Fringe subtrees and additive functionals

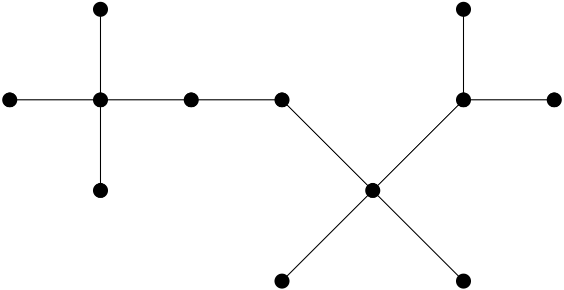
Recurring themes in the study of random trees

Stephan Wagner

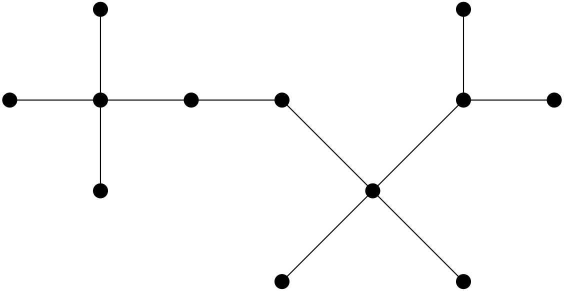
Uppsala University
Department of Mathematics

Taipei, 28 June 2023

Why study trees?

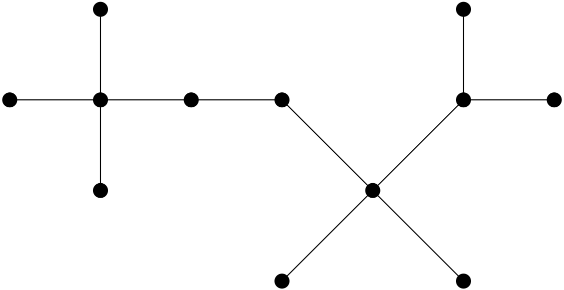


Why study trees?



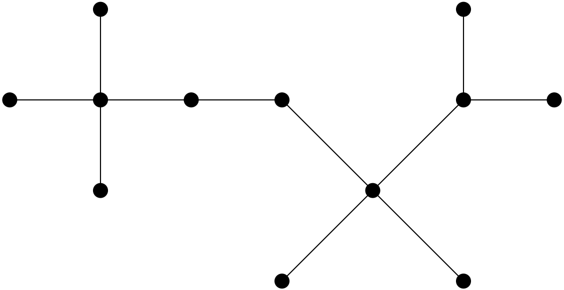
- They are simple.

Why study trees?



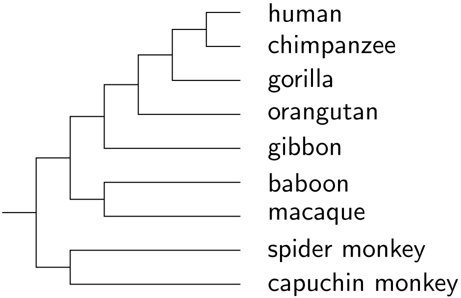
- They are simple.
- They have many nice properties.

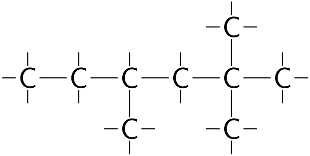
Why study trees?

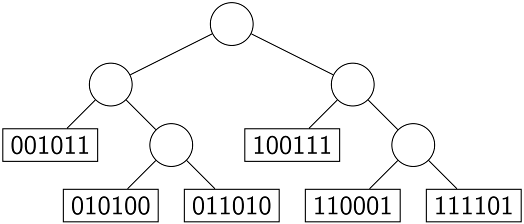


- They are simple.
- They have many nice properties.
- They are useful.

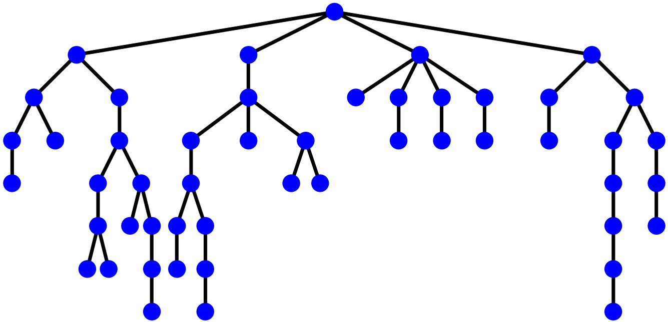
Trees are useful







Random trees



A random tree with 50 vertices.

Random tree models

Random trees play a role in many areas, from computational biology (phylogenetic trees) to the study of algorithms (search trees).

Depending on the specific application, various random models have been brought forward, such as:

Random tree models

Random trees play a role in many areas, from computational biology (phylogenetic trees) to the study of algorithms (search trees).

Depending on the specific application, various random models have been brought forward, such as:

- Uniform models (e.g. uniformly random unlabelled trees),

Random tree models

Random trees play a role in many areas, from computational biology (phylogenetic trees) to the study of algorithms (search trees).

Depending on the specific application, various random models have been brought forward, such as:

- Uniform models (e.g. uniformly random unlabelled trees),
- Branching processes (e.g. Galton–Watson trees),

Random tree models

Random trees play a role in many areas, from computational biology (phylogenetic trees) to the study of algorithms (search trees).

Depending on the specific application, various random models have been brought forward, such as:

- Uniform models (e.g. uniformly random unlabelled trees),
- Branching processes (e.g. Galton–Watson trees),
- Increasing tree models (e.g. recursive trees),

Random tree models

Random trees play a role in many areas, from computational biology (phylogenetic trees) to the study of algorithms (search trees).

Depending on the specific application, various random models have been brought forward, such as:

- Uniform models (e.g. uniformly random unlabelled trees),
- Branching processes (e.g. Galton–Watson trees),
- Increasing tree models (e.g. recursive trees),
- Models based on random strings or permutations (e.g. tries, binary search trees).

Uniform models

The simplest type of model uses the uniform distribution on the set of trees of a given order within a specified family (e.g. the family of all labelled trees, all unlabelled trees or all binary trees).

Uniform models

The simplest type of model uses the uniform distribution on the set of trees of a given order within a specified family (e.g. the family of all labelled trees, all unlabelled trees or all binary trees).

The analysis of such models often involves exact counting and generating functions.

Uniform models

The simplest type of model uses the uniform distribution on the set of trees of a given order within a specified family (e.g. the family of all labelled trees, all unlabelled trees or all binary trees).

The analysis of such models often involves exact counting and generating functions.

In particular, this is the case for *simply generated families of trees*.

Simply generated families

On the set of all rooted ordered (plane) trees, we impose a weight function by first specifying a sequence $1 = w_0, w_1, w_2, \ldots$ and then setting

$$w(T) = \prod_{i \geq 0} w_i^{N_i(T)},$$

where $N_i(T)$ is the number of vertices of outdegree i in T. Then we pick a tree of given order n at random, with probabilities proportional to the weights. For instance,

Simply generated families

On the set of all rooted ordered (plane) trees, we impose a weight function by first specifying a sequence $1 = w_0, w_1, w_2, \ldots$ and then setting

$$w(T) = \prod_{i \geq 0} w_i^{N_i(T)},$$

where $N_i(T)$ is the number of vertices of outdegree i in T. Then we pick a tree of given order n at random, with probabilities proportional to the weights. For instance,

- $w_0 = w_1 = w_2 = \cdots = 1$ generates random plane trees,

Simply generated families

On the set of all rooted ordered (plane) trees, we impose a weight function by first specifying a sequence $1 = w_0, w_1, w_2, \ldots$ and then setting

$$w(T) = \prod_{i \geq 0} w_i^{N_i(T)},$$

where $N_i(T)$ is the number of vertices of outdegree i in T. Then we pick a tree of given order n at random, with probabilities proportional to the weights. For instance,

- $w_0 = w_1 = w_2 = \cdots = 1$ generates random plane trees,
- $w_0 = w_2 = 1$ (and $w_i = 0$ otherwise) generates random binary trees,

Simply generated families

On the set of all rooted ordered (plane) trees, we impose a weight function by first specifying a sequence $1 = w_0, w_1, w_2, \ldots$ and then setting

$$w(T) = \prod_{i \geq 0} w_i^{N_i(T)},$$

where $N_i(T)$ is the number of vertices of outdegree i in T. Then we pick a tree of given order n at random, with probabilities proportional to the weights. For instance,

- $w_0 = w_1 = w_2 = \cdots = 1$ generates random plane trees,
- $w_0 = w_2 = 1$ (and $w_i = 0$ otherwise) generates random binary trees,
- $w_i = \frac{1}{i!}$ generates random rooted labelled trees.

Branching processes

A classical branching model to generate random trees is the *Galton–Watson tree model*: fix a probability distribution on the set $\{0, 1, 2, \ldots\}$.

Branching processes

A classical branching model to generate random trees is the *Galton–Watson tree model*: fix a probability distribution on the set $\{0, 1, 2, \ldots\}$.

- Start with a single vertex, the root.

Branching processes

A classical branching model to generate random trees is the *Galton–Watson tree model*: fix a probability distribution on the set $\{0, 1, 2, \dots\}$.

- Start with a single vertex, the root.
- At time t, all vertices at level t (i.e., distance t from the root) produce a number of children, independently at random according to the fixed distribution (some of the vertices might therefore not have children at all).

Branching processes

A classical branching model to generate random trees is the *Galton–Watson tree model*: fix a probability distribution on the set $\{0, 1, 2, \dots\}$.

- Start with a single vertex, the root.
- At time t, all vertices at level t (i.e., distance t from the root) produce a number of children, independently at random according to the fixed distribution (some of the vertices might therefore not have children at all).
- A random Galton–Watson tree of order n is obtained by conditioning the process.

Branching processes

A classical branching model to generate random trees is the *Galton–Watson tree model*: fix a probability distribution on the set $\{0, 1, 2, \dots\}$.

- Start with a single vertex, the root.
- At time t, all vertices at level t (i.e., distance t from the root) produce a number of children, independently at random according to the fixed distribution (some of the vertices might therefore not have children at all).
- A random Galton–Watson tree of order n is obtained by conditioning the process.

Simply generated trees and Galton–Watson trees are essentially equivalent. For example, a geometric distribution for branching will result in a random plane tree, a Poisson distribution in a random rooted labelled tree.

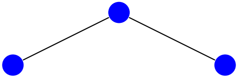
Branching processes

Construction of a random binary tree according to the Galton–Watson model: each vertex has either no children or precisely two.

$t = 0$

Branching processes

Construction of a random binary tree according to the Galton–Watson model: each vertex has either no children or precisely two.

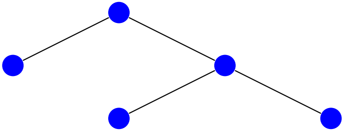


$t = 0$

$t = 1$

Branching processes

Construction of a random binary tree according to the Galton–Watson model: each vertex has either no children or precisely two.



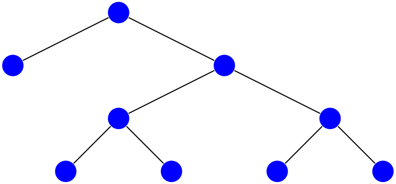
$t = 0$

$t = 1$

$t = 2$

Branching processes

Construction of a random binary tree according to the Galton–Watson model: each vertex has either no children or precisely two.



$t = 0$

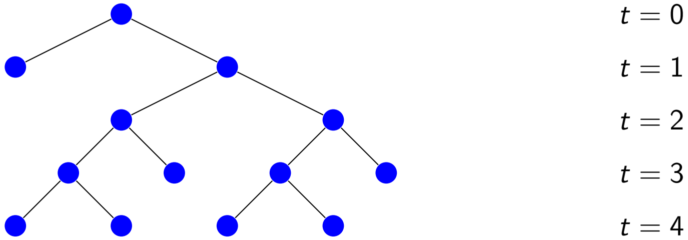
$t = 1$

$t = 2$

$t = 3$

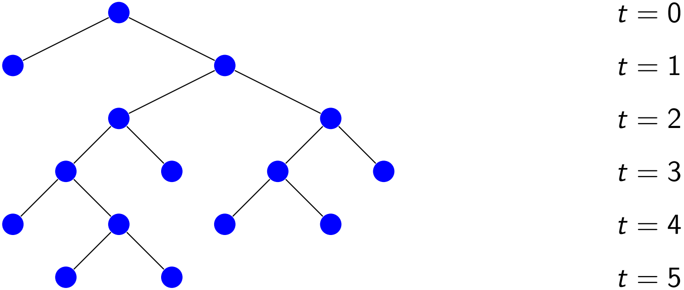
Branching processes

Construction of a random binary tree according to the Galton–Watson model: each vertex has either no children or precisely two.



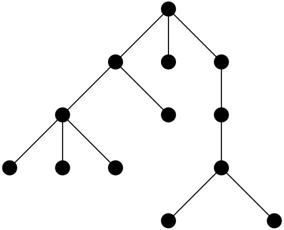
Branching processes

Construction of a random binary tree according to the Galton–Watson model: each vertex has either no children or precisely two.



Simply generated and Galton–Watson trees

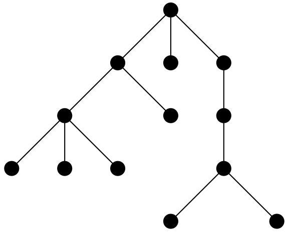
An example:



Consider the Galton–Watson process based on a geometric distribution with $P(X = k) = pq^k$ (where $p = 1 - q$).

Simply generated and Galton–Watson trees

An example:



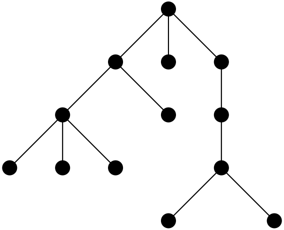
Consider the Galton–Watson process based on a geometric distribution with $P(X = k) = pq^k$ (where $p = 1 - q$).

The tree above has probability

$$p^7(pq)^2(pq^2)^2(pq^3)^2 = p^{13}q^{12},$$

Simply generated and Galton–Watson trees

An example:



Consider the Galton–Watson process based on a geometric distribution with $P(X = k) = pq^k$ (where $p = 1 - q$).

The tree above has probability

$$p^7(pq)^2(pq^2)^2(pq^3)^2 = p^{13}q^{12},$$

as does *every* rooted ordered tree with 13 vertices.

Random increasing trees

Another random model that produces very different shapes uses the following simple process, which generates *random recursive trees*:

Random increasing trees

Another random model that produces very different shapes uses the following simple process, which generates *random recursive trees*:

- Start with the root, which is labelled 1.

Random increasing trees

Another random model that produces very different shapes uses the following simple process, which generates *random recursive trees*:

- Start with the root, which is labelled 1.
- The n-th vertex is attached to one of the previous vertices, uniformly at random.

Random increasing trees

Another random model that produces very different shapes uses the following simple process, which generates *random recursive trees*:

- Start with the root, which is labelled 1.
- The n-th vertex is attached to one of the previous vertices, uniformly at random.

In this way, the labels along any path that starts at the root are increasing. Clearly, there are $(n-1)!$ possible recursive trees of order n, and there are indeed interesting connections to permutations.

Random increasing trees

Another random model that produces very different shapes uses the following simple process, which generates *random recursive trees*:

- Start with the root, which is labelled 1.
- The n-th vertex is attached to one of the previous vertices, uniformly at random.

In this way, the labels along any path that starts at the root are increasing. Clearly, there are $(n-1)!$ possible recursive trees of order n, and there are indeed interesting connections to permutations.

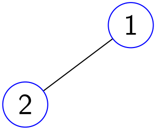
The model can be modified by not choosing a parent uniformly at random, but depending on the current outdegrees (to generate, for example, binary increasing trees or preferential attachment trees).

Random increasing trees

Construction of a recursive tree with 10 vertices:

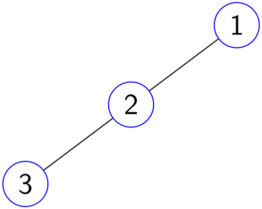
Random increasing trees

Construction of a recursive tree with 10 vertices:



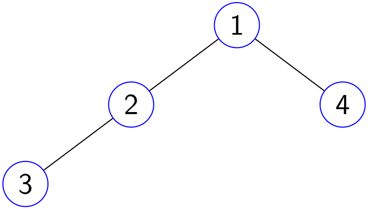
Random increasing trees

Construction of a recursive tree with 10 vertices:



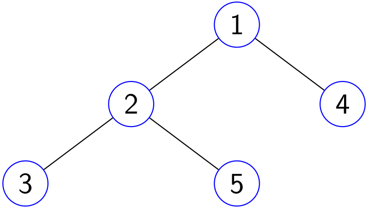
Random increasing trees

Construction of a recursive tree with 10 vertices:



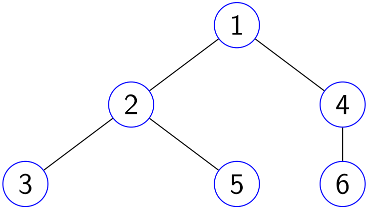
Random increasing trees

Construction of a recursive tree with 10 vertices:



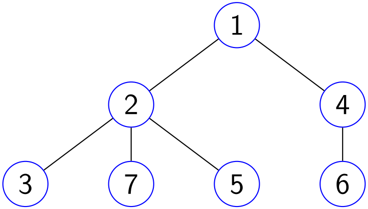
Random increasing trees

Construction of a recursive tree with 10 vertices:



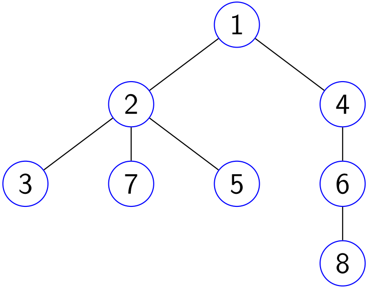
Random increasing trees

Construction of a recursive tree with 10 vertices:



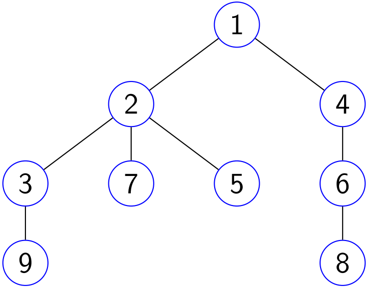
Random increasing trees

Construction of a recursive tree with 10 vertices:



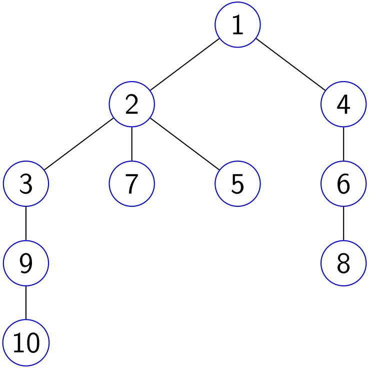
Random increasing trees

Construction of a recursive tree with 10 vertices:



Random increasing trees

Construction of a recursive tree with 10 vertices:



Processes based on random strings

In computer science, tries (short for *retrieval trees*) are a popular data structure for storing strings over a finite alphabet. A random binary trie is obtained as follows:

Processes based on random strings

In computer science, tries (short for *retrieval trees*) are a popular data structure for storing strings over a finite alphabet. A random binary trie is obtained as follows:

- Create n random binary strings of sufficient length, so that they are all distinct (for all practical purposes, one can assume that their length is infinite).

Processes based on random strings

In computer science, tries (short for *retrieval trees*) are a popular data structure for storing strings over a finite alphabet. A random binary trie is obtained as follows:

- Create n random binary strings of sufficient length, so that they are all distinct (for all practical purposes, one can assume that their length is infinite).
- All strings whose first bit is 0 are stored in the left subtree, the others in the right subtree.

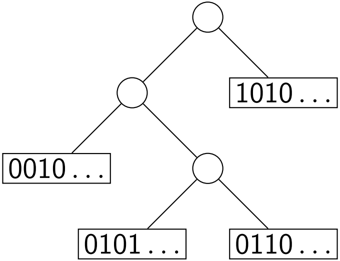
Processes based on random strings

In computer science, tries (short for *retrieval trees*) are a popular data structure for storing strings over a finite alphabet. A random binary trie is obtained as follows:

- Create n random binary strings of sufficient length, so that they are all distinct (for all practical purposes, one can assume that their length is infinite).
- All strings whose first bit is 0 are stored in the left subtree, the others in the right subtree.
- This procedure is repeated recursively.

Processes based on random strings

An example of a trie:



Tree parameters

Many different parameters of trees have been studied in the literature, such as

Tree parameters

Many different parameters of trees have been studied in the literature, such as

- the number of leaves,

Tree parameters

Many different parameters of trees have been studied in the literature, such as

- the number of leaves,
- the number of vertices of a given degree,

Tree parameters

Many different parameters of trees have been studied in the literature, such as

- the number of leaves,
- the number of vertices of a given degree,
- the path length (total distance of all vertices from the root),

Tree parameters

Many different parameters of trees have been studied in the literature, such as

- the number of leaves,
- the number of vertices of a given degree,
- the path length (total distance of all vertices from the root),
- the Wiener index (sum of distances between all pairs of vertices),

Tree parameters

Many different parameters of trees have been studied in the literature, such as

- the number of leaves,
- the number of vertices of a given degree,
- the path length (total distance of all vertices from the root),
- the Wiener index (sum of distances between all pairs of vertices),
- the independence number, domination number, and similar quantities,

Tree parameters

Many different parameters of trees have been studied in the literature, such as

- the number of leaves,
- the number of vertices of a given degree,
- the path length (total distance of all vertices from the root),
- the Wiener index (sum of distances between all pairs of vertices),
- the independence number, domination number, and similar quantities,
- the order of the automorphism group,

Tree parameters

Many different parameters of trees have been studied in the literature, such as

- the number of leaves,
- the number of vertices of a given degree,
- the path length (total distance of all vertices from the root),
- the Wiener index (sum of distances between all pairs of vertices),
- the independence number, domination number, and similar quantities,
- the order of the automorphism group,
- the number of subtrees,

Tree parameters

Many different parameters of trees have been studied in the literature, such as

- the number of leaves,
- the number of vertices of a given degree,
- the path length (total distance of all vertices from the root),
- the Wiener index (sum of distances between all pairs of vertices),
- the independence number, domination number, and similar quantities,
- the order of the automorphism group,
- the number of subtrees,
- the number of independent sets or matchings,

Tree parameters

Many different parameters of trees have been studied in the literature, such as

- the number of leaves,
- the number of vertices of a given degree,
- the path length (total distance of all vertices from the root),
- the Wiener index (sum of distances between all pairs of vertices),
- the independence number, domination number, and similar quantities,
- the order of the automorphism group,
- the number of subtrees,
- the number of independent sets or matchings,
- the spectrum.

A general question

Given a family of trees (a random tree model) and a tree parameter, what can we say about . . .

A general question

Given a family of trees (a random tree model) and a tree parameter, what can we say about . . .

- . . . the average value of the parameter among all trees with n vertices?

A general question

Given a family of trees (a random tree model) and a tree parameter, what can we say about . . .

- . . . the average value of the parameter among all trees with n vertices?
- . . . the variance or higher moments?

A general question

Given a family of trees (a random tree model) and a tree parameter, what can we say about ...

- ... the average value of the parameter among all trees with n vertices?
- ... the variance or higher moments?
- ... the distribution?

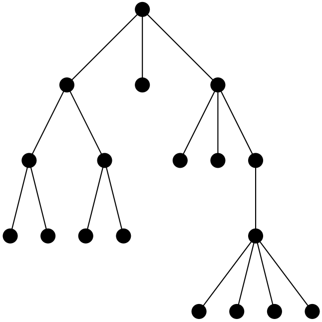
A general question

Given a family of trees (a random tree model) and a tree parameter, what can we say about ...

- ... the average value of the parameter among all trees with n vertices?
- ... the variance or higher moments?
- ... the distribution?

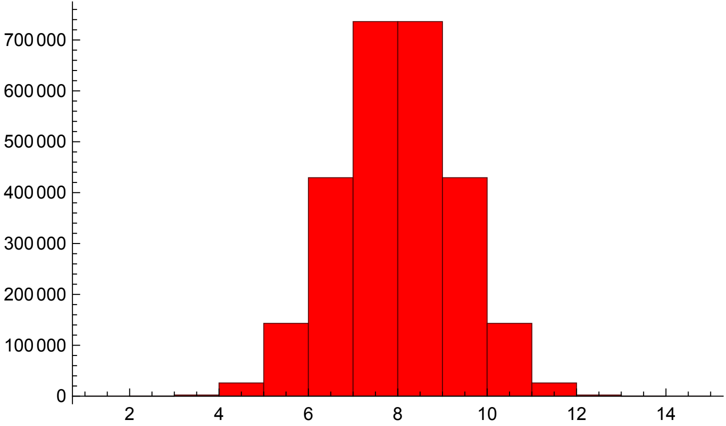
These questions become particularly relevant when n is large.

Some examples of parameters



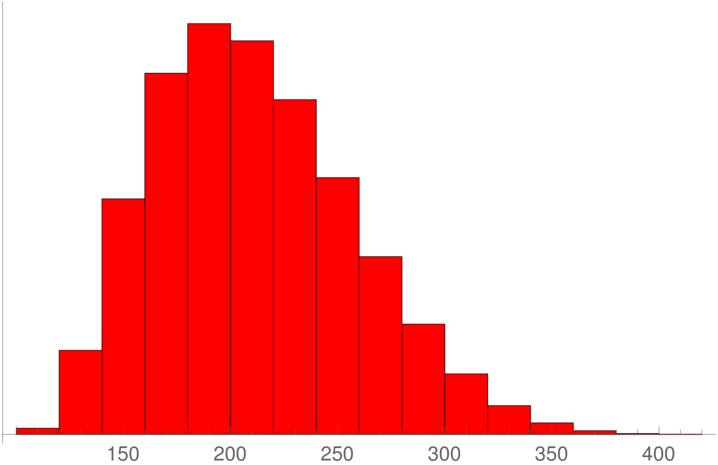
The tree above has 11 leaves, 2 “cherries”, path length 44, 384 automorphisms and 3945 subtrees.

Distribution of parameters: some examples



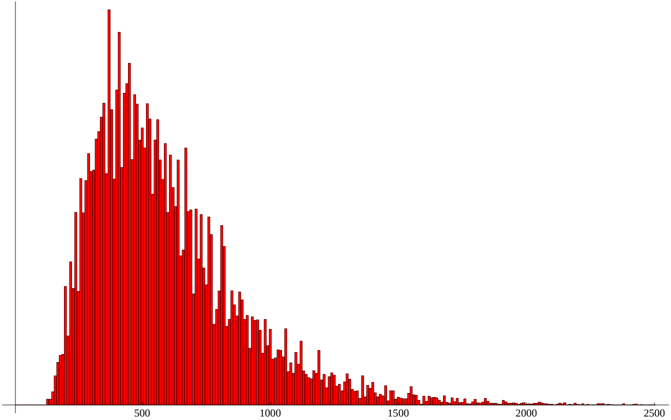
Distribution of the number of leaves in plane trees with 15 vertices. Plane trees with n vertices and k leaves are counted by the Narayana numbers $N_{n,k} = \frac{1}{n-1}\binom{n-1}{k}\binom{n-1}{k-1}$.

Distribution of parameters: some examples



Distribution of the path length in (pruned) binary trees with 30 vertices.

Distribution of parameters: some examples



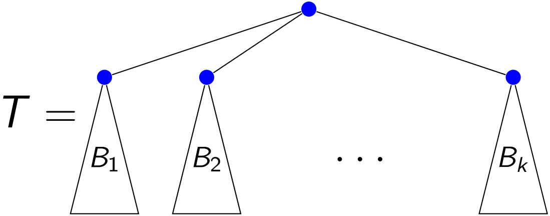
Distribution of the number of subtrees in labelled trees with 15 vertices.

Additive functionals: a general concept

A tree parameter is called an *additive functional* if it can be computed by adding its values for all the branches and adding a "toll function" that also depends on the tree.

Additive functionals: a general concept

A tree parameter is called an *additive functional* if it can be computed by adding its values for all the branches and adding a "toll function" that also depends on the tree.

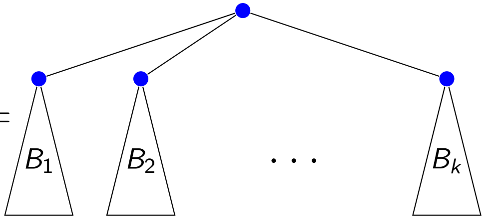


$$F(T) = F(B_1) + F(B_2) + \cdots + F(B_k) + f(T).$$

Additive functionals: a general concept

A tree parameter is called an *additive functional* if it can be computed by adding its values for all the branches and adding a "toll function" that also depends on the tree.

$T =$



$$F(T) = F(B_1) + F(B_2) + \cdots + F(B_k) + f(T).$$

Remark

The recursion remains true for the tree $T = \bullet$ of order 1 if we assume without loss of generality that $f(\bullet) = F(\bullet)$.

An equivalent definition

The fringe subtree T_v associated with a vertex v of a tree T is the subtree consisting of v and all its descendants.

An equivalent definition

The fringe subtree T_v associated with a vertex v of a tree T is the subtree consisting of v and all its descendants.

One can see by induction that the recursion

$$F(T) = F(B_1) + F(B_2) + \cdots + F(B_k) + f(T)$$

is equivalent to the formula

$$F(T) = \sum_v f(T_v).$$

Some examples

- The number of leaves, corresponding to the toll function

$$f(T) = \begin{cases} 1 & |T| = 1, \\ 0 & \text{otherwise.} \end{cases}$$

Some examples

- The number of leaves, corresponding to the toll function

$$f(T) = \begin{cases} 1 & |T| = 1, \\ 0 & \text{otherwise.} \end{cases}$$

- More generally, the number of occurrences of a fixed rooted tree H:

$$f(T) = \begin{cases} 1 & T \simeq H, \\ 0 & \text{otherwise.} \end{cases}$$

Some examples

- The number of leaves, corresponding to the toll function
$$f(T) = \begin{cases} 1 & |T| = 1, \\ 0 & \text{otherwise.} \end{cases}$$
- More generally, the number of occurrences of a fixed rooted tree H:
$$f(T) = \begin{cases} 1 & T \simeq H, \\ 0 & \text{otherwise.} \end{cases}$$
- The number of vertices whose outdegree is a fixed number k:
$$f(T) = \begin{cases} 1 & \text{if the root of } T \text{ has outdegree } k, \\ 0 & \text{otherwise.} \end{cases}$$

Some more examples

- The path length, i.e., the sum of the distances from the root to all vertices, can be obtained from the toll function $f(T) = |T| - 1$:

$$P(T) = \sum_{i=1}^{k} (P(B_i) + |B_i|) = |T| - 1 + \sum_{i=1}^{k} P(B_i).$$

Some more examples

- The path length, i.e., the sum of the distances from the root to all vertices, can be obtained from the toll function $f(T) = |T| - 1$:

$$P(T) = \sum_{i=1}^{k} (P(B_i) + |B_i|) = |T| - 1 + \sum_{i=1}^{k} P(B_i).$$

- The log-product of the subtree sizes, also called the "shape functional", corresponds to $f(T) = \log |T|$. It is related to the number of linear extensions:

$$\mathrm{LE}(T) = \binom{|T| - 1}{|B_1|, |B_2|, \dots, |B_k|} \prod_{i=1}^{k} \mathrm{LE}(B_i),$$

Some more examples

- The path length, i.e., the sum of the distances from the root to all vertices, can be obtained from the toll function $f(T) = |T| - 1$:

$$P(T) = \sum_{i=1}^{k} (P(B_i) + |B_i|) = |T| - 1 + \sum_{i=1}^{k} P(B_i).$$

- The log-product of the subtree sizes, also called the "shape functional", corresponds to $f(T) = \log|T|$. It is related to the number of linear extensions:

$$\mathrm{LE}(T) = \binom{|T|-1}{|B_1|, |B_2|, \ldots, |B_k|} \prod_{i=1}^{k} \mathrm{LE}(B_i),$$

thus

$$\log \frac{|T|!}{\mathrm{LE}(T)} = \log|T| + \sum_{i=1}^{n} \log \frac{|B_i|!}{\mathrm{LE}(B_i)}.$$

Even more examples

- The size of the automorphism group: if $c_1, c_2, \ldots, c_r$ are the multiplicities of the different isomorphism classes of branches, we have

$$|\operatorname{Aut}(T)| = \prod_{i=1}^{k} |\operatorname{Aut}(B_i)| \cdot \prod_{j=1}^{r} c_j!,$$

Even more examples

- The size of the automorphism group: if $c_1, c_2, \dots, c_r$ are the multiplicities of the different isomorphism classes of branches, we have

$$|\operatorname{Aut}(T)| = \prod_{i=1}^{k} |\operatorname{Aut}(B_i)| \cdot \prod_{j=1}^{r} c_j!,$$

thus

$$\log|\operatorname{Aut}(T)| = \sum_{i=1}^{k} \log|\operatorname{Aut}(B_i)| + \sum_{j=1}^{r} \log(c_j!).$$

Even more examples

- The size of the automorphism group: if $c_1, c_2, \ldots, c_r$ are the multiplicities of the different isomorphism classes of branches, we have

$$|\operatorname{Aut}(T)| = \prod_{i=1}^{k} |\operatorname{Aut}(B_i)| \cdot \prod_{j=1}^{r} c_j!,$$

  thus

$$\log|\operatorname{Aut}(T)| = \sum_{i=1}^{k} \log|\operatorname{Aut}(B_i)| + \sum_{j=1}^{r} \log(c_j!).$$

- The multiplicity of some eigenvalue λ:

Even more examples

- The size of the automorphism group: if $c_1, c_2, \dots, c_r$ are the multiplicities of the different isomorphism classes of branches, we have

$$|\operatorname{Aut}(T)| = \prod_{i=1}^{k} |\operatorname{Aut}(B_i)| \cdot \prod_{j=1}^{r} c_j!,$$

thus

$$\log|\operatorname{Aut}(T)| = \sum_{i=1}^{k} \log|\operatorname{Aut}(B_i)| + \sum_{j=1}^{r} \log(c_j!).$$

- The multiplicity of some eigenvalue λ:

$$N_\lambda(T) = \sum_{i=1}^{k} N_\lambda(B_i) + \epsilon_\lambda(T),$$

where $\epsilon_\lambda(T) \in \{-1, 0, 1\}$.

Back to AofA 2012 . . .

The number of subtrees (connected induced subgraphs) was first studied for simply generated trees by Meir and Moon in 1983. For labelled trees, the average number of subtrees is asymptotically equal to $(e/(e-1))^{3/2}e^{n/e}$.

Back to AofA 2012 . . .

The number of subtrees (connected induced subgraphs) was first studied for simply generated trees by Meir and Moon in 1983. For labelled trees, the average number of subtrees is asymptotically equal to $(e/(e-1))^{3/2}e^{n/e}$.

The asymptotic behaviour of the variance can be computed along the same lines to be of asymptotic order K^n for a constant $K \approx 2.15483 > e^{2/e}$, so the "standard" normalisation does not yield a limiting distribution.

The number of subtrees

However, the logarithm of the number of subtrees fits the framework of additive functionals!

The number of subtrees

However, the logarithm of the number of subtrees fits the framework of additive functionals!

It is somewhat more convenient to work with the number $s_1(T)$ of subtrees that contain the root.

The number of subtrees

However, the logarithm of the number of subtrees fits the framework of additive functionals!

It is somewhat more convenient to work with the number $s_1(T)$ of subtrees that contain the root.
The following recursion in terms of the branches $B_1, B_2, \dots, B_k$ holds:

$$s_1(T) = \prod_{i=1}^{k} (1 + s_1(B_i)).$$

The number of subtrees

However, the logarithm of the number of subtrees fits the framework of additive functionals!

It is somewhat more convenient to work with the number $s_1(T)$ of subtrees that contain the root.
The following recursion in terms of the branches $B_1, B_2, \ldots, B_k$ holds:

$$s_1(T) = \prod_{i=1}^{k} (1 + s_1(B_i)).$$

Hence

$$\log(1 + s_1(T)) = \sum_{i=1}^{k} \log(1 + s_1(B_i)) + \log(1 + s_1(T)^{-1}).$$

The number of subtrees

$$\log(1 + s_1(T)) = \sum_{i=1}^{k} \log(1 + s_1(B_i)) + \log(1 + s_1(T)^{-1}).$$

This means that $\log(1 + s_1(T))$ is additive with toll function $f(T) = \log(1 + s_1(T)^{-1})$.

The number of subtrees

$$\log(1 + s_1(T)) = \sum_{i=1}^{k} \log(1 + s_1(B_i)) + \log(1 + s_1(T)^{-1}).$$

This means that $\log(1 + s_1(T))$ is additive with toll function $f(T) = \log(1 + s_1(T)^{-1})$.

- A priori estimates show that $f(T)$ is exponentially small on average if T is a large tree.

The number of subtrees

$$\log(1 + s_1(T)) = \sum_{i=1}^{k} \log(1 + s_1(B_i)) + \log(1 + s_1(T)^{-1}).$$

This means that $\log(1 + s_1(T))$ is additive with toll function $f(T) = \log(1 + s_1(T)^{-1})$.

- A priori estimates show that $f(T)$ is exponentially small on average if T is a large tree.
- This suffices to show that $\log(1 + s_1(T))$ satisfies a central limit theorem, along with convergence of all moments.

The number of subtrees

$$\log(1 + s_1(T)) = \sum_{i=1}^{k} \log(1 + s_1(B_i)) + \log(1 + s_1(T)^{-1}).$$

This means that $\log(1 + s_1(T))$ is additive with toll function $f(T) = \log(1 + s_1(T)^{-1})$.

- A priori estimates show that $f(T)$ is exponentially small on average if T is a large tree.
- This suffices to show that $\log(1 + s_1(T))$ satisfies a central limit theorem, along with convergence of all moments.
- For the total number of subtrees $s(T)$, we have $\log s(T) = \log(1 + s_1(T)) + O(\log |T|)$.

The number of subtrees

Theorem (SW, AofA 2012)

The logarithm $\log s(\mathcal{T}_n)$ *of the number of subtrees of a random labelled tree* $\mathcal{T}_n$ *of order* n *is asymptotically normally distributed, with mean and variance asymptotically equal to* μn *and* $\sigma^2 n$ *respectively, where the numerical values of* μ *and* σ^2 *are* $\mu \approx 0.35$ *and* $\sigma^2 \approx 0.04$*, respectively.*

The number of subtrees

Theorem (SW, AofA 2012)

The logarithm $\log s(\mathcal{T}_n)$ *of the number of subtrees of a random labelled tree* $\mathcal{T}_n$ *of order* n *is asymptotically normally distributed, with mean and variance asymptotically equal to* μn *and* $\sigma^2 n$ *respectively, where the numerical values of* μ *and* σ^2 *are* $\mu \approx 0.35$ *and* $\sigma^2 \approx 0.04$*, respectively.*

Remark

Computing the constants is surprisingly tricky: they are given by infinite sums that converge poorly.

The number of subtrees

The method also applies to a related parameter, the *average subtree size*: it can also be seen as an additive parameter with (on average) exponentially small toll function.

The number of subtrees

The method also applies to a related parameter, the *average subtree size*: it can also be seen as an additive parameter with (on average) exponentially small toll function.

This condition also applies to other natural examples, but turns out to be a lot stronger than necessary.

General results

General results

Theorem (SW 2012/2015, Holmgren + Janson 2014/2015, Janson 2016, Holmgren + Janson + Šileikis 2016/2017, Ralaivaosaona + SW 2016/2019, Ralaivaosaona + Šileikis + SW 2018/2020, Janson 2022)

Under suitable technical conditions, an additive functional F on a family of trees satisfies a central limit theorem:

General results

Theorem (SW 2012/2015, Holmgren + Janson 2014/2015, Janson 2016, Holmgren + Janson + Šileikis 2016/2017, Ralaivaosaona + SW 2016/2019, Ralaivaosaona + Šileikis + SW 2018/2020, Janson 2022)

Under suitable technical conditions, an additive functional F on a family of trees satisfies a central limit theorem:

There exist constants μ and σ^2 such that mean and variance of $F(\mathcal{T}_n)$ for a random tree $\mathcal{T}_n$ with n vertices are $\mu_n \sim \mu n$ and $\sigma_n^2 \sim \sigma^2 n$.

General results

Theorem (SW 2012/2015, Holmgren + Janson 2014/2015, Janson 2016, Holmgren + Janson + Šileikis 2016/2017, Ralaivaosaona + SW 2016/2019, Ralaivaosaona + Šileikis + SW 2018/2020, Janson 2022)

Under suitable technical conditions, an additive functional F on a family of trees satisfies a central limit theorem:

There exist constants μ and σ^2 such that mean and variance of $F(\mathcal{T}_n)$ for a random tree $\mathcal{T}_n$ with n vertices are $\mu_n \sim \mu n$ and $\sigma_n^2 \sim \sigma^2 n$.

Moreover, the renormalised random variable

$$X_n = \frac{F(\mathcal{T}_n) - \mu n}{\sqrt{\sigma^2 n}}$$

converges weakly to a standard normal distribution.

General results

What are “suitable technical conditions”?

General results

What are "suitable technical conditions"?

There are three main types of conditions:

- The toll function f only depends on the size of the tree.

General results

What are “suitable technical conditions”?

There are three main types of conditions:

- The toll function f only depends on the size of the tree.
- The toll function f is “small” (at least on average) for large trees.

General results

What are "suitable technical conditions"?

There are three main types of conditions:

- The toll function f only depends on the size of the tree.
- The toll function f is "small" (at least on average) for large trees.
- The toll function f is "local" (only depends on a small neighbourhood of the root), at least approximately.

General results

Proofs involve:

- combinatorial techniques (explicit counting, generating functions, analytic combinatorics, . . .)

General results

Proofs involve:

- combinatorial techniques (explicit counting, generating functions, analytic combinatorics, ...)
- probabilistic techniques (growth processes, urn models, method of moments, ...)

Examples covered

Many different examples are covered by one or more of the conditions:

(N) = normal (L) = lognormal

Examples covered

Many different examples are covered by one or more of the conditions:

- the number of leaves (N),

(N) = normal   (L) = lognormal

Examples covered

Many different examples are covered by one or more of the conditions:

- the number of leaves (N),
- the number of vertices of degree k (N),

(N) = normal   (L) = lognormal

Examples covered

Many different examples are covered by one or more of the conditions:

- the number of leaves (N),
- the number of vertices of degree k (N),
- the number of fringe subtrees of a given type (N),

(N) = normal (L) = lognormal

Examples covered

Many different examples are covered by one or more of the conditions:

- the number of leaves (N),
- the number of vertices of degree k (N),
- the number of fringe subtrees of a given type (N),
- the number of subtrees (L),

(N) = normal   (L) = lognormal

Examples covered

Many different examples are covered by one or more of the conditions:

- the number of leaves (N),
- the number of vertices of degree k (N),
- the number of fringe subtrees of a given type (N),
- the number of subtrees (L),
- the number of independent sets (L),

(N) = normal (L) = lognormal

Examples covered

Many different examples are covered by one or more of the conditions:

- the number of leaves (N),
- the number of vertices of degree k (N),
- the number of fringe subtrees of a given type (N),
- the number of subtrees (L),
- the number of independent sets (L),
- the number of matchings (L),

(N) = normal (L) = lognormal

Examples covered

Many different examples are covered by one or more of the conditions:

- the number of leaves (N),
- the number of vertices of degree k (N),
- the number of fringe subtrees of a given type (N),
- the number of subtrees (L),
- the number of independent sets (L),
- the number of matchings (L),
- the independence number (N),

(N) = normal (L) = lognormal

Examples covered

Many different examples are covered by one or more of the conditions:

- the number of leaves (N),
- the number of vertices of degree k (N),
- the number of fringe subtrees of a given type (N),
- the number of subtrees (L),
- the number of independent sets (L),
- the number of matchings (L),
- the independence number (N),
- the domination number (N),

(N) = normal (L) = lognormal

Examples covered

Many different examples are covered by one or more of the conditions:

- the number of leaves (N),
- the number of vertices of degree k (N),
- the number of fringe subtrees of a given type (N),
- the number of subtrees (L),
- the number of independent sets (L),
- the number of matchings (L),
- the independence number (N),
- the domination number (N),
- the average subtree size (N),

(N) = normal (L) = lognormal

Examples covered

Many different examples are covered by one or more of the conditions:

- the number of leaves (N),
- the number of vertices of degree k (N),
- the number of fringe subtrees of a given type (N),
- the number of subtrees (L),
- the number of independent sets (L),
- the number of matchings (L),
- the independence number (N),
- the domination number (N),
- the average subtree size (N),
- the number of automorphisms (L),

(N) = normal (L) = lognormal

Examples covered

Many different examples are covered by one or more of the conditions:

- the number of leaves (N),
- the number of vertices of degree k (N),
- the number of fringe subtrees of a given type (N),
- the number of subtrees (L),
- the number of independent sets (L),
- the number of matchings (L),
- the independence number (N),
- the domination number (N),
- the average subtree size (N),
- the number of automorphisms (L),
- the multiplicity of eigenvalues (N).

(N) = normal (L) = lognormal

Non-Gaussian limits

When the toll function is not sufficiently small, then non-Gaussian limit distributions can be observed. Perhaps the most prominent examples are the *path length* (sum of distances from the root) and the related *Wiener index* (sum of all distances).

Non-Gaussian limits

When the toll function is not sufficiently small, then non-Gaussian limit distributions can be observed. Perhaps the most prominent examples are the *path length* (sum of distances from the root) and the related *Wiener index* (sum of all distances).

For simply generated trees/conditioned Galton–Watson trees and Pólya trees (rooted unordered trees), limits that can be expressed in terms of a Brownian excursion are observed.

Path length and Wiener index

Theorem (Takács 1993, Janson 2003, SW 2012)

For conditioned Galton–Watson trees (whose offspring distribution has finite variance) and for uniformly random Pólya trees, there exists a constant $\mu > 0$ *such that the path length* $D(\mathcal{T}_n)$ *and the Wiener index* $W(\mathcal{T}_n)$ *of a random tree* $\mathcal{T}_n$ *with* n *vertices have means* $\mu_n^D \sim \mu n^{3/2}$ *and* $\mu_n^W \sim \frac{\mu}{2} n^{5/2}$ *respectively.*

Path length and Wiener index

Theorem (cont.)

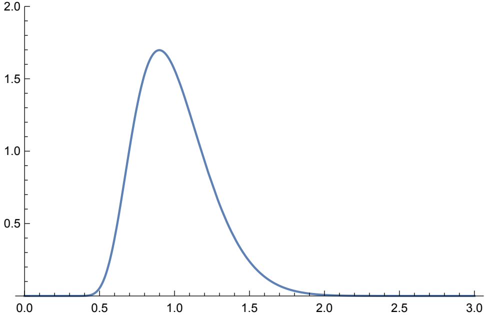
Moreover, the random variables

$$X_n = \frac{D(\mathcal{T}_n)}{\mu n^{3/2}} \qquad \text{and} \qquad Y_n = \frac{W(\mathcal{T}_n)}{\mu n^{5/2}}$$

converge weakly to random variables given in terms of a normalised Brownian excursion $e(t)$ on $[0,1]$:

$$\sqrt{\frac{8}{\pi}}\int_0^1 e(t)\,dt \quad \text{and} \quad \sqrt{\frac{8}{\pi}}\iint_{0<s<t<1}\Big(e(s)+e(t)-2\min_{s\le u\le t} e(u)\Big)\,ds\,dt.$$

Path length and Wiener index



The Airy distribution: limiting distribution of the path length.

Power toll functions

The path length corresponds essentially to the toll function $f(T) = |T|$.

Power toll functions

The path length corresponds essentially to the toll function $f(T) = |T|$.

The more general case $f(T) = |T|^\alpha$ has been studied in detail as well, as discussed in Jim Fill's talk (Fill + Kapur 2004, Fill + Flajolet + Kapur 2005, Delmas + Dhersin + Sciauveau 2018, Abraham + Delmas + Nassif 2022, Fill + Janson 2022, Fill + Janson + SW 2023+).

Power toll functions

The path length corresponds essentially to the toll function $f(T) = |T|$.

The more general case $f(T) = |T|^\alpha$ has been studied in detail as well, as discussed in Jim Fill's talk (Fill + Kapur 2004, Fill + Flajolet + Kapur 2005, Delmas + Dhersin + Sciauveau 2018, Abraham + Delmas + Nassif 2022, Fill + Janson 2022, Fill + Janson + SW 2023+).

A phase transition can be observed at $\mathrm{Re}\,\alpha = 0$.

Some more recent applications of additive functionals

- Hackl + Heuberger + Kropf + Prodinger 2018: cutting and pruning procedures

Some more recent applications of additive functionals

- Hackl + Heuberger + Kropf + Prodinger 2018: cutting and pruning procedures
- Gołębiewski + Magner + Szpankowski 2019: entropy of random tree classes

Some more recent applications of additive functionals

- Hackl + Heuberger + Kropf + Prodinger 2018: cutting and pruning procedures
- Gołębiewski + Magner + Szpankowski 2019: entropy of random tree classes
- Komjáthy + Ódor 2021: metric dimension

Some more recent applications of additive functionals

- Hackl + Heuberger + Kropf + Prodinger 2018: cutting and pruning procedures
- Gołębiewski + Magner + Szpankowski 2019: entropy of random tree classes
- Komjáthy + Ódor 2021: metric dimension
- Disanto + Fuchs + Paningbatan + Rosenberg 2022: ancestral configurations of species trees

Some more recent applications of additive functionals

- Hackl + Heuberger + Kropf + Prodinger 2018: cutting and pruning procedures
- Gołębiewski + Magner + Szpankowski 2019: entropy of random tree classes
- Komjáthy + Ódor 2021: metric dimension
- Disanto + Fuchs + Paningbatan + Rosenberg 2022: ancestral configurations of species trees
- Seelbach-Benkner + SW 2022: tree compression algorithms

Directions for future research

- Random tree models that have not been covered yet,

Directions for future research

- Random tree models that have not been covered yet,
- Tree-like graph classes,

Directions for future research

- Random tree models that have not been covered yet,
- Tree-like graph classes,
- Parameters that are not covered by any of the existing general conditions,

Directions for future research

- Random tree models that have not been covered yet,
- Tree-like graph classes,
- Parameters that are not covered by any of the existing general conditions,
- General schemes for additive parameters whose limit distributions are not normal,

Directions for future research

- Random tree models that have not been covered yet,
- Tree-like graph classes,
- Parameters that are not covered by any of the existing general conditions,
- General schemes for additive parameters whose limit distributions are not normal,
- Parameters that follow different types of recursion (e.g.: max instead of $\sum$).