On the Natural Growth of Random Forests

Theodore Johnson, University of Florida
Harold S. Stone, IBM T. J. Watson Center

July 29, 1992

Abstract

We explore several aspects of the natural growth of random forests. We show that if initially there are k single-node forests and N nodes to add to the forest, and every forest node is equally likely to be connected to the next node added, then the tree that receives the last node will have an expected $2N/k$ nodes. We next explore the growth of a minimum weight forest. We show that the probability that a node in the forest is connected to the next node added to the forest depends primarily upon how recently the forest node was added to the forest, and that this distribution approaches a limit distribution. We obtain the surprising result that when a node is added to the forest, it is adjacent to the most recently added node 50% of the time.

1 Introduction

In this paper, we examine several distributions associated with randomly grown forests. We assume that there are k initial trees in the forest, each of which consists of a single node, and N nodes to be added to the forest. The forest grows by repeatedly adding a node not in the forest to a tree. The trees are never merged.

Randomly grown and minimum weight forests are used in many algorithms. Our results will provide a useful insight on their expected size and performance. This work was motivated by the desire to analyze the expected time complexity of a 2-matching algorithm [13]. Other works that use minimum weight spanning trees or forests include Held and Karp's Traveling Salesman Problem algorithms [4, 5], and the Matching algorithm of Desler and Hakami [2].

Our first growth model (the *uniform growth* model) assumes that when a node is added to the forest, the edge that connects it to the forest is equally likely to be adjacent to every node already in the forest. We show that this growth assumption leads to an easily solved system of equations that describe the distribution of tree sizes.

In our second growth model (the *minimum weight* model) we assume that all nodes in the graph are connected by weighted edges. We then seek to characterize the growth of a minimum weight forest. We

initially propose the uniform growth model as a tractable approximation to the minimum weight model. We show that the uniform growth model is a poor approximation, however, because the uniform growth model predicts short, bushy trees, while the minimum weight model predicts long, thin trees.

Previous work in characterizing the growth of forests has concentrated on forests in random graphs (see [1] for a survey), or on the natural growth of search trees (see [8] for a survey). The question of the weight of the minimum spanning tree of a randomly weighted graph has been addressed by Frieze [3] and Steele [12]. In this work, however, we are interested in the growth of random forests, and are not concerned about their weight.

2 Uniform Growth Model

In this section, we analyze the growth of random forests that have a tractable growth model. When we build the forest, we add to the forest an edge that connects a forest node to a nonforest node. We say that the forest node adjacent to the new edge *receives* the edge, and that edge and the nonforest node are *added* to the tree. In the *uniform growth* model, every forest node is equally likely to receive the next edge added to the forest.

We calculate the distribution of the tree sizes, and the size of the tree that receives the j^{th} node added to the forest.

Theorem 1 *The probability that the node that receives the j^{th} edge added to the forest is part of a tree that contains i nodes is*

$$(k + 2j + 1)/(k + 1)$$

Proof: We define $T_i(j)$ to be the expected number of trees with i nodes after j nodes have been added to the initial forest. The forest starts with k trees each of which contain a single node, so the initial conditions are:

$$T_1(0) = \quad k$$

$$T_i(0) = \quad 0 \quad \text{for } i > 1$$

Since all trees start with 1 node, after $j - 1$ nodes have been added, no tree has more than j nodes. The produces the additional constraint:

$$T_i(j-1) = \quad 0 \quad \text{for } i > j$$

Each time a node is added, it is added to some tree with i nodes, so the number of trees with i nodes decreases by 1, and the number of trees with $i+1$ nodes increases by 1. The probability of selecting a tree with i nodes at the j^{th} node selection is proportional to the total number of nodes in trees with i nodes. Before the j^{th} selection there are $k+j-1$ nodes in all of the trees, so that the probability of selecting a tree with i nodes is given by

$$\Pr[\text{Selecting a node in a tree with } i \text{ nodes at step } j] = \frac{iT_i(j-1)}{k+j-1}$$

This distribution yields the following equations for the tree distribution after the j^{th} selection in terms of trees that exist before the j^{th} selection:

$$\begin{aligned} T_1(j) &= T_1(j-1) - \frac{T_1(j-1)}{k+j-1} \\ &= T_1(j-1)\left(\frac{k+j-2}{k+j-1}\right) \end{aligned}$$

and

$$\begin{aligned} T_i(j) &= T_i(j-1) - \frac{iT_i(j-1)}{k+j-1} + \frac{(i-1)T_{i-1}(j-1)}{k+j-1} \\ &= T_i(j-1)\left(\frac{k+j-i-1}{k+j-1}\right) + T_{i-1}(j-1)\left(\frac{i-1}{k+j-1}\right) \end{aligned}$$

The solution to these equations with these initial conditions is:

$$T_i(j) = \frac{k\,(k-1)\,j\,!(k+j-i-1)!}{(j+1-i)!(k+j-1)!}$$

which can be shown by induction.

The next step is to compute the expected size of a tree that receives the j^{th} node. The probability of adding the j^{th} node to a tree of size i is proportional to the number of nodes in trees of size i, and therefore is $iT_i(j)/(k+j)$. As a result, the expected size of the tree that receives the j^{th} node is:

$$\left(\frac{1}{k+j}\right)\sum_{i=0}^{j+1} i^2 T_i(j)$$

We can show that the general form of the expected value is

$$(k+2j+1)/(k+1)$$

by using induction to show that:

$$\sum_{i=1}^{j+1} i^2 T_i(j) = \frac{(k+j)(k+2j+1)}{k+1}\bullet$$

If k is large, and $j \gg k$, then the size of this tree is approximately $2j/k$ – twice as large as the average tree size.

3 Minimum Weight Model

The uniform growth assumption is convenient for analytical tractability and is realistic for some situations. We also want to analyze minimum weight forests, and we would like to approximate the minimum weight model by the uniform growth model. Unfortunately, the approximation is inaccurate, and the actual distribution is quite complex. Our objective in this section is to show what relation exists between the two growth models.

In the *minimum weight* model, our starting assumptions again are that we are growing a forest, starting with k initial (isolated) nodes in the forest and N nodes not in the forest. We denote the set of forest nodes by F, and the set of nonforest nodes by $V - F$. There are $M = \binom{N}{2} + Nk$ edges connecting all pairs of nodes in the graph, except for pairs where both nodes are in F. There is a function defined on each edge, $w(e)$, the weight of edge e. At each step, we add a new node to the forest by finding the lowest weight edge that connects a node in the forest to a node not in the forest, then adding the edge and the attached node (i.e, apply Dijkstra's algorithm [9]). We assume that all edge weights are independently and identically chosen from a continuous distribution over the real numbers, $F(x)$. That is, $\Pr[w(e) < x | w(e_1) = x_1, w(e_2) = x_2, \ldots, w(e_l) = x_l] = \Pr[w(e) < x] = F(x)$. See Figure 1.

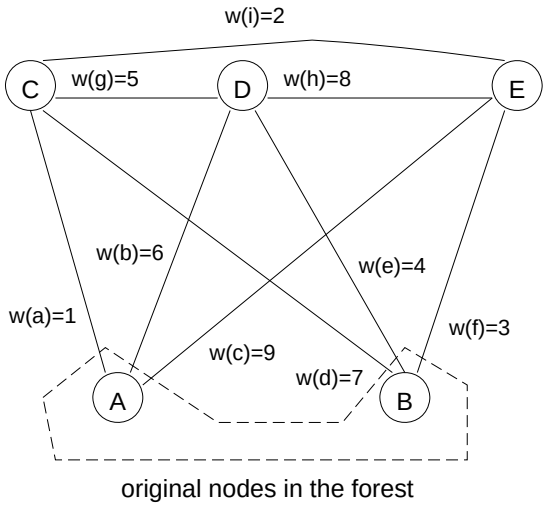


Figure 1: Initial graph (A and B are the roots of the forest)

We define a function of the nodes, $l(n)$, the *label* of the node. The original k forest nodes are labeled 1 through k, and the i^{th} node added to the forest is labeled $k + i$, $i = 1, \ldots, N$. We also label the edges

$e = (n_1, n_2)$ with $l(e) = \min(l(n_1), l(n_2))$.

We can classify the edges in the graph at any point in the forest growth as follows: Edges that connect two nodes in F (two forest nodes) are in E_F, edges that connect two nodes in $V - F$ (not in the forest) are in E_{V-F}, and edges that connect a node in F to a node in $V - F$ are in E_{cross} (cross edges). Edges in E_F can be subclassified as being either *tree* edges or *nontree* edges depending on whether they are part of the minimum weight forest.

To add the next node to the forest, the minimum weight edge in E_{cross} is chosen and made into a tree edge. At this point, we can label the new tree node and the unlabeled edges incident to the new tree node using the following algorithm:

```
min_weight_forest(V,F,E)
   Label the vertices in F by 1 through |F|.
   E_cross = edges in E incident to a vertex in F.
   Label the edges in E_cross by the label of the node in F.
   E_F = ∅.
   E_V-F = E − E_cross.
   i=0.
   While V-F is not empty do
      let e_min be the minimum weight edge in E_cross.
      Let u be the node incident to e_min that is in V-F.
      Remove from E_cross all edges incident to u, put them in E_F.
      Label u by k+i.
      Remove from E_V-F all edges incident to u, label them with k+i,
       and put them in E_cross.
```

The `min_weight_forest` algorithm is illustrated in Figures 1 through 6. Figure 1 shows the initial forest. Figure 2 shows the eligible edges (the cross edges) when the first edge is selected. The edges incident on the root nodes are labeled with the node's label. Figure 3 shows the cross edges before the second edge has been selected. Node C was added to the forest, and was labeled with 3. Edges leading from C to nodes not yet in the forest are also labeled with 3. Figure 4 shows the graph before last edge

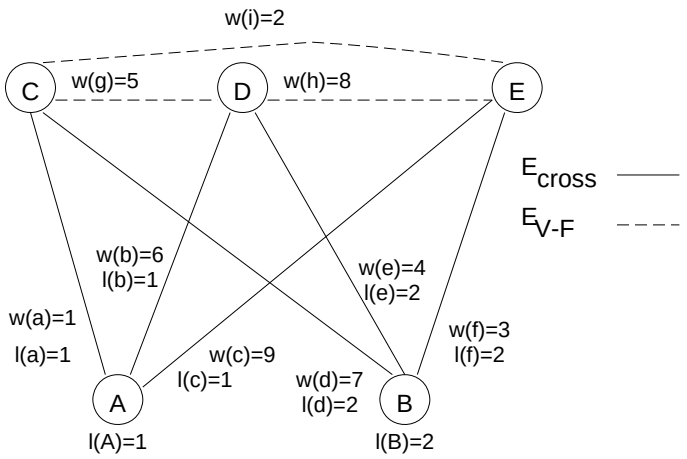


Figure 2: Selecting the first edge

has been selected, and Figure 5 shows the final forest.

To aid in calculating the distribution of the probability that a given node in the forest receives the next edge, we list the edges in the graph sorted by the edge weights. If the `min_weight_forest` algorithm chooses an edge labeled i as the next edge to add to the forest, then the node labeled i receives the edge. Therefore, we identify the edges in the list by their labels only. Figure 6 shows the list of edges with their corresponding edge labels, sorted by edge weight, that corresponds to our running example. We next prove this list of edges has a tractable distribution.

Definition: A $(N_1, N_2, \ldots, N_j)-$*permutation* is an ordered list of N_1 items labeled 1, N_2 items labeled 2, etc.

Lemma 1 *Label the edges using the edge labeling algorithm, then sort the edges by their weights. The resulting* $(\underbrace{N, N, \cdots, N}_{k \text{ times}}, N-1, N-2, \cdots, 2, 1)$*-permutation, L is uniformly randomly chosen from the set of all* $(N, N, \cdots, N, N-1, N-2, \cdots, 2, 1)$*-permutations.*

Proof: First, L is indeed a $(N, N, \cdots, N, N-1, N-2, \cdots, 2, 1)$-permutation, since the original k nodes each label N edges, and the node added at the i^{th} labeling step labels $N-i$ edges.

We consider the algorithm for labeling edges. The set of edges, E is partitioned into E_F, E_{V-F}, and E_{cross}. At each edge selection step, the algorithm searches E_{cross} and picks the edge with the lowest weight, e_{min}. Based on e_{min}, the algorithm removes some edges from E_{cross}, puts them in E_F, and also

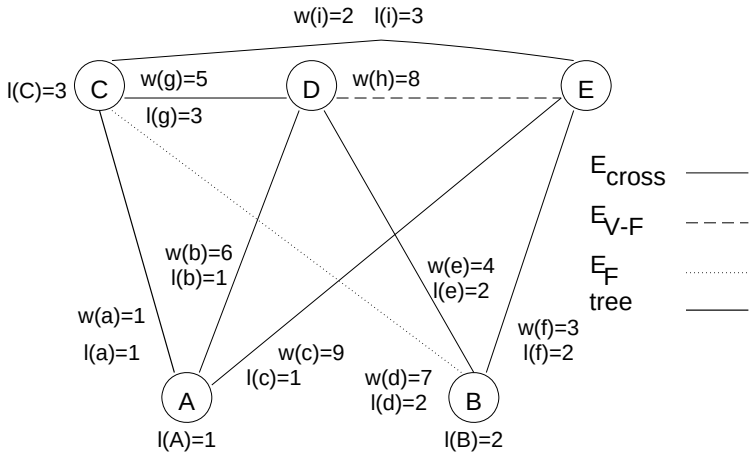


Figure 3: Selecting the second edge

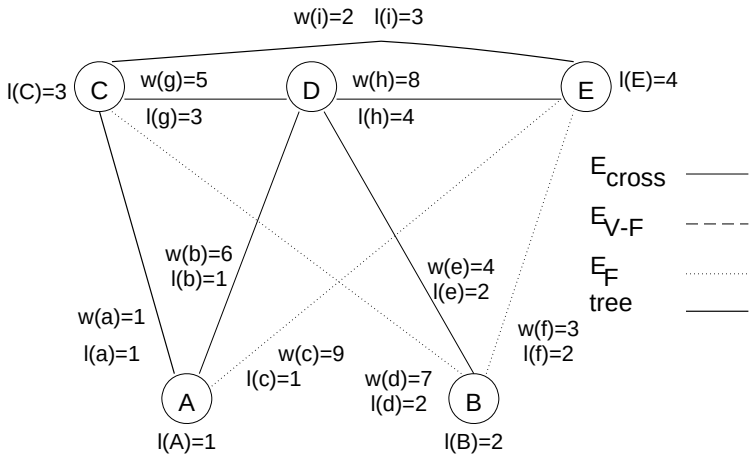


Figure 4: Selecting the third edge

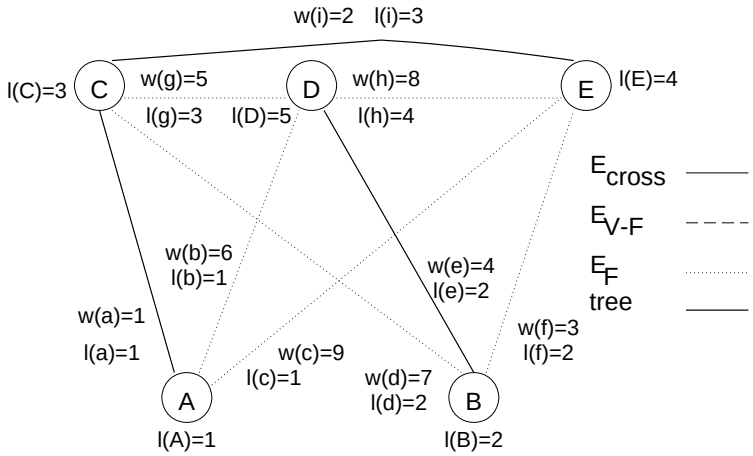
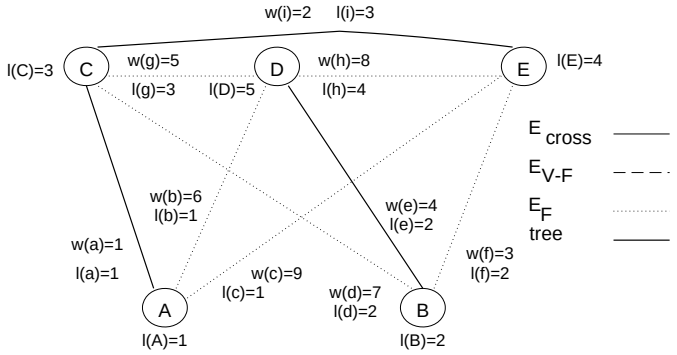


Figure 5: Final forest



edge	a	i	f	e	g	b	d	h	c
weight	1	2	3	4	5	6	7	8	9
label	1	3	2	2	3	1	2	4	1

Figure 6: Final forest and corresponding list of labeled edges

removes some edges from E_{V-F} and puts them in E_{cross}. At this point, the new edges that are placed into E_{cross} from E_{V-F} are labeled. This step repeats until E_{cross} is empty.

An edge that is removed from E_{V-F} is never returned to E_{V-F}. The edges are labeled only when they are removed from E_{V-F} and placed into E_{cross}. The weights of these edges are examined by the algorithm for the first time when the next edge is selected. Since the edge weights are independent and the weight of an edge is first read by the algorithm after the edge is labeled, the edge weight is independent of the edge label.

All edge weights have the same distribution, regardless of their labels. The permutation, L, is sorted by edge weight, so every edge is equally likely to be in every position. The resulting permutation of the edges is equally likely to be chosen from the set of all possible permutations, so L is a uniformly randomly chosen $(N, N, \cdots, N, N-1, N-2, \cdots, 2, 1)$-permutation•

The underlying combinatorial object is a matrix of independent and identically distributed edge weights. So far, we have made a transformation from the matrix of edge weights to a list of edge labels, and have shown that the list is uniformly distributed. At any step in the forest growing algorithm, only a subset of the edges are examined in order to find a least weight edge – those in E_{cross}. To find the edge that the `min_weight_forest` algorithm adds, we take the list that corresponds to E and remove the edges in E_F and E_{V-F} while preserving the order of the remaining edges. We then select the first edge on the list. We next prove some properties of the lists with removed edges:

Lemma 2 *For every $k \in 1, \ldots, N_1 + \cdots + N_j = M$, the probability that the k^{th} entry in a uniformly selected $(N_1, N_2, \ldots, N_j)$-permutation is labeled r, $r \in 1, \ldots, j$ is N_r/M.*

Proof: If we pick edge $e \in E$ as the edge in the k^{th} entry of the permutation, there are $(M-1)!$ ways to choose the remainder of the permutation. There are N_r ways to choose an edge e with a label r, so there are $N_r(M-1)!$ permutations with an edge labeled r in the k^{th} position. There are $M!$ possible permutations of the edges, so the probability that the edge in the k^{th} position is labeled r is N_r/M•

Lemma 3 *Let $k, l \in 1, \ldots, N_1 + \cdots + N_j = M$, $k \neq l$, $M > 1$. If the k^{th} entry of a uniformly selected $(N_1, \ldots, N_j)$-permutation is identified as being labeled r, $r \in 1, \ldots, j$, then the probability that the l^{th} entry is labeled s, $s \in 1, \ldots, j$ is*

$$\frac{N_s}{M-1} \quad r \neq s$$

$$\frac{N_s - 1}{M - 1} \quad r = s$$

Proof: We fix edge e in position r, then apply lemma 2.

Corollary 1 *Let P be a uniformly chosen $(N_1, \ldots, N_j)$-permutation, and identify the r^{th} entry as being labeled l, $r \in 1, \ldots, N_1 + \cdots + N_j$, $l \in 1, \ldots, j$. Create P' by taking the first $r-1$ and the last $N_1 + \cdots + N_j - r$ entries of P and concatenating them together. Then P' is equally likely to be any one of the $(N_1, \ldots, N_l - 1, \ldots, N_j)$-permutations.*

Proof: Apply lemma 3•

Suppose that we have run the algorithm once, labeled the edges, and created the list L. Suppose that we rerun the `min_weight_forest` algorithm on the same graph that resulted in L. By examining L, we want to determine what choices the algorithm makes. We then use the uniform randomness of L to make an expected case analysis of the forest growth.

At any point in the `min_weight_forest` algorithm, the algorithm examines a subset of the edges (those in E_{cross}), and chooses the lowest-weight edge to add to the forest. The list L is sorted by edge weight, so if we rule out the ineligible edges (those in E_{V-F} or in E_F), the first edge in L is the edge that the algorithm picks. This edge's label is the label of the forest node it is incident on, so we can keep track of how the forest grows.

At each edge selection step, we choose an edge that connects a node in the forest to a node not in the forest (i.e., an edge in E_{cross}). If we select the i^{th} edge, then nodes numbered $1, \ldots, k+i-1$ are in the forest. Nodes numbered $k+i, \ldots, k+N$ are not. The fact that we are selecting the i^{th} edge immediately gives us some information: edges labeled $i, \ldots, N$ are in E_{V-F}, and so are not eligible. Thus, we can simplify L by removing edges that will not be chosen.

Definition: L_i is L with edges labeled $i+k, \ldots, N+k$ removed. The list L_i contains exactly the edges in E_{cross} and E_F when the i^{th} edge is chosen.

Corollary 2 *L_i is equally likely to be any one of the $(\underbrace{N, \ldots, N}_{k \text{ times}}, N-1, \ldots, N-i+1)$-permutations.*

Proof: We generate L_i from L by removing edges labeled i or larger, then apply Corollary 1 repeatedly•

Even though L_i is a refinement of L, L_i still contains many edges that the `min_weight_forest` algorithm does not examine: the edges in E_F. We have not kept track of the edge destinations, so

we have no information yet about which edges are in E_F and which are in E_{cross}. Fortunately, the information necessary for an expected case analysis is implicitly stored in L_i.

To extract this information, we look at the first entry in L_i, e_1. If this edge is labeled $k+i-1$ (that is, it leads from the most recently added node), then it has never been considered before. Therefore, e_1 must be in E_{cross}, so the `min_weight_forest` algorithm will select it as the i^{th} edge in the forest. If the label of the edge, $l(e_1)$, is less than $k+i-1$, then e_1 was also the first edge in L_{i-1} since edges with labels greater than or equal to $k+i-1$ are removed from L to create L_{i-1}. Thus, at some previous selection j, $l(e_1) < j < i$, edge e_1 was the lowest weight edge in e_{cross}, so the `min_weight_forest` algorithm chose it as the j^{th} edge to add to the forest. Therefore, $e_1 \in E_F$ (in fact, e_1 is a tree edge), so e_1 is ineligible.

If the first edge on L_i is in E_F, we need to continue scanning L_i for the first eligible edge. The discovery that e_1 is not an eligible edge tells us that some other edges in L_i are ineligible also: the edges that lead to the same destination as e_1 does. We are not keeping any destination information in L, but we assume for now that we have some way of determining destinations (we address this assumption later). In the analysis that follows, we will want to scan over tree edges only, so we remove edges from $L_i = L_{i,1}$ that have the same destination as e_1 to form $L_{i,2}$.

Suppose that e_2 is the first edge in $L_{i,2}$ (i.e, the second eligible or tree edge in L_i). If e_2 is labeled $k+i-1$, the `min_weight_forest` algorithm has never examined e_2 before, so e_2 is chosen by the algorithm. Suppose that e_2 is labeled $k+i-2$. If e_1 is labeled $k+i-2$ then e_1 must have been chosen when the $i+k-1$-st edge was added to the forest. An edge labeled $i+k-2$ can be selected as the next forest edge only when nodes labeled $i+k-1$ or higher are added to the forest. Since we know that e_1 added the node labeled $i+k-1$ to the forest, e_2 must add the node labeled $i+k$, and so the `min_weight_forest` algorithm chooses edge e_2 on the i^{th} edge selection step. If e_2 is labeled $k+i-3$ or lower, then e_2 was chosen as a forest edge on a previous iteration regardless of the label of e_1. In general,

Theorem 2 *Let e_j be the first edge on $L_{i,j}$, and suppose that we have not found in $\{e_1, \ldots, e_{j-1}\}$ the edge that adds the node labeled $k+i$ to the forest.*

If $l(e_j) = k+i-s$ and $s < i$, then e_j adds the node labeled $k+i$ to the forest if and only if we have already found in $\{e_1, \ldots, e_{j-1}\}$ the edges that added the nodes labeled $k+i-s+1$ through $k+i-1$ to the forest.

If $s \geq i$, then e_j adds the node labeled $k+i$ to the forest if and only if we have already found in

$\{e_1, \ldots, e_{j-1}\}$ *the edges that added the nodes labeled* $k+1$ *through* $k+i-1$ *to the forest.*

Proof: Assume that $s < i$. Suppose that we have found the edges that added the nodes labeled $k+i-s+1$ through $k+i-1$ to the forest. Since edge e_j can only add nodes labeled $k+i-s+1$ and higher to the forest and all of the previous possibilities have been covered already, e_j can not be a tree edge. Since we throw away nontree edges in E_F, e_j must be a cross edge. Since it is the first cross edge we have found, it must have the lowest weight among all cross edges. Therefore, the tree growing algorithm picks it to add the node labeled $k+i$ to the forest.

Suppose that we have not yet found all of the edges that add the nodes labeled $k+i-s+1$ through $k+i-1$ to the forest. Let the lowest labeled node that is unaccounted for be labeled $k+r$.

Claim: Edge e_j added the node labeled $k+r$ to the forest.

Proof: Consider what occurs when the node labeled $k+r$ is added to the the forest. We can retrace the forest generating algorithm's decision by examining the list L_r. Edge e_j in L_i was considered as some edge $e_{j'}$ in L_r, because we had not accounted for the edge that added node $k+r$ in L_i before we came to e_j. When edge $e_{j'}$ in L_r is considered, all nodes with labels between $k+i-s+1$ and $k+r-1$ are accounted for. As we have shown, $e_{j'}$ must be the lowest weight cross edge, so the forest growing algorithm picks it to add the node labeled $k+r$ to the forest•

Since e_j adds the node labeled $k+r$ to the forest, it can not add node $k+i$ to the forest.

If $s \geq i$, we only need to account for the fact that any edge labeled 1 through k can add the node labeled $k+1$ to the forest•

Suppose that we have an algorithm for determining which node a tree edge adds to the forest. Then the algorithm for using L to calculate which node received the i^{th} edge is:

```
procedure retrace(L)
   for i=1 to to N do
      L_i is L with edges labeled k+i or greater removed.
      j=1.
      L_{i,j}=L_i.
      while the edge that adds node k+i to the forest is not found
         e_j=the first edge in L_{i,j}.
```

```
if all nodes between k + l(e_j) + 1 and k + i − 1 are accounted for
   output(i,l(e_j)).
else
   account for the node that e_j adds.
   L_{i,j+1}=L_{i,j} with edges that lead to the same destination as e_j removed.
   increment j.
```

Theorem 2 gives us an algorithm for determining which node a tree edge adds to the forest. When edge e_j is considered, the algorithm looks to see if node $k + l(e_j) + 1$ is accounted for. If the node is accounted for, the algorithm looks at node $k + l(e_j) + 2$, and so on until an unaccounted for node is found. Edge e_j adds that node (possibly node $k + i$), so the algorithm accounts for it.

We use a 'marked boxes' technique to account for the nodes. When sublist L_i is examined, the algorithm initializes i boxes, numbered 1 through i. If the edge e_j is labeled $k + i - s$, the algorithm looks for an unmarked box starting at box s and working towards box 1. The algorithm marks the first unmarked box that is found. If box 1 is marked, the algorithm concludes that edge e_j adds node $k + i$.

```
procedure markbox(L_i)
   initialize box[1..i] as unmarked.
   j=1.
   while box[1] is unmarked
      get e_j.
      s=k+i-l(e_j).
      if s>i then s=i.
      while box[s] is marked
         s=s-1.
      mark box[s].
      j=j+1.
   return(j-1).
```

Executions of `retrace` and `markbox` procedures are illustrated in Table 1. The sorted list of edges L is the same as that in Figure 6. The list $L_{1,1}$ is the list of edge choices when node 3 is selected (all edges

	a	i	f	e	g	b	d	h	c	1	2	3
$L =$	1	3	2	2	3	1	2	4	1			
$L_{1,1} =$	1		2	2		1	2		1	X		
$L_{2,1} =$	1	3	2	2	3	1	2		1		X	
$L_{2,2} =$		3	2	2	3	1			1	X	X	
$L_{3,1} =$	1	3	2	2	3	1	2	4	1			X
$L_{3,2} =$		3	2	2	3	1		4	1		X	X
$L_{3,3} =$				2	3	1		4		X	X	X

Table 1: The sorted list of edges, and finding the selected edge.

labeled 3 and larger are deleted). The first edge on the list is labeled $1 = 3 - 2$, so the algorithm marks box 1 and selects that edge (since $2 > 1$). The list $L_{2,1}$ is the initial list of edges when node 4 is selected. The first edge on $L_{2,1}$ is labeled $1 = 4 - 3$. Since $3 > 2$, the algorithm marks box 2. It generates $L_{2,2}$ by removing the initial edge on $L_{2,1}$ and all edges that lead to the same node. The first edge on $L_{2,2}$ is labeled $3 = 4 - 1$, so the algorithm marks box 1 and selects that edge. Finally, list $L_{3,1}$ has all edges on it, for there are no edges in E_{V-F}. The first edge on $L_{3,1}$ is labeled $1 = 5 - 4$, so the algorithm marks box 3. The first edge on $L_{3,2}$ is labeled $3 = 5 - 2$, so the algorithm marks box 2. The first edge on $L_{3,3}$ is labeled $2 = 5 - 3$, so it tries to mark box 3. Box 3 is already marked (since a different edge chose the third node), so it tries to mark box 2. Box 2 is also marked, so it marks box 1, and select that edge. Note that this procedure selects the same edges (a, i, e) that the simulation of Dijkstra's algorithm does.

Before we continue, we need to determine the underlying distributions of the sublists. Lemma 1 and Corollary 2 show that L and L_i are uniformly randomly chosen permutations. The algorithm to recreate the `min_weight_forest` algorithm's choices from L requires the use of the $L_{i,j}$ sublists. The $L_{i,j}$ sublists are created by determining edge destinations and removing some edges. We do not explicitly keep edge destination information in L, but we show that this is not a problem for the analysis.

Corollary 3 *The lists $L_{i,j}$ are uniformly randomly distributed permutations.*

Proof: $L_{i,1} = L_i$, so $L_{i,1}$ is a uniformly randomly chosen permutation. Since $L_{i,1}$ is a uniformly randomly chosen permutation, we see that $L_{i,2}$ is a uniformly randomly chosen permutation by repeated applications of Corollary 1. Proceeding inductively, we see that every $L_{i,j}$ is a uniformly randomly chosen permutation•

3.1 Relation to Uniform Growth Model

The algorithm for determining the `min_weight_forest` algorithm's choices from L shows that edges with different labels have different probabilities of being chosen to add the i-th node. An edge labeled $k+i-1$ can be selected from every $L_{i,j}$, while an edge labeled 1 through k can only be selected from $L_{i,i}$. We will return to the issue of the edge label of the edge that is selected, but here we want to determine the relation of the minimum weight model to the uniform growth model.

We observe that if the edge that adds the node labeled $k+i$ to the forest is not found in $L_{i,1}$ through $L_{i,i-1}$, then the edge will be found in $L_{i,i}$ since every edge in $L_{i,i}$ is a cross edge. Every forest node is incident to the same number of cross edges $(N-i+1)$, so by lemma 3, every forest node is equally likely to receive the next edge. Thus, the minimum weight model is equivalent to the uniform growth model on the occasions when $L_{i,i}$ must be searched to find the minimum weight cross edge. The following theorem states the probability of this event.

Theorem 3 *Suppose that there are k trees, and the* `min_weight_forest` *algorithm is adding the node labeled $k+i$ to the forest. If $i \ll N$, then the probability that the first edge in $L_{i,j}$, e_j, is the minimum weight cross edge is approximately $1/(k+i-1)$ if $j<i$ and $k/(k+i-1)$ if $j=i$.*

Proof: In this proof, we use the marked boxes mechanism described in the previous section. The input to the marked boxes algorithm is the sequence $(l(e_1), l(e_2), \ldots, l(e_n))$. We transform this sequence into the more convenient sequence $\{(x_1, x_2, \ldots, x_n)\}$, where each $x_r \in \{1, \ldots, i\}$ and $1 \leq n \leq i$, by the following function:

$$x_j = \begin{cases} k+i-l(e_j) & \text{if } l(e_j) > k \\ i & \text{if } l(e_j \leq k \end{cases}$$

If $x_j = r$, then x_j represents an edge labeled $(k+i-r)$ and refers to box r in the `markbox` procedure, if $r < i$. If $x_j = i$, then x_j represents to an edge labeled 1 through k and refers to box i.

In L_i, there are N edges labeled 1 through k, and $N-s$ edges labeled $k+s$ for $s = 1, \ldots, i-1$. Since $N \gg i$, there are about the same number of edges labeled $k+i-1$ in L_i as there are edges labeled 1. Therefore, we will assume that each x_j in a sequence is independent, and has value r with probability $1/(i+k-1)$ if $r < i$, and value i with probability $k/(k+i-1)$. We want to determine the probability that the length of the minimum length sequence which marks box 1 is n.

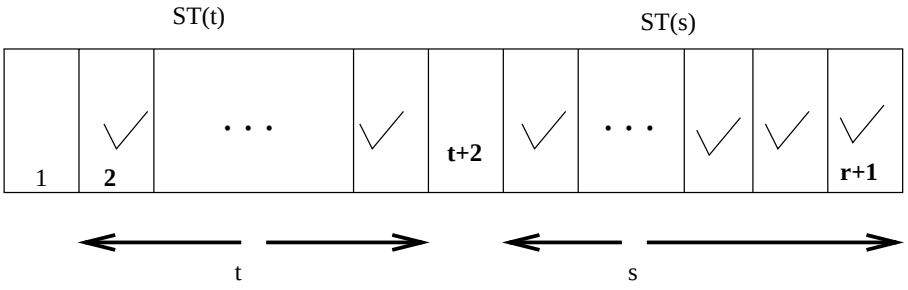


Figure 7: Last entry in st_r

We define a *safe trigger of length* r to be a sequence st_r which is of length r, does not mark box 1, but (st_r j) marks 1 if $1 \leq j \leq r+1$. For example, (2 3) is a safe trigger of length 2. We define $st(r)$ to be the set of all safe triggers of length r, and $ST(r) = |st(r)|$. For example, $st(3) = \{(2\ 3), (3\ 2), (3\ 3)\}$, and $ST(3) = 3$. We define $ST(0) = 1$.

Lemma 4 $ST(r) = (r+1)^{r-1}$

Proof: We first derive a recurrence for $ST(r)$, then solve the recurrence. For the starting values of the recurrence, we observe that $st(0) = \{\emptyset\}$ and $st(1) = \{(2)\}$, so $ST(0) = ST(1) = 1$.

Consider the last entry added to a sequence in $st(r)$. The last entry marks a box, so we assume that it marks the box numbered $t+2$ (see Figure 7). To the right of the last marked box is a sequence in $st(t)$ and to the left is a sequence that corresponds to a sequence in $st(s)$ by translation. There are $\binom{s+t}{s} = \binom{r-1}{s}$ ways to combine an s length sequence with a t length sequence. The last entry in the safe trigger of length r can be any number between $t+2$ and $r+1$ inclusive, so there are $s+1$ possibilities. Therefore:

$$ST(r) = \sum_{s=0}^{r-1} (s+1) \binom{r-1}{s} ST(s) ST(r-s-1) \tag{1}$$

The form of the recurrence is similar to the convolution of an exponential generating function, so we let $Y = Y(z) = \sum ST(r) z^r / r!$ be the exponential generating function of $ST(r)$. Then Y satisfies the differential equation

$$(1 - Yz) Y' = Y^2 \tag{2}$$

To solve this equation, we look for a solution of the form:

$$Y = e^{\alpha(Y) z}$$

Then

$$\begin{aligned} Y' = &\quad Y(Y'\alpha'(Y)z + \alpha(Y)) \\ Y'(1 - Y\alpha'(Y)z) = &\quad Y\alpha(Y) \end{aligned}$$

Setting $\alpha(Y) = Y$ satisfies the differential equation. The generating function of $Y = e^{Yz}$ is (see [7], page 392)

$$Y(z) = \sum_{r \geq 0} \frac{(r+1)^{r-1}}{r!} z^r$$

so that

$$ST(r) = \sum_{s=0}^{r-1} (s+1) \binom{r-1}{s} ST(s)ST(r-s-1) = (r+1)^{r-1} \bullet$$

Let us count the number of ways that edge e_j marks box 1 (i.e, adds the node labeled $k+i$ to the forest). Suppose after processing x_{j-1}, box 1 is unmarked, and the t subsequent boxes are marked. If we examine which boxes are marked after $j-1$ steps, box 1 is unmarked, boxes 2 through $t+1$ are marked, and box $t+2$ is unmarked. Of the remaining $i-t-2$ boxes, numbered $t+3$ through i, exactly $j-t-1$ are marked (see Figure 8). In order to make the analysis simpler, we aggregate all of the x_l values that can not mark box 1 on the j^{th} step (those values between $j+1$ and i) and renumber them by $j+1$. That is,

$$x'_l = \begin{cases} x_l & \text{if } x_l \leq j \\ j+1 & \text{if } x_l > j \end{cases}$$

In the transformed sequence, all the boxes are marked except for boxes 1 and $t+2$. When x'_j is chosen to augment this sequence, there are $i+k-j-1$ ways to choose it to be $j+1$, and 1 way to choose it to be n, $n = 1, \ldots, j$. With this configuration, any x'_j numbered between 1 and $t+1$ will mark box 1. Let $R(s)$ be the number of ways to mark the last $s = j-t-1$ boxes. The number of ways that an edge will be selected on the j^{th} examination step is therefore

$$Stop(i) = \sum_{t=0}^{j-1} (t+1) \binom{j-1}{t} ST(t)R(j-1-t) \tag{3}$$

In order to solve equation (3), we need to determine the value of $R(s)$. We can quickly derive a recursive formula that is similar to equation (1). In Figure 7, the number of sequences that can fill the right hand side is $R(s)$ instead of $ST(s)$. The sequence will be completed if $x'_j \in t+2, .., r+1$. The x'_j

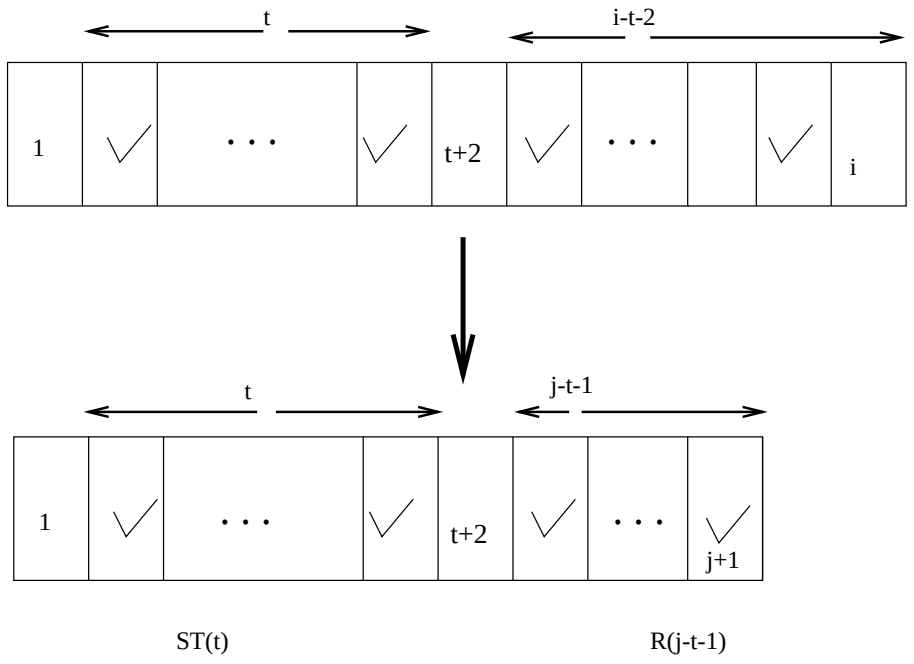


Figure 8: Transformation for i^{th} selection

can be chosen to have the value r in $X = i + k - j - 1$ ways, All other numbers can only be chosen one way. Thus, the number of ways to complete the sequence is

$$\begin{aligned} R(r) = &\quad \textstyle\sum_s (X+s)\binom{r-1}{s} ST(r-1-s)R(s) \\ = &\quad \textstyle\sum_s (X+s)\binom{r-1}{s}(r-s)^{r-s-2}R(s) \end{aligned}$$

Lemma 5 $R(r) = X(X+r)^{r-1}$

We prove the lemma by induction. It is easy to see that $R(0) = 1$ and $R(1) = X$. For the inductive step, we need to show that

$$X(X+r)^{r-1} = \textstyle\sum_s (X+s)\binom{r-1}{s}(r-s)^{r-s-2}X(X+s)^{s-1}$$

or that

$$(X+r)^{r-1} = \textstyle\sum_s \binom{r-1}{s}(r-s)^{r-s-2}(X+s)^s$$

Let

$$T(r) = R(r+1)/X = \sum_s \binom{r}{s}(r+1-s)^{r-s-1}(X+s)^s$$

Then, from Riordan [11], page 18, we see that $T(r)$ is an instance of the generalization of Abel's binomial formula, $T(r) = A_r(X, 1, 0, -1) = (X + 1 + r)^r$, which is exactly what we need to show •

Lemma 6 *Stop*$(j) = (X + j)^{j-1} = (i + k - 1)^{j-1}$

To prove this lemma, we apply lemma 5 to equation (3):

$$\begin{aligned} \mathit{Stop}(j+1) = & \quad \textstyle\sum_s (j+1-s)\binom{j}{s} ST(j-s)R(s) \\ = & \quad X \textstyle\sum_s \binom{j}{s} (j+1-s)^{j-s}(X+s)^{s-1} \\ = & \quad X A_j(X, 1, -1, 0) \\ = & \quad X \frac{(X+j+1)^j}{X} \\ = & \quad (X+j+1)^j \end{aligned}$$

where the third step is another application of the generalization of Abel's binomial formula•

To prove the theorem, we note that on the j^{th} examination step there are $(i + k - 1)^j$ possible sequences. The `min_weight_forest` algorithm chooses an edge on the j^{th} examination step on exactly $(i + k - 1)^{j-1}$ of these sequences, so the probability that a selection is made on the j^{th} examination step is $1/(i + k - 1)$ if $j < i$. A selection will be made by the i^{th} step, so the probability of selecting an edge on the i^{th} step is the remaining probability, proving the theorem•

As a corollary to theorem 3, we find the relationship between the uniform growth model and the minimum weight model.

Corollary 4 *Let k be the number of trees in the forests, and N the number of nodes to be added to the forest. Consider the act of adding the i^{th} edge to the minimum weight forest. If $i \ll N$, then*

$$\Pr[\text{every node in the forest is equally likely to receive the } i^{\text{th}} \text{ edge}] = k/(i + k - 1)$$

Proof: Every forest node is equally represented in $L_{i,i}$ •

3.1.1 A Limiting Distribution

The results we used to prove Corollary 4 let us calculate for the minimum weight model the probability that a forest node receives the next edge. In Table 2, we list the possibilities by which an edge at the head of $L_{i,j}$ can be chosen.

examination step	Size of safe trigger 0	1	2	...
1	$ST(0)\binom{0}{0}R^1(0)/Z$ $= 1/Z$			
2	$ST(0)\binom{1}{0}R^2(1)/Z^2$ $= (Z-2)/Z^2$	$ST(1)\binom{1}{1}R^2(0)/Z^2$ $= 1/Z^2$		
3	$ST(0)\binom{2}{0}R^3(2)/Z^3$ $= (Z-3)(Z-1)/Z^3$	$ST(1)\binom{2}{1}R^3(1)/Z^3$ $= 2(Z-3)/Z^3$	$ST(2)\binom{2}{2}R^3(0)/Z^3$ $= 3/Z^3$	
4	$ST(0)\binom{3}{0}R^4(3)/Z^4$ $= (Z-4)(Z-1)^2/Z^4$	$ST(1)\binom{3}{1}R^4(2)/Z^4$ $= 3(Z-4)(Z-2)/Z^4$	$ST(2)\binom{3}{2}R^4(1)/Z^4$ $= 9(Z-4)/Z^4$	...
5	$ST(0)\binom{4}{0}R^5(4)/Z^5$ $= (Z-5)(Z-1)^3/Z^5$	$ST(1)\binom{4}{1}R^5(3)/Z^5$ $= 4(Z-5)(Z-2)^2/Z^5$	$ST(2)\binom{4}{2}R^5(2)/Z^5$ $= 18(Z-5)(Z-3)/Z^5$	...
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Table 2: Probability that an edge is selected, given an examination step and a safe trigger size

In Table 2, the number of nodes in the forest is $Z = k + i - 1$. The number of identifiable elements grows as the list is searched further. On the j^{th} selection step, only edges labeled $(k+i)-1$ through $(k+i)-j$ can be cross edges. Since the number of unselectable edges changes with the selection step, we must modify our definition of $R(n)$, which represents the edges that will not help cause a selection on step j. We replace X in $R(n)$ by $Z-j$, and write the new function as $R^j(n) = (Z-j)(Z-j+n)^{n-1}$.

In the previous section, we proved Theorem 3 by weighting the entries in a column by the number of edge labels that can cause a selection and summing across the rows of the table. In order to calculate the probability distributions that a node with a given label receives the edge, we sum down the columns. Each entry in the table represents a different possibility by which an edge can be chosen as the next edge added to the forest. The s^{th} column is the probability that an s-safe trigger causes the selection. The r^{th} most recently added node will receive the next edge only through a safe trigger of size r or larger. So, the s^{th} column sum is the difference between the probability that the s^{th} and the $s+1^{\text{st}}$ most recently added node receives the next edge. Let us define $A_i(j)$ to be the probability that the forest node labeled $k+i-j$ receives the i^{th} edge added to the forest.

Theorem 4 *If the i^{th} edge is being added to the forest and i is large compared to the number of forests but small compared the the number of nodes in the graph, then the probability that a node receives an edge approaches the following limiting distribution:*

$$A(j) = \sum_{t=j}^{\infty} \frac{t^{t-2}}{t!} e^{-t} \tag{4}$$

Proof: Let $C(t)$ be the sum of the t^{th} column of Table 2. Then, ignoring the possibility that $L_{i,i}$ must be examined in order to make a selection,

$$A_i(j) = \sum_{t=j}^{i} C(t)$$

Lemma 7

$$C(t+1) \approx \frac{(t+1)^{t-1}}{(t+1)!} e^{-(t+1)}$$

It is easily seen from the preceding arguments that column sum has the following value:

$$\begin{aligned} C(t+1) &= \frac{ST(t)}{Z^{t+1}} \sum_{j=0}^{Z-t} \binom{j+t}{t} R^{j+t+1}(j)/Z^j \\ &= \frac{ST(t)}{Z^{t+1}t!} \sum_{j=0}^{Z-t} (j+t)^{(t)}(Z-j-(t+1))(Z-j-(t+1)+j)^{j-1}/Z^j \\ &= \frac{ST(t)}{Z^{t+1}t!} \sum_{j=0}^{Z-t} \left[(j+t)^{(t)} - \frac{(j+t)^{(t+1)}}{Z-(t+1)}\right] \left(\frac{Z-(t+1)}{Z}\right)^j \end{aligned}$$

Here, $x^{(n)} = x(x-1)(x-1)\ldots,(x-n+1)$ is the falling factorial function. In order to solve this sum, we apply Theorem 1.6 from Micken (page 36) [10], which states

$$\sum \lambda^j P(j) = \frac{\lambda^j}{\lambda - 1} \left(1 + \frac{\lambda}{1-\lambda}\Delta + \frac{\lambda^2}{(1-\lambda)^2}\Delta^2 + \cdots\right) P(j)$$

where $\sum$ is indefinite summation and Δ is the forward difference.

In our equation, $\lambda = (Z-(t+1))/Z$, so that $1/(\lambda-1) = -Z/(t+1)$ and $\lambda/(1-\lambda) = ((Z-(t+1))/(t+1)$. For $C(t+1)$, $P(j) = P^t(j) = (j+t)^{(t)} - (j+t)^{(t+1)}/(Z-(t+1))$. Therefore,

$$\Delta^r P^t(j) = \begin{cases} t^{(r)}(j+t)^{(t-r)} - \frac{(t+1)^{(r)}(j+t)^{(t+1-r)}}{Z-(t+1)} & r \le t \\ \frac{(t+1)!}{Z-(t+1)} & r = t+1 \end{cases}$$

Putting these into our formula for $C(t+1)$, we get:

$$\begin{aligned} C(t+1) = \; & \frac{ST(t)}{Z^{t+1}t!} \frac{-Z}{t+1} \left(\frac{Z-(t+1)}{Z}\right)^j \left(\sum_{r=0}^{t} \left(\frac{Z-(t+1)}{t+1}\right)^r \left[t^{(r)}(j+t)^{(t-r)} - \frac{(t+1)^{(r)}(j+t)^{(t+1-r)}}{m-(t+1)}\right]\right. \\ & \left. - \left(\frac{Z-(t+1)}{t+1}\right)^{t+1} \frac{(t+1)!}{Z-(t+1)}\right)\Bigg|_{j=0}^{j=Z-t+1} \\ = \; & \frac{ST(t)}{Z^{t+1}t!} \frac{-Z}{t+1} \left(\frac{Z-(t+1)}{Z}\right)^j \left(\sum_{r=0}^{t} \left(\frac{Z-(t+1)}{t+1}\right)^r t^{(r)}(j+t)^{(t-r)} - \right. \\ & \left. \sum_{r=1}^{t+1} \left(\frac{Z-(t+1)}{t+1}\right)^r \frac{(t+1)^{(r)}(j+t)^{(t+1-r)}}{m-(t+1)} - \frac{(j+t)^{(t+1)}}{Z-(t+1)}\right)\Bigg|_{j=0}^{j=Z-t+1} \\ = \; & \frac{ST(t)}{Z^{t+1}t!} \frac{-Z}{t+1} \left(\frac{Z-(t+1)}{Z}\right)^j \left(\sum_{r=0}^{t} \left(\frac{Z-(t+1)}{t+1}\right)^r t^{(r)}(j+t)^{(t-r)} - \right. \\ & \left. \sum_{r=0}^{t} \left(\frac{Z-(t+1)}{t+1}\right)^r (t)^{(r)}(j+t)^{(t-r)} - \frac{(j+t)^{(t+1)}}{Z-(t+1)}\right)\Bigg|_{j=0}^{j=Z-t+1} \end{aligned}$$

$$= \frac{ST(t)}{Z^{t+1}t!}\frac{Z}{t+1}\left(\frac{Z-(t+1)}{Z}\right)^{j}\frac{(j+t)^{(t+1)}}{Z-(t+1)}\Bigg|_{j=0}^{j=Z-t+1}$$

$$= \frac{ST(t)}{Z^{t+1}t!}\frac{Z}{t+1}\left(\frac{Z-(t+1)}{Z}\right)^{Z-t+1}\frac{(Z-1)^{(t+1)}}{Z-(t+1)}$$

If i is large then Z is large, so that $((1-(t+1)/Z)^{Z-t+1} \approx e^{-(t+1)}$ and

$$C(t+1) \approx \frac{(t+1)^{t-1}}{(t+1)!}e^{-(t+1)}\bullet$$

To finish the proof of the theorem, we note that the the value of $C(t)$ rapidly decreases to zero, so that

$$A(j) \approx \sum_{t=j}^{\infty}\frac{(t)^{t-2}}{t!}e^{-t}\bullet$$

For large j, we can find an approximate value of the sum. If we use Stirling's approximation for the factorial, we find that

$$A(j) \approx \sum_{t=j}^{\infty}\frac{(t)^{t-2}}{\sqrt{2\pi}t^{t+1/2}e^{-t}}e^{-t}$$

$$\approx \sum_{t=j}^{\infty}\frac{1}{\sqrt{2\pi}t^{5/2}}$$

We can approximate the sum with an integral using Euler's summation formula [6] to get:

$$A(j) \approx \frac{2}{3\sqrt{2\pi}}\left[j^{-3/2}-\frac{3}{4}j^{-5/2}\right]\left(1+O(j^{-2})\right) \tag{5}$$

Table 3 lists the first few values of the distribution $A(j)$. We summed the first 400 values of $C(t)$ and then added the approximation of $A_i(401)$ to calculate $A_i(1)$. We calculated the remaining entries in Table 3 by subtracting the appropriate values of $C(t)$. Note that $A_i(j)$ depends on j but not i, so the probability that a forest node receives the next edge depends primarily on how recently it was added to the tree. The analysis shows that the most recently added forest node receives the next edge 50% of the time, and the ten most recently added forest nodes receive the next edge 83% of the time.

The analysis that leads to the distribution A depends on the assumption that $k \ll i \ll N$. In order to determine how strongly the analysis depends on the assumptions, we wrote a simulator that generated minimum weight random forests using the `min_weight_forest` algorithm. We generated a minimum weight random forest using the parameters $k = 5$ and $N = 94$ 10,000 times. For each edge selection step i, we recorded the number of times that an edge labeled $k+i-s$ was selected. We used this information to estimate $\hat{A}_i(j)$, which we list in table 3 for $i = 10, 20, 40,$ and 60. For small i ($i = 10$), there is still

j	1	2	3	4	5	6	7	8	9	10
$A(j)$	.5000	.1321	.0645	.0396	.0273	.0203	.0159	.0128	.0106	.0090
$\hat{A}_{10}(j)$	.4808	.1227	.0642	.0428	.0373	.0320	.0285	.0 273	.0271	
$\hat{A}_{20}(j)$	.4811	.1209	.0627	.0442	.0318	.0254	.0233	.0172	.0157	.0142
$\hat{A}_{40}(j)$	.4527	.1122	.0597	.0302	.0234	.0198	.0160	.0126	.0140	.0080
$\hat{A}_{60}(j)$	.3993	.0914	.0425	.0252	.0142	.0142	.0122	.0114	.0089	.0082

Table 3: Probability that the j^{th} most recently added forest node receives the next edge.

a large chance (5/14) that every node in the forest will receive the next edge, so the tail of A_{10} is not well approximated by A. For large i ($i = 60$), there are significantly fewer edges labeled $k + i - 1$ than labeled 1 through k. As a result, $A_i(1)$ becomes smaller than $A(1)$. For moderate i ($i = 20$ and $i = 40$), A is a good approximation to A_i.

4 Conclusions

We have examined two models of naturally growing forests, one in which every node in the forest is equally likely to be be adjacent to the next node added to the forest (uniform growth model), and a model in which the forest is constructed in order to create a minimum weight forest (minimum weight model).

We found that the uniform growth model is analytically tractable. We calculated the distribution of tree sizes, and found that the tree that receives the last node will contain about $2N/k$ nodes, where k is the number of trees in the forest and N nodes are added to the forest.

We found the minimum weight model to be far more difficult. We examined the relationship between the uniform growth model and the minimum weight model and found that the uniform growth model holds in the minimum weight model with probability $k/(k+i-1)$ when adding the i^{th} node to the forest. The probability that a forest node receives the next edge added to the forest depends primarily on how recently that node was added to the forest. The most recently added node receives the next edge about 50% of the time, and one of the ten most recently added nodes receives the next edge about 82% of the time.

5 Acknowledgements

We'd like to thank Kevin Donovan and Yogin Campbell for their help and advice on solving the problems in this paper.

References

[1] B. Bollobas. *Random Graphs.* Academic Press, 1985.

[2] J.F. Desler and S.L. Hakami. A graph-theoretic approach to a class of integer-programming problems. *Operations Research*, 17:1017–1033, 1969.

[3] A.M. Frieze. On the value of a random minimum spanning tree problem. *Discrete Applied Marthematics*, 10:47–56, 1985.

[4] M. Held and R.M. Karp. The traveling salesman problem and minimum spanning trees. *Operations Research*, 18:1138–1162, 1970.

[5] M. Held and R.M. Karp. The traveling salesman problem and minimum spanning trees. *Mathematical Programming*, 1:6–25, 1971.

[6] Micha Hofri. *Probabilistic Analysis of Algorithms.* Springer-Verlag, 1987.

[7] D. Knuth. *The Art of Computer Programming*, volume 1. Addison-Wesley, 1968.

[8] H.M. Mahmoud. *Evolution of Random Search Trees.* Wiley-Interscience, 1992.

[9] Udi Manber. *Introduction to Algorithms.* Addison Wesley, 1989.

[10] R. Mickens. *Difference Equations.* Van Nostrand Reinhold, New York, 1987.

[11] J. Riordan. *Combinatorial Identities.* Robert E. Kreiger, Huntington, NY, 1979.

[12] J.M. Steele. On Frieze's $\varsigma(3)$ limit of lengths of minimal spanning trees. *Discrete Applied Mathematics*, 18:99–103, 1987.

[13] H.S. Stone. An algorithm for finding a minimum weighted 2-matching. Technical Report RC 15014, IBM T.J. Watson Research Center, 189.