonormal basis of order n . L is not invertible, owing to a unique minimal eigen value 0, if the graph G is simple, undirected and connected. The *Moore-Penrose Pseudo-Inverse* of L , denoted by L^+ is a generalized inverse [11, 17, 32, 34, 35]. Both L , in a modified form akin to the Tikhonov Regularization [28], as well as its pseudo-inverse L^+ yield distance functions which we exploit in §3 to construct our *iteratively greedy rank-and-regress* algorithm for generating the most and least compact spanning trees of a given G .

2.2 Compactness of a Spanning Tree

Definition 1 *Spanning Tree of a graph*: A spanning tree $T(G)$ of the graph $G(V, E, W)$ is a connected acyclic spanning sub-graph of G with exactly $n - 1$ edges in it.

The term *spanning* applied to a sub-graph simply means that it has all the vertices of the graph G in it. In Figure 1, both S_4 and P_4 represent spanning trees of K_4 . As we shall see later, $T(G)$ is also a special kind of spanning forest of G (one with **exactly one** tree in it). A *Minimum Spanning Tree* (MST) of a graph is a spanning tree with the least sum of edge weights across all possible spanning trees. But when G is unweighted, all edges have the same unit weight. Clearly, the volume of any $T(G)$ is given by:

$$\text{Vol}(T(G)) = 2(n - 1), \quad \forall T(G) \in \mathcal{T}(G) \quad (2)$$

which is an invariant across the set of spanning trees of a given graph. Therefore, algorithms like Prim's or Kruskal's cannot differentiate between them. In contrast, we define the following as a *compactness* metric that can:

Definition 2 *The Compactness Metric*:

$$\mathcal{C}(T(G)) = \sum_{i=1}^n \sum_{j=1}^n \mathcal{D}(v_i, v_j) \quad (3)$$

where $\mathcal{D}(v_i, v_j)$ is the length of the path connecting v_i to v_j (or, *vice versa*) in the tree $T(G)$. Given that $G(V, E, W)$ is unweighted, $\mathcal{D}(v_i, v_j)$ is simply the *hop count* from v_i to v_j . With this in view, we define the MCST ($T^*(G)$) and LCST ($T^\#(G)$) respectively as:

Definition 3 *Most Compact Spanning Tree of a graph (MCST)*:

$$T^*(G) = \underset{T(G) \in \mathcal{T}(G)}{\text{argmin}} \mathcal{C}(T(G)) \quad (4)$$

Definition 4 *Least Compact Spanning Tree of a graph (LCST)*:

$$T^\#(G) = \underset{T(G) \in \mathcal{T}(G)}{\text{argmax}} \mathcal{C}(T(G)) \quad (5)$$

Note again in Figure 1, the star graph S_4 and the path graph P_4 respectively constitute an MCST and an LCST in K_4 . In what follows, we describe an iteratively greedy *rank-and-regress* algorithm that eliminates $[m - (n - 1)]$ edges of a graph to produce at least one $T^*(G)$ (or, $T^\#(G)$); depending upon the nature of the greedy choice).

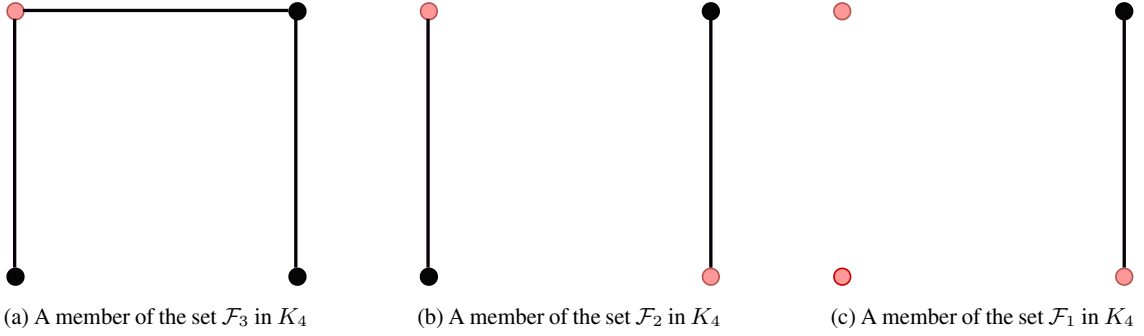


Figure 2: Some spanning rooted forests of K_4 : (a) a spanning rooted forest with 3 edges (or, one tree), which is the same as a spanning tree, with one root colored in red (top-left), (b) a spanning rooted forest with 2 edges (or, two trees), with one vertex in each tree colored in red (top vertex in the left tree, and bottom vertex in the right), (c) a spanning rooted forest with 1 edge (or, three trees), in which two isolated vertices represent two trees (top left, bottom left), each colored in red as they are roots by default; and one tree on the right, represented by the sole edge, with the bottom right vertex colored in red as root. A member of \mathcal{F}_0 in K_4 is simply a collection of four isolated vertices (i.e. no edges), and each vertex represents a trivial tree with itself as root. So, each tree of a spanning rooted forest has one node designated as root in it.

3 Methodology

3.1 Spanning Rooted Forests and a Pair of Distances: $[\Delta_{ij}, \Omega_{ij}]$

In this section, we introduce a pair of distance functions called the *forest distance* and the *effective resistance distance*, based on which we define our *iteratively greedy rank-and-regress* algorithm. But first, a word on a special class of spanning sub-graphs called the *spanning rooted forests* of G is due.

Definition 5 *Spanning Rooted Forest with k edges*: A spanning acyclic sub-graph of G with k edges, or, equivalently, $n - k$ trees in it, in which each tree has a designated root vertex.

Computationally, the number of *spanning rooted forests* of G are enumerated using the elements of a single matrix $Q \in \mathbb{R}^{n \times n}$:

$$Q = (I + L)^{-1} \quad (6)$$

where $I \in \mathbb{R}^{n \times n}$ is the *identity* matrix whose columns form the standard orthonormal basis in an n -dimensional space, and L is the *combinatorial Laplacian* of G that was introduced in §2.1.2³. The matrix Q is referred to as the matrix of *relative forest accessibilities* (or, the *forest matrix* for brevity) and has some very interesting properties and applications [20, 21, 22]. For a simple, connected, undirected and unweighted graph $G(V, E, W)$, Q is square, symmetric, positive-definite and doubly stochastic; i.e. every row or column of Q sums up to 1.

Next, we connect the elements of Q to the *spanning rooted forests* in G . Let \mathcal{F}_k^{ij} denote the set of spanning rooted forests in G with k edges - or, $n - k$ trees - in which v_i and v_j belong to the same tree and v_i is the *root* of that tree. \mathcal{F}_k , similarly, represents the set of all spanning rooted forests in G with exactly k edges (or, $n - k$ trees, each with one root in it). Then,

$$Q_{ij} = \sum_{k=0}^{n-1} \varepsilon(\mathcal{F}_k^{ij}) / \sum_{k=0}^{n-1} \varepsilon(\mathcal{F}_k) = Q_{ji} \quad (7)$$

The operator $\varepsilon(\cdot)$ above simply represents the cardinality (count of the elements) in its operand set, because all edges have a unit weight. For weighted graphs, it would represent a sum of products of edge weights in a forest. For details, the reader is referred to [21, 22]. Of particular interest to us is the *Forest Distance* function, defined over all pairs of vertices $(i, j) \in V(G) \times V(G)$:

$$\Delta_{ij} = Q_{ii} - Q_{ij} - Q_{ji} + Q_{jj} = \Delta_{ji} \quad (8)$$

Finally, we come to the matrix L^+ , the *Moore-Penrose Pseudo-Inverse* of L , that was also introduced in §2.1.2. Analogous to Q , is related to the *dense spanning rooted forests* of G [21, 22]. A dense spanning rooted forest contains

³Note the similarity to the Tikhonov regularization [28] here. It is easy to see that the form $(I + L)$ simply shifts all the eigen values of L by 1 and makes it invertible.

exactly $(n - 1)$ or $(n - 2)$ edges in it; or, equivalently, has either one or two trees, respectively. To be specific,

$$L_{ij}^+ = \frac{\varepsilon(\mathcal{F}_{n-2}^{ij}) - \frac{1}{n}\varepsilon(\mathcal{F}_{n-2})}{\varepsilon(\mathcal{F}_{n-1})} = L_{ji}^+ \quad (9)$$

Once again, the elements of L^+ yield the *effective resistance distance* between any pair of vertices $(i, j) \in V(G) \times V(G)$ [33, 41, 42, 50]:

$$\Omega_{ij} = L_{ii}^+ - L_{ij}^+ - L_{ji}^+ + L_{jj}^+ = \Omega_{ji} \quad (10)$$

In particular, we use Ω_{ij} to detect a *bridge edge* - i.e. an edge which upon deletion renders a graph disconnected. If $\Omega_{ij} = 1$ and $e_{ij} \in E(G)$, then it is easy to show that e_{ij} must be a bridge edge [42]. In the next sub-section, we use the values of Δ_{ij} and Ω_{ij} to construct our *iteratively greedy rank-and-regress* algorithm for extracting an MCST/LCST.

3.2 An Iteratively Greedy Rank-And-Regress Algorithm

#	Pseudo-Code	Comment
1	init: $G(V, E, W)$; $n \leftarrow V(G) $; $E(G) \leftarrow G(V, E, W)$;	G is simple, connected, undirected, and unweighted. G is a graph of order n . Initialize $E(G)$, the set of edges of G .
2	while $ E(G) > (n - 1)$:	Ensure that G is not a tree already. If not, start iteration.
3	$Q = (I + L)^{-1}$; $L^+ = \text{pinv}(L)$;	Compute the Forest matrix and L^+ for current G .
4	init: $\Delta \leftarrow [0]$; $\Omega \leftarrow [0]$; $\delta^* \leftarrow 0$; $e^* \leftarrow \phi$; for each e_{ij} in $E(G)$: $\Delta_{ij} = Q_{ii} - Q_{ij} - Q_{ji} + Q_{jj}$ $\Omega_{ij} = L_{ii}^+ - L_{ij}^+ - L_{ji}^+ + L_{jj}^+$ if $\Omega_{ij} == 1$: continue ; else if $\Delta_{ij} > \delta^*$: $\delta^* \leftarrow \Delta_{ij}$; $e^* \leftarrow e_{ij}$; end for each	Initialize the Forest and Resistance distance matrices. Iterate over the edges in G . Compute the Forest distance. Compute the Resistance distance. Check if e_{ij} is a <i>bridge edge</i> . Check if e_{ij} improves on the extremality criterion? e_{ij} is the new candidate for being e^* . Loop back for the next edge.
5	$E(G) \leftarrow E(G) - e^*$ end while	Delete e^* from the graph G . Loop back to step 2 for the next iteration.
6	return $[T^*(G) \leftarrow E(G)]$;	The remaining $(n - 1)$ edges in $E(G)$ are the MCST.

Table 1: **An iteratively greedy rank-and-regress algorithm for constructing an MCST:** Returns one of the *Most Compact Spanning Trees* of the graph G . The variant for finding at least one **LCST** simply requires a reversal in the *extremality* criteria. Instead of δ^* selecting an edge with maximum Δ_{ij} for deletion in each iteration (c.f. step 4 above), it will select an edge with the minimum Δ_{ij} .

Table 1 presents the pseudo-code for our *iteratively greedy rank-and-regress* algorithm. Although the code targets the extraction of an MCST ($T^*(G)$), with one slight modification, it would extract an LCST ($T^\#(G)$) instead. Note:

- a. **Regress, Not Selection:** Unlike the *Minimum Spanning Tree* algorithms, or, the random/shortest-path spanning tree generation, where the tree is constructed by greedily selecting edges, our method eliminates one *extremal* edge at a time. Hence, the qualifier *regress*.
- b. **Number of Iterations:** The algorithm begins by initializing the set $E(G)$ of edges and deleting **exactly one** edge from the graph in each iteration until the number of edges in $E(G)$ reduces to $(n - 1)$. Hence, if the number of edges in G originally (at the beginning) is m , the algorithm concludes in exactly $m - (n - 1)$ iterations.
- c. **Maintaining Connectedness:** The graph G , through this regress, always stays connected. This is ensured in step 4 by skipping any extremal edge e_{ij} which is also a *bridge* edge at that stage; i.e. $\Omega_{ij} = 1$.
- d. **The Tree Property:** By ensuring connectedness during all iterations, as stated in [c.] above, and noting the fact that there are exactly $(n - 1)$ edges remaining in the end, our algorithm ensures that the resulting $E(G)$ in step 6 is indeed a spanning tree.
- e. **The Iteratively Greedy Nature:** The *rank* order of the edges remaining in the graph in each iteration may differ across iterations. This can be observed in steps 3 and 4 where $\{Q, L^+, \Delta, \Omega\}$ are recomputed in each iteration. Hence, the qualifier *iteratively greedy*.

3.3 Analysis: Why would it work?

In order to make sense of why the greedy choice on $\Delta_{ij} : e_{ij} \in E(G)$ works for attaining the objectives for MCST and LCST - as defined in (4) and (5) respectively - we first need to understand the constituents of Δ_{ij} . Recall, from (8):

$$\Delta_{ij} = Q_{ii} - Q_{ij} - Q_{ji} + Q_{jj} = \Delta_{ji} \quad (11)$$

The term $[Q_{ii} - Q_{ij}]$, based on the definition in (7), yields:

$$\begin{aligned} Q_{ii} - Q_{ij} &= \frac{\sum_{k=0}^{n-1} [\varepsilon(\mathcal{F}_k^{ii}) - \varepsilon(\mathcal{F}_k^{ij})]}{\sum_{k=0}^{n-1} \varepsilon(\mathcal{F}_k)} \\ &= \frac{[\varepsilon(\mathcal{F}_0^{ii}) - \varepsilon(\mathcal{F}_0^{ij})] + [\varepsilon(\mathcal{F}_1^{ii}) - \varepsilon(\mathcal{F}_1^{ij})] + \dots + [\varepsilon(\mathcal{F}_{n-2}^{ii}) - \varepsilon(\mathcal{F}_{n-2}^{ij})] + [\varepsilon(\mathcal{F}_{n-1}^{ii}) - \varepsilon(\mathcal{F}_{n-1}^{ij})]}{\sum_{k=0}^{n-1} \varepsilon(\mathcal{F}_k)} \end{aligned}$$

The term $[Q_{jj} - Q_{ji}]$ can be similarly decomposed as:

$$\begin{aligned} Q_{jj} - Q_{ji} &= \frac{\sum_{k=0}^{n-1} [\varepsilon(\mathcal{F}_k^{jj}) - \varepsilon(\mathcal{F}_k^{ji})]}{\sum_{k=0}^{n-1} \varepsilon(\mathcal{F}_k)} \\ &= \frac{[\varepsilon(\mathcal{F}_0^{jj}) - \varepsilon(\mathcal{F}_0^{ji})] + [\varepsilon(\mathcal{F}_1^{jj}) - \varepsilon(\mathcal{F}_1^{ji})] + \dots + [\varepsilon(\mathcal{F}_{n-2}^{jj}) - \varepsilon(\mathcal{F}_{n-2}^{ji})] + [\varepsilon(\mathcal{F}_{n-1}^{jj}) - \varepsilon(\mathcal{F}_{n-1}^{ji})]}{\sum_{k=0}^{n-1} \varepsilon(\mathcal{F}_k)} \end{aligned}$$

The denominator $\sum_{k=0}^{n-1} \varepsilon(\mathcal{F}_k)$ is an invariant for a given graph; and plays no role in determining the relativities across the members of the set $E(G)$. The numerator, on the other hand, certainly does. We go over each term in the numerator, backwards, starting from $[\varepsilon(\mathcal{F}_{n-1}^{ii}) - \varepsilon(\mathcal{F}_{n-1}^{ij})]$ and going all the way to $[\varepsilon(\mathcal{F}_0^{ii}) - \varepsilon(\mathcal{F}_0^{ij})]$ which is a constant ($= 1$).

3.3.1 Contribution of spanning rooted forests with $(n - 1)$ edges to Δ_{ij}

Consider an individual spanning rooted forest $F_{n-1}^{ii} \in \mathcal{F}_{n-1}^{ii}$: a spanning forest with $v_i \in V(G)$ as the root of the tree in which it belongs, and $(n - 1)$ edges. Clearly, F_{n-1}^{ii} is a spanning tree of G ; since, F_{n-1}^{ii} is spanning, acyclic and has $(n-1)$ edges - with the only difference being that v_i is distinguished as the root of this tree (say, colored *red*). Therefore:

$$\varepsilon(\mathcal{F}_{n-1}^{ii}) = |\mathcal{T}(G)| \quad (12)$$

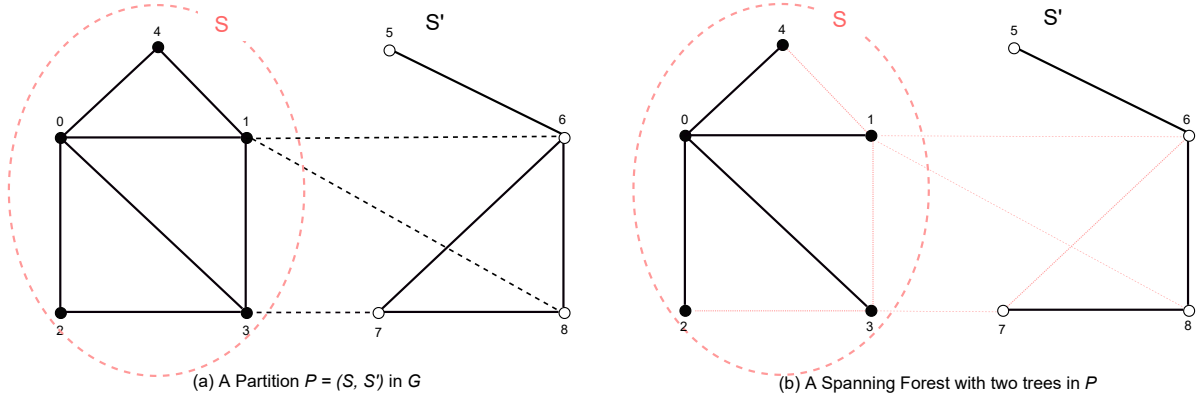


Figure 3: **Bi-Partition of a graph** $G(V, E, W)$: (a) $G(V, E, W)$ has nine vertices (v_0 to v_8) and fourteen edges $E(G) = \{e_{0,1}, e_{0,2}, e_{0,3}, e_{0,4}, e_{1,3}, e_{1,4}, e_{1,6}, e_{1,8}, e_{2,3}, e_{3,7}, e_{5,6}, e_{6,7}, e_{6,8}, e_{7,8}\}$. S and S' are the two sub-graphs of G that constitute the bi-partition $P = (S, S')$, shown in (a), constructed by deleting edges $\{e_{1,6}, e_{1,8}, e_{3,7}\}$. $V(S) = \{v_0, v_1, v_2, v_3, v_4\}$ and $V(S') = \{v_5, v_6, v_7, v_8\}$; $E(S) = \{e_{0,1}, e_{0,2}, e_{0,3}, e_{0,4}, e_{1,3}, e_{1,4}, e_{2,3}\}$, $E(S') = \{e_{5,6}, e_{6,7}, e_{6,8}, e_{7,8}\}$ and $E(S, S') = \{e_{1,6}, e_{1,8}, e_{3,7}\}$. (b) $[T(S), T(S')]$ represent one spanning forest of G , subtended by the partition $P = (S, S')$. The edges in $T(S)$ are $\{e_{0,1}, e_{0,2}, e_{0,3}, e_{0,4}\}$ while those in $T(S')$ are $\{e_{5,6}, e_{6,7}, e_{6,8}, e_{7,8}\}$. The number of spanning rooted forests represented in this configuration $[T(S), T(S')]$ in (b), for the members of the sub-graph S is 4, while those for the members of the sub-graph S' is 5 (c.f. [41]).

where $\mathcal{T}(G)$ is the set of all the spanning trees in G and $|\mathcal{T}(G)|$ is their count. Note also, that in a spanning tree, all the vertices of G are present by definition. So,

$$\varepsilon(\mathcal{F}_{n-1}^{ij}) = |\mathcal{T}(G)| \quad (13)$$

Therefore, $[\varepsilon(\mathcal{F}_{n-1}^{ii}) - \varepsilon(\mathcal{F}_{n-1}^{jj})] = 0$. By symmetry, $[\varepsilon(\mathcal{F}_{n-1}^{jj}) - \varepsilon(\mathcal{F}_{n-1}^{ii})] = 0$. So, the set \mathcal{F}_{n-1} does not contribute to the numerator of Δ_{ij} at all.

3.3.2 Contribution of spanning rooted forests with $(n - 2)$ edges to Δ_{ij}

In order to analyze the counts $[\varepsilon(\mathcal{F}_{n-2}^{ii}), \varepsilon(\mathcal{F}_{n-2}^{jj}), \varepsilon(\mathcal{F}_{n-2}^{ij}) \& \varepsilon(\mathcal{F}_{n-2}^{ji})]$, we need to revisit the *bi-partitions* of a graph introduced in [41].

Definition 6 *Bi-partition* ($P = (S, S')$): A cut of the graph G which contains exactly two mutually exclusive and exhaustive connected sub-graphs S and S' (c.f. Figure 3).

Let, $V(S)$ and $V(S')$ be the mutually exclusive and exhaustive subsets of $V(G)$, $E(S)$ and $E(S')$, the sets of edges in the respective components S and S' of P and $E(S, S')$, the set of edges that *violate* P i.e. have one end in S and the other in S' . Also, let $\mathcal{T}(S)$ and $\mathcal{T}(S')$ be the set of spanning trees in the respective component sets S and S' . We denote by $\mathcal{P}(G)$, the set of all bi-partitions of $G(V, E)$. Clearly, a given $P = (S, S')$ represents a sub-graph from which $E(S, S')$ have edges have been deleted. A vertex $v_i \in V(S)$ stays connected to $|V(S)| - 1$ other vertices and gets disconnected from $|V(S')|$ vertices; while a vertex $v_j \in V(S')$ stays connected to $|V(S')| - 1$ other vertices and gets disconnected from $|V(S)|$ vertices. For convenience, we will always call the sub-graph in which v_i is as S . Then:

$$\varepsilon(\mathcal{F}_{n-2}^{ii})_{|P(S, S')} = |\mathcal{T}(S)||\mathcal{T}(S')||V(S')| \quad v_i \in V(S) \quad (14)$$

and,

$$\begin{aligned} \varepsilon(\mathcal{F}_{n-2}^{ij})_{|P(S, S')} &= |\mathcal{T}(S)||\mathcal{T}(S')||V(S')| && \text{if } v_j \in V(S), \\ &= 0 && \text{otherwise.} \end{aligned}$$

So:

$$\varepsilon(\mathcal{F}_{n-2}^{ii}) - \varepsilon(\mathcal{F}_{n-2}^{jj}) = \sum_{P \in \mathcal{P}(G)} \sum_{v_i \in V(S), v_j \in V(S')} |\mathcal{T}(S)||\mathcal{T}(S')||V(S')| \quad (15)$$

Note that the condition $v_i \in V(S) \& v_j \in V(S')$ maps to those partitions in $\mathcal{P}(G)$ in which the edge $e_{ij} \in E(S, S')$.

By symmetry:

$$\varepsilon(\mathcal{F}_{n-2}^{jj}) - \varepsilon(\mathcal{F}_{n-2}^{ji}) = \sum_{P \in \mathcal{P}(G)} \sum_{v_j \in V(S'), v_i \in V(S)} |\mathcal{T}(S)| |\mathcal{T}(S')| |V(S)| \quad (16)$$

which yields:

$$\begin{aligned} \varepsilon(\mathcal{F}_{n-2}^{ii}) - \varepsilon(\mathcal{F}_{n-2}^{ij}) + \varepsilon(\mathcal{F}_{n-2}^{jj}) - \varepsilon(\mathcal{F}_{n-2}^{ji}) &= \sum_{P \in \mathcal{P}(G)} \sum_{v_i \in V(S), v_j \in V(S')} |\mathcal{T}(S)| |\mathcal{T}(S')| (|V(S')| + |V(S)|) \\ &= n \sum_{P \in \mathcal{P}(G)} \sum_{v_i \in V(S), v_j \in V(S')} |\mathcal{T}(S)| |\mathcal{T}(S')| \end{aligned}$$

The final term in the *RHS* above is of great interest. Henceforth, we denote it by τ_{ij} for convenience. Given that e_{ij} is a cut edge in $E(S, S')$, it is easy to see that:

- Handling Bridge Edges:** If e_{ij} is a *bridge* edge in a graph G , then there is only one bi-partition P in which it is present in the set $E(S, S')$. Algorithm MCST/LCST never deletes it, anyway.
- Contribution to Connectivity between vertices in S & S' :** No edge in $E(S, S')$ contributes to inter-vertex connectivity or to the spanning trees defined over the vertices of the sub-graph S , or the vertices of the sub-graph S' . $|\mathcal{T}(S)|$ is a measure of connectivity in the sub-graph S ; and $|\mathcal{T}(S')|$ a measure of connectivity in the sub-graph S' . Higher the value of $|\mathcal{T}(S)|$ and $|\mathcal{T}(S')|$, shorter the inter-vertex distances in S and S' respectively (owing to greater path diversity); and lower the impact of the deletion of e_{ij} in inter-vertex path diversity. By extension, higher the value of τ_{ij} , lower the impact of a potential deletion of e_{ij} from the graph during an iteration of the algorithm MCST.
- Contribution to Connectivity in Spanning Trees:** If we choose a $T(S) \in \mathcal{T}(S)$, an edge in $E(S, S')$ and a $T(S') \in \mathcal{T}(S')$, together they form a spanning tree $T(G) \in \mathcal{T}(G)$. In fact, all the spanning trees of G can be obtained by combining elements from $\mathcal{T}(S), \mathcal{T}(S')$ & $E(S, S')$ for any given partition $P \in \mathcal{P}(G)$. Note, as we only regress over e_{ij} that are not bridge edges, each $e_{ij} \in E(S, S')$ is *covered* by at least one other edge in $E(S, S')$ which can be used as an alternative to construct spanning trees across the bi-partition P . Higher the value of τ_{ij} , greater the fraction of spanning trees of G in which e_{ij} is covered by at least one other edge; and its deletion has a lower impact.

Analogously, for LCST, edges with lower τ_{ij} are suitable candidates. The rest of the argument follows *as is*.

3.3.3 Contribution of spanning rooted forests with $(n - 3)$ edges to Δ_{ij}

Extending the argument to *tri-partitions* (c.f. Fig. 4): three mutually exclusive and exhaustive sub-graphs of G , denoted by S, S' and S'' , we obtain the following:

$$\varepsilon(\mathcal{F}_{n-3}^{ii}) - \varepsilon(\mathcal{F}_{n-3}^{ij}) + \varepsilon(\mathcal{F}_{n-3}^{jj}) - \varepsilon(\mathcal{F}_{n-3}^{ji}) = n \sum_{P \in \mathcal{P}(G)} \sum_{v_i \in V(S), v_j \in V(S')} |\mathcal{T}(S)| |\mathcal{T}(S')| |\mathcal{T}(S'')| \quad (17)$$

By abuse of notation, $P = (S, S', S'')$ is a tri-partition in this case. All the definitions from the previous section follow as is: with $V(S), V(S'), V(S'')$ representing the sets of vertices in each sub-graph, $E(S), E(S'), E(S'')$ represent the edges in the respective sub-graphs, and $E(S, S'), E(S, S''), E(S', S'')$ represent the edges between the sub-graphs. In fact, the argument is easily generalized for any value of $(n - \kappa)$ where $\kappa = \{2, 3, 4, \dots, n\}$.

3.3.4 Contribution of spanning rooted forests with 1 edge to Δ_{ij}

Clearly, $\varepsilon(\mathcal{F}_1^{ii}) = d(v_i) + 2 \cdot (|E(G)| - d(v_i)) = 2 \cdot m - d(v_i)$, where $d(v_i)$ is the degree of the vertex v_i , and $m = |E(G)|$ is the total number of edges in G . Similarly, $\varepsilon(\mathcal{F}_1^{ij}) = 1$ if $e_{ij} \in E(G)$, 0 otherwise. As we are only concerned with vertex pairs $(i, j) \in V(G) \times V(G) : e_{ij} \in E(G)$, we obtain:

$$\begin{aligned} [\varepsilon(\mathcal{F}_1^{ii}) - \varepsilon(\mathcal{F}_1^{ij})] + [\varepsilon(\mathcal{F}_1^{jj}) - \varepsilon(\mathcal{F}_1^{ji})] &= [2 \cdot m - d(v_i) - 1] + [2 \cdot m - d(v_j) - 1] \\ &= 4 \cdot m - [d(v_i) + d(v_j)] - 2 \end{aligned}$$

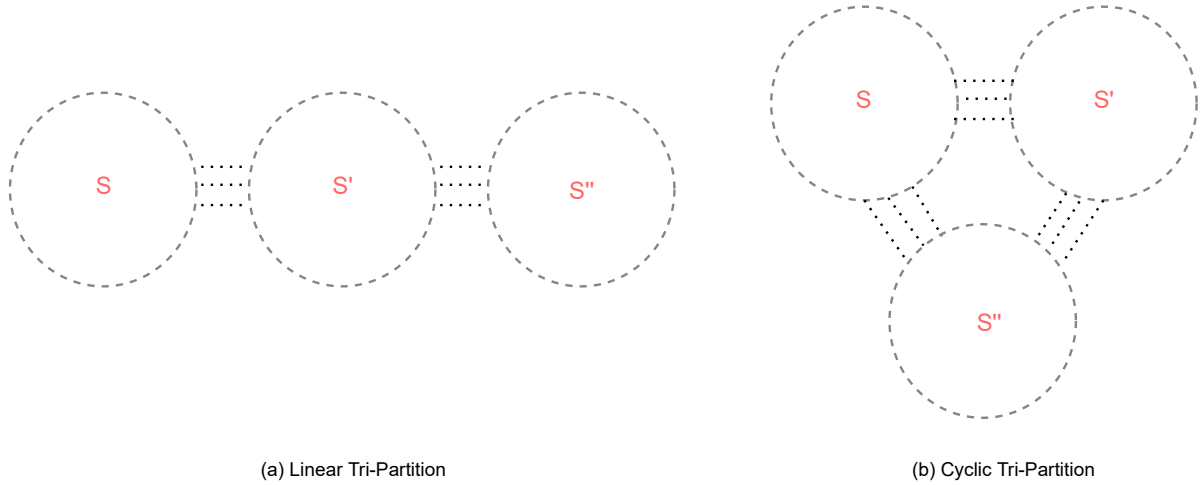


Figure 4: **Tri-Partitions of a graph** $G(V, E, W)$: A schematic showing the linear and cyclic variants. The labels $\{S, S', S''\}$ are arbitrarily assigned here. From the point of view of an edge, it can only belong to one of the sets: $E(S), E(S'), E(S''), E(S, S'), E(S', S'')$ in the linear variant; and $E(S), E(S'), E(S''), E(S, S'), E(S', S''), E(S'', S)$ in the cyclic variant. It is easy to see that the linear configuration in (a) is simply a special case of the more general cyclic configuration in (b); induced by the condition $E(S, S'') = \phi$. Higher order partitions can similarly have multiple variants depending upon the structure of the graph.

3.3.5 Contribution of spanning rooted forests with 0 edges to Δ_{ij}

Finally, we evaluate the term $[\varepsilon(\mathcal{F}_0^{ii}) - \varepsilon(\mathcal{F}_0^{jj})]$ and $[\varepsilon(\mathcal{F}_0^{jj}) - \varepsilon(\mathcal{F}_0^{ii})]$. There can only be one spanning rooted forest of G with 0 edges: a forest in which all the n vertices of G are isolated, single-vertex trivial trees and serve as roots. Clearly, in a forest of this kind, the question of (i, j) being in the same tree, doesn't arise. Hence, $\forall i \in V(G) : \varepsilon(\mathcal{F}_0^{ii}) = 1$ and $\forall (i, j) \in V(G) \times V(G) : \varepsilon(\mathcal{F}_0^{ij}) = 0$. Hence, the term contributes exactly $+2$ to Δ_{ij} for any pair (i, j) : $+1$ for v_i and $+1$ for v_j ; and is an invariant.

To summarize, therefore, the edge e_{ij} with the maximum Δ_{ij} in a graph contributes the least inter-vertex connectivity and represents an ideal candidate for elimination during a regress step during the MCST algorithm in Table 1. This is owing to the fact that the value itself is based on the number of spanning trees that remain intact in the graph despite the removal of the e_{ij} . It is easy to see that the same argument follows in the case of LCST construction.

4 Empirical Evaluation

4.1 The Complete Graph K_n of Order n

The complete graph K_n , by definition contains every graph of order n as a sub-graph in it. Hence, all possible trees of order n are sub-graphs of K_n as well. If the algorithm in Table 1 indeed generates an MCST ($T^*(G)$) - or, an LCST ($T^\#(G)$), based on the choice of the extremality criteria - we would expect it to return an S_n - or, a P_n - in step 6 for any $G(V, E, W) = K_n$ that is input in step 1. Figure 5 demonstrates that this is indeed the case for K_4 . When the MCST and LCST variants are run over K_4 , we attain the expected output. Following are worth noting:

- Number of Iterations:** Both sequences $\{(a) \rightarrow (d)\}$ and $\{(e) \rightarrow (h)\}$ conclude in three regress steps each; leaving behind exactly three *bridge* edges connecting the four vertices ⁴.
- Maintaining Connectedness:** The intermediary graphs always stay connected. The bridge edge $e_{0,3}$ in (c) is retained in the regress (c) \rightarrow (d) despite of having the highest $\Delta_{0,3} = 0.6$, amongst all the edges in (c). The edge $e_{1,2}$ is deleted instead, which has the highest $\Delta_{1,2} = 0.5$, amongst the *non-bridge* edges.
- Breaking Ties:** When more than one edge has the maximum (MCST) - or, minimum (LCST) - value of Δ_{ij} in an iteration, any one - and exactly one - of them is deleted.

⁴Equivalently, the algorithm can be concluded when $\forall (i, j) \in V(G) \times V(G) \ \& \ e_{ij} \in E(G) : \Omega_{ij} = 1$.

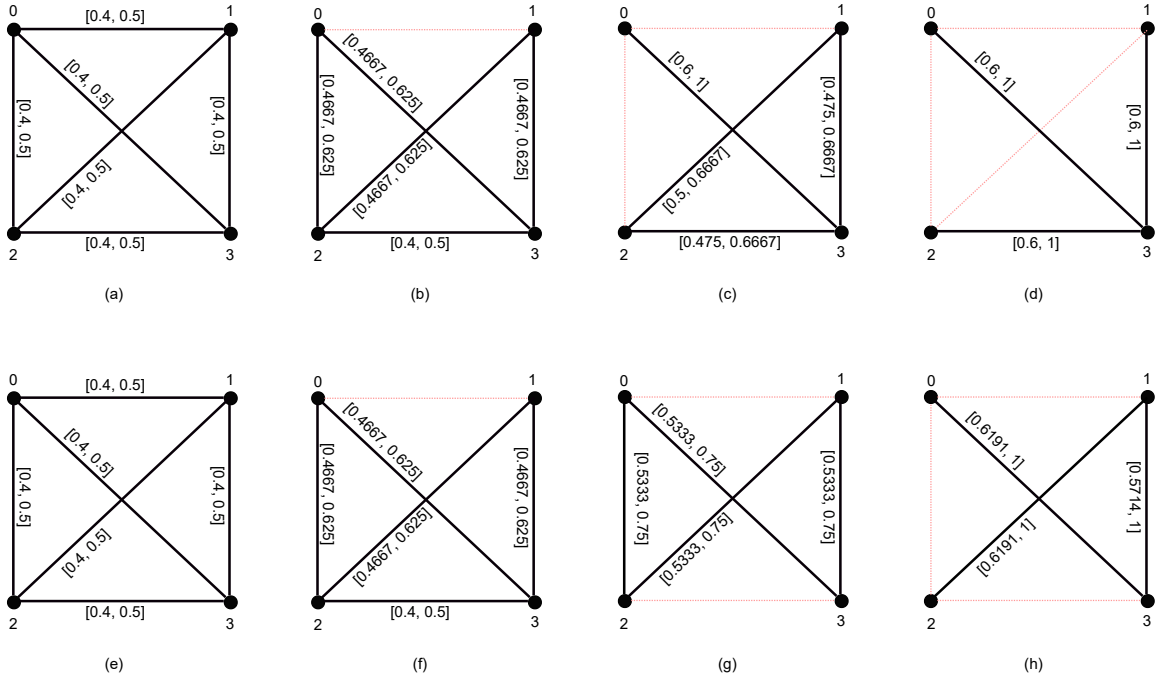


Figure 5: **Algorithm MCST and LCST over K_4** : Sequence (a) \rightarrow (d) represents the iteratively greedy MCST extraction; while, sequence (e) \rightarrow (h) represents the iteratively greedy LCST extraction. Note that the sub-graph in (d) and (h) are indeed S_4 (one of the four possible) and P_4 (one of the twelve possible). The tuple on the edges represents the pair $[\Delta_{ij}, \Omega_{ij}]$ for the edge e_{ij} in each iteration. Red dotted edges represent all the deleted edges until a given iteration.

But K_4 is a special graph with only S_4 (four different trees) and P_4 (twelve different trees) as its spanning trees. In order to ensure that this is indeed the case, we repeat the experiment for $K_n : n = 5, 6, 7, 8, 9, 10$; and each time obtain S_n and P_n as the end result. Note, we are aware that these illustrations over K_n are necessary to support our claims, but not sufficient. They are simply meant to be basic sanity checks.

4.2 The Erdős-Renyi Random Graphs

Next, we generate over 500 Erdős-Renyi random graphs (ER-graphs) [14] of varying orders: $n \in [10, 99]$ and edge densities $\rho \in [0.25, 0.50]$. ER-graphs are considered as benchmarks in many studies in literature and hence make for a natural choice in ours too. For each graph in this set, we generate:

- One MCST ($T^*(G)$) and LCST ($T^\#(G)$) pair.
- A set of random spanning trees [9, 16, 49]. We use the version in [49] to generate n random spanning trees for a graph of order n (by starting the algorithm once from each vertex).

Figure 6 shows comparative results on the average inter-vertex shortest path distances in the original graph (G), the MCST ($T^*(G)$), the LCST ($T^\#(G)$) and the mean across all random spanning trees generated (c.f. [b.] above). The graphs are arranged first by their respective orders in ascending order and then for each order, by ascending orders of the edge density (ρ). The results validate our claim that the MCST and LCST produced by our algorithm indeed are *extremal* in compactness. As one would expect, minor variations in absolute values do occur when n is kept fixed and ρ is varied; but these do not seem to change the overall pattern; the relativity of compactness between the various tree classes (c.f. APPENDIX for a discussion on a couple of minor exceptions). For completeness, we also study the diameters of the trees thus produced (c.f. Figure 6(b)), and notice the same general trends.

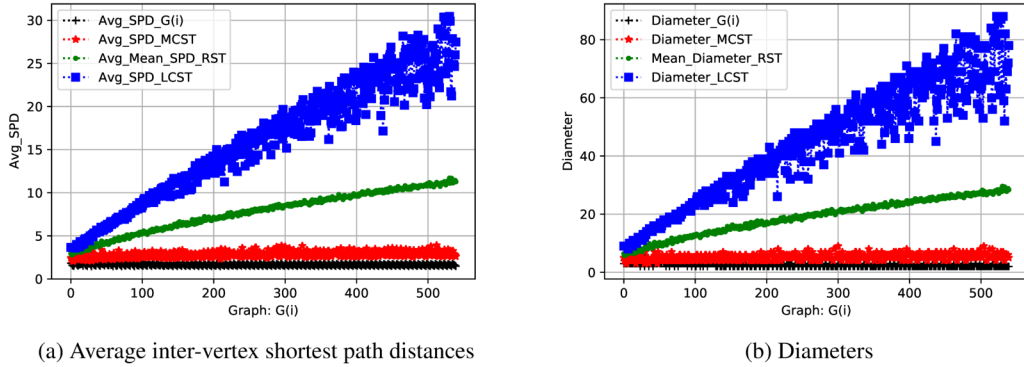


Figure 6: **Algorithm MCST and LCST over a set of ER-graphs:** (a) average shortest path distances in G , MCST, LCST and mean of average shortest path distance across all the random spanning trees, (b) diameter of G , MCST, LCST and mean diameter across all the random spanning trees. The number of random spanning trees for each graph G is $n = |V(G)|$; we generate one random spanning tree per vertex in $V(G)$ for this analysis.

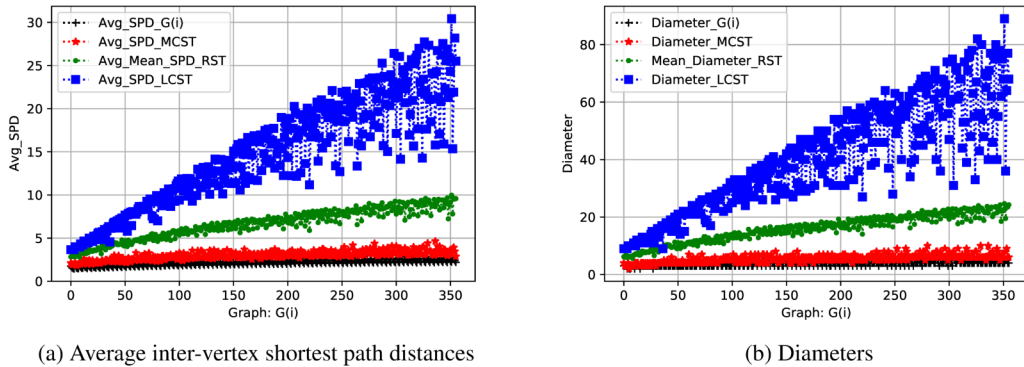


Figure 7: **Algorithm MCST and LCST over a set of BA-graphs:** (a) average shortest path distances in G , MCST, LCST and mean of average shortest path distance across all the random spanning trees, (b) diameter of G , MCST, LCST and mean diameter across all the random spanning trees. The number of random spanning trees for each graph G is $n = |V(G)|$; we generate one random spanning tree per vertex in $V(G)$ for this analysis.

4.3 The Barabási-Albert Graphs

Finally, we look at the performance of the MCST and LCST algorithms over a set of about 360 Barabási-Albert graphs (BA-graph) [8, 10, 24] of varying orders and edge densities: $n \in [10, 99]$ and the average degree per node varying between $[2, 5]$. A BA-graph is a result of the so called *preferential attachment* generative process in which each in-coming node attaches itself with pre-existing nodes in the erstwhile graph with a constant number of edges, and the *preference* of attachment is proportional to the node degrees of the other vertices. This class of graphs also finds widespread use in literature; and hence makes for an important baseline class to validate our work on. Once again, the compactness relationships hold between the graph, the MCST, the LCST and the random spanning trees; both for the average inter-vertex shortest path distances as well as the diameters. We note that as average vertex degree increases (for the same order), the gap between the three classes - the MCST, the LCST and the random spanning trees - does seem to reduce. One would expect this given the so called *small-world* nature of the BA-graphs (c.f. Figure 7). This is largely due to short diameters which in turn is owed to the *power-law scale-free* degree distributions and the existence of *rich club connectivity* (RCC) by the virtue of which a few *influential* vertices (vertices of high degrees), are inter-connected with each other with high probability, and also provide connectivity between any arbitrary vertex-pair. We refer interested readers to the original sources [8, 10, 24] and the references therein, for details.

5 Complexity Analysis

Complexity analysis of the algorithm in Table 1 is relatively straightforward. The *while* loop in step 2 runs $m - (n - 1)$ times; m being the total number of edges and n , the order of the original G (number of nodes). So, the computational complexity of the outer loop is $O(m)$. Each *while* iteration, in turn, involves the computation of a matrix inverse (Q) and a pseudo-inverse (L^+). These matrices belong to $\mathbb{R}^{n \times n}$; so, both operations have $O(n^3)$ complexity [29]. In addition, the *for each* loop within the *while* runs over the set of edges again; and has an $O(m)$ complexity again. Overall complexity of the algorithm is:

$$O(m * [(2 \cdot n^3) + m]) = O((2 \cdot m \cdot n^3) + (m^2)) = O(m \cdot n^3) \quad (18)$$

Therefore, our proposed *iteratively greedy rank-and-regress* algorithm is still polynomial time; albeit a relatively higher order polynomial. In the worst case, $m = O(n^2)$, which yields a further reduction of the form in (18) to $O(n^5)$. For most real world graphs, m is often $O(n)$ as opposed to $O(n^2)$; which reduces the complexity by an order of magnitude to $O(n^4)$ for real world graphs.

Further reduction in time complexity can be attained by using the incremental computation methods for Q and L^+ [18, 22, 42]; as well as, hardware accelerators (e.g. GPUs) [44, 47].

6 Related Work

The literature on spanning trees of graphs is vast. While it is impossible - or, at least, impractical - to provide a comprehensive survey, we give a brief summary of some of the most commonly used variants here. As stated earlier in this work, *Minimum Spanning Trees* (MST) are the most pervasively known class of spanning trees; and the *Prim's* and *Kruskal's* algorithms the most widely used for constructing MSTs [48]. The MST is, alas, most meaningful for weighted graphs; and, at best, of limited use in the unweighted case. Another class of spanning trees are the shortest path trees [1] which try to construct compact spanning trees starting from a particular vertex. The spanning trees thus generated are not compact from the standpoint of the graph; and have found much use in fields of communication networks and distributed systems [2, 4]. For unweighted graphs, *breadth first* and *depth first* searches can also be used to generate spanning trees of extremal compactness from the standpoint of a vertex (the root) [19, 48]. Once again, these are not designed for - nor do they attain - desired compactness from the point of view of the graph itself. A population of random spanning trees of the graph is indeed more suitable for a statistical study. The most prominent work on random spanning trees is due to Aldous and Border, who individually studied this problem [9, 16].

Similar to the study of spanning trees, there is a thriving tradition in literature of extracting other sub-graphs of desired compactness from a graph. We refer the reader to [15], and the references there in, where the authors propound on graph *sparification* and *coarsening* techniques and present a unified theory of *sparification* and *coarsening*. Antecedents of such work are also found in [25, 26, 27, 39, 40].

And the world of directed graphs is not entirely untouched from this sort of activity either. All the aforementioned algorithms have some or the other analogue for the case of directed graphs. The spanning tree, in the case of a digraph, simply becomes a spanning rooted *arborescence*. Much of the algebraic and topological/graph-theoretic paraphernalia extends to the world of digraphs [6, 7, 13].

7 Conclusion and Future Work

In this work, we introduced the concept of *Most, and Least, Compact Spanning Trees* - viz. $T^*(G)$ and $T^\#(G)$ - of a simple, connected, undirected and unweighted graph $G(V, E, W)$. We then presented an *iteratively greedy rank-and-regress* method that produces at least one $T^*(G)$ or one $T^\#(G)$ by eliminating one extremal edge per iteration. The rank function for performing the elimination is based on making a greedy choice over the elements of the matrix of *relative forest accessibilities* (Q) of a graph and the related *forest distance* associated with the edges of G . The algorithm guarantees polynomial time convergence ($O(m \cdot n^3)$) in the general case, and ensures that at all intermediary stages of the *regress*, the graph stays connected. We provided empirical evidence in support of our methodology; and discussed potentials for computational efficiencies, along with relevant trade-offs, to enable extraction of $T^*(G)$ and $T^\#(G)$ within reasonable time limits on standard computational platforms.

While useful for practical purposes already, establishing formal bounds on the level of compactness produced by our proposed algorithm - akin to those available for random spanning trees [23] - is desirable. Also, the question of extending the proposed methodology to the case of directed graphs - or, digraphs - is tantalizing in its own right. Indeed, there exists an analogous matrix of *relative forest accessibilities* for the directed case [6, 7, 13]. We motivate all these for future work.

Acknowledgment

The authors would like to thank Dr. Daniel Boley and Dr. Zhi-Li Zhang for preliminary discussions related to this work.

References

- [1] https://en.wikipedia.org/wiki/shortest-path_tree.
- [2] https://link.springer.com/referenceworkentry/10.1007/0-306-48332-7_462.
- [3] <https://www.baeldung.com/cs/minimum-spanning-vs-shortest-path-trees>.
- [4] <https://www.sciencedirect.com/topics/computer-science/shortest-path-tree>.
- [5] http://wiki.gis.com/wiki/index.php/shortest_path_tree.
- [6] R. Agaev and P. Chebotarev. The matrix of maximum out forests of a digraph and its applications. *Automation and Remote Control*, 61(9):1424–1450, 2000.
- [7] R. Agaev and P. Chebotarev. Spanning forests of a digraph and their applications. *Automation and Remote Control*, 62(3):443–466, 2001.
- [8] R. Albert, H. Jeong, and A. L. Barabási. Error and attack tolerance of complex networks. *Nature*, 406(6794):378–382, 2000.
- [9] D. J. Aldous. The random walk construction of uniform spanning trees and uniform labelled trees. *SIAM Journal on Discrete Mathematics*, 3:450–465, 1990.
- [10] A. L. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999.
- [11] A. Ben-Israel and T. Greville. *Generalized Inverses: Theory and Applications*, 2nd edition. Springer-Verlag, 2003.
- [12] N. Biggs. *Algebraic Graph Theory*. Cambridge University Press, 1993.
- [13] D. Boley, G. Ranjan, and Z.-L. Zhang. Commute times for a directed graph using an asymmetric laplacian. *Linear Algebra and its Applications*, 435(2):224–242, 2011.
- [14] B. Bollobás. *Random Graphs*. Cambridge University Press, 2001.
- [15] G. Bravo Hermsdorff and L. Gunderson. A unifying framework for spectrum-preserving graph sparsification and coarsening. In H. Wallach, H. Larochele, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.
- [16] A. Broder. Generating random spanning trees. In *Proc. of the 30th Annual Symposium on Foundations of Computer Science*, pages 442–447, 1989.
- [17] S. Campbell and C. D. Meyer. *Generalized Inverses of Linear Transformations*. Pitman Publishing Company, 1979.
- [18] Y. E. Campbell and T. A. Davis. Computing the sparse inverse subset: An inverse multifrontal approach. *Tech. report TR-95-021, Univ. of Florida, Gainesville*, 1995.
- [19] N. Cesa-bianchi, C. Gentile, F. Vitale, and G. Zappella. Active learning on graphs via spanning trees. In *In NIPS Workshop on Networks across Disciplines*, page 2010, 2010.
- [20] P. Chebotarev and E. Shamis. The matrix-forest theorem and measuring relations in small social groups. *Automation and Remote Control*, 58(9):1505–1514, 1997.
- [21] P. Chebotarev and E. Shamis. On proximity measures for graph vertices. *Automation and Remote Control*, 59(10):1443–1459, 1998.
- [22] P. Chebotarev and E. Shamis. The forest metrics for graph vertices. *Electronic Notes in Discrete Mathematics*, 11:98–107, 2002.
- [23] F. Chung, P. Horn, and L. Lu. Diameter of random spanning trees in a given graph. *Journal of Graph Theory*, 69(3):223–240, 2012.
- [24] I. J. Farkas, I. Derényi, A. L. Barabási, and T. Vicsek. Spectra of real world graphs: Beyond the semicircle law. *Physical Review E*, 64(2), 2001.
- [25] F. Fouss, A. Pirotte, J. M. Renders, and M. Saerens. A novel way of computing similarities between nodes of a graph, with application to collaborative recommendation. In *Proc. 2005 IEEE/WIC/ACM Int’l Joint Conf. Web Intelligence*, pages 550–556, 2005.

- [26] F. Fouss, A. Pirotte, J. M. Renders, and M. Saerens. Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 19, 2007.
- [27] F. Fouss, L. Yen, A. Pirotte, and M. Saerens. An experimental investigation of graph kernels on a collaborative recommendation task. In *Proc. of the 6th International Conference on Data Mining*, 2006.
- [28] I. Herstein and D. Winter. *Matrix Theory and Linear Algebra*. Maxwell Macmillan International Editions, 1988.
- [29] R. A. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge University Press, Cambridge, 1985.
- [30] D. Kimovski, A. Marosi, S. Gec, N. Saurabh, A. Kertesz, G. Kecskemeti, V. Stankovski, and R. Prodan. Distributed environment for efficient virtual machine image management in federated cloud architectures. *Concurrency and Computation: Practice and Experience*, 30(20):e4220, 2018. e4220 cpe.4220.
- [31] D. Kimovski, N. Saurabh, S. Gec, V. Stankovski, and R. Prodan. Multi-objective optimization framework for vmi distribution in federated cloud repositories. In *Euro-Par 2016: Parallel Processing Workshops*, pages 236–247, Cham, 2017. Springer International Publishing.
- [32] S. J. Kirkland, M. Neumann, and B. L. Shader. Distances in weighted trees and group inverse of laplacian matrices. *SIAM J. Matrix Anal. Appl.*, 18:827–841, 1997.
- [33] D. J. Klein and M. Randić. Resistance distance. *J. Math. Chemistry*, 12:81–95, 1993.
- [34] C. D. Meyer. Generalized inversion of modified matrices. *SIAM Journal on Applied Mathematics*, 24(3):315–323, 1973.
- [35] C. D. Meyer. The role of the group generalized inverse in the theory of finite markov chains. *SIAM Rev.*, 17:443–464, 1975.
- [36] B. Mohar. The Laplace spectrum of graphs. In *Graph Theory, Combinatorics, and Applications*, volume 2, pages 871–898. Wiley, 1991.
- [37] B. Mohar. Laplace eigenvalues of graphs - a survey. *Discrete Math.*, 109:171–183, 1992.
- [38] B. Mohar. Some applications of Laplace eigenvalues of graphs. In *Graph Symmetry: Algebraic Methods and Applications*, volume NATO ASI Ser C 497, pages 225–275. Kluwer, 1997.
- [39] G. Ranjan, T. N. Nguyen, H. Mekky, and Z.-L. Zhang. On virtual id assignment in networks for high resilience routing: a theoretical framework. In *Proc. of the IEEE Global Communications Conference, GLOBECOM*, 2020.
- [40] G. Ranjan and Z.-L. Zhang. How to glue a robust smart-grid: A finite network theory of inter-dependent networks (extended abstract). In *Proc. of the 7th (2011) Cyber Security and Information Intelligence Research Workshop: THEME – Energy Infrastructure Cyber Protection (CSIIRW7)*, 2011.
- [41] G. Ranjan and Z.-L. Zhang. Geometry of complex networks and topological centrality. *Physica A: Statistical Mechanics and its Applications*, 392(17):3833–3845, 2013.
- [42] G. Ranjan, Z.-L. Zhang, and D. Boley. Incremental computation of pseudo-inverse of laplacian. In *Combinatorial Optimization and Applications*, pages 729–749, Cham, 2014. Springer International Publishing.
- [43] Z. N. Samani, N. Saurabh, and R. Prodan. Multilayer resource-aware partitioning for fog application placement. In *2021 IEEE 5th International Conference on Fog and Edge Computing (ICFEC)*, pages 9–18, 2021.
- [44] N. Saurabh. *Improving the Performance of Moore-Penrose pseudo-inverse for a Graph’s Laplacian Using GPU*. PhD thesis, 08 2014.
- [45] N. Saurabh, S. Benedict, J. G. Barbosa, and R. Prodan. Expelliarnus: Semantic-centric virtual machine image management in IaaS clouds. *Journal of Parallel and Distributed Computing*, 146:107–121, 2020.
- [46] N. Saurabh, J. Remmers, D. Kimovski, R. Prodan, and J. G. Barbosa. Semantics-aware virtual machine image management in IaaS clouds. In *2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 418–427, 2019.
- [47] N. Saurabh, A. L. Varbanescu, and G. Ranjan. Computing the pseudo-inverse of a graph’s laplacian using gpus. In *2015 IEEE International Parallel and Distributed Processing Symposium Workshop*, pages 265–274, 2015.
- [48] D. B. West. *Introduction to Graph Theory*, 2nd edition. Prentice Hall, 2001.
- [49] D. B. Wilson. Generating random spanning trees more quickly than the cover time. *Proc. of the twenty-eighth annual ACM symposium on Theory of Computing*, pages 296–303, 1996.
- [50] W. Xiao and I. Gutman. Resistance distance and laplacian spectrum. *Theoretical Chemistry Accounts*, 110:284–289, 2003.

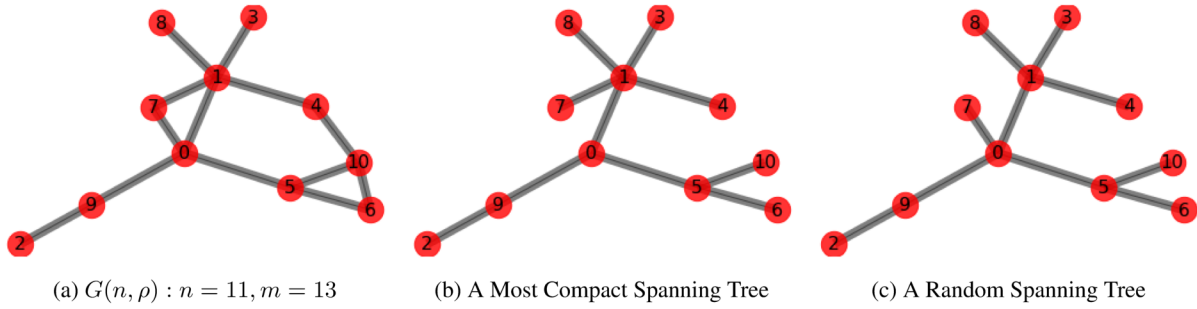


Figure 8: **ER-Graph I:** The MCST and RST differ in exactly one edge ($e_{0,7}$ vs. $e_{1,7}$). The resulting discrepancy in the average shortest path lengths is within a $\approx 1.2\%$ margin, and hence, for all practical purposes, negligible. The diameters of all three are exactly the same ($= 4$).

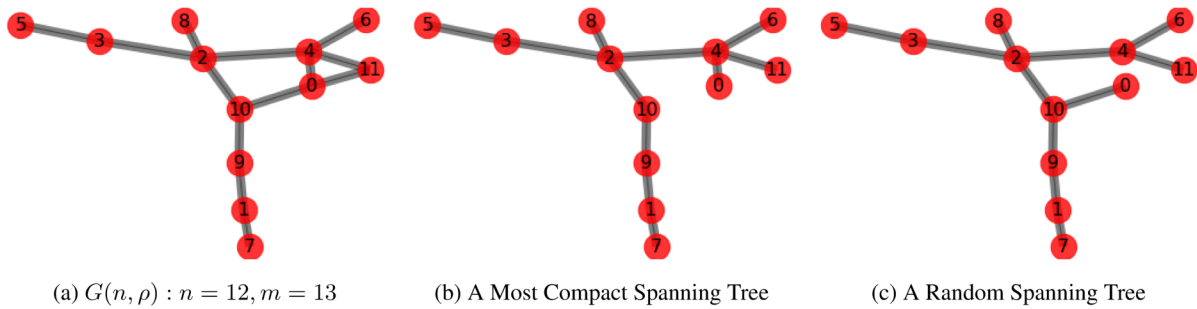


Figure 9: **ER-Graph II:** Once again, the MCST and the RST differ in exactly one edge ($e_{0,4}$ vs. $e_{0,10}$). The discrepancy in this case is even smaller; within a $\approx 1\%$ margin. The diameters in this case are also the same for all three ($= 6$).

APPENDIX

A Couple of Exceptions

During the experiments on ER-Graphs (c.f. §4.2), we encountered two samples where the MCST had a higher value for the average inter-vertex shortest path lengths than at least one of the random spanning trees. Figures 8 and 9, show the respective graphs and their spanning trees. The difference between the average shortest path lengths (compactness) in the MCST and one of the random trees in both cases is negligible. Nevertheless, it is there to be seen; and hence must be accounted for. Hence, we motivate obtaining formal bounds on the output of our algorithm - akin to those for random spanning trees [23] - in future work.