Algorithmica (2023) 85:1738–1753
https://doi.org/10.1007/s00453-022-01024-9

A Stronger Lower Bound on Parametric Minimum Spanning Trees

David Eppstein[1]

Received: 27 August 2021 / Accepted: 4 August 2022 / Published online: 22 August 2022

Abstract
We prove that, for an undirected graph with n vertices and m edges, each labeled with a linear function of a parameter λ, the number of different minimum spanning trees obtained as the parameter varies can be $\Omega(m \log n)$.

Keywords Parametric optimization · Minimum spanning tree · Parametric minimum spanning tree · Lower bounds

1 Introduction

In the *parametric minimum spanning tree problem* [19], the input is a graph G whose edges are labeled with linear functions of a parameter λ. For any value of λ, one can obtain a spanning tree Δ_λ as the minimum spanning tree of the weight functions, evaluated at λ. Varying λ continuously from $-\infty$ to ∞ produces in this way a discrete sequence of trees, each of which is minimum within some range of values of λ. How many different spanning trees can belong to this sequence, for a worst case graph, and how can we construct them all efficiently? This problem was introduced by Gusfield in 1979, who proved that a graph with n vertices and m edges can have at most $O(m\sqrt{n})$ different parametric spanning trees [19]; Gusfield's bound was improved to $O(mn^{1/3})$ in 1997 by Dey [7]. The only nontrivial lower bound known until now dates from 1995: there exist instances of the parametric minimum spanning tree problem for which the number of trees is $\Omega\big(m\alpha(n)\big)$, where α is the inverse Ackermann function [9]. It is a special case of a parametric matroid problem that also includes the k-set problem from discrete geometry; for parametric matroids, $O(mn^{1/3})$ is a tight bound,

A preliminary version of this paper was published in the Proceedings of the 17th Algorithms and Data Structures Symposium (WADS 2021).

✉ David Eppstein
eppstein@uci.edu

1 Computer Science Department, University of California, Irvine, Irvine, USA

but the instances proving this do not come from the parametric minimum spanning tree problem [9]. In this paper we improve the 25-year-old lower bound on the number of parametric minimum spanning trees from $\Omega\big(m\alpha(n)\big)$ to $\Omega(m \log n)$.

One motivation for understanding the combinatorial complexity of the parametric minimum spanning tree problem comes from its applications in bicriterion optimization, where each edge of a graph has two real weights of different types (say, investment cost and eventual profit) and one wishes to find a tree optimizing a nonlinear combination of the sums of these two weights (such as the ratio of total profit to total investment cost, the return on the investment). Each spanning tree of G may be represented by a planar point whose Cartesian coordinates are the sums of its two kinds of weights, giving an exponentially large cloud of points, one per tree. The convex hull of this point cloud has as its vertices the parametric minimum spanning trees and maximum spanning trees for linear weight functions obtained from the pair of weight values on each edge by using these values as coefficients. This construction of weight functions from pairs of weights can be understood as a form of projective duality transforming points into lines, under which the equivalence between the convex hull of the points representing trees into the lower envelope of lines representing their total weight is a standard reflection of that projective duality. Any bicriterion optimization problem that can be expressed as maximizing a quasiconvex function, or minimizing a quasiconcave function, of the two kinds of total weight automatically has its optimum at a convex hull vertex. It can be solved by constructing the sequence of parametric minimum spanning trees and evaluating the combination of weights for each one [22, 24]. Beyond spanning trees, other combinatorial optimization problems that have been considered from the same parametric and bicriterion point of view include shortest paths [3–5, 11], optimal subtrees of rooted trees [2], minimum-weight bases of matroids [9, 20], maximum flows and minimum cuts [15, 17], minimum-weight closures of directed graphs [10], sequence alignment [18], and the knapsack problem [8, 16, 21].

The algorithmic problem of constructing the sequence of all parametric minimum spanning trees, used in these bicriterion optimization applications, can be solved in time $O(mn \log n)$ [13] or in time $O(n^{2/3} \log^{O(1)} n)$ per tree [1].The question of which of these two solutions is asymptotically better remains open, because we do not know whether the $O(mn^{1/3})$ combinatorial bound on the number of trees is tight. Although our new lower bound does not resolve this question, it slightly reduces the large gap between the upper and lower bounds for this problem, and makes more significant progress in separating the combinatorial complexity of this problem from the trivial linear lower bound. Faster algorithms are also known for parametric minimum spanning trees in planar graphs [1] or for related optimization problems that construct only a single tree in the parametric sequence [6, 12, 25].

The main idea behind our new lower bound is a recursive construction of a family of graphs, formed by repeated replacement of edges by triangles (Fig. 1). The resulting graphs are planar, and more specifically, 2-trees. We also determine the parametric weight functions of these graphs by a separate recursive construction (Fig. 3). The number of parametric spanning trees produced by this construction can be analyzed via a standard divide-and-conquer recurrence, producing an $\Omega(n \log n)$ lower bound. The graphs constructed in this way are necessarily sparse, with $O(n)$ edges. To obtain our full $\Omega(m \log n)$ lower bound we use an additional packing argument, in which we find a dense graph containing many copies of our sparse lower bound construction, each contributing its own subsequence of parametric minimum spanning trees to the total.

2 Background and Preliminaries

The *minimum spanning tree* of a connected undirected graph with real-valued edge weights is a tree formed as a subgraph of the given graph, having the minimum possible total edge weight. As outlined by Tarjan [28], standard methods for constructing minimum spanning trees are based on two rules, stated most simply for the case when all edge weights are distinct:

- The *cut rule* concerns cuts in the graph, partitions of the vertices into two subsets; an edge *spans* a cut when its two endpoints are in different subsets. The cut rule states that (for distinct edge weights) the minimum-weight edge spanning any given cut in a graph belongs to its unique minimum spanning tree. Algorithmically, this can be used to identify edges that belong to the tree, and add them one by one to a forest until the result is a tree.
- The *cycle rule*, on the other hand, states that (again for distinct edge weights) the maximum-weight edge in any cycle of the graph does not belong to its unique spanning tree. This is used, for instance, in the linear-time randomized minimum-spanning tree algorithm of Karger, Klein, and Tarjan [23], to remove edges from dense graphs in order to make them more sparse.

As well as their algorithmic application, these rules are a frequent component of proofs involving minimum spanning trees, and we will use them for that purpuse. One consequence of these rules is that the minimum spanning tree depends only on the sorted ordering of the edge weights, rather than on more detailed properties of their numeric values.

Definition 1 The *parametric minimum spanning tree problem* is a computational problem whose input consists of an undirected connected graph with edges labeled by linear functions of a parameter λ (rather than with real numbers). For any value of λ, plugging λ into these functions produces a system of real weights for the edges, and therefore a minimum spanning tree Δ_λ. Different values of λ may produce different trees, and the task is to obtain a complete description of which tree is minimum for each possible value of λ. The output should be an ordered list of intervals of λ within which a given tree is minimum, and the tree that is minimum within that interval. The *combinatorial complexity* of a parametric minimum spanning tree instance is the number of intervals in this output.

If we plot the graphs of the linear functions of a parametric minimum spanning tree instance, as lines in the (λ, weight) plane, then the geometric properties of this *arrangement of lines* are closely related to the combinatorial properties of the parametric minimum spanning tree problem. In particular, we have:

Observation 2 Every parametric minimum spanning tree instance has a finite sequence of output intervals, whose endpoints are λ-coordinates of the crossing points of the arrangement of lines described above.

Proof As λ varies continuously, the sorted ordering of the weights will remain unchanged except when λ passes through one of these crossing points, where the set of lines involved in any crossing will reverse their weight order. The result follows from the well-known fact that, among the spanning trees of any fixed graph, the choice of which one is the minimum spanning tree for a system of edge weights on the graph depends only on the sorted ordering of the weights. □

In particular, m lines have $O(m^2)$ crossings and there can be only $O(m^2)$ distinct trees in the sequence of parametric minimum spanning trees. However, a stronger bound, $O(mn^{1/3})$, is known [7].

Observation 3 The worst-case instances of the parametric minimum spanning tree problem, the ones with the most trees for their numbers of edges and vertices, can be assumed to have distinct edge weight functions whose arrangement of lines has only *simple crossings*, crossings of exactly two lines.

Proof In any other instance, perturbing the edge weight functions by a small amount will preserve the ordering of weights away from the crossings of its lines, and therefore will preserve its sequence of trees away from these crossings, while only possibly increasing the number of breakpoints near perturbed crossings of multiple lines, which become multiple simple crossings. □

Therefore, in constructing lower bound examples for the parametric minimum spanning tree problem, we will only need to consider instances in which each weight function corresponds to a distinct line, and in which the lines have only simple crossings.

Observation 4 For a parametric minimum spanning tree instance whose arrangement of lines has only simple crossings, the only possible change to the minimum spanning tree at a breakpoint is a *swap*, a change to the tree in which one edge (corresponding to one of the two crossing lines at a simple crossing) is removed, and the other edge (corresponding to the other of the two crossing lines) is added in its place.

Proof The sorted ordering of other pairs of edges does not change between parameter values before the crossing and after it. Every edge of the minimum spanning tree before the crossing, other than one of the two edges whose lines cross, can only be the shortest edge across the cut between the two subtrees formed by removing it from the tree, by the cut rule and the fact that it is the only tree edge that crosses this cut. For parameter values after the crossing, its status as shortest across the same cut does not change, so it is still part of the minimum spanning tree. A symmetric argument shows that every tree edge after the crossing, other than the two edges involved in the crossing, must be part of the tree prior to the crossing as well. Therefore, the only edges that can change from being part of the tree to being out of it are the two involved in the crossing. □

For more details on this correspondence between the geometry of line arrangements and the sequence of parametric minimum spanning trees, and generalizations of this correspondence to other matroids than the matroid of spanning trees, see our previous paper on this topic [9].

3 Replacing Edges by Triangles

A *2-tree* is a graph obtained from the two-vertex one-edge graph K_2 by repeatedly adding new degree-two vertices, adjacent to pairs of adjacent earlier vertices. Equivalently, they are obtained by repeatedly replacing edges by triangles. These graphs are planar and include the maximal outerplanar graphs [26]; their subgraphs are the *partial 2-trees*, graphs of treewidth ≤ 2 [30]. The graphs we use in our lower bound are a special case of this construction where we apply this edge replacement process simultaneously to all edges in a smaller graph of the same type.

Definition 5 We define the first graph Δ_0 in our sequence of graphs to be the graph K_2, and then for all $i > 0$ we define Δ_i to be the graph obtained by replacing each edge of Δ_{i-1} by a triangle, consisting of the replaced edge and a two-edge path through a new vertex.

It seems natural to call these *complete 2-trees*, by analogy to complete trees (whose leaves are repeatedly replaced by stars for a given number of levels) but we have been unable to find this usage in the literature. The graphs Δ_i for $i \leq 3$ are depicted in Fig. 1.

Observation 6 Δ_i has 3^i edges and $(3^i + 3)/2$ vertices.

Proof The bound on the number of edges follows by induction from the fact that each replacement of edges by triangles triples the number of edges. Similarly, the bound on the number of vertices follows by induction on i, using the observations that each edge of Δ_{i-1} leads to a newly added vertex in Δ_i and that $(3^{i-1}+3)/2+3^{i-1} = (3^i+3)/2$. □

What happens when we perform the same sort of replacement, of an edge by a triangle, in a spanning tree problem? For a non-parametric minimum spanning tree, the answer is given by the following lemma.

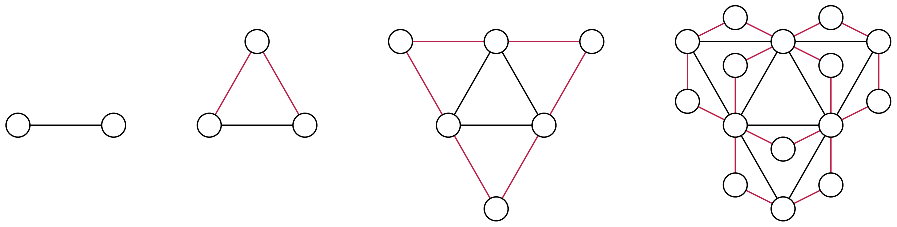
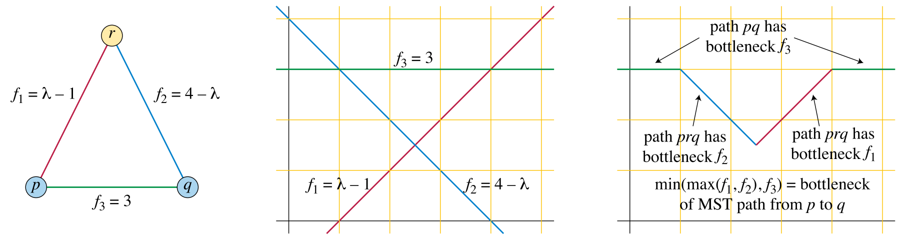


Fig. 1 Recursively constructing a family of 2-trees Δ_i (here, $i = 0, 1, 2, 3$ in left-to-right order) by repeatedly replacing every edge of Δ_{i-1} by a triangle, according to Definition 5



r
p
q
$f_1 = \lambda - 1$
$f_2 = 4 - \lambda$
$f_3 = 3$
$f_3 = 3$
$f_1 = \lambda - 1$
$f_2 = 4 - \lambda$
path pq has bottleneck f_3
path prq has bottleneck f_2
path prq has bottleneck f_1
$\min(\max(f_1, f_2), f_3)$ = bottleneck of MST path from p to q

Fig. 2 A parametric spanning tree problem on a single triangle pqr, and the graph of the bottleneck edge weight on the path from p to q in the parametric spanning tree, as a function of the parameter λ

Definition 7 Let G be a connected graph or subgraph with distinct edge weights, and let p and q be any two vertices in it. We define the *bottleneck edge* from p to q in G to be the maximum-weight edge e in the path from p to q in the minimum spanning tree of G.

Lemma 8 *Let graph G contain edge pq, and replace this edge by a triangle pqr to form a larger graph G^+. Suppose that the edges in G^+ have distinct edge weights, and use these weights to assign weights to the edges in G, with the following exception: in G, give edge pq the weight of the bottleneck edge from p to q in the subgraph formed by triangle pqr instead of the weight of pq. Then, the minimum spanning tree of G^+ has the same set of edge weights as the minimum spanning tree of G, together with the minimum weight of a non-bottleneck edge in triangle pqr.*

Proof If pq is the heaviest edge in pqr then the path from p to q in the minimum spanning tree of pqr passes through r, the bottleneck edge is the heavier of the two edges on this path, and the minimum non-bottleneck edge is the lighter of its two edges. Otherwise, pq is the bottleneck edge and again the minimum non-bottleneck edge is the lighter of the two remaining edges incident to r. Applying the cut rule to the cut separating r from the rest of the graph shows that the minimum non-bottleneck edge is an edge of the minimum spanning tree of G^+. Since we did not include its edge weight in the weights for G, its weight is not included in the set of edge weights of the minimum spanning tree for G.

Contracting this minimum non-bottleneck edge in G^+ produces a multigraph with two copies of edge pq, the lighter of which is the bottleneck edge. Therefore, if we keep only the lighter of the two edges, we obtain the weighting on G as a contraction of a minimum spanning tree edge in G^+. This contraction preserves the set of remaining minimum spanning tree weights, as the lemma states. □

It follows that in the parametric case, replacing an edge pq by a triangle pqr through a new vertex r, with linear parametric weights on each triangle edge, causes that edge to behave as if it has a nonlinear piecewise linear weight function attached to it, the function mapping the parameter λ to the bottleneck edge weight from p to q in triangle pqr. Figure 2 shows an example of three parametric weights on a triangle pqr and this bottleneck weight function, with the weights chosen so that the function has three breakpoints. This example function will be important for our construction; note that if we perturb these three weight functions within small neighborhoods of their coefficients, we will obtain a qualitatively similar bottleneck weight function, with three breakpoints at nearby points.

4 Weighted 2-Trees

We now describe how to assign parametric weights to the edges of Δ_i to obtain our $\Omega(n \log n)$ lower bound. As a base case, we may use any linear function as the weight of the single edge of Δ_0; it can have only one spanning tree, regardless of this choice. For Δ_i, with $i > 0$, we perform the following steps to assign its weights:

- Construct the weight functions for the edges of Δ_{i-1}, recursively.
- Apply a linear transformation to the parameter of these weight functions (the same transformation for each edge) so that, in the arrangement of lines representing the graphs of these weight functions, all crossings occur in the interval [0, 1] of λ-coordinates. Additionally, scale these weight functions by a sufficiently small factor ϵ so that, within this interval, they are close enough to the λ-axis, for a meaning of "close enough" to be specified below.
- Construct Δ_i by replacing each edge pq in Δ_{i-1} by a triangle pqr, with a new vertex for each triangle. Color the three edges of each triangle red, blue, and green, as in Fig. 2(left), with pq colored green and the other two edges colored red and blue (choosing arbitrarily which one to color red and which one to color blue).
- Give each edge of Δ_i a transformed copy of the weight function of the corresponding edge of Δ_{i-1}, transformed as follows:
  - For a green edge pq, corresponding to an edge of Δ_{i-1} with weight function $f(\lambda)$, give pq the weight function $f(\lambda - 4.5) + 3$. The addition of 3 lifts the green lines (the graphs of the weight functions of green edges) to be close to the green line of Fig. 2(center), and offsetting λ by 4.5 shifts the range of values of λ within which these green lines cross each other to [4.5, 5.5], a subinterval of the λ-extent $[4, \infty)$ of the right green segment of Fig. 2(right).
  - For a red edge pr, corresponding to an edge pq of Δ_{i-1} with weight function $f(\lambda)$, give pr the weight function $f(3.75 - \lambda) + \lambda - 1$. This transformation causes the red lines to lie close to the red line of Fig. 2(center). The range of values of λ where they cross is shifted to [2.75, 3.75], which lies within the λ-extent [2.5, 4] of the red segment of Fig. 2(right). Because the transformation negates λ in the argument to f, it reverses the ordering of the crossings within that range.
  - For a blue edge qr, corresponding to an edge pq of Δ_{i-1} with weight function $f(\lambda)$, give qr the weight function $f(\lambda - 1.25) + 4 - \lambda$. This transformation causes the blue lines to lie close to the blue line of Fig. 2(center). The range of values of λ where they cross is shifted to [1.25, 2.25], which lies within the λ-extent [1, 2.5] of the red segment of Fig. 2(right).
- Perturb all of the weight functions, if necessary, so that all crossings of two weight functions have different λ-coordinates, without changing the left-to-right ordering of the crossings between any one weight function and the rest of them.

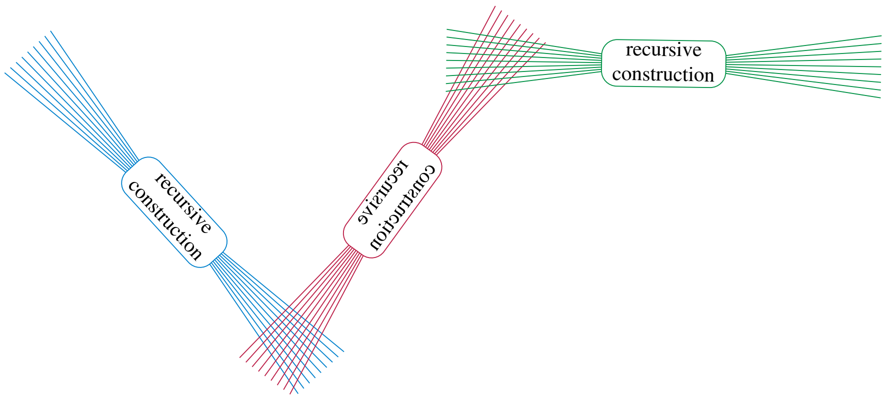


Fig. 3 Recursive construction for the parametric weight functions of the graphs Δ_i, shown here as an arrangement of lines in a plane whose horizontal coordinate is the parameter λ and whose vertical coordinate is the edge weight at that parameter value. The reversed text in the central recursive construction indicates that the construction is reversed left-to-right relative to the other two copies

This construction is depicted schematically, in the (λ, weight) plane, in Fig. 3. We are now ready to define what it means for the weight scaling factor ϵ to be small enough, so that the scaled weight functions are "close enough" to the λ axis. Intuitively, what we mean by this is that, with this choice, the left-to-right ordering of crossing points, according to their λ-coordinates, is the same as in the figure. That is:

1. All crossings of blue with green lines are first, to the left of other types of crossings.
2. All crossings of two blue lines, in one copy of the recursively constructed blue subarrangement of lines, are next.
3. All crossings of blue with red lines are third.
4. All crossings of two red lines, in a second (reversed) copy of the recursively constructed subarrangement, are fourth.
5. All crossings of red with green lines are fifth.
6. All crossings of two green lines, in the third copy of the recursively constructed subarrangement, are last, to the right of all other types of crossings.

Lemma 9 *For sufficiently small choices of ϵ, the crossings in our construction will have the ordering described above.*

Proof Our construction automatically places all monochromatic crossings into disjoint unit-length intervals with these orderings, in the same order that they had in the recursive construction. The bichromatic crossings of Fig. 2 are separated from these unit-length intervals by a horizontal distance of at least 0.25, and sufficiently small values of ϵ will cause the bichromatic crossings of Δ_i to be arbitrarily close to the positions of the crossings with the same color in Fig. 2. □

Figure 4 depicts this construction for Δ_2, with the crossing ordering described above.

Lemma 10 *Let Δ_i be weighted by the recursive construction above, with a small-enough choice of ϵ to satisfy Lemma 9. Then, within each of the unit-length intervals*

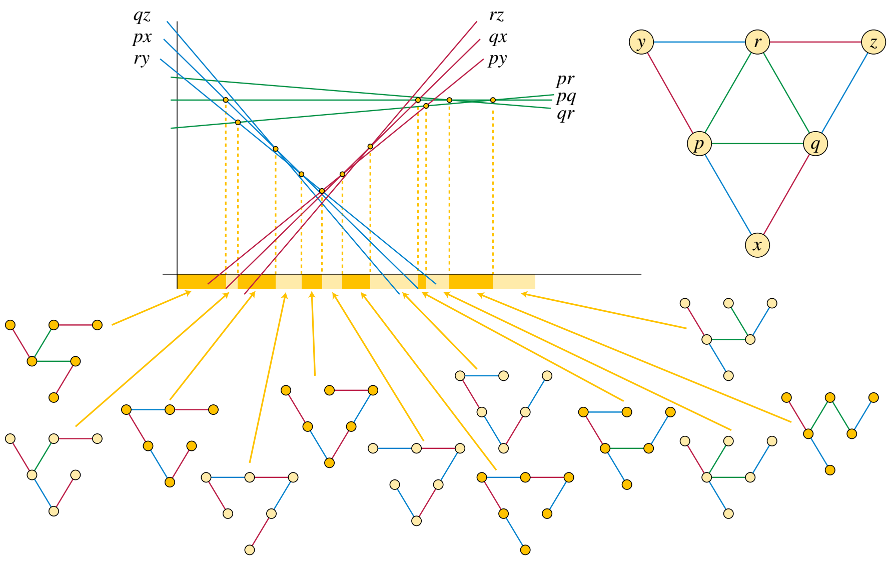


Fig. 4 Δ_2 (upper right) as parametrically weighted in our construction, with the graphs of each weight function shown as lines in the (λ, w) plane (upper right), and the resulting sequence of 12 parametric minimum spanning trees (bottom). The marked crossings of pairs of lines correspond to breakpoints in the sequence of trees (Color figure online)

containing a copy of the recursive construction, the changes in the sequence of parametric minimum spanning trees within these intervals exactly correspond to the changes in the trees of Δ_{i-1} *from the recursive construction.*

Proof Consider any one of these intervals, for the copy of the recursive construction with color c, and any triangle pqr of the construction of Δ_i. By Lemma 9, the three weight functions of pqr form a graph qualitatively similar to that of Fig. 2, with the interval contained within a segment of that graph within which the bottleneck edge has color c. Therefore, by Lemma 8, for each value of λ within this interval, the minimum spanning tree of Δ_i corresponds to the minimum spanning tree of Δ_{i-i} according to the weight of this c-colored edge, together with the minimum non-bottleneck edge in each triangle. The changes to the minimum spanning tree of Δ_i are exactly those from the recursive construction. Because of the crossing order described above, the choice of which edge in each triangle is the minimum non-bottleneck edge does not change within this interval, so the only breakpoints in the interval are those coming from the recursive construction. □

Lemma 11 *For weights constructed as above, the number of distinct parametric minimum spanning trees for* Δ_i *is at least as large as*

$$N(i) = \frac{i3^i}{2} + \frac{3^i + 3}{4}.$$

Proof We prove by induction on i that the number of trees is at least as large as the solution to the recurrence

$$N(i) = 3N(i-1) + \frac{3^i - 3}{2}.$$

To prove this, it is easier to count the number of breakpoints, values of λ at which the tree structure changes; the number of trees is the number of breakpoints plus one. In each copy of the recursive construction, this number of breakpoints is exactly $N(i-1) - 1$, so by Lemma 10 the total number of breakpoints appearing in these three copies is $3N(i-1) - 3$.

Additional breakpoints happen within the ranges of values for λ at which (in the (λ, weight) plane) pairs of lines of two different colors cross. Because of the reversal of the red copy of the recursive construction, the minimum spanning trees immediately to the left and right of these regions of bichromatic crossings correspond to the same trees in Δ_{i-1}: the bottleneck edges that are included in these minimum spanning trees come from the same triangles, but with different colors. In the regions where the green lines cross lines of other colors, the minimum non-bottleneck edge in each triangle does not change, so each green bottleneck edge in the minimum spanning tree must be exchanged for a red or blue one. Each change to a tree within this crossing region removes a single edge from the minimum spanning tree and replaces it with another single edge, the two edges whose two lines cross at the λ-coordinate of that change. Therefore, no matter what sequence of changes is performed, to exchange all green bottleneck edges for all red or blue ones requires a number of crossings equal to the number of edges in the minimum spanning tree of Δ_{i-1}, which is $(3^{i-1} + 1)/2$ by Lemma 6. We get this number of breakpoints at the region where the green and blue lines cross, and the same number at the region where the red and green lines cross.

The analysis of the number of breakpoints at the region where the blue and red lines cross is similar, but slightly different. Immediately to the left and right of this region, the the bottleneck edge in each triangle and the minimum non-bottleneck edge in the triangle are red and blue, but in a different order to the left and to the right. Therefore, in triangles where the bottleneck edge is part of the minimum spanning tree (as is always the minimum non-bottleneck edge), nothing changes. However, in triangles where the bottleneck edge is *not* part of the minimum spanning tree, there is a change, to the minimum non-bottleneck edge, from before this crossing region to after it. These triangles correspond to edges of Δ_{i-1} which do not belong to the minimum spanning tree (for the parameter values in this range), of which there are $(3^{i-1} - 1)/2$ by Lemma 6. By the same argument as before, the crossing region must contain at least this many breakpoints.

Adding together the $3N(i-1)-3$ breakpoints from the recursive copies, the $(3^{i-1} + 1)/2$ breakpoints for the green–red and green–blue crossing regions, the $(3^{i-1} - 1)/2$ breakpoints for the red–blue crossing region, and $+1$ to convert numbers of breakpoints to numbers of distinct trees, and simplifying, gives the right hand side of the recurrence. A straightforward induction shows that the solution to the recurrence is the formula given in the statement of the lemma. □

For $i = 0, 1, 2, \ldots$ the number of trees given by this formula is

$$1, 3, 12, 48, 183, 669, 2370, 8202, 27885, 93495 \ldots$$

For instance, Δ_1 has three trees with the weighting given in Fig. 2: the bottleneck function shown in the figure has four linear pieces, but the red and blue pieces both correspond to the same tree, with a different edge on the path pqr as the bottleneck edge. Figure 4 shows the 12 trees for Δ_2.

5 Packing into Dense Graphs

The lower bound obtained from Lemma 11 applies only to sparse graphs, where the numbers of vertices and edges are within constant factors of each other. However, we want a bound that applies more generally, for graphs with significantly more edges than vertices. The other direction, for graphs with significantly fewer edges than vertices, is less interesting. To achieve many fewer edges than vertices, it is necessary to allow disconnected graphs, and consider minimum spanning forests instead of minimum spanning trees; but with these modifications one can obtain a lower bound simply by including isolated vertices in the counting argument of Lemma 11.

To achieve many more edges than vertices, we use the following construction for packing many instances of a sparse lower bound graph into a single denser graph. It does not require any detailed knowledge of the structure of the sparse graph.

Lemma 12 *Let G be a parametrically weighted graph with N vertices and M edges, whose sequence of parametric minimum spanning trees has length T, and let k be a positive integer satisfying $k \le M$. Then there is a parametrically weighted graph H with $N + 3M$ vertices and $(2k + 2)M$ edges whose sequence of parametric minimum spanning trees has length at least $2kT$.*

Proof We construct H from G in the following steps, illustrated in Fig. 5.

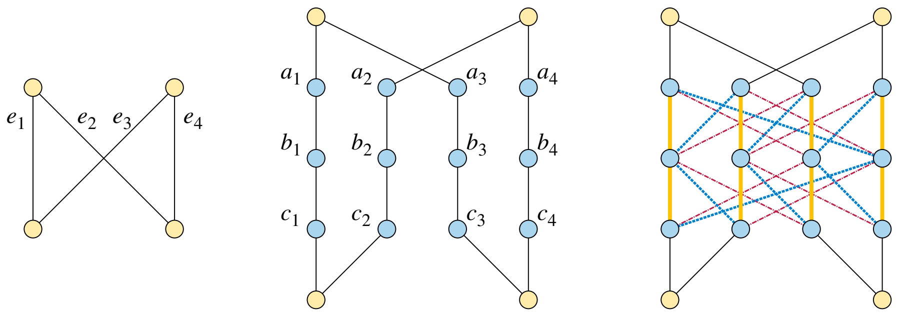


Fig. 5 The construction of Lemma 12, applied to a graph G with four vertices and four edges (left), with the parameter $k = 3$. The central graph is a subdivision of each edge of this graph into a four-edge path, with vertices labeled as shown, and the graph on the right is the final construction H, with the colors and textures of edges indicating the partition of its edges into four subgraphs H_0 (thin black edges), H_1 (thick yellow edges), H_2 (dotted blue edges), and H_3 (dashed red edges) (Color figure online)

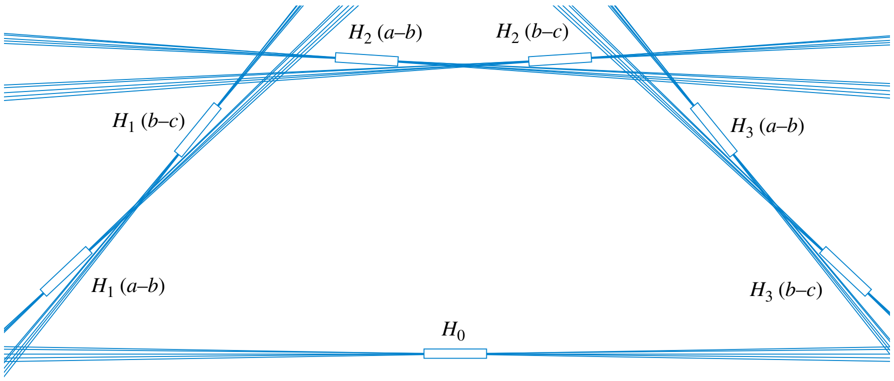


Fig. 6 An arrangement of lines for the weight functions of Lemma 12 with $k = 3$. The small rectangles indicate transformed neighborhoods of the unit λ-interval, containing all crossings of the bundle of lines associated with each subgraph

- Number the edges of G as $e_0, e_2, \ldots e_{M-1}$ arbitrarily.
- Subdivide each edge e_i of G, connecting two vertices u and v, into a four-edge path u–a_i–b_i–c_i–v. (It is arbitrary which vertex of this path we call a_i and which we call c_i.)
- Add additional edges from b_i to a_j and c_j, for each i and each $j = i + 1, i + 2, \ldots i + k - 1 \bmod M$.

Given this construction, we define subgraphs H_j as follows:

- H_0 consists of all edges connecting vertices of G to new vertices a_i or c_i.
- H_j consists of all edges from b_i to a_{i+j-1} or c_{i+j-1}, for all i, with indexes taken modulo m.

Then, for $i = 1, 2, \ldots k$, the graph $H_0 \cup H_i$ is isomorphic to a subdivision of G, with $H_0 \cup H_1$ being the subdivision we used to construct H and the others obtained in the same way but with permuted connections.

As in Lemma 11, we flatten the arrangement of lines for the weighting of G so that its crossings all lie within a small neighborhood of the unit interval of the λ-axis, without changing its sequence of parametric minimum spanning trees. We then apply linear transformations to the system of weights for the edges in each copy H_j with $j > 0$, as detailed below, while using small-enough weights for all edges in H_0 so that these edges belong to all minimum spanning trees for parameters in the range covered by the transformed unit intervals shown in Fig. 6. More specifically, for each $j > 0$ we use one transformed copy of the weights in G for the a–b edges in H_j, and a second transformed copy for the b–c edges, arranged so that the transformed unit intervals containing the crossings within each copy project to disjoint intervals of the λ-axis, and so that all crossings of the a–b edges appear above all lines for the b–c edges and vice versa. Therefore, in the graph $H_0 \cup H_j$, the parametric trees in the parameter range where the a–b edges cross each other consist of all b–c edges (because those have smaller weight than the a–b edges in each path) together with a subset of the a–b edges corresponding to a spanning tree of G. Because we copied and transformed the weights of G for the a–b edges in this parameter range, we obtain T distinct trees of this type. To arrange the a–b and b–c parameter weights for H_i in this fashion, we transform them so that the a–b weights lie near the line $w = 3 - \lambda$, with crossings in the range $\lambda \in [1, 2]$, and so that the b–c weights lie near the line $w = \lambda - 3$, with crossings in the range $\lambda \in [4, 5]$. Then, we transform and flatten these combined weights of H_i, so that they again lie near the λ-axis with all crossings of edges of either type in the range $[0, 1]$.

We arrange the sets of lines associated with H_1, H_2, etc., so that the lines from each H_j pass above the crossings for each other H'_j, $j \neq j'$, and so that the range of parameters within which H_j has the lowest lines contains the two subranges where its a–b lines cross and where its b–c lines cross, again as shown in the figure. We may do this by finding a convex-downward polygonal chain with k sides (for instance the upper part of a regular $2k$-gon), in which all sides project to a range of λ-coordinates of more than unit length, and by transforming the weights of each H_i so that the unit interval of the λ-axis, near which all crossings of these weights occur, is transformed to the interior of one of the sides of this polygonal chain. Figure 6 shows the weights for three subgraphs H_1, H_2, and H_3, transformed in this way so that they are near the upper three sides of a hexagon. The weights for H_0 can be chosen to be near a horizontal line, below all crossings of the other weight functions, as also shown in the figure.

Therefore, within these subranges, the parametric minimum spanning trees for all of H will be the same as the trees for $H_0 \cup H_j$, because $H_0 \cup H_j$ spans H and has lower edge weights than any of the remaining edges. With this arrangement, we get $2kT$ distinct parametric minimum spanning trees, $2T$ for each H_j with $j > 0$, as well as additional trees that are not counted in the lemma. □

With this, we are ready to prove our main result:

Theorem 1 *There exists a constant C such that the following is true. Let n and m be integers with $n > 0$ and $2n - 3 \leq m \leq \binom{n}{2}$. Then there exists a parametrically weighted graph with n vertices and m edges, with at least $Cm \log n$ parametric minimum spanning trees.*

Proof Let $G = \Delta_i$, $N = (3^i + 3)/2$, and $M = 3^i$, with i chosen as large as possible so that $N + 3M \leq n$ and $4M \leq m$, and choose k as large as possible so that $(2k+2)M \leq m$; then $N = \Theta(n)$ and $M = \Theta(m/n)$. Apply Lemma 11 to give weights to G so that it has $\Omega(n \log n)$ parametric minimum spanning trees, and apply Lemma 12 to construct a parametrically weighted graph H with $N + 3M$ vertices and $(2k + 2)M$ edges that has $\Omega(m \log n)$ parametric minimum spanning trees. If necessary, add leaf vertices to H to increase its number of vertices to n, and then add high-weight edges to increase its number of edges to m without affecting this sequence of parametric spanning trees. □

6 Conclusions and Open Problems

We have shown that instances of parametric minimum spanning tree problem with m vertices and n edges can have $\Omega(m \log n)$ different trees for different parameter values in the worst case, improving a 25-year-old $\Omega\big(m\alpha(n)\big)$ lower bound. Because of the structure of the graphs used in our lower bound construction, the new lower bound applies as well to the special cases of planar graphs and of bounded-treewidth graphs, both of which can have $\Omega(n \log n)$ parametric minimum spanning trees. However, our new lower bound is still far from the $O(mn^{1/3})$ upper bound, so there is plenty of room for additional improvement.

The parametric minimum spanning tree problem is a special case of parametric matroid optimization, on graphic matroids. The k-set problem in discrete geometry can be interpreted as a parametric matroid optimization problem on a different class of matroids, the uniform matroids. The best known upper bound for it has the same form as for parametric minimum spanning trees, $O(nk^{1/3})$, with the same proof [7], but the best known lower bound is stronger than our new parametric spanning tree lower bound, and takes the form $nc^{\sqrt{\log k}}$ for a constant c [29]. Bounds of this form make a natural target for the next improvement in the parametric minimum spanning tree problem.

Another related open problem concerns the *parametric bottleneck shortest path problem*, a parametric version of the problem of finding a path between two specified vertices that minimizes the maximum edge weight on the path. In the non-parametric version of the problem, a minimum spanning tree path is an optimal path, although faster algorithms are possible and the problem is also of interest in the case of directed graphs [14]. The same problem is also known in the equivalent maximin form as the widest path problem, where one possible optimal solution can be found as the path in a maximum spanning tree [27]. The parametric versions of minimum spanning trees and bottleneck shortest paths differ somewhat: a breakpoint in the piecewise linear parametric minimum spanning tree function (the function mapping the parameter value λ to the weight of its minimum spanning tree) might not be a breakpoint in the bottleneck shortest path problem (the maximum weight of an edge on the bottleneck shortest path problem) or vice versa. However, the bottleneck breakpoints that look locally like the minimum of two linear functions do correspond to breakpoints of the minimum spanning tree problem. For this reason, any asymptotic lower bound on the parametric bottleneck shortest path problem would also be a lower bound for parametric minimum spanning trees, and any asymptotic upper bound on the parametric minimum spanning tree problem is also an upper bound on parametric bottleneck shortest paths. Therefore, the known $O(mn^{1/3})$ upper bound on parametric minimum spanning trees applies as well to parametric bottleneck shortest paths. Additionally, our previous $\Omega\big(m\alpha(n)\big)$ lower bound also applies to parametric bottleneck shortest paths, but our new $\Omega(m \log n)$ bound does not. Can we strengthen the $\Omega\big(m\alpha(n)\big)$ bound for this problem?

Acknowledgements This research was supported in part by the US National Science Foundation under Grant CCF-2212129. The author has no financial or proprietary interests in any material discussed in this article.

References

1. Agarwal, P.K., Eppstein, D., Guibas, L.J., Henzinger, M.R. : Parametric and kinetic minimum spanning trees. In: Proceedings of the 39th IEEE Symposium on Foundations of Computer Science (FOCS'98), pp. 596–605 (1998). https://doi.org/10.1109/SFCS.1998.743510
2. Carlson, J., Eppstein, D.: The weighted maximum-mean subtree and other bicriterion subtree problems. In: Proceedings of the 10th Scandinavian Workshop Algorithm Theory (SWAT 2006), volume 4059 of Lecture Notes in Computer Science, pp. 397–408. Springer (2006). https://doi.org/10.1007/11785293_37
3. Carstensen, P.J.: Parametric cost shortest path problems. Unpublished memo, Bellcore (1984)
4. Castelli, L., Labbé, M., Violin, A.: Network pricing problem with unit toll. Networks **69**(1), 83–93 (2017). https://doi.org/10.1002/net.21701
5. Chakraborty, S., Fischer, E., Lachish, O., Yuster, R.: Two-phase algorithms for the parametric shortest path problem. In: Marion, J.-Y., Schwentick, T. (eds.) Proceedings of the 27th International Symposium on Theoretical Aspects of Computer Science (STACS 2010), volume 5 of LIPIcs, pp. 167–178. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2010) https://doi.org/10.4230/LIPIcs.STACS.2010.2452
6. Chan, T.M.: Finding the shortest bottleneck edge in a parametric minimum spanning tree. In: Proceedings of the 16th ACM-SIAM Symposium on Discrete Algorithms (SODA 2005), pp. 917–918. SIAM, (2005). https://dl.acm.org/citation.cfm?id=1070432.1070561
7. Dey, T.K.: Improved bounds for planar k-sets and related problems. Discrete Comput. Geom. **19**(3), 373–382 (1998). Announced at FOCS (1997) https://doi.org/10.1007/PL00009354
8. Eben-Chaime, M.: Parametric solution for linear bicriteria knapsack models. Manag. Sci. **42**(11), 1565–1575 (1996). https://doi.org/10.1287/mnsc.42.11.1565
9. Eppstein, D.: Geometric lower bounds for parametric matroid optimization. Discrete Comput. Geom. **20**(4), 463–476 (1998). Announced at STOC (1995) https://doi.org/10.1007/PL00009396
10. Eppstein, D.: The parametric closure problem. ACM Trans. Algorithms **14**(1), A2:1-A2:22 (2018). https://doi.org/10.1145/3147212
11. Erickson, J.: Maximum flows and parametric shortest paths in planar graphs. In: Charikar, M. (ed.) Proceedings of the 21st ACM-SIAM Symposium on Discrete Algorithms (SODA 2010), pp. 794–804. SIAM (2010). https://doi.org/10.1137/1.9781611973075.65
12. Fernández-Baca, D., Slutzki, G.: Linear-time algorithms for parametric minimum spanning tree problems on planar graphs. Theoret. Comput. Sci. **181**(1), 57–74 (1997). https://doi.org/10.1016/S0304-3975(96)00262-9
13. Fernández-Baca, D., Slutzki, G., Eppstein, D.: Using sparsification for parametric minimum spanning tree problems. Nordic J. Comput. **3**(4), 352–366 (1996)
14. Gabow, H.N., Tarjan, R.E.: Algorithms for two bottleneck optimization problems. J. Algorithms **9**(3), 411–417 (1988). https://doi.org/10.1016/0196-6774(88)90031-4
15. Gallo, G., Grigoriadis, M.D., Tarjan, R.E.: A fast parametric maximum flow algorithm and applications. SIAM J. Comput. **18**(1), 30–55 (1989). https://doi.org/10.1137/0218003
16. Giudici, A., Halffmann, P., Ruzika, S., Thielen, C.: Approximation schemes for the parametric knapsack problem. Inform. Process. Lett. **120**, 11–15 (2017). https://doi.org/10.1016/j.ipl.2016.12.003
17. Granot, F., Thomas McCormick, S., Queyranne, M., Tardella, F.: Structural and algorithmic properties for parametric minimum cuts. Math. Program. **135**(1–2, Ser. A), 337–367 (2012). https://doi.org/10.1007/s10107-011-0463-1
18. Gusfield, D., Balasubramanian, K., Naor, D.: Parametric optimization of sequence alignment. Algorithmica **12**(4–5), 312–326 (1994). https://doi.org/10.1007/BF01185430
19. Gusfield, D.: Bounds for the parametric minimum spanning tree problem. In: Proceedings of the West Coast Conference on Combinatorics, Graph Theory and Computing (Humboldt State Univ., Arcata, Calif., 1979), volume 26 of Congress. Numer., Winnipeg, Manitoba, pp. 173–181 (1980) Utilitas Math

20. Hassin, R., Tamir, A.: Maximizing classes of two-parameter objectives over matroids. Math. Oper. Res. **14**(2), 362–375 (1989). https://doi.org/10.1287/moor.14.2.362
21. Holzhauser, M., Krumke, S.O.: An FPTAS for the parametric knapsack problem. Inform. Process. Lett. **126**, 43–47 (2017). https://doi.org/10.1016/j.ipl.2017.06.006
22. Ishii, H., Shiode, S., Nishida, T., Namasuya, Y.: Stochastic spanning tree problem. Discrete Appl. Math. **3**(4), 263–273 (1981). https://doi.org/10.1016/0166-218X(81)90004-4
23. Karger, D.R., Klein, P.N., Tarjan, R.E.: A randomized linear-time algorithm to find minimum spanning trees. J. ACM **42**(2), 321–328 (1995). https://doi.org/10.1145/201019.201022
24. Katoh, N.: Bicriteria network optimization problems. IEICE Trans. Fundam. Electron. Commun. Comput. Sci. **E75–A**, 321–329 (1992)
25. Katoh, N., Tokuyama, T.: Notes on computing peaks in k-levels and parametric spanning trees. In: Souvaine, D.L. (ed.) Proceedings of the 17th Symposium on Computational Geometry (SoCG 2001), pp. 241–248. ACM (2001) https://doi.org/10.1145/378583.378675
26. Mitchell, S.L.: Linear algorithms to recognize outerplanar and maximal outerplanar graphs. Inf. Process. Lett. **9**(5), 229–232 (1979). https://doi.org/10.1016/0020-0190(79)90075-9
27. Pollack, M.: The maximum capacity route through a network. Oper. Res. **8**, 733–736 (1960). https://doi.org/10.1287/opre.8.5.733
28. Tarjan, R.E.: Data structures and network algorithms, volume 44 of CBMS-NSF regional conference series in applied mathematics. Society for Industrial and Applied Mathematics, Philadelphia (1983). https://doi.org/10.1137/1.9781611970265
29. Tóth, G.: Point sets with many k-sets. Discrete Comput. Geom. **26**(2), 187–194 (2001). https://doi.org/10.1007/s004540010022
30. Wald, J.A., Colbourn, C.J.: Steiner trees, partial 2-trees, and minimum IFI networks. Networks **13**(2), 159–167 (1983). https://doi.org/10.1002/net.3230130202