Journal of Combinatorics
Volume 3, Number 2, 207–223, 2012

Trees with the most subtrees – an algorithmic approach

Xiu-Mei Zhang*, Xiao-Dong Zhang*, Daniel Gray and Hua Wang†

When considering the number of subtrees of trees, the extremal structures which maximize this number among binary trees and trees with a given maximum degree lead to some interesting facts that correlate to other graphical indices in applications. The number of subtrees in the extremal cases constitute sequences which are of interest to number theorists. The structures which maximize or minimize the number of subtrees among general trees, binary trees and trees with a given maximum degree have been identified previously. Most recently, results of this nature are generalized to trees with a given degree sequence. In this note, we characterize the trees which maximize the number of subtrees among trees of a given order and degree sequence. Instead of using theoretical arguments, we take an algorithmic approach that explicitly describes the process of achieving an extremal tree from any random tree. The result also leads to some interesting questions and provides insight on finding the trees close to extremal and their numbers of subtrees.

2000 Mathematics Subject Classification: 05C05, 05C07, 05C35, 05C85.
Keywords and phrases: Tree, subtrees, extremal.

1. Introduction and terminology

For a *tree* $T = (V, E)$ with vertex set $V(T)$ and edge set $E(T)$, $d_T(u)$ denotes the *degree* of vertex u. $P_T(u, v)$ and $d_T(u, v)$ denote the *path* connecting two vertices $u, v \in V(T)$ and the *distance* between them. The degree sequence of a graph is the multiset containing the degrees of all non-leaf vertices in descending order.

*Supported by NNSFC (#10971137 and 11271256, to X.-M. Zhang and X.-D. Zhang).
†This work was partially supported by a grant from the Simons Foundation (#245307 to Hua Wang).

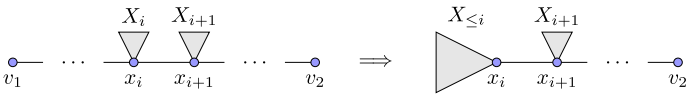

Figure 1: $X_{\leq i}$ in T.

We denote a tree T *rooted* at $r \in V(T)$ as (T, r). Let $h_T(u) = d_T(r, u)$ and $h(T) = \max_{u \in V(T)} h_T(u)$ be the *height* of u and T, respectively. Also let $f(T)$ denote the number of subtrees of T and $f_u^T(X)$ denote the number of subtrees containing u in a subgraph X of T. If $h_T(u) < h_T(v)$ then we say u is an *ancestor* of v or v is a *descendant* of u. If $h_T(u) = h_T(v) - 1$ for two vertices u and v, then we say u is the *parent* of v and v is the *child* of u. When two vertices share the same parent we call them *siblings*. We sometimes omit the subscript or superscript if there is no ambiguity.

Suppose that $v_1x_1x_2x_3 \dots x_nv_2$ is a path in T with v_1 and v_2 being leaves. After the removal of all edges on the path there still remain connected components, each containing one of $x_1, x_2, \dots, x_n$. Label these components as $X_1, X_2, \dots, X_n$, respectively. Also, let $X_{\leq i}$ ($X_{\geq j}$) be the component containing x_i (x_j) in $T - x_ix_{i+1}$ ($T - x_{j-1}x_j$). Fig. 1 shows an example of such labellings.

The subtrees of trees have been studied in [5] and some general properties were provided. A nice coincidence was found between the binary trees which maximize the number of subtrees and the binary trees which minimize the *Wiener index* ([7], defined as the sum of all pairwise distances between vertices), a chemical index widely used in biochemistry. In the same paper, formulas are given to calculate the number of subtrees of these extremal binary trees. The sequence of the number of subtrees of these extremal binary trees are found to be novel [4]. These formulas use a new representation of integers as a sum of powers of 2. Number theorists have already started investigating this new binary representation [2]. The results were extended to trees with a given maximum degree [3] and the extremal structures once again coincide with the ones found for some other topological indices. Yan and Yeh [8] presented an algorithm for counting the number of subtrees of a tree. The correlations of different graphical indices that share the same extremal structures, including the number of subtrees, have also been considered [6]. Moreover, there is a relation between the greedoid Tutte polynomial of a tree and the number of subtrees [1]. We will examine trees with a given *degree sequence*.

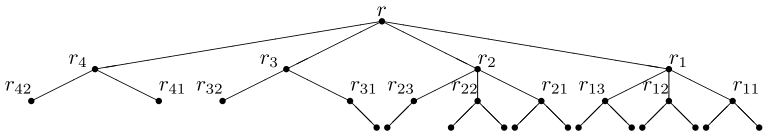

Figure 2: A greedy tree.

Definition 1.1. Let M be a multiset whose elements are in descending order. The greedy tree with degree sequence M can be achieved through the following 'greedy algorithm':

i) Label the vertex with the largest degree as r (the root);
ii) Label the children of r as $r_1, r_2, \ldots$, assign the largest degrees available to them such that $d(r_1) \geq d(r_2) \geq \cdots$;
iii) Label the children of r_1 as $r_{11}, r_{12}, \ldots$ such that $d(r_{11}) \geq d(r_{12}) \geq \cdots$, then do the same for $r_2, r_3, \ldots$, respectively;
iv) Repeat (iii) for all the newly labeled vertices, always start with the children of the labeled vertex with largest degree whose neighbors are not labeled yet.

Fig. 2 shows a greedy tree with degree sequence $\{4, 4, 4, 3, 3, 3, 3, 3, 3, 3, 2, 2\}$.

The greedy tree is shown to maximize the number of subtrees [9]. It is nice to note that the greedy tree once again coincides with the extremal structures of other graphical indices. In this note, we will employ an algorithmic approach that defines the exact operations needed to achieve the extremal tree from any tree and vice versa. With this approach, it is possible to identify trees that are 'close' to extremal and study their numbers of subtrees. In Section 2 we introduce several 'switching operations' and algorithms that increase the number of subtrees in every step. In Section 3 we prove the main result and discuss the second or kth extremal tree in general. A summary and some questions/conjectures are posted in Section 4.

2. Algorithms on switching components

For the following definitions, we consider a path $v_1x_1x_2 \ldots x_nv_2$ labeled as in Fig. 1.

Definition 2.1. A 'component-switch', denoted by $S^T_{v_1,v_2}(X_i, X_j)$ (Fig. 3), is to interchange X_i and X_j.

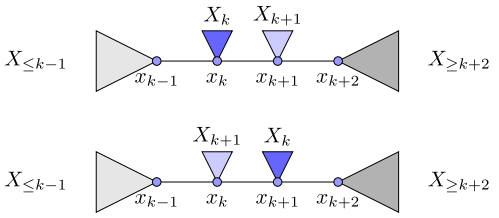

Figure 3: T (on top), before switching, and S (at bottom), after switching.

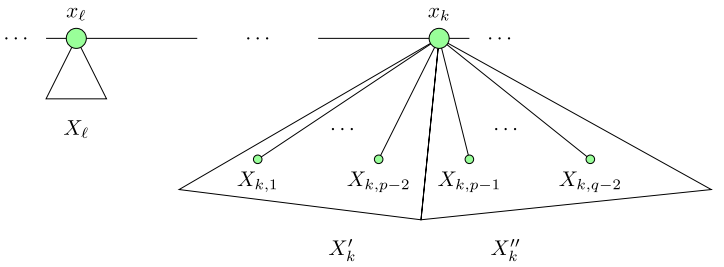

Figure 4: A 'degree switch' when $p = d(x_\ell) < d(x_k) = q$.

Remark 2.2. For the purpose of better illustrating the idea, in most of the figures we use 'darker' colors for 'larger' components in the sense of having larger values for $f_u^T(X)$ (for the corresponding u and X).

Definition 2.3. A 'tail-switch', denoted by $S^T_{v_1,v_2}(X_{\leq i}, X_{\geq j})$, is to switch the components $X_{\leq i}$ and $X_{\geq j}$.

Definition 2.4. Suppose that $p = d_T(x_\ell) < d_T(x_k) = q$ in Fig. 4. Let $x_{k,i}$ $(i = 1, 2, \ldots, q-2)$ denote the neighbors of x_k that are not on $P_T(v_1, v_2)$. Let $X_{k,1}, X_{k,2}, \ldots, X_{k,q-2}$ denote the corresponding components, ordered from smallest to largest according to the value of $f^T_{x_{k,i}}(X_{k,i})$. A 'degree-switch' $S^T_{v_1,v_2}(\emptyset_{x_\ell}, X''_k) := R^T_{v_1,v_2}(x_\ell, x_k)$ is to move the largest $q - p$ components (denoted by X''_k in Fig. 4) in X_k from x_k to x_ℓ.

2.1. Phase I of the switching algorithm

For convenience let C^T_k, $C^T_{\leq k}$ and $C^T_{\geq k}$ denote $f^T_{x_k}(X_k)$, $f^T_{x_k}(X_{\leq k})$ and $f^T_{x_k}(X_{\geq k})$ respectively. Then $C_{\leq 0} = C_{\geq n+1} = 1$.

Lemma 2.5. *If*

$$C^T_{\leq k-1} < C^T_{\geq k+2} \quad \textit{and} \quad C^T_k > C^T_{k+1},$$

then performing $S^T_{v_1,v_2}(X_k, X_{k+1})$ *will increase the number of subtrees.*

Proof. Let T and S denote the trees before and after switching respectively as in Fig. 3.

Note that from T to S, the numbers of subtrees that contain both or neither x_k and x_{k+1} stay the same. So we only need to consider the number of subtrees containing exactly one of x_k and x_{k+1}.

In T, the number of subtrees containing x_k but not x_{k+1} is

$$C^T_{\leq k} = C^T_k(1 + C^T_{\leq k-1})$$

and the number of subtrees containing x_{k+1} but not x_k is

$$C^T_{\geq k+1} = C^T_{k+1}(1 + C^T_{\geq k+2}).$$

Similarly, these two numbers in S are $C^T_{k+1}(1 + C^T_{\leq k-1})$ and $C^T_k(1 + C^T_{\geq k+2})$. Consequently, we have

$$f(S) - f(T) = (C^T_k - C^T_{k+1})(C^T_{\geq k+2} - C^T_{\leq k-1}) > 0. \qquad \square$$

The following is a result of "applying" bubble sort algorithm with component-switches.

Algorithm 2.6 (Phase I Switching Algorithm)**.** For a tree T and leaves $v_1, v_2 \in V(T)$. Let **Label 1** be the labeling of the path from v_1 to v_2 denoted by $v_1x_1x_2\ldots x_nv_2$, and let **Label 2** be a re-labeling of the path from v_2 to v_1 given by $v_2x_1x_2\ldots x_nv_1$. $S = P^T_1(v_1, v_2)$ is given below.

$S = (\emptyset, \emptyset)$ {Empty graph}
$R = T$
`while` $S \neq R$ `do`
  $k = 1$
  $S = R$ {Terminate program if no change in R}
  `Label` x_i`'s according to Label 1.`
  `while` $C^R_{\leq k-1} < C^R_{\geq k+2}$ `do`
    `if` $C^R_k > C^R_{k+1}$ `then`
      {Conditions of Lemma 2.5}
      $R = S^R_{v_1,v_2}(X_k, X_{k+1})$ {Increases # of subtrees}
    `end if`
    $k = k + 1$
  `end while`
  $k = 1$
  `Relabel` x_i`'s according to Label 2.` {Repeat for **Label 2**}

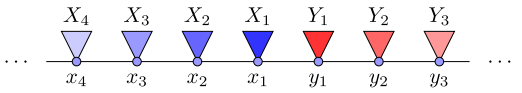

Figure 5: Re-labeling of path after the Phase I algorithm.

```
    while C^R_{≤k−1} < C^R_{≥k+2} do
      if C^R_k > C^R_{k+1} then
        R = S^R_{v2,v1}(X_k, X_{k+1})
      end if
      k = k + 1
    end while
  end while
```

In Algorithm 2.6, R updates if and only if there is an increase in the number of subtrees. Hence the algorithm must terminate at some point. Simple calculations shows the following in a tree resulted from the Phase I algorithm.

Corollary 2.7. *Let a path from v_1 to v_2 be sorted by the Phase I algorithm. Relabel the vertices such that x_1 is the vertex on the path with the largest component, and label the rest of the vertices as in Fig. 5. Without loss of generality, let y_1 and x_2 be labeled so that $D_1^T \geq C_2^T$ where $C_i^T = f_{x_i}^T(X_i)$ and $D_k^T = f_{y_k}^T(Y_k)$. Then*

$$C_1^T \geq C_2^T \geq \cdots \quad \text{and} \quad D_1^T \geq D_2^T \geq \cdots .$$

2.2. Phase II of the switching algorithm

For the lemmas that follow, we assume that our path is sorted by the Phase I algorithm and labeled as in Fig. 5. Firstly, the reader can easily verify the following technical observation.

Lemma 2.8. *If $C_1^T \geq D_1^T \geq C_2^T \geq \cdots \geq C_{k-1}^T \geq D_{k-1}^T$, let*

$$C := \sum_{i=1}^{k-1} \prod_{j=1}^{i} C_{k-j}^T + C_{k-1}^T \ldots C_1^T \sum_{i=1}^{k-1} \prod_{j=1}^{i} D_j^T$$

and

$$D := \sum_{i=1}^{k-1} \prod_{j=1}^{i} D_{k-j}^T + D_{k-1}^T \ldots D_1^T \sum_{i=1}^{k-1} \prod_{j=1}^{i} C_j^T,$$

then $C \geq D$.

Similarly, if $C_1^T \geq D_1^T \geq C_2^T \geq \cdots \geq C_{k-1}^T \geq D_{k-1}^T \geq C_k^T$, *let*

$$C' := \sum_{i=0}^{k-1} \prod_{j=0}^{i} C_{k-j}^T + C_k^T \ldots C_1^T \sum_{i=1}^{k-1} \prod_{j=1}^{i} D_j^T$$

and

$$D' := \sum_{i=1}^{k-1} \prod_{j=1}^{i} D_{k-j}^T + D_{k-1}^T \ldots D_1^T \sum_{i=1}^{k-1} \prod_{j=1}^{i} C_j^T,$$

then $D' \geq C'$.

Remark 2.9. Here C and D (C' and D') are the numbers of subtrees containing exactly one of x_{k-1} and y_{k-1} in the tree induced by the unions of X_i and Y_i ($i = 1, 2, \ldots$). Note that we have strict inequalities in both cases of Lemma 2.8 if

$$C_j > D_j \quad (D_j > C_{j+1}) \quad \text{for some } j. \tag{1}$$

For the simplicity of statements, we assume (1) in Lemmas 2.10, 2.11, 2.13. Note that in the case of $C_j = D_j$ or $D_j = C_{j+1}$ for every j, we can simply relabel the vertices/components while keeping the same tree.

Lemma 2.10. *If* $C_1^T \geq D_1^T \geq C_2^T \geq \cdots \geq C_{k-1}^T \geq D_{k-1}^T$ *and* $D_{\geq k}^T > C_{\geq k}^T$ *for some* k, *then performing* $S_{v_1,v_2}^T(X_{\geq k}, Y_{\geq k})$ *will increase the number of subtrees.*

If $C_1^T \geq D_1^T \geq C_2^T \geq \cdots \geq D_{k-1}^T \geq C_k^T$ *and* $C_{\geq k+1}^T > D_{\geq k}^T$, *then performing* $S_{v_1,v_2}^T(X_{\geq k+1}, Y_{\geq k})$ *will increase the number of subtrees.*

Proof. We show the first case (the other one is similar). Similar to Lemma 2.5, let T be the tree prior to switching and S the tree after switching. We only need to consider the subtrees that contain exactly one of x_k and y_k, yielding

$$\begin{aligned} f(S) - f(T) &= D_{\geq k}^T(1 + C) + C_{\geq k}^T(1 + D) - C_{\geq k}^T(1 + C) - D_{\geq k}^T(1 + D) \\ &= (D_{\geq k}^T - C_{\geq k}^T)(C - D) + 0 > 0 \end{aligned}$$

where C and D are as defined in Lemma 2.8. □

Lemma 2.11. *If* $C_1^T \geq D_1^T \geq C_2^T \geq D_2^T \geq \cdots \geq D_{k-1}^T \geq C_k^T$, $C_{\geq k+1}^T \geq D_{\geq k+1}^T$ *and* $D_k^T > C_k^T$, *then performing* $S_{v_1,v_2}^T(X_k, Y_k)$ *will increase the number of subtrees.*

If $C_1^T \geq D_1^T \geq C_2^T \geq D_2^T \geq \cdots \geq C_k^T \geq D_k^T$, $D_{\geq k+1}^T \geq C_{\geq k+2}^T$ *and* $C_{k+1}^T > D_k^T$, *then performing* $S_{v_1,v_2}^T(X_{k+1}, Y_k)$ *will increase the number of subtrees.*

Proof. Similar to Lemma 2.10, we consider the first case. This time we have

$$\begin{aligned} f(S) - f(T) &= D_k^T(1 + C_{\geq k+1}^T)(1 + C) + C_k^T(1 + D_{\geq k+1}^T)(1 + D) \\ &\quad - C_k^T(1 + C_{\geq k+1}^T)(1 + C) - D_k^T(1 + D_{\geq k+1}^T)(1 + D) \\ &= (D_k^T - C_k^T)[(1 + C_{\geq k+1})(1 + C) - (1 + D_{\geq k+1}^T)(1 + D)] > 0 \end{aligned}$$

where C and D are as defined in Lemma 2.8. □

With Lemmas 2.10 and 2.11, we combine component-switches and tail-switches to sort a path. The following algorithm will terminate if the number of non-empty subtrees in both tails are equal to 1, indicating that the ends of the path have been reached.

Algorithm 2.12 (Phase II Switching Algorithm)**.** For a tree T and leaves v_1, v_2. Choose x_1 to be the vertex with largest component on $P_T(v_1, v_2)$. Let y_1 be the neighbor of x_1 on $P_T(v_1, v_2)$ with larger component. Label the other vertices according to Fig. 5. Then $S = P_2^T(v_1, v_2)$ is given below.

$S = (\emptyset, \emptyset)$
$R = T$
$k = 1$
`while` $C_{\geq k+1}^R \neq 1$ `OR` $D_{\geq k+1}^R \neq 1$ `do`
{Terminate program if leaf vertex}
  `if` $C_k^R < D_k^R$ `then`
{Conditions of Lemma 2.11}
    `if` $C_{\geq k+1}^R < D_{\geq k+1}^R$ `then`
{Conditions of Lemma 2.10}
      $R = S_{v_1,v_2}^R(X_{\geq k}, Y_{\geq k})$
    `else`
      $R = S_{v_1,v_2}^R(X_k, Y_k)$
    `end if`
  `end if`
  `if` $D_k^R < C_{k+1}^R$ `then`
{Repeat for y_k and x_{k+1}}
    `if` $D_{\geq k+1}^R < C_{\geq k+2}^R$ `then`
      $R = S_{v_1,v_2}^R(Y_{\geq k}, X_{\geq k+1})$
    `else`

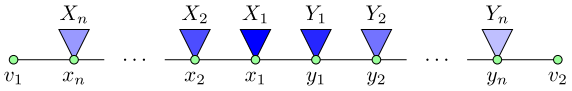

Figure 6: Path of length $2n+1$ after the Phase II algorithm.

$R = S^R_{v_1,v_2}(Y_k, X_{k+1})$
`end if`
`end if`
$k = k + 1$
`end while`
$S = R$

After the Phase II algorithm, the path from v_1 to v_2 is labeled in such a way that

$$C_1^T \geq D_1^T \geq C_2^T \geq \cdots \geq C_n^T \geq D_n^T$$
$$\text{and} \quad C_{\geq 1}^T \geq D_{\geq 1}^T \geq C_{\geq 2}^T \geq \cdots \geq C_{\geq n}^T \geq D_{\geq n}^T$$

for a path of odd length $2n+1$ (Fig. 6) and

$$C_1^T \geq D_1^T \geq C_2^T \geq \cdots \geq D_n^T \geq C_{n+1}^T$$
$$\text{and} \quad C_{\geq 1}^T \geq D_{\geq 1}^T \geq C_{\geq 2}^T \geq \cdots \geq D_{\geq n}^T \geq C_{\geq n+1}^T$$

for a path of even length $2n+2$.

2.3. Phase III of the switching algorithm

The final phase of the switching algorithm uses degree-switches to sort the degrees of the vertices on the path.

Lemma 2.13. *Consider $P_T(v_1, v_2)$ that has been sorted by the Phase I and Phase II algorithms and has been labeled as in Fig. 6. For $k = 1, 2, \ldots, n$, if $d(x_k) < d(y_k)$ or $d(y_k) < d(x_{k+1})$, performing a corresponding degree-switch (Definition 2.4) will increase the number of subtrees.*

Proof. We show only the case where $d(x_k) < d(y_k)$. Let S be the tree after moving Y_k'' to x_k. Also, let $D_k' = f_{y_k}^T(Y_k')$ and $D_k'' = f_{y_k}^T(Y_k'')$.

Notice that $D_k^T = D'_k D''_k$, then $f(S) - f(T)$ is

$$C_k^T D''_k(1 + C_{\geq k+1}^T)(1 + C) + D'_k(1 + D_{\geq k+1}^T)(1 + D)$$
$$- C_k^T(1 + C_{\geq k+1}^T)(1 + C) - D'_k D''_k(1 + D_{\geq k+1}^T)(1 + D)$$
$$= (D''_k - 1)(C_k^T - D'_k)[(1 + C_{\geq k+1}^T)(1 + C) - (1 + D_{\geq k+1}^T)(1 + D)] > 0.$$

□

The following algorithm applies Lemma 2.13. Note that the algorithm terminates after any degree-switch since the conditions for an increase in the number of subtrees as a result of a degree-switch may no longer be certain after the switch. Consequently the path must be resorted by the Phase I and II algorithms after every degree-switch. However this process will stop since every switch increases the number of subtrees.

Algorithm 2.14 (Phase III Switching Algorithm)**.** Given $P_T(v_1, v_2)$ that has been sorted by the Phase I and Phase II algorithms. Then $S = P_3^T(v_1, v_2)$ is given below.

```
for i = 1 to n do
  if d(x_i) < d(y_i) then
    S = R^T_{v1,v2}(x_i, y_i)
    break                                   {Terminate algorithm}
  end if
  if d(y_i) < d(x_{i+1}) then
    S = R^T_{v1,v2}(y_i, x_{i+1})
    break                                   {Terminate algorithm}
  end if
end for
```

2.4. The complete algorithm

We now introduce our final algorithm which encompasses all three previously discussed phases, wherein every step of the algorithm increases the number of subtrees. Therefore the algorithm will terminate after finitely many steps.

Algorithm 2.15 (Switching Algorithm)**.** Let T be a tree with leaf vertices $v_1, v_2, \ldots, v_\ell$. Then $S = SA(T)$ is given below.

$S = (\emptyset, \emptyset)$
$J = T$
$R = (\emptyset, \emptyset)$

```
while S ≠ J do
                                        {Terminate when every path is sorted}
  S = J
  J = (∅, ∅)
  for i = 1 to ℓ do
    for j = 1 to ℓ do
      while J ≠ R do
                                        {Terminate when path is sorted}
        J = R
        R = P_1^R(v_i, v_j)                                   {Phase I}
        R = P_2^R(v_i, v_j)                                   {Phase II}
        R = P_3^R(v_i, v_j)                                   {Phase III}
      end while
    end for
  end for
end while
```

Remark 2.16. Suppose $S = SA(T)$ where T is a tree with given degree sequence. Then for any path in S from one leaf to another, we can label the vertices on the path according to Fig. 6. On such paths we have,

$$\begin{aligned}
&C_1^S \geq D_1^S \geq C_2^S \geq \cdots \geq C_n^S \geq D_n^S,\\
&C_{\geq 1}^S \geq D_{\geq 1}^S \geq C_{\geq 2}^S \geq \cdots \geq C_{\geq n}^S \geq D_{\geq n}^S,\\
&d_S(x_1) \geq d_S(y_1) \geq d_S(x_2) \geq \cdots \geq d_S(x_n) \geq d_S(y_n)
\end{aligned}$$

for paths of odd length $2n-1$, and

$$\begin{aligned}
&C_1^S \geq D_1^S \geq C_2^S \geq \cdots \geq D_n^S \geq C_{n+1}^S,\\
&C_{\geq 1}^S \geq D_{\geq 1}^S \geq C_{\geq 2}^S \geq \cdots \geq D_{\geq n}^S \geq C_{\geq n+1}^S,\\
&d_S(x_1) \geq d_S(y_1) \geq d_S(x_2) \geq \cdots \geq d_S(y_n) \geq d_S(x_{n+1})
\end{aligned}$$

for paths of even length $2n$.

If the first two conditions imply the third one, then the Phase III algorithm would be unnecessary. We post it as a question in Section 4.

3. The extremal trees

The following observation is an immediate consequence from the definition of the greedy tree.

Lemma 3.1. *A rooted tree T with a given degree sequence is a greedy tree if:*

i) the root r has the largest degree;
ii) the heights of any two leaves differ by at most 1;
iii) for any two vertices u and w, if $h_T(w) < h_T(u)$, then $d(w) \geq d(u)$;
iv) for any two vertices u and w of the same height, $d(u) > d(w) \Rightarrow d(u') \geq d(w')$ for any successors u' of u and w' of w that are of the same height;
v) for any two vertices u and w of the same height, $d(u) > d(w) \Rightarrow d(u') \geq d(w')$ and $d(u'') \geq d(w'')$ for any siblings u' of u and w' of w or successors u'' of u' and w'' of w' of the same height.

3.1. The maximal tree

Theorem 3.2. *Given the degree sequence, the greedy tree maximizes the number of subtrees.*

Proof. First note that the maximal tree must satisfy the conditions in Remark 2.16 or we could increase the number of subtrees.

On the other hand, we will show that these conditions in Remark 2.16 imply the properties (i–v) of the greedy tree listed in Lemma 3.1. It was shown in [5] that the set of vertices that is contained in most subtrees consists of one or two adjacent vertices. We illustrate the idea by considering the first case with this vertex designated as the root.

Root the tree at the vertex r contained in the most subtrees. Consider a path between two leaf vertices that contains r and let v be a vertex on the path adjacent to r. Then, since the number of subtrees containing r is larger than the number of subtrees containing v, the tail containing r is larger than the tail containing v. By Remark 2.16, we can label, on this path, the root r as x_1 and the rest of the vertices according to Fig. 6 with v labeled as y_1 or x_2 (depending on which neighbor of r on this path has larger tail). Thus, the root has the largest degree of all vertices in the tree.

Let u and w be vertices such that $h(u) < h(w)$, and let s be the common ancestor of u and w with the greatest height. Then $h(u) = d(u, s) + h(s) < d(w, s) + h(s)$ implies that any path containing u or w and the root must also contain s. Furthermore, by Remark 2.16 again, $d(s) \geq d(u) \geq d(w)$. Therefore, vertices which are closer to the root will have greater degree.

This fact will force vertices which are closer to the root to be associated with the largest component and the largest degree on any path that we are studying. Then the properties (i–v) of Lemma 3.1 follows from considering the path through appropriate vertices. □

3.2. Trees close to being maximal

We have shown that the switching algorithm always converges to the maximal tree. Hence, by back-tracking the algorithm, the candidates for the kth maximal tree can be easily identified. Then the following corollaries are immediate consequences along this line:

Corollary 3.3. *Starting with the greedy tree, any tree with the same degree sequence can be achieved in a finite number of switchings wherein every switching results in a decrease in the number of subtrees.*

Corollary 3.4. *The kth maximal tree can be achieved with at most $k-1$ switches from the greedy tree.*

Consequently, the second maximal tree is exactly one switch away from the greedy tree. To illustrate the insights provided by the algorithm, we examine the second maximal tree in a bit more detail. The following observation helps us to identify the 'smaller' switches in the sense that it produces a smaller change in the number of subtrees.

Lemma 3.5. *Given the greedy tree T, for any component-switch or degree-switch, there exists a tail-switch which produces a decrease in the number of subtrees at least as small as that of the component-switch or degree-switch.*

Proof. **Case 1**. Consider the inverse of a component-switch in Phase I and suppose that we wish to switch X_k and X_{k+1} to decrease the number of subtrees. The switch is the same as performing a tail-switch with $X_{\leq k-1}$ and $X_{\geq k+2}$.

Case 2. Consider the inverse of a component-switch in Phase II as discussed in Lemma 2.11, that we wish to switch X_k and Y_k to decrease the number of subtrees. Note that, from the definition of a greedy tree, we already know that each branch of X_k and the tail $X_{\geq k+1}$ is at least as large as each branch of Y_k and the tail $Y_{\geq k+1}$. Consider each branch of X_k as a tail and label the tails at x_k as $X_{k,1}, X_{k,2}, \ldots, X_{k,p-1} := X_{\geq k+1}$, where p is the degree of x_k. Similarly, label the tails at y_k as $Y_{k,1}, Y_{k,2}, \ldots, Y_{k,q-1} := Y_{\geq k+1}$ where q is the degree of y_k. Let H be the tree produced by the switch $S(X_k, Y_k)$ and let H' be the tree produced by the tail switch $S(X_{k,j}, Y_{k,l})$, with $X_{k,j}$ ($Y_{k,l}$) being the smallest (largest) tail among $X_{k,1}, X_{k,2}, \ldots, X_{k,p-1}$ ($Y_{k,1}, Y_{k,2}, \ldots, Y_{k,q-1}$) (Fig. 7).

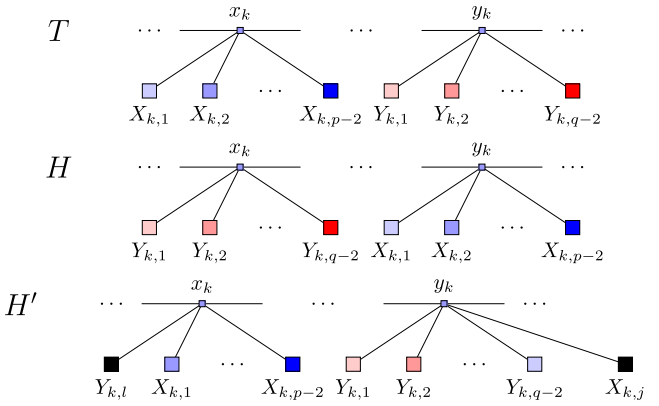

Figure 7: T—before switching, H—after component switch, H'—after tail switch.

Then

$$
\begin{aligned}
&f(H') - f(H)\\
&\quad = \left(\prod_{\substack{i=1\\ i\neq j}}^{p-1}(C_{k,i}+1)\right)(D_{k,l}+1)(1+C) + \left(\prod_{\substack{i=1\\ i\neq l}}^{q-1}(D_{k,i}+1)\right)(C_{k,j}+1)(1+D)\\
&\qquad - \left(\prod_{i=1}^{p-2}(C_{k,i}+1)\right)(D_{k,q-1}+1)(1+D)\\
&\qquad - \left(\prod_{i=1}^{q-2}(D_{k,i}+1)\right)(C_{k,p-1}+1)(1+C)\\
&\quad = \left(\prod_{\substack{i=1\\ i\neq j}}^{p-2}(C_{k,i}+1) - \prod_{\substack{i=1\\ i\neq l}}^{q-2}(D_{k,i}+1)\right)\\
&\qquad \times ((C_{k,p-1}+1)(D_{k,l}+1)(1+C) - (C_{k,j}+1)(D_{k,q-1}+1)(1+D))\\
&\quad > 0
\end{aligned}
$$

from the fact that $C_k > D_k$ by considering two cases (we skip the calculations). Here $C_{k,i}$ ($D_{k,i}$) is the number of subtrees in the component $X_{k,i}$ ($Y_{k,i}$) containing the corresponding root. Note that one could have $j = p-1$ or $l = q-1$ and the argument is still true.

Therefore such a tail switch is at least as 'small' as the component switch. Similarly for switching Y_k with X_{k+1}.

Case 3. Consider the inverse of a degree-switch in Phase III as discussed in Lemma 2.13, with $d(x_k) > d(y_k)$ ($d(y_k) > d(x_{k+1})$). Similar to Case 2, one can show that switching the smallest branch of X_k (Y_k) with the largest branch of Y_k (X_{k+1}) will result in a decrease at least as small. □

4. Summary and questions

In summary, we characterize the trees with given degree sequence that maximize the number of subtrees. The proof is displayed with an algorithm consisting a sequence of 'subtree-switchings' that increase the number of subtrees. By back-tracking from the maximal tree along these switchings provides some insights on how to find the second maximal tree and kth maximal tree in general. Some interesting questions arise in this study.

Firstly, one may be able to show that the 'degree-switches' are not necessary, by showing the following, namely that what was achieved in Phase III is forced to be true after Phases I and II.

Conjecture 4.1. *Suppose that H is a tree such that for any path between leaf vertices we have*

$$C_1^H \geq D_1^H \geq C_2^H \geq \cdots \geq C_n^H \geq D_n^H$$

for a path of odd length $2n+1$, and

$$C_1^H \geq D_1^H \geq C_2^H \geq \cdots \geq D_n^H \geq C_{n+1}^H$$

for a path of even length $2n+2$. Then we must have that

$$d(x_1) \geq d(y_1) \geq d(x_2) \geq \cdots \geq d(x_n) \geq d(y_n)$$

for the path of odd length, and

$$d(x_1) \geq d(y_1) \geq d(x_2) \geq \cdots \geq d(y_n) \geq d(x_{n+1})$$

for the path of even length.

Secondly, as part of the effort to study trees that are close to maximal and their numbers of subtrees, one encounters several questions regarding the switch operations on a greedy tree.

Problem 4.1. *For two vertices x_k and y_k on the path $\ldots x_2x_1(z)y_1y_2\ldots$, will one particular type of switch be smaller and which one?*

Problem 4.2. *If w is a common ancestor with the greatest height to leaf vertices v_1, v_2, v_3, and $f_T(x_i) \geq f_T(y_i) \geq f_T(z_i) \geq f_T(x_{i+1})$ for all i in the labellings of $wx_1x_2 \ldots v_1$, $wy_1y_2 \ldots v_2$ and $wz_1z_2 \ldots v_3$. Will there always be a switch on y_i and z_i that is smaller than any switch on x_i and z_i?*

Problem 4.3. *In a path labeled as $\ldots x_2x_1(z)y_1y_2 \ldots$, if $j < k$, will a switch on x_k and y_k always be smaller than the same switch on x_j and y_j?*

These seemingly elementary questions do not seem to have a straight forward answer. However, the answers to these questions will provide characterizations of the close-to-maximal trees.

References

[1] D. Eisenstata and G. Gordonb (2006). Non-isomorphic caterpillars with identical subtree data, *Discrete Math.* **306** 827–830. MR2234989

[2] C. Heuberger and H. Prodinger (2007). On α-greedy expansions of numbers, *Advances in Applied Mathematics* **38** (4) 505–525. MR2311049

[3] R. Kirk and H. Wang (2008). Largest number of subtrees of trees with a given maximum degree, *SIAM J. Discrete Mathematics* **22**(3) 985–995. MR2424834

[4] The On-Line Encyclopedia of Integer Sequences, A092781. `http://www.research.att.com/~njas/sequences`

[5] L.A. Székely and H. Wang (2005). On subtrees of trees, *Adv. Appl. Math.* **34** 138–155. MR2102279

[6] S. Wagner (2007). Correlation of graph-theoretical indices, *SIAM Journal on Discrete Math.* **21** (1) 33–46. MR2299692

[7] H. Wiener (1947). Structural determination of paraffin boiling points, *J. Amer. Chem. Soc.* **69** 17–20.

[8] W.G. Yan and Y.N. Yeh (2006). Enumeration of subtrees of trees, *Theoretical Comuter Science*, **369** 256–268. MR2277574

[9] X-M Zhang, X-D Zhang, D. Gray, and H. Wang (2012). The number of subtrees of trees with given degree sequence, *Journal of Graph Theory*, to appear.

Xiu-Mei Zhang
Department of Mathematics
Shanghai Jiao Tong University
800 Dongchuan road, Shanghai, 200240
P. R. China
E-mail address: wfluckyzxm@163.com

Xiao-Dong Zhang
Department of Mathematics, and MOE-LSC
Shanghai Jiao Tong University
800 Dongchuan road, Shanghai, 200240
P. R. China
E-mail address: xiaodong@sjtu.edu.cn

Daniel Gray
Department of Mathematics
University of Florida
Gainesville, FL, 32611
USA
E-mail address: dgray1@ufl.edu

Hua Wang
Department of Mathematical Sciences
Georgia Southern University
Statesboro, GA 30460
USA
E-mail address: hwang@georgiasouthern.edu

Received November 2, 2011