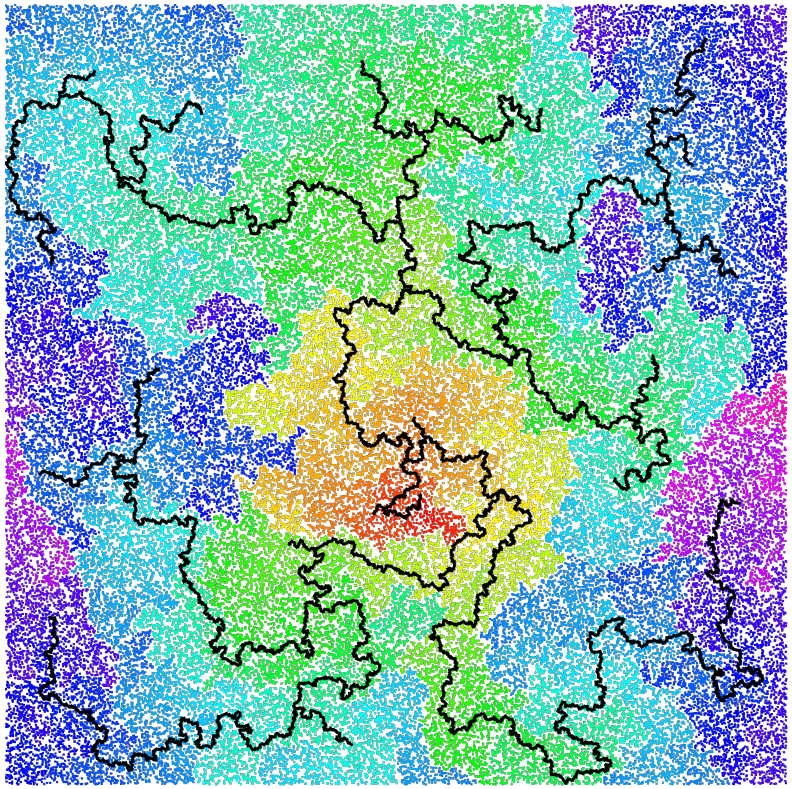
The critical random graph



L. Addario-Berry

Random Structures and Dynamics

Oxford

April 11-14, 2011

Parts are joint with Nicolas Broutin, Luc Devroye, Christina Goldschmidt, Svante Janson, Grégory Miermont

Limit for a component of $G_{n,1/n}$

A component of size about $cn^{2/3}$ has distributional limit:

Limit for a component of $G_{n,1/n}$

A component of size about $cn^{2/3}$ has distributional limit:

- Tree coded by a Brownian excursion of length c, with exponential tilt; plus

Limit for a component of $G_{n,1/n}$

A component of size about $cn^{2/3}$ has distributional limit:

- Tree coded by a Brownian excursion of length c, with exponential tilt; plus
- identifications given by Poisson random points under the excursion.

Limit for a component of $G_{n,1/n}$

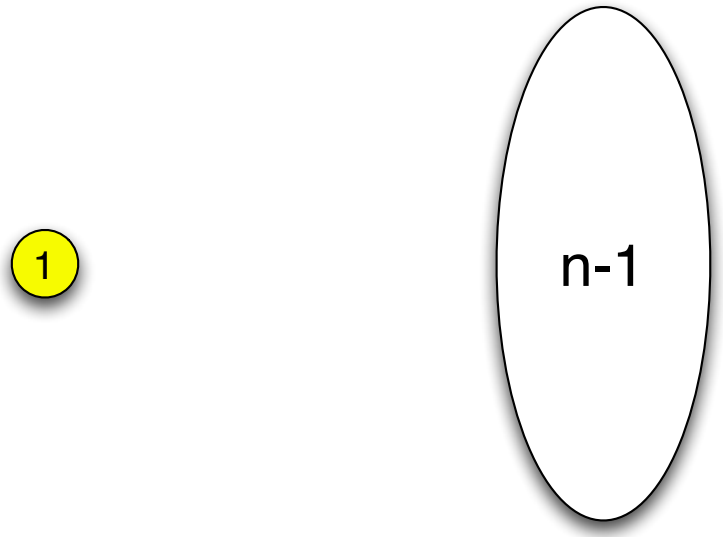
A component of size about $cn^{2/3}$ has distributional limit:

- Tree coded by a Brownian excursion of length c, with exponential tilt; plus
- identifications given by Poisson random points under the excursion.

Now have an asymptotic description of the structure of each "macroscopic" component of the critical random graph.

Brownian motion with drift?

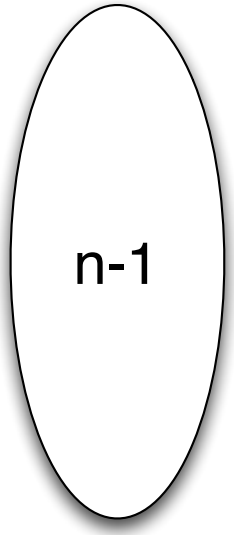
Mean number of neighbours:



Brownian motion with drift?

Mean number of neighbours:

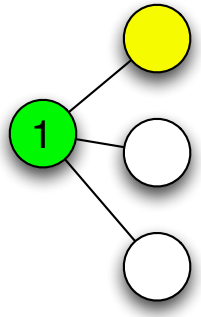
- $\mathrm{Bin}(n-1, 1/n) = 1 - 1/n$ at step 1.

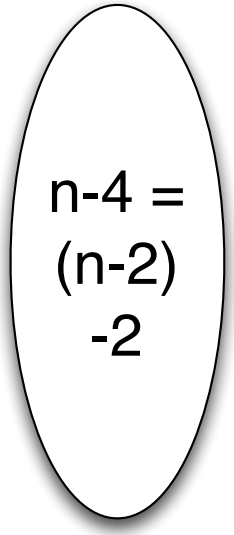


Brownian motion with drift?

Mean number of neighbours:

- $\mathrm{Bin}(n-1, 1/n) = 1 - 1/n$ at step 1.
- $1 - 4/n$ at step 2.

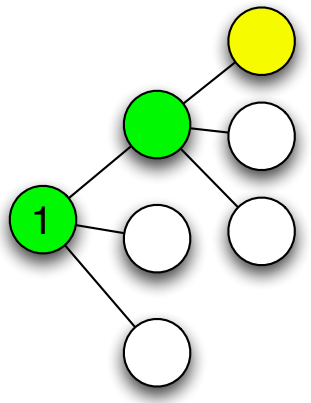


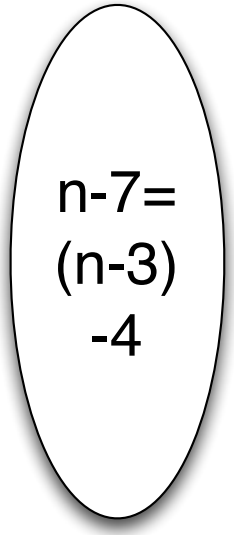


Brownian motion with drift?

Mean number of neighbours:

- $\mathrm{Bin}(n-1, 1/n) = 1 - 1/n$ at step 1.
- $1 - 4/n$ at step 2.
- $1 - 7/n$ at step 3.

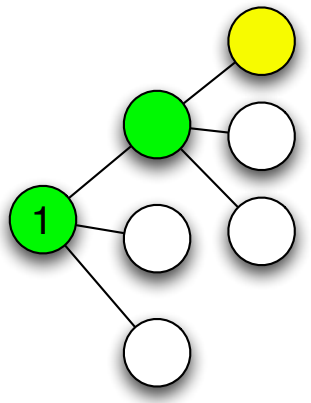


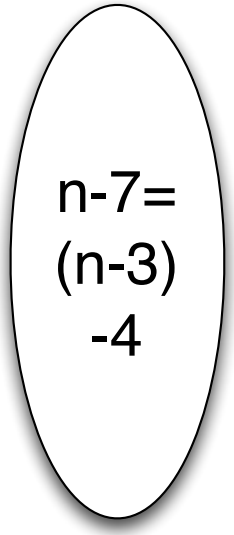


Brownian motion with drift?

Mean number of neighbours:

- $\mathrm{Bin}(n-1, 1/n) = 1 - 1/n$ at step 1.
- $1 - 4/n$ at step 2.
- $1 - 7/n$ at step 3.



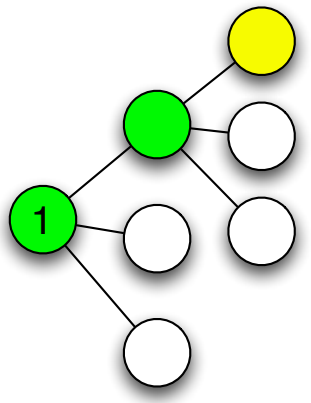


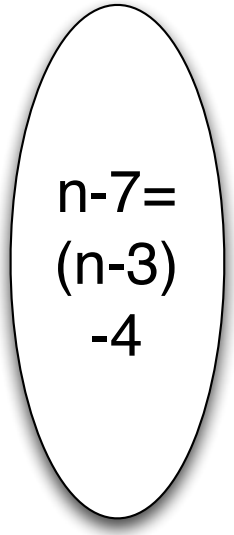
- At step i: infinitesimal drift $-i/n - (\#\text{ of active vertices})/n = -(i + O(\sqrt{i}))/n$.

Brownian motion with drift?

Mean number of neighbours:

- $\mathrm{Bin}(n-1, 1/n) = 1 - 1/n$ at step 1.
- $1 - 4/n$ at step 2.
- $1 - 7/n$ at step 3.



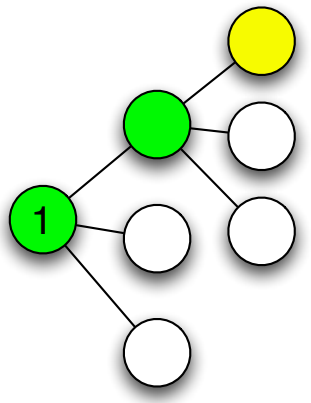


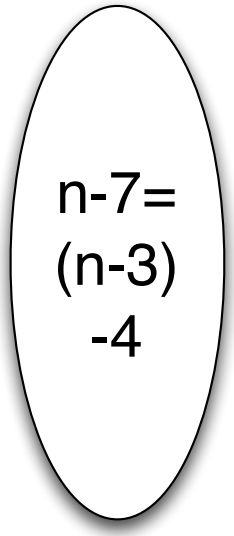
- At step i: infinitesimal drift $-i/n - (\#\text{ of active vertices})/n = -(i + O(\sqrt{i}))/n$.
- Accumulated drift about $-i^2/2n$, or $-n^{1/3}$ at time $n^{2/3}$.

Brownian motion with drift?

Mean number of neighbours:

- $\mathrm{Bin}(n-1, 1/n) = 1 - 1/n$ at step 1.
- $1 - 4/n$ at step 2.
- $1 - 7/n$ at step 3.





- At step i: infinitesimal drift $-i/n - (\#\text{ of active vertices})/n = -(i + O(\sqrt{i}))/n$.
- Accumulated drift about $-i^2/2n$, or $-n^{1/3}$ at time $n^{2/3}$.
- The first component is explored when discrete walk hits -1; start again from 0 and explore another component! This is why reflection at running infimum.

Need to know

- What the MSTs of these components look like.

Need to know

- What the MSTs of these components look like.
- How these MSTs hook together.

How the MSTs hook together

List the components of $G_{n,1/n}$ as $C_1, C_2, \ldots$.

How the MSTs hook together

List the components of $G_{n,1/n}$ as $C_1, C_2, \ldots$.
Recall that in terms of their vertex sets, these are also the components of $T_{n,1/n}$.

How the MSTs hook together

List the components of $G_{n,1/n}$ as $C_1, C_2, \ldots$.
Recall that in terms of their vertex sets, these are also the components of $T_{n,1/n}$.

- The next edge that joints two components will join a given C_i and C_j with probability proportional to $|C_i| \cdot |C_j|$.

How the MSTs hook together

List the components of $G_{n,1/n}$ as $C_1, C_2, \ldots$.
Recall that in terms of their vertex sets, these are also the components of $T_{n,1/n}$.

- The next edge that joints two components will join a given C_i and C_j with probability proportional to $|C_i| \cdot |C_j|$.
- In the limit, the merging of components is governed by the multiplicative coalescent. (Aldous 1997.)

How the MSTs hook together

List the components of $G_{n,1/n}$ as $C_1, C_2, \ldots$.
Recall that in terms of their vertex sets, these are also the components of $T_{n,1/n}$.

- The next edge that joints two components will join a given C_i and C_j with probability proportional to $|C_i| \cdot |C_j|$.
- In the limit, the merging of components is governed by the multiplicative coalescent. (Aldous 1997.)
- It is a fair bit of work to define the multiplicative coalescent dynamics on spaces of sequences of measured metric spaces, and show that the resulting dynamics yield the required closure properties.

How the MSTs hook together

List the components of $G_{n,1/n}$ as $C_1, C_2, \ldots$.
Recall that in terms of their vertex sets, these are also the components of $T_{n,1/n}$.

- The next edge that joints two components will join a given C_i and C_j with probability proportional to $|C_i| \cdot |C_j|$.
- In the limit, the merging of components is governed by the multiplicative coalescent. (Aldous 1997.)
- It is a fair bit of work (ongoing) to define the multiplicative coalescent dynamics on spaces of sequences of measured metric spaces, and show that the resulting dynamics yield the required closure properties.

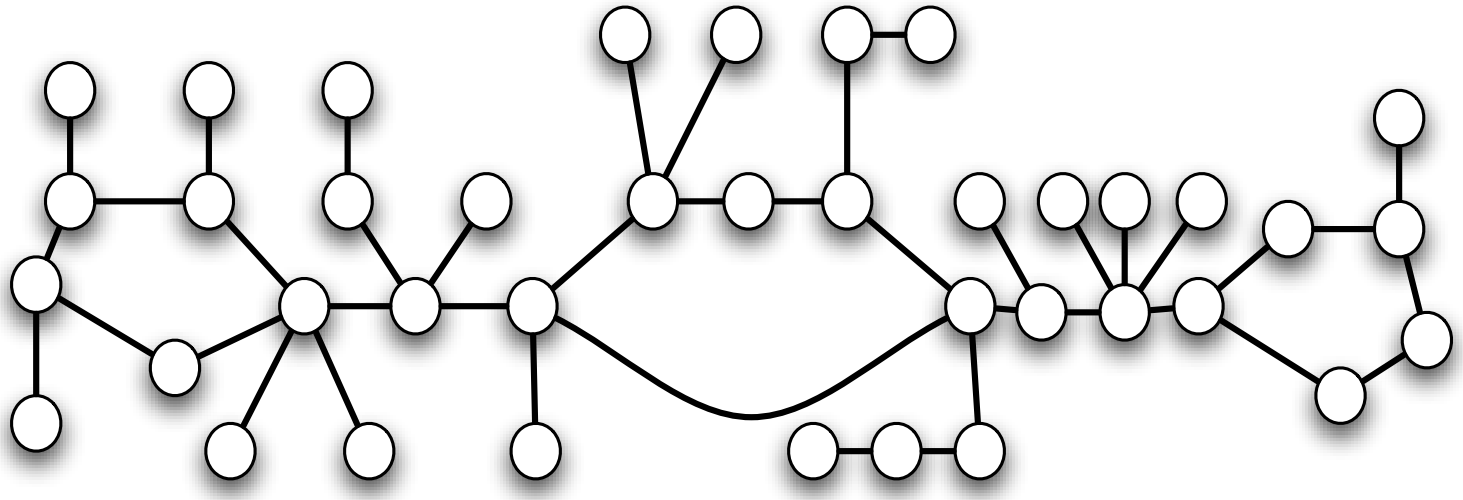
What the MSTs look like

- We will use the cycle breaking algorithm to figure this out.

What the MSTs look like

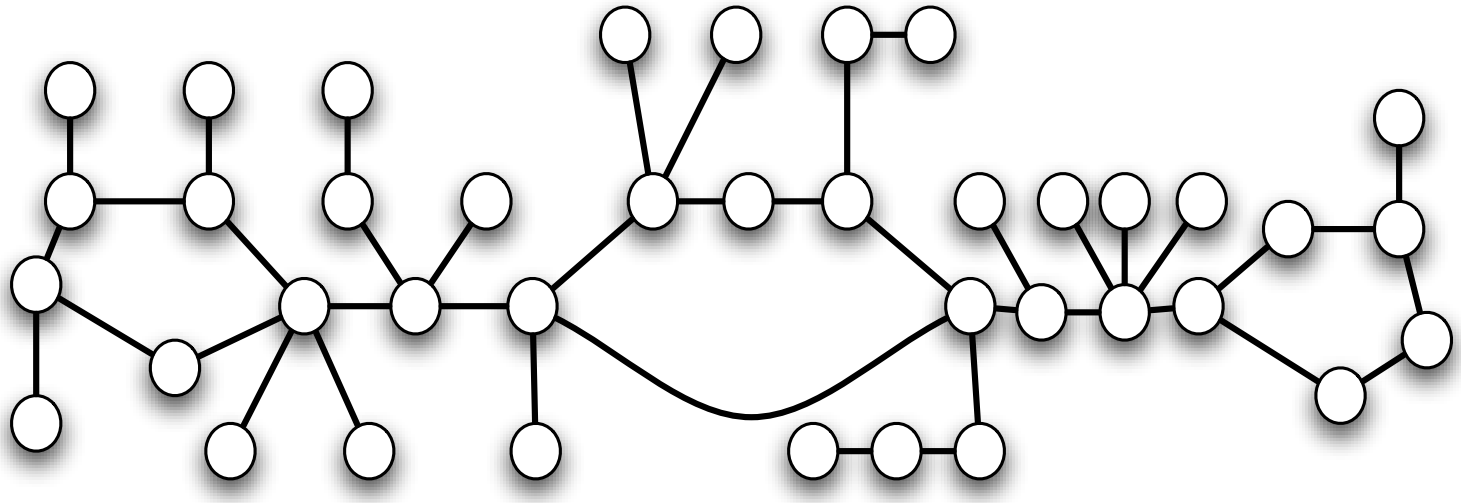
- We will use the cycle breaking algorithm to figure this out.
- First need to explain another description of a component of $G_{n,1/n}$, so we can better see the geometry (cycle structure).

The cycle structure of a connected graph.



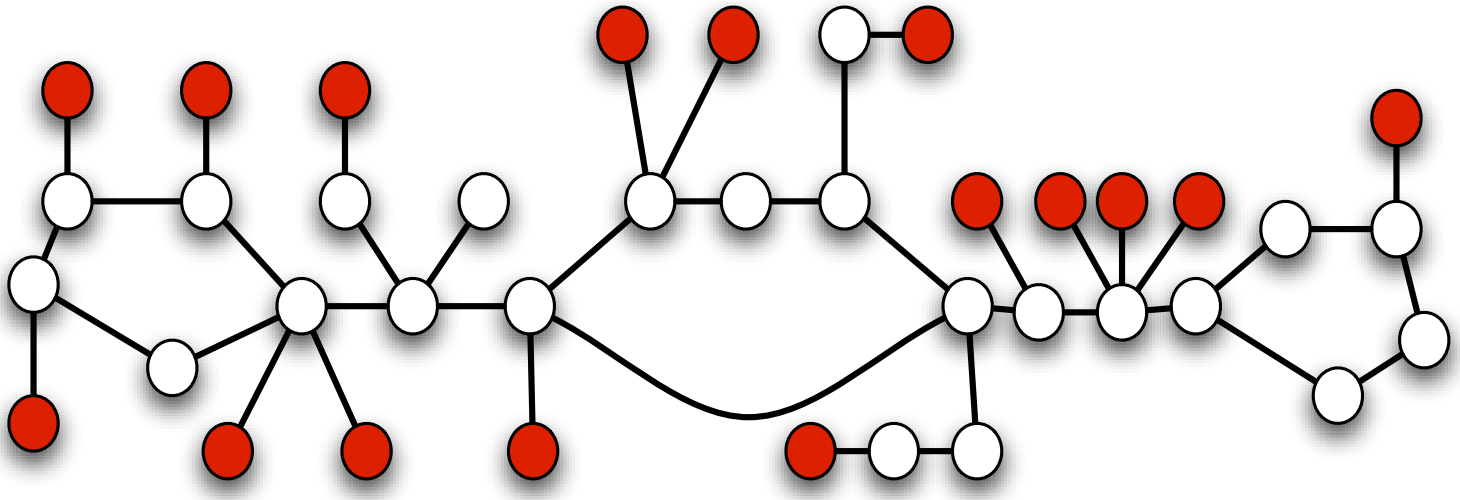
- Vertex labels suppressed so you can see what's going on.

The cycle structure of a connected graph.



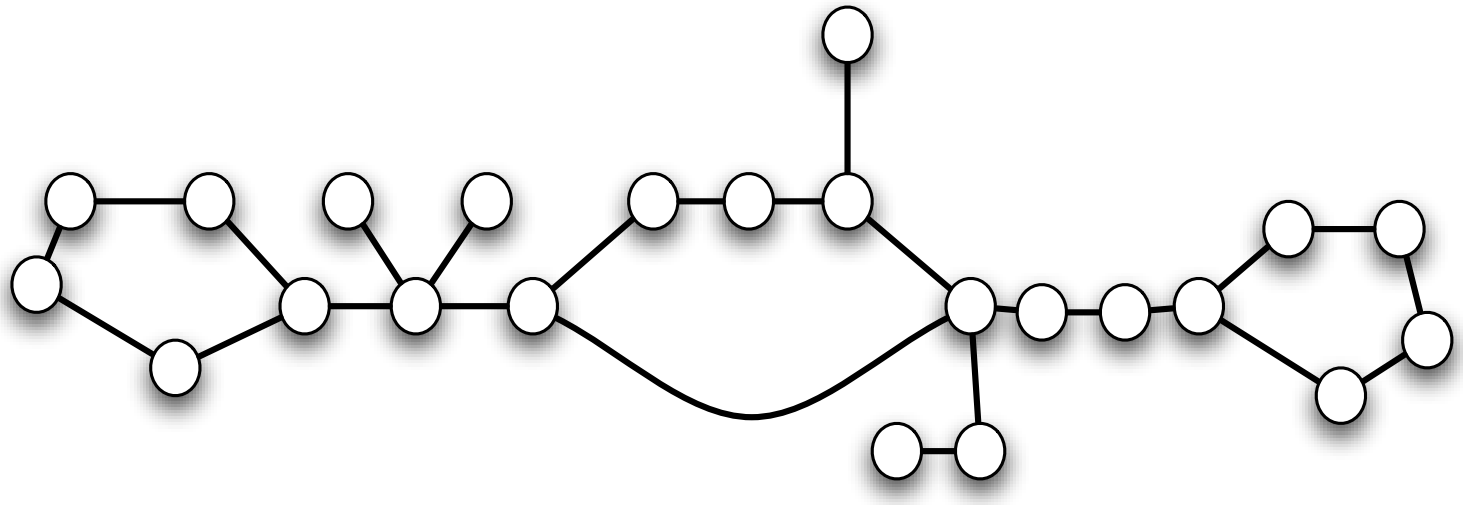
- Vertex labels suppressed so you can see what's going on.
- Leaves (nodes of degree 1) aren't in cycles, so don't affect the cycle structure – remove them!

The cycle structure of a connected graph.



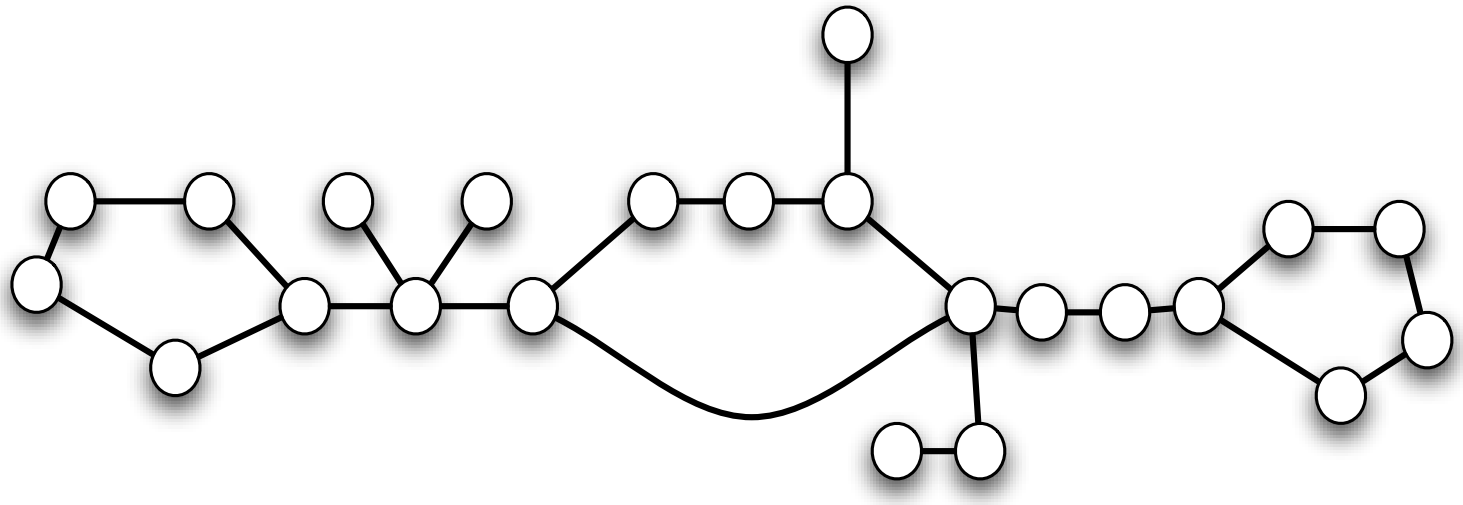
- Vertex labels suppressed so you can see what's going on.
- Leaves (nodes of degree 1) aren't in cycles, so don't affect the cycle structure – remove them!

The cycle structure of a connected graph.



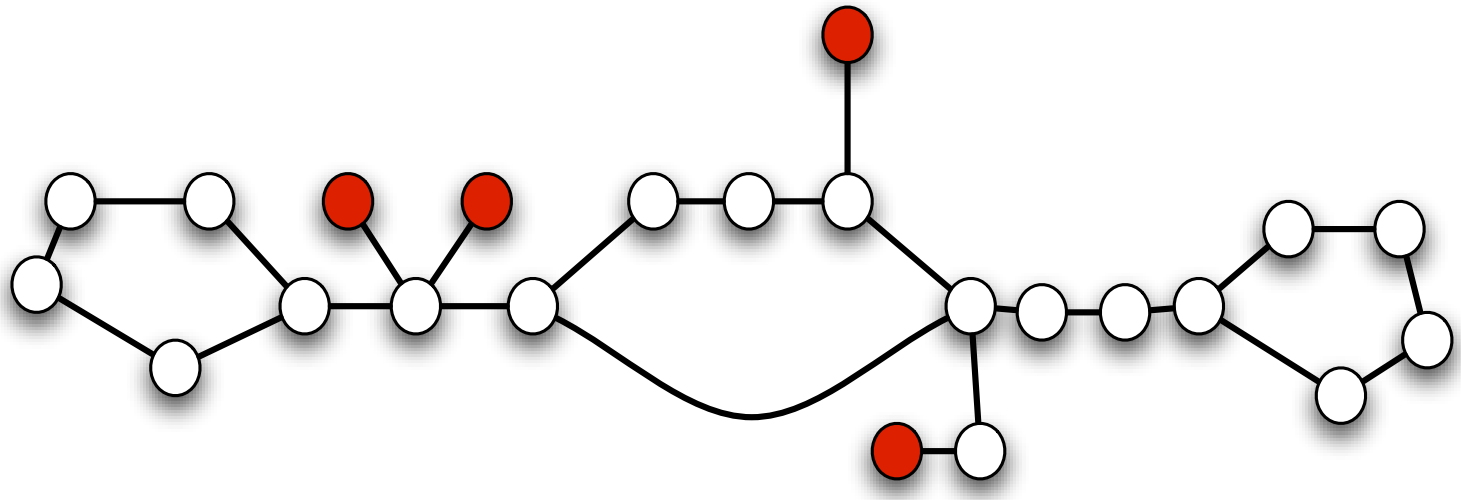
- Vertex labels suppressed so you can see what's going on.
- Leaves (nodes of degree 1) aren't in cycles, so don't affect the cycle structure – remove them!

The cycle structure of a connected graph.



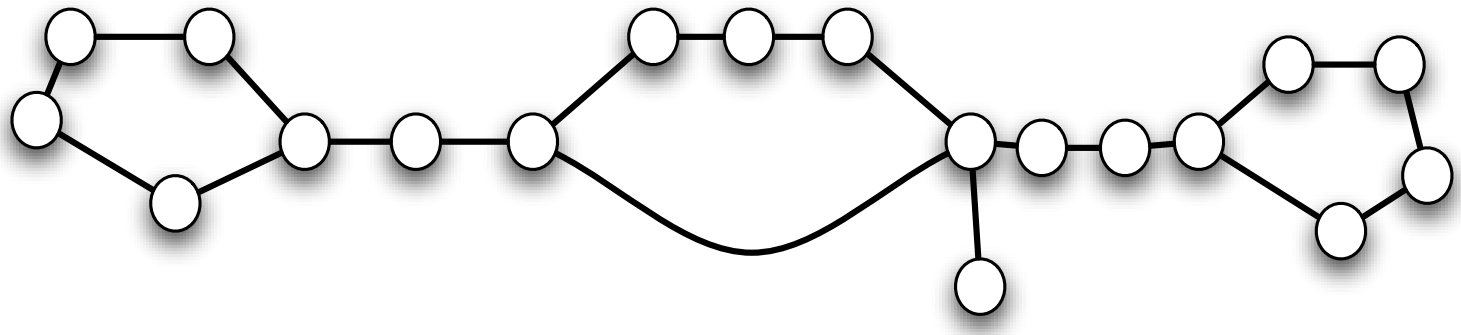
- Vertex labels suppressed so you can see what's going on.
- Leaves (nodes of degree 1) aren't in cycles, so don't affect the cycle structure – remove them!
- More leaves appeared!

The cycle structure of a connected graph.



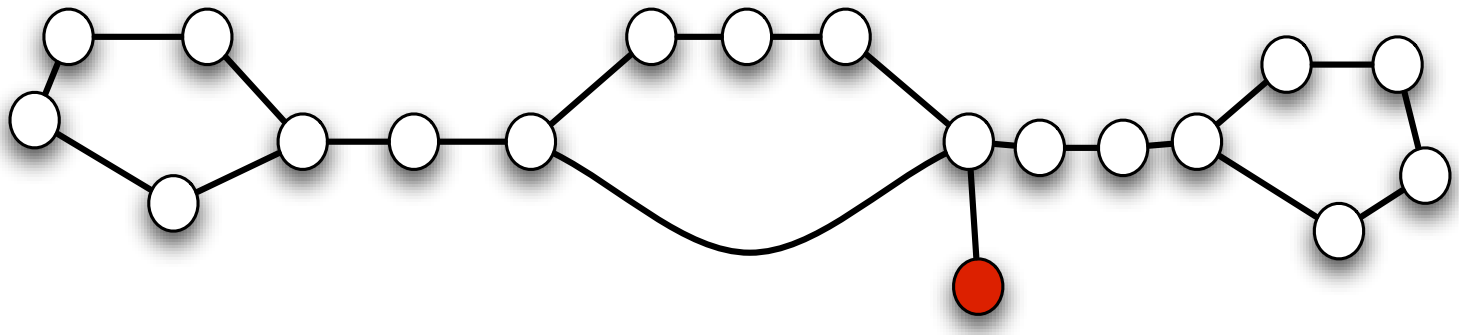
- Vertex labels suppressed so you can see what's going on.
- Leaves (nodes of degree 1) aren't in cycles, so don't affect the cycle structure – remove them!
- More leaves appeared!

The cycle structure of a connected graph.



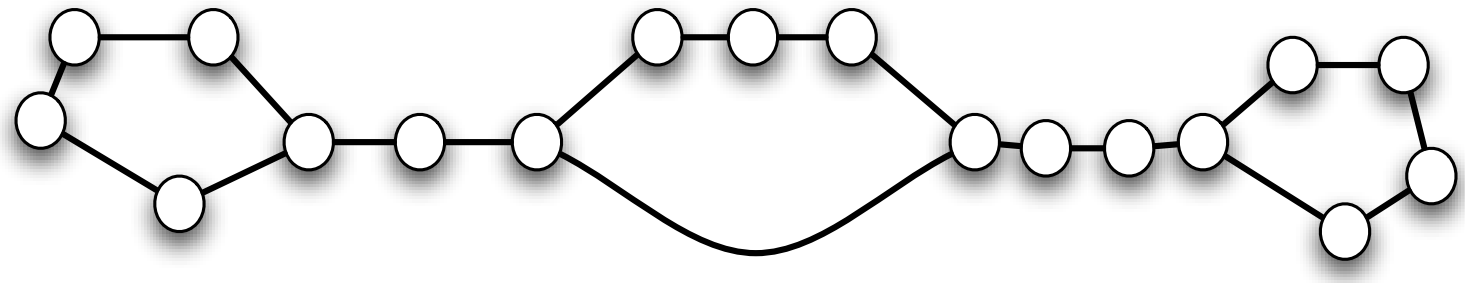
- Vertex labels suppressed so you can see what's going on.
- Leaves (nodes of degree 1) aren't in cycles, so don't affect the cycle structure – remove them!
- More leaves appeared!

The cycle structure of a connected graph.



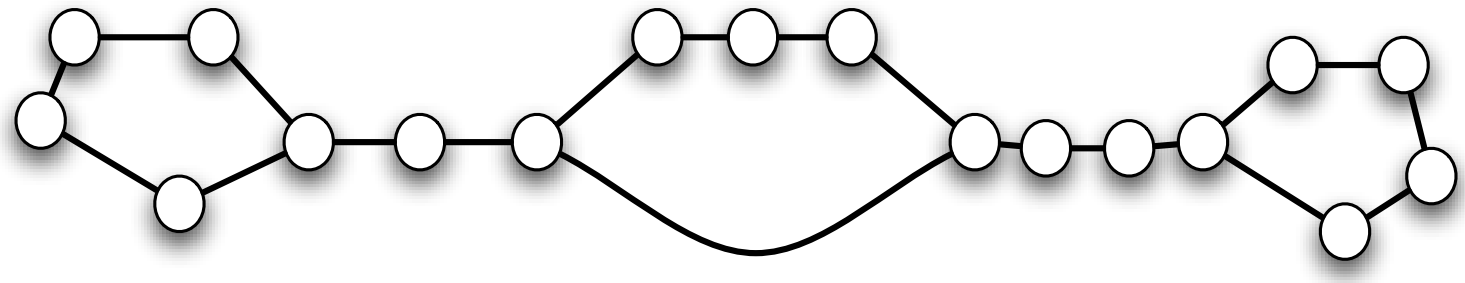
- Vertex labels suppressed so you can see what's going on.
- Leaves (nodes of degree 1) aren't in cycles, so don't affect the cycle structure – remove them!
- More leaves appeared!

The cycle structure of a connected graph.



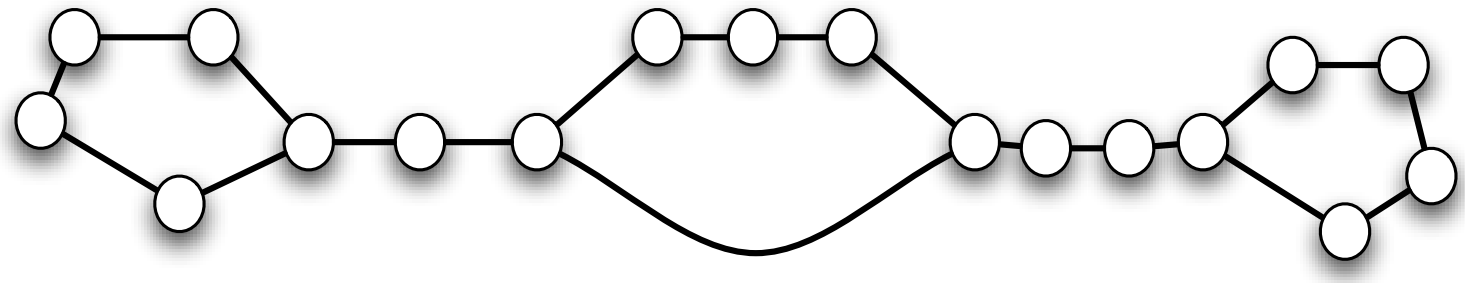
- Vertex labels suppressed so you can see what's going on.
- Leaves (nodes of degree 1) aren't in cycles, so don't affect the cycle structure – remove them!
- More leaves appeared!

The cycle structure of a connected graph.



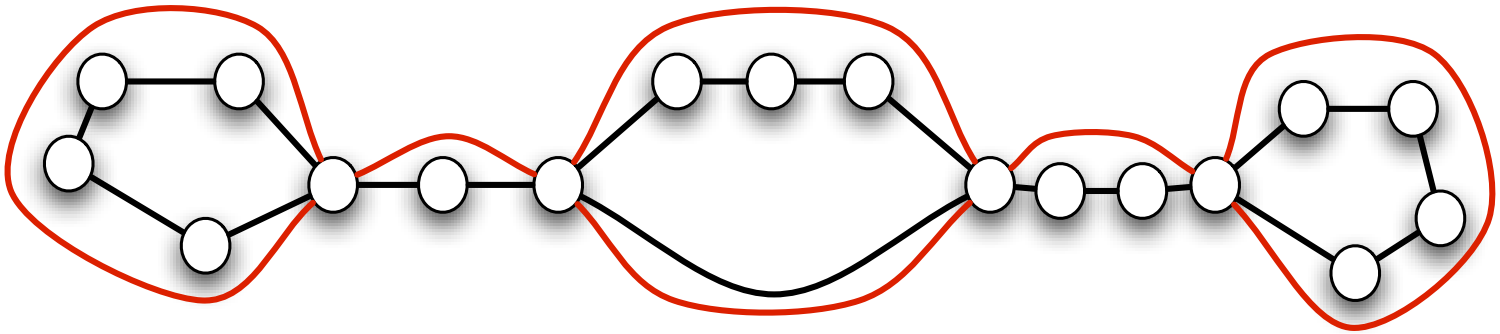
- Vertex labels suppressed so you can see what's going on.
- Leaves (nodes of degree 1) aren't in cycles, so don't affect the cycle structure – remove them!
- What's left when all leaves are gone is called the *core*.

The cycle structure of a connected graph.



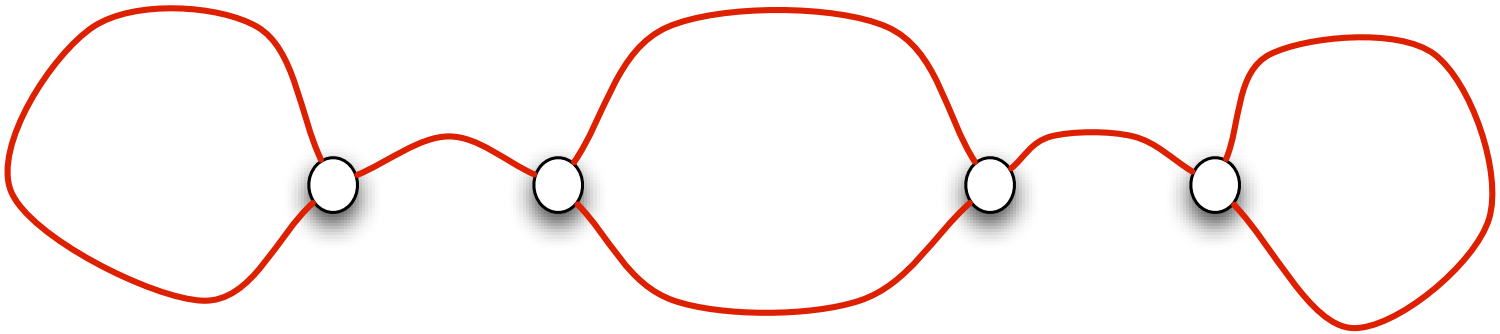
- Vertex labels suppressed so you can see what's going on.
- Leaves (nodes of degree 1) aren't in cycles, so don't affect the cycle structure – remove them!
- What's left when all leaves are gone is called the *core*.
- Vertices of degree 2 in whats left just form paths – suppress them!

The cycle structure of a connected graph.



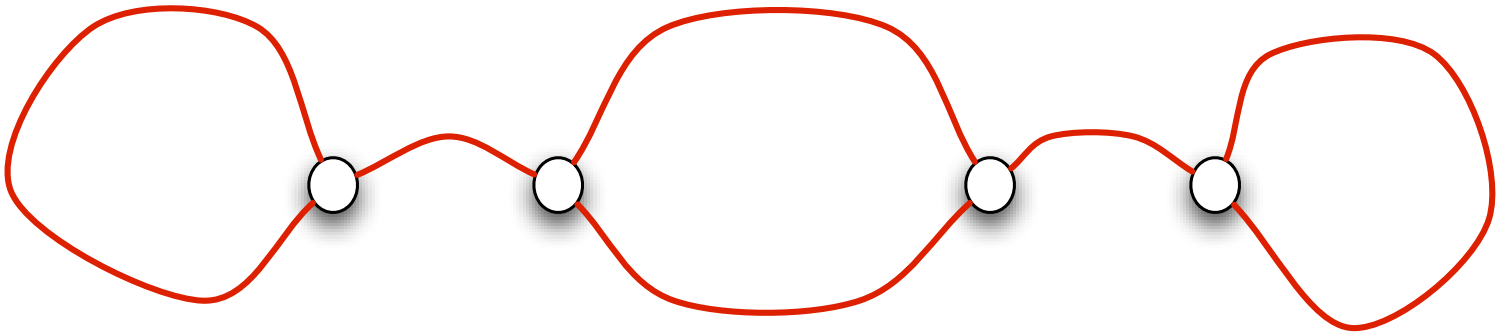
- Vertex labels suppressed so you can see what's going on.
- Leaves (nodes of degree 1) aren't in cycles, so don't affect the cycle structure – remove them!
- What's left when all leaves are gone is called the *core*.
- Vertices of degree 2 in whats left just form paths – suppress them!

The cycle structure of a connected graph.



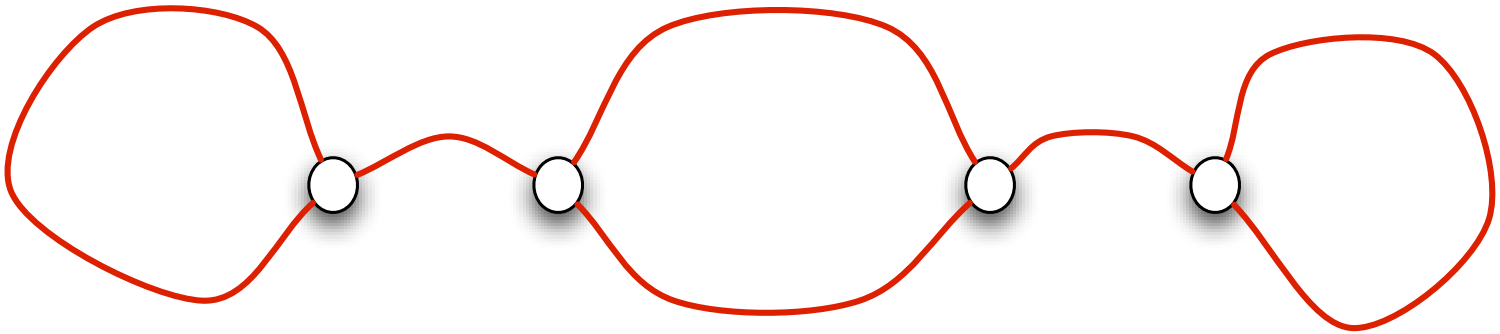
- Vertex labels suppressed so you can see what's going on.
- Leaves (nodes of degree 1) aren't in cycles, so don't affect the cycle structure – remove them!
- What's left when all leaves are gone is called the *core*.
- Vertices of degree 2 in whats left just form paths – suppress them!

The cycle structure of a connected graph.



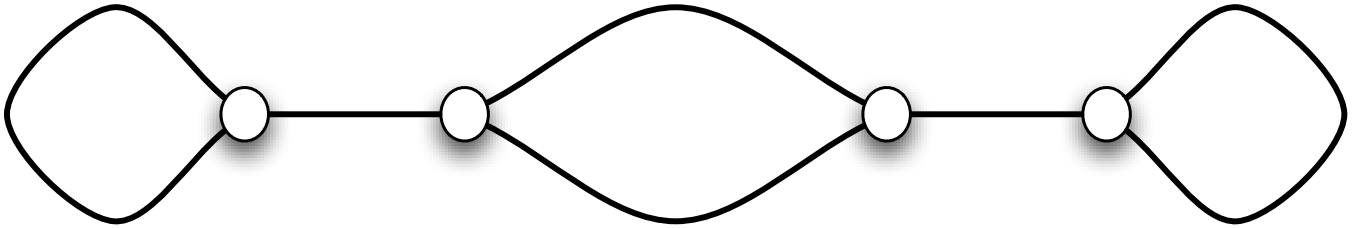
- Vertex labels suppressed so you can see what's going on.
- Leaves (nodes of degree 1) aren't in cycles, so don't affect the cycle structure – remove them!
- What's left when all leaves are gone is called the *core*.
- Vertices of degree 2 in whats left just form paths – suppress them!
- Result after suppressing degree-2 vertices is called the *kernel*.

The cycle structure of a connected graph.



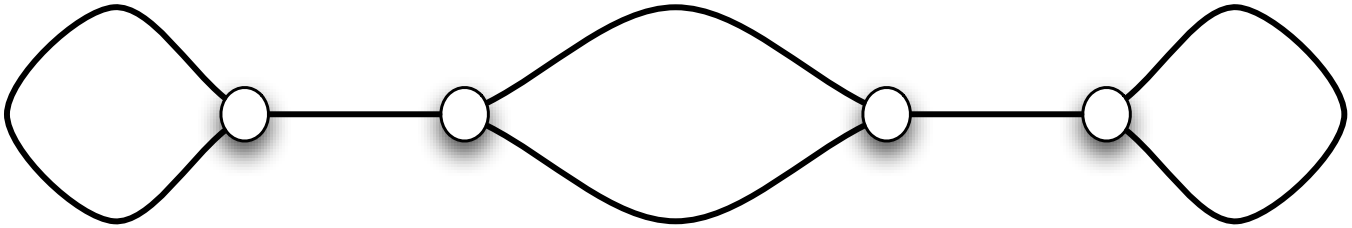
- Vertex labels suppressed so you can see what's going on.
- Leaves (nodes of degree 1) aren't in cycles, so don't affect the cycle structure – remove them!
- What's left when all leaves are gone is called the *core*.
- Vertices of degree 2 in whats left just form paths – suppress them!
- Result after suppressing degree-2 vertices is called the *kernel*.
- Kernel has same surplus as the original graph.

The kernel



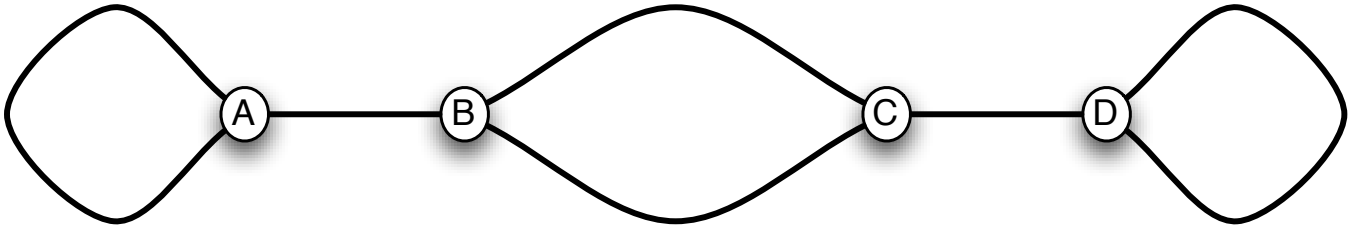
- The kernel is a connected, unlabelled multigraph with minimum degree 3.

The kernel



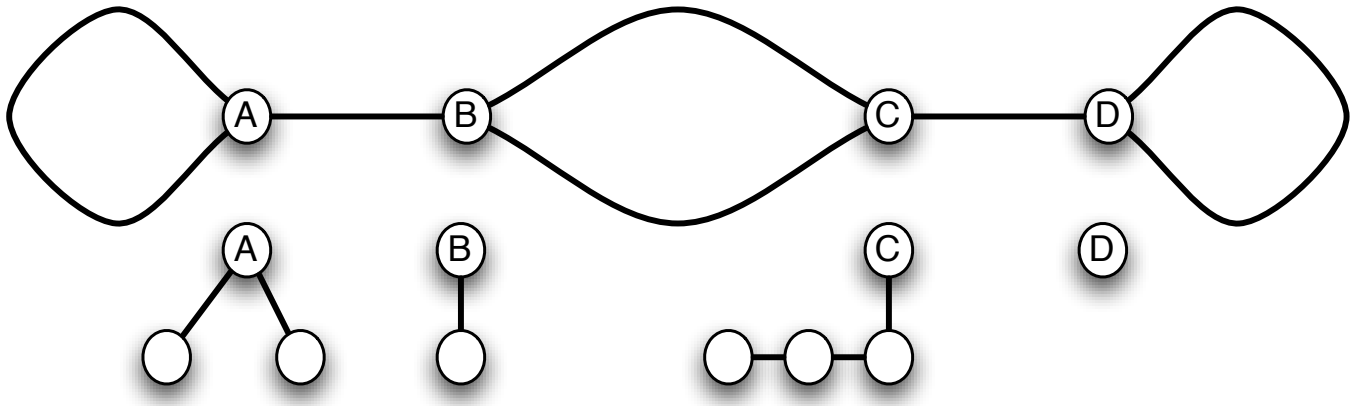
- The kernel is a connected, unlabelled multigraph with minimum degree 3.
- Given the kernel, we can reconstruct the graph by specifying a *marked tree* for each vertex of the kernel, and a *doubly-marked tree* for each edge of the kernel.

The kernel



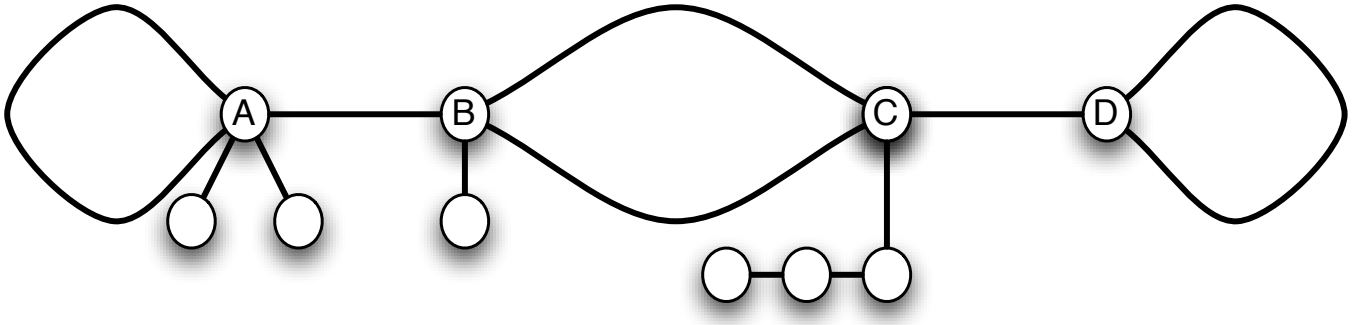
- The kernel is a connected, unlabelled multigraph with minimum degree 3.
- Given the kernel, we can reconstruct the graph by specifying a *marked tree* for each vertex of the kernel, and a *doubly-marked tree* for each edge of the kernel.

The kernel



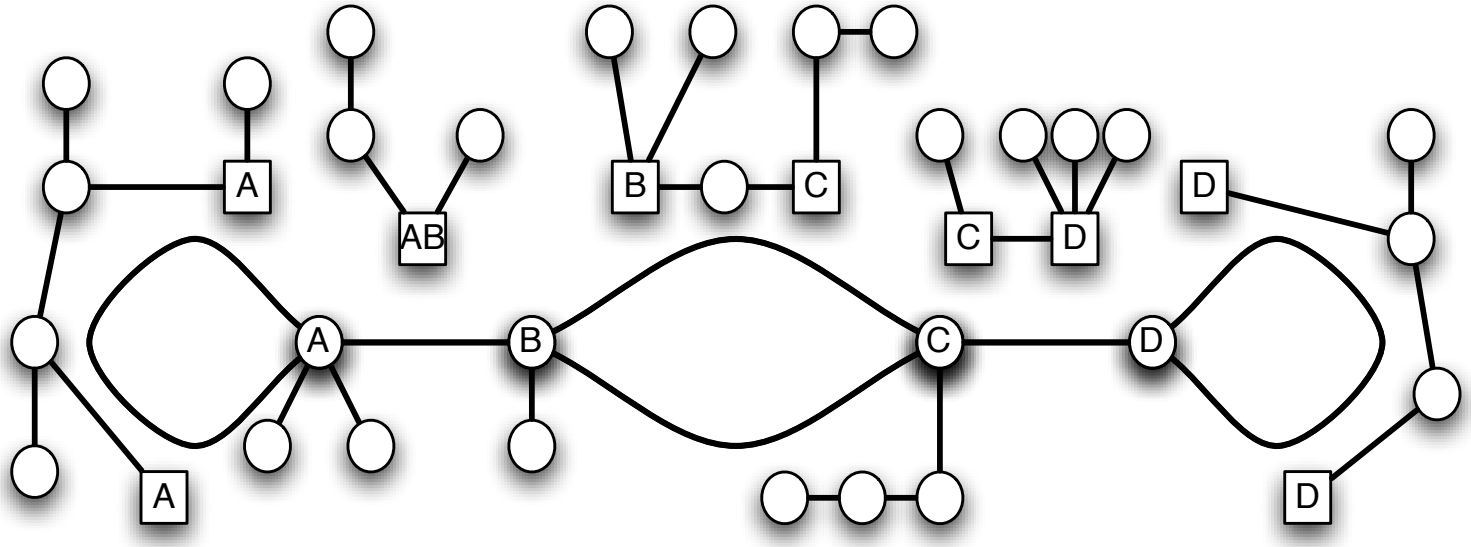
- The kernel is a connected, unlabelled multigraph with minimum degree 3.
- Given the kernel, we can reconstruct the graph by specifying a *marked tree* for each vertex of the kernel, and a *doubly-marked tree* for each edge of the kernel.

The kernel



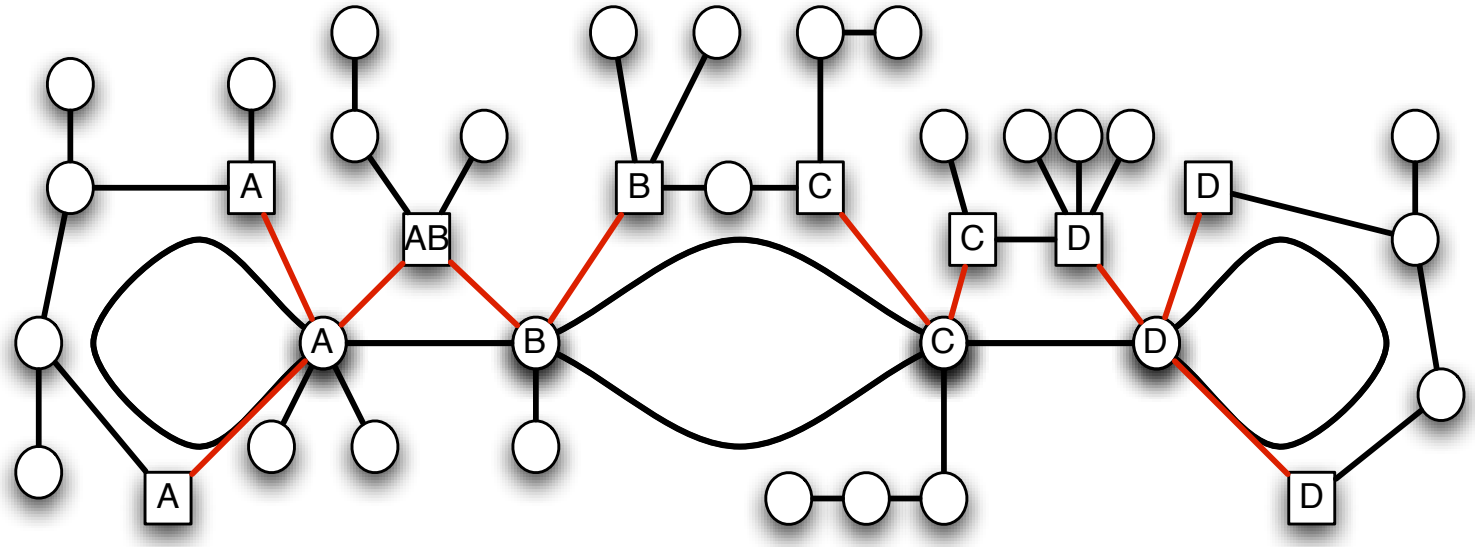
- The kernel is a connected, unlabelled multigraph with minimum degree 3.
- Given the kernel, we can reconstruct the graph by specifying a *marked tree* for each vertex of the kernel, and a *doubly-marked tree* for each edge of the kernel.

The kernel



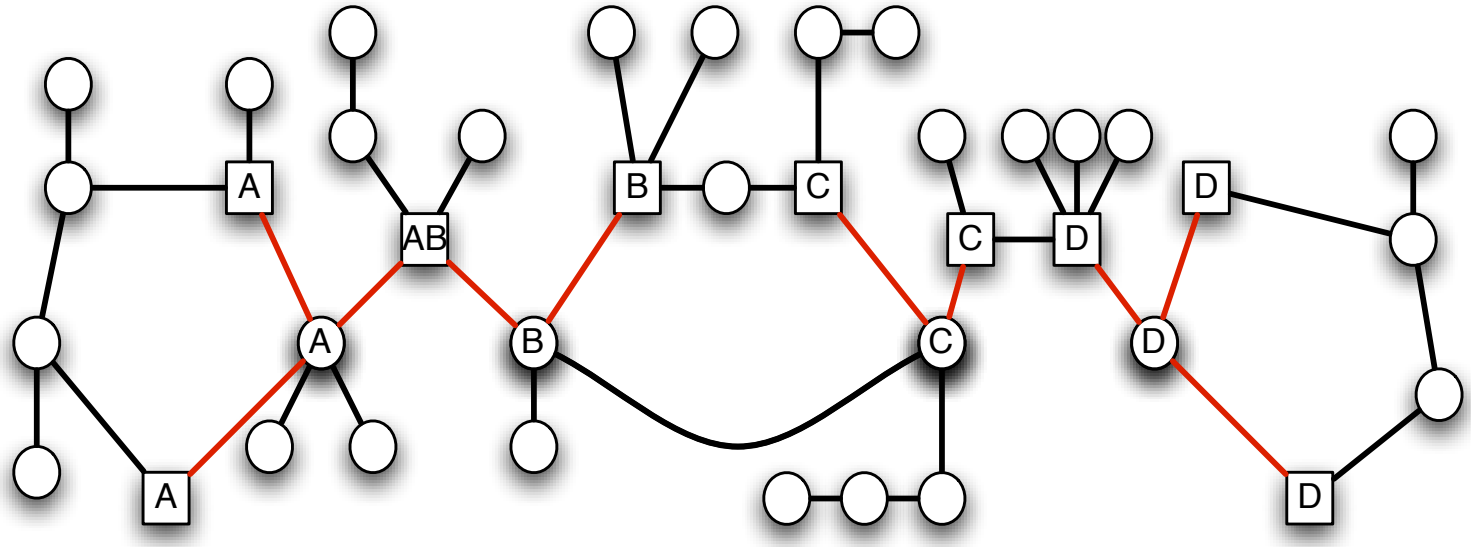
- The kernel is a connected, unlabelled multigraph with minimum degree 3.
- Given the kernel, we can reconstruct the graph by specifying a *marked tree* for each vertex of the kernel, and a *doubly-marked tree* for each edge of the kernel.

The kernel



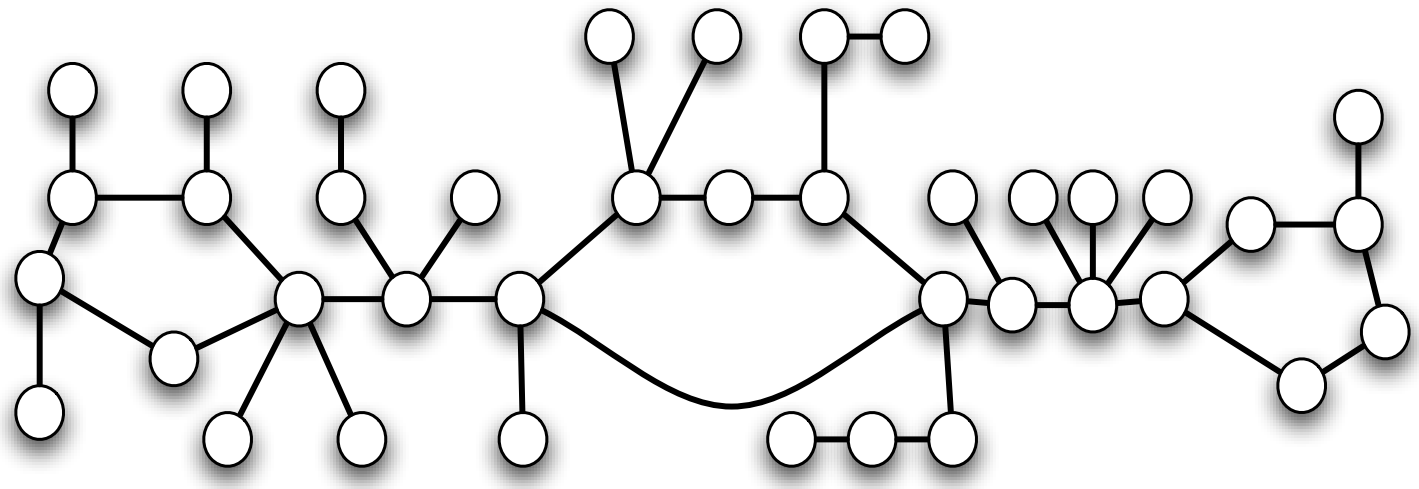
- The kernel is a connected, unlabelled multigraph with minimum degree 3.
- Given the kernel, we can reconstruct the graph by specifying a *marked tree* for each vertex of the kernel, and a *doubly-marked tree* for each edge of the kernel.

The kernel



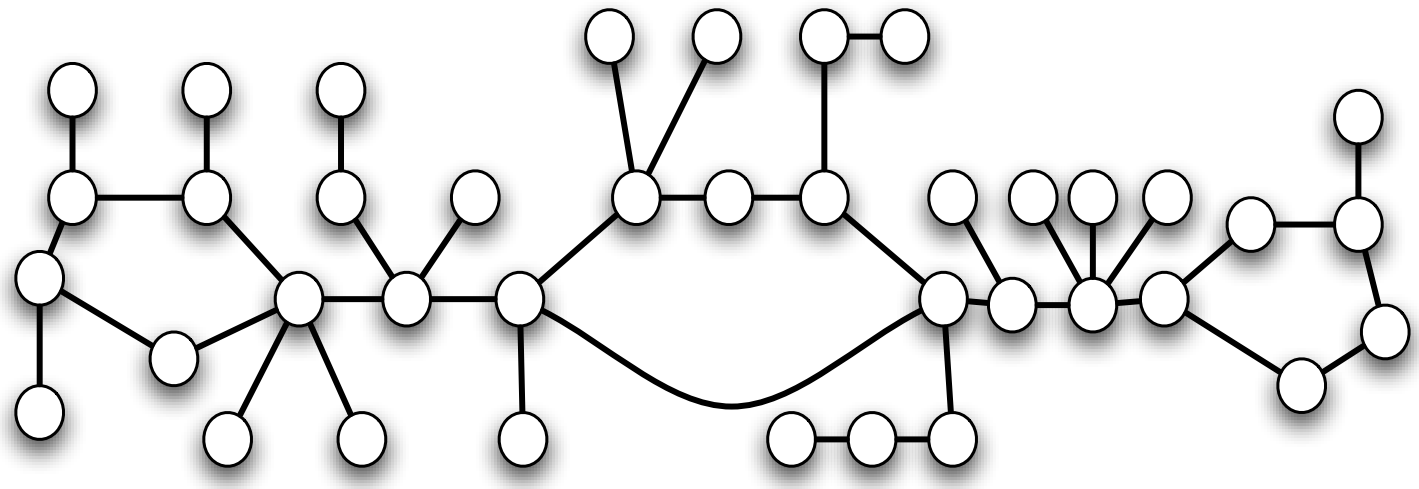
- The kernel is a connected, unlabelled multigraph with minimum degree 3.
- Given the kernel, we can reconstruct the graph by specifying a *marked tree* for each vertex of the kernel, and a *doubly-marked tree* for each edge of the kernel.

The kernel



- The kernel is a connected, unlabelled multigraph with minimum degree 3.
- Given the kernel, we can reconstruct the graph by specifying a *marked tree* for each vertex of the kernel, and a *doubly-marked tree* for each edge of the kernel.

The kernel



- The kernel is a connected, unlabelled multigraph with minimum degree 3.
- Given the kernel, we can reconstruct the graph by specifying a *marked tree* for each vertex of the kernel, and a *doubly-marked tree* for each edge of the kernel.
- Symmetry factors: $1/2$ for loops, $1/k!$ for each k-multiedge.

The kernel of a component

Consider a random connected graph with vertices $1, \ldots, m$ and surplus $s \geq 2$. For s fixed, m large, the structure is as follows.

The kernel of a component

Consider a random connected graph with vertices $1, \ldots, m$ and surplus $s \geq 2$. For s fixed, m large, the structure is as follows.

- With probability $1 - o(1)$ the kernel is 3-regular (all nodes degree three), so has $2(s-1)$ vertices and $3(s-1)$ edges.

The kernel of a component

Consider a random connected graph with vertices $1, \ldots, m$ and surplus $s \geq 2$. For s fixed, m large, the structure is as follows.

- With probability $1 - o(1)$ the kernel is 3-regular (all nodes degree three), so has $2(s-1)$ vertices and $3(s-1)$ edges.
- The probability of any given kernel is the same (up to the symmetry factors mentioned before).

The kernel of a component

Consider a random connected graph with vertices $1, \ldots, m$ and surplus $s \geq 2$. For s fixed, m large, the structure is as follows.

- With probability $1 - o(1)$ the kernel is 3-regular (all nodes degree three), so has $2(s-1)$ vertices and $3(s-1)$ edges.
- The probability of any given kernel is the same (up to the symmetry factors mentioned before).
- "Vertex trees" have $O(1)$ vertices in probability (they disappear in the limit).

The kernel of a component

Consider a random connected graph with vertices $1, \ldots, m$ and surplus $s \geq 2$. For s fixed, m large, the structure is as follows.

- With probability $1 - o(1)$ the kernel is 3-regular (all nodes degree three), so has $2(s-1)$ vertices and $3(s-1)$ edges.
- The probability of any given kernel is the same (up to the symmetry factors mentioned before).
- "Vertex trees" have $O(1)$ vertices in probability (they disappear in the limit).
- List the random "edge trees" as $T_1, \ldots, T_{3(s-1)}$; conditional on their sizes, these trees are *independent uniformly random trees*, so look like rescaled Brownian CRTs in the limit.

The kernel of a component

Consider a random connected graph with vertices $1, \ldots, m$ and surplus $s \geq 2$. For s fixed, m large, the structure is as follows.

- With probability $1 - o(1)$ the kernel is 3-regular (all nodes degree three), so has $2(s-1)$ vertices and $3(s-1)$ edges.
- The probability of any given kernel is the same (up to the symmetry factors mentioned before).
- "Vertex trees" have $O(1)$ vertices in probability (they disappear in the limit).
- List the random "edge trees" as $T_1, \ldots, T_{3(s-1)}$; conditional on their sizes, these trees are *independent uniformly random trees*, so look like rescaled Brownian CRTs in the limit.
- The vector $(|T_1|/m, \ldots, |T_{3(s-1)}|/m)$ has distributional limit Dirichlet$(1/2, \ldots, 1/2)$.

The limiting picture

In the limit, then, a random component can be built by gluing *randomly rescaled Brownian CRTs* along the edges of a random kernel.

How to find a tree in the forest

To find the distribution of MST of a limiting component, use *cycle breaking on the core*.

How to find a tree in the forest

To find the distribution of MST of a limiting component, use *cycle breaking on the core*.
Remark: In the limit, core = kernel with edge lengths, is a random metric space, and edge removal becomes point removal.

How to find a tree in the forest

To find the distribution of MST of a limiting component, use *cycle breaking on the core*.
Remark: In the limit, core = kernel with edge lengths, is a random metric space, and edge removal becomes point removal.
As before, random connected graph with vertices $1, \ldots, m$ and surplus $s \geq 2$.

How to find a tree in the forest

To find the distribution of MST of a limiting component, use *cycle breaking on the core*.
Remark: In the limit, core = kernel with edge lengths, is a random metric space, and edge removal becomes point removal.
As before, random connected graph with vertices $1, \ldots, m$ and surplus $s \geq 2$.

- Write C for the number of vertices in the core – then $C = O(\sqrt{m})$ in expectation and $C/\sqrt{m}$ has a non-degenerate limit.

How to find a tree in the forest

To find the distribution of MST of a limiting component, use *cycle breaking on the core*.
Remark: In the limit, core = kernel with edge lengths, is a random metric space, and edge removal becomes point removal.
As before, random connected graph with vertices $1, \ldots, m$ and surplus $s \geq 2$.

- Write C for the number of vertices in the core – then $C = O(\sqrt{m})$ in expectation and $C/\sqrt{m}$ has a non-degenerate limit.
- Write $P_1, \ldots, P_{3(s-1)}$ for the core paths corresponding to kernel edges – then $(|P_1|/C, \ldots, |P_{3(s-1)}/C)$ has distributional limit Dirichlet$(1, \ldots, 1)$.

How to find a tree in the forest

To find the distribution of MST of a limiting component, use *cycle breaking on the core*.
Remark: In the limit, core = kernel with edge lengths, is a random metric space, and edge removal becomes point removal.
As before, random connected graph with vertices $1, \ldots, m$ and surplus $s \geq 2$.

- Write C for the number of vertices in the core – then $C = O(\sqrt{m})$ in expectation and $C/\sqrt{m}$ has a non-degenerate limit.
- Write $P_1, \ldots, P_{3(s-1)}$ for the core paths corresponding to kernel edges – then $(|P_1|/C, \ldots, |P_{3(s-1)}/C)$ has distributional limit Dirichlet$(1, \ldots, 1)$.
- In other words, core path lengths are just given by splittings of an interval by uniforms.

Cycle breaking on the core

Choose the largest weight edge in the core, try to remove it.

Cycle breaking on the core

Choose the largest weight edge in the core, try to remove it.
This edge is equally likely to be any edge of the core.

Cycle breaking on the core

Choose the largest weight edge in the core, try to remove it.
This edge is equally likely to be any edge of the core.
In the limit, edge lengths shrink to zero and we're choosing a random point on the core.

Cycle breaking on the core

Choose the largest weight edge in the core, try to remove it.
This edge is equally likely to be any edge of the core.
In the limit, edge lengths shrink to zero and we're choosing a random point on the core.
This may or may not break a cycle.

Cycle breaking on the core

Choose the largest weight edge in the core, try to remove it.
This edge is equally likely to be any edge of the core.
In the limit, edge lengths shrink to zero and we're choosing a random point on the core.
This may or may not break a cycle.
But it certainly breaks an edge tree in two.

Cycle breaking on the core

Choose the largest weight edge in the core, try to remove it.
This edge is equally likely to be any edge of the core.
In the limit, edge lengths shrink to zero and we're choosing a random point on the core.
This may or may not break a cycle.
But it certainly breaks an edge tree in two.
Resulting trees are still (rescaled) Brownian CRTs!

Cycle breaking on the core

Choose the largest weight edge in the core, try to remove it.
This edge is equally likely to be any edge of the core.
In the limit, edge lengths shrink to zero and we're choosing a random point on the core.
This may or may not break a cycle.
But it certainly breaks an edge tree in two.
Resulting trees are still (rescaled) Brownian CRTs!
Vector of edge tree masses still Dirichlet$(1/2, \ldots, 1/2)$ with length increased by one.

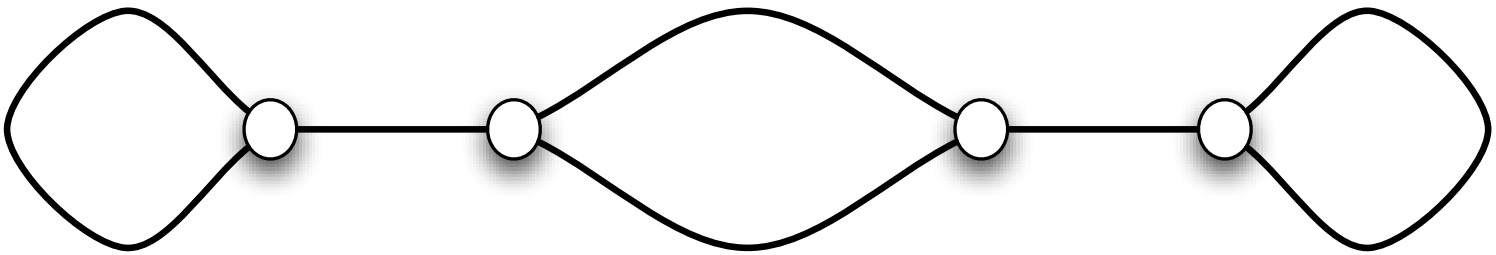
Cycle breaking on the core

Choose the largest weight edge in the core, try to remove it.
This edge is equally likely to be any edge of the core.
In the limit, edge lengths shrink to zero and we're choosing a random point on the core.
This may or may not break a cycle.
But it certainly breaks an edge tree in two.
Resulting trees are still (rescaled) Brownian CRTs!
Vector of edge tree masses still Dirichlet$(1/2, \ldots, 1/2)$ with length increased by one.
Vector of core lengths still Dirichlet$(1, \ldots, 1)$ with length increased by one.

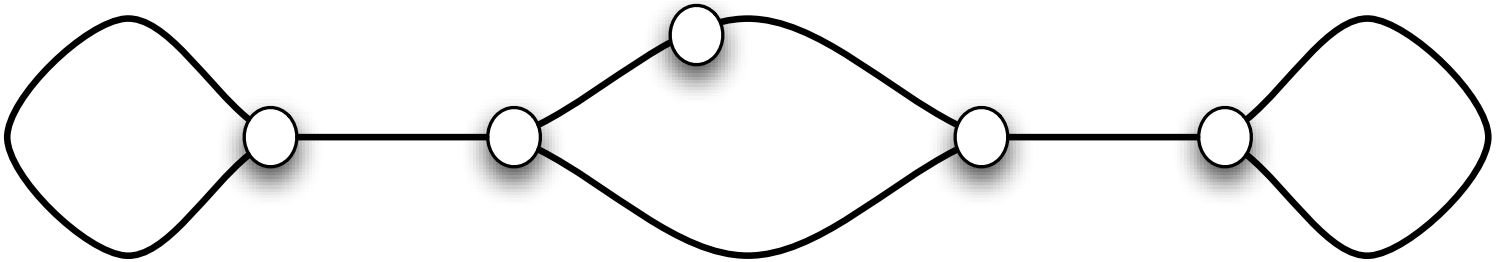
Cycle breaking on the core

Choose the largest weight edge in the core, try to remove it.
This edge is equally likely to be any edge of the core.
In the limit, edge lengths shrink to zero and we're choosing a random point on the core.
This may or may not break a cycle.
But it certainly breaks an edge tree in two.
Resulting trees are still (rescaled) Brownian CRTs!
Vector of edge tree masses still Dirichlet$(1/2, \ldots, 1/2)$ with length increased by one.
Vector of core lengths still Dirichlet$(1, \ldots, 1)$ with length increased by one.
Repeat until all cycles are broken.

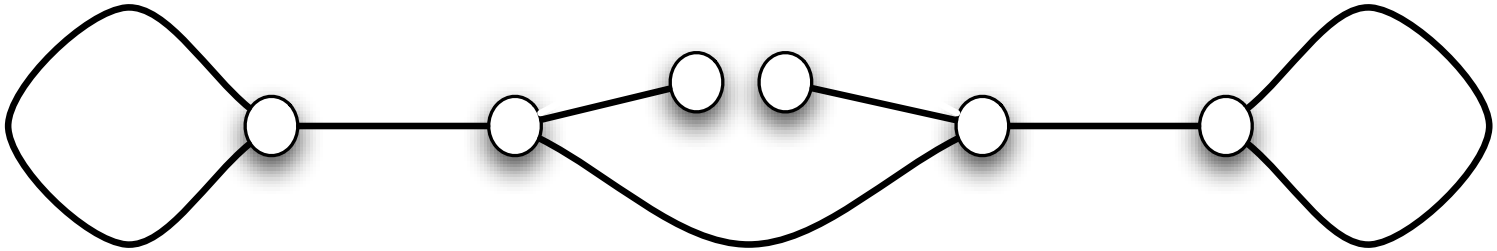
Cycle breaking on the core



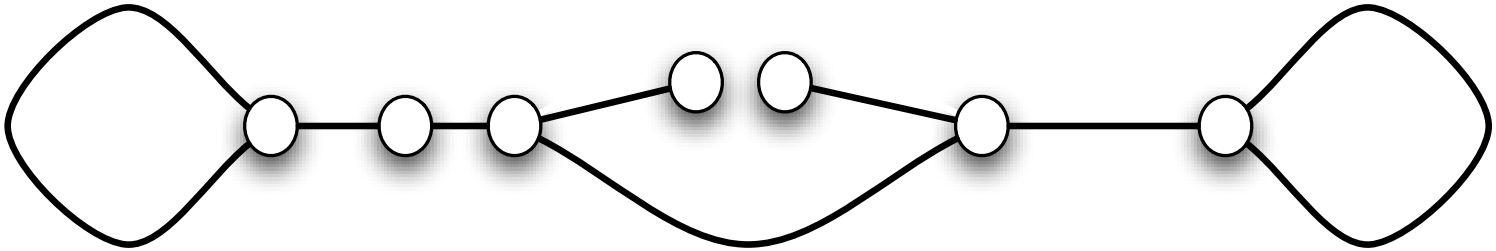
Cycle breaking on the core



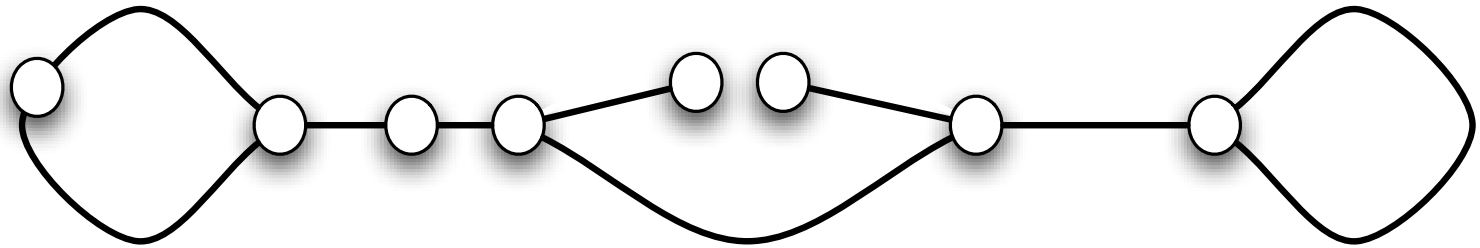
Cycle breaking on the core



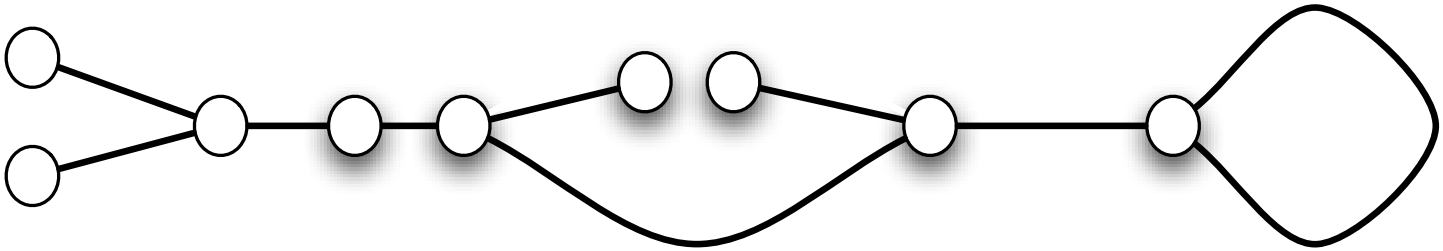
Cycle breaking on the core



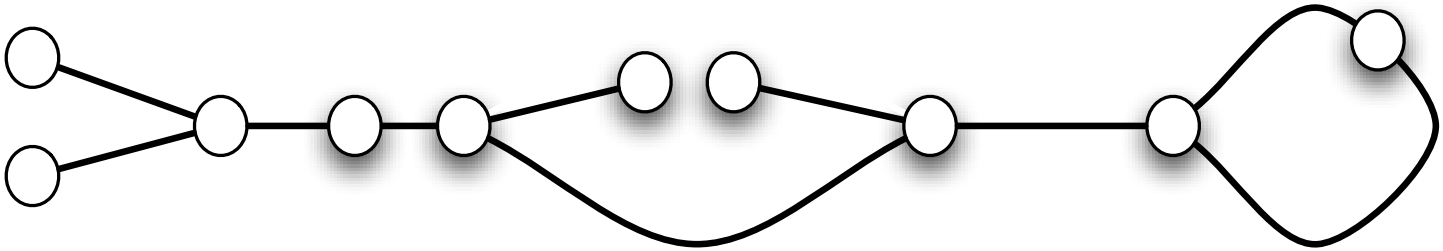
Cycle breaking on the core



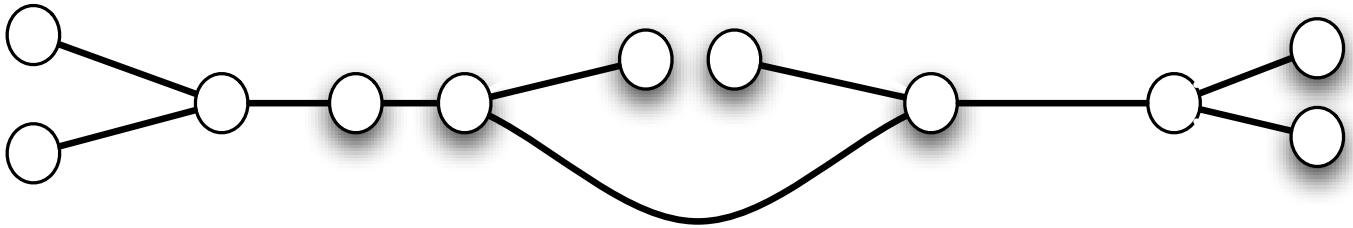
Cycle breaking on the core



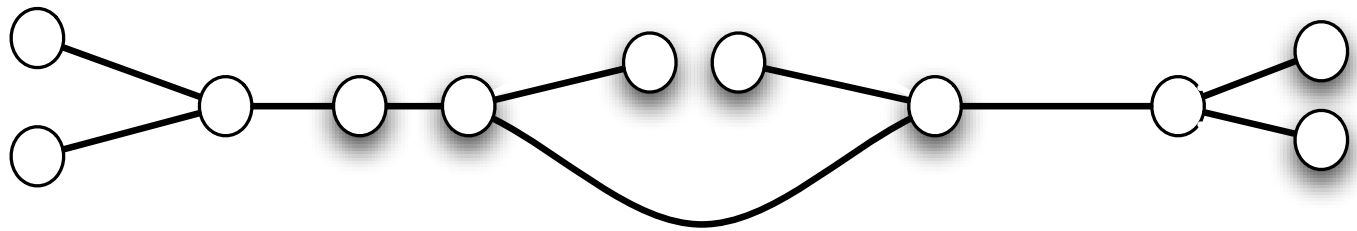
Cycle breaking on the core



Cycle breaking on the core

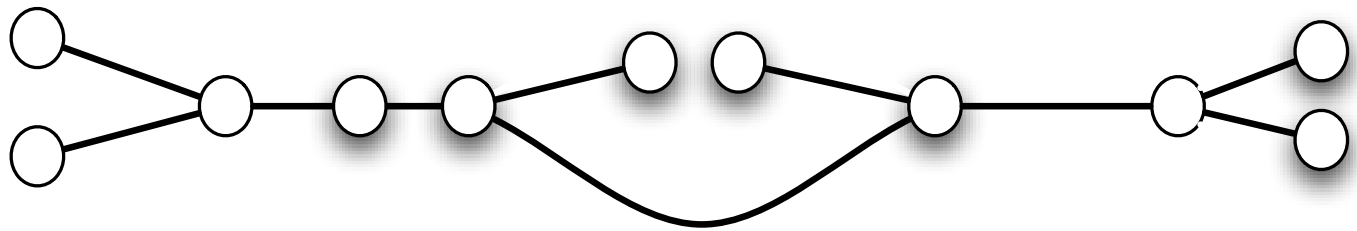


Cycle breaking on the core



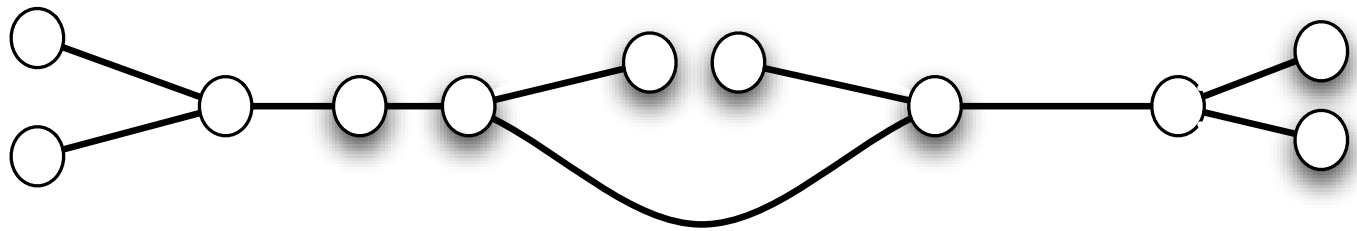
Result: MST of a random connected graph with a fixed surplus s, can be built (in the limit) by gluing randomly rescaled CRTs along the edges of a random discrete tree

Cycle breaking on the core



Result: MST of a random connected graph with a fixed surplus s, can be built (in the limit) by gluing randomly rescaled CRTs along the edges of a random discrete tree whose internal nodes all have degrees 2 or 3.

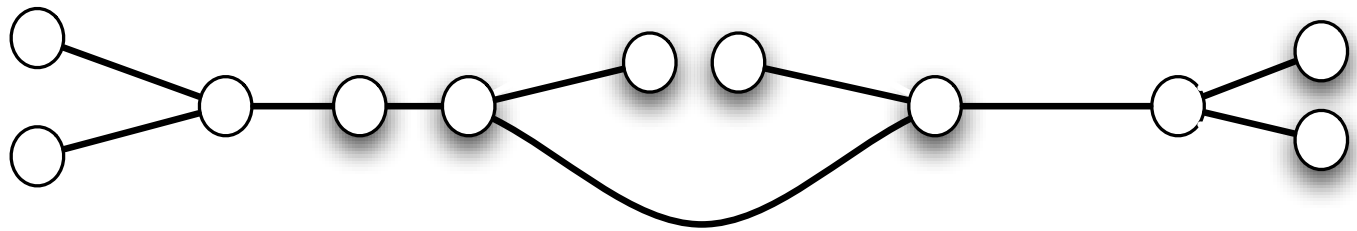
Cycle breaking on the core



Result: MST of a random connected graph with a fixed surplus s, can be built (in the limit) by gluing randomly rescaled CRTs along the edges of a random discrete tree whose internal nodes all have degrees 2 or 3.

We have not found a nice formula for the distribution of this discrete tree.

Cycle breaking on the core



Result: MST of a random connected graph with a fixed surplus s, can be built (in the limit) by gluing randomly rescaled CRTs along the edges of a random discrete tree whose internal nodes all have degrees 2 or 3.

We have not found a nice formula for the distribution of this discrete tree.

In the $s \to \infty$ limit, it must have height of order $s^{1/3}$, because in the $s \to \infty$ limit it encodes the structure of the MST of K_n, and this is known to have order $n^{1/3}$.

Putting it together

- This gives a GHP limit for $T_{n,1/n}$, which is a sequence of random real trees (measured metric spaces).

Putting it together

- This gives a GHP limit for $T_{n,1/n}$, which is a sequence of random real trees (measured metric spaces).
- Running the multiplicative coalescent on this sequence, as $t \to \infty$ all elements of the sequence disappear except the largest.

Putting it together

- This gives a GHP limit for $T_{n,1/n}$, which is a sequence of random real trees (measured metric spaces).
- Running the multiplicative coalescent on this sequence, as $t \to \infty$ all elements of the sequence disappear except the largest.
- The largest has mass tending to infinity but its diameter stays bounded.

Putting it together

- This gives a GHP limit for $T_{n,1/n}$, which is a sequence of random real trees (measured metric spaces).
- Running the multiplicative coalescent on this sequence, as $t \to \infty$ all elements of the sequence disappear except the largest.
- The largest has mass tending to infinity but its diameter stays bounded.
- The limiting object is the distributional GH limit of the MST of K_n (with distances rescaled by $n^{1/3}$). (*)


COMPLETELY
DIFFERENT

Universality

- The limiting sequence of metric spaces and the limiting MST tree are candidates for describing the limiting behaviour of several models.

Universality

- The limiting sequence of metric spaces and the limiting MST tree are candidates for describing the limiting behaviour of several models.
- High-dimensional percolation on a torus.

Universality

- The limiting sequence of metric spaces and the limiting MST tree are candidates for describing the limiting behaviour of several models.
- High-dimensional percolation on a torus. Too hard, for now!

Universality

- The limiting sequence of metric spaces and the limiting MST tree are candidates for describing the limiting behaviour of several models.
- High-dimensional percolation on a torus. Too hard, for now!
- Percolation on the hypercube.

Universality

- The limiting sequence of metric spaces and the limiting MST tree are candidates for describing the limiting behaviour of several models.
- High-dimensional percolation on a torus. Too hard, for now!
- Percolation on the hypercube. Also too hard!

Universality

- The limiting sequence of metric spaces and the limiting MST tree are candidates for describing the limiting behaviour of several models.
- High-dimensional percolation on a torus. Too hard, for now!
- Percolation on the hypercube. Also too hard!
- For now models with substantial local geometry seem hard to treat at this level of detail.

Universality

- The limiting sequence of metric spaces and the limiting MST tree are candidates for describing the limiting behaviour of several models.
- High-dimensional percolation on a torus. Too hard, for now!
- Percolation on the hypercube. Also too hard!
- For now models with substantial local geometry seem hard to treat at this level of detail.
- One model we think we can handle: random graphs with a given degree sequence.

Random graphs with a given degree sequence

- Fix a degree sequence $\mathbf{d} = (d_1, \ldots, d_n)$.

Random graphs with a given degree sequence

- Fix a degree sequence $\mathbf{d} = (d_1, \ldots, d_n)$.
- $G_{\mathbf{d}}$ uniformly random among simple graphs on $\{1, \ldots, n\}$ with node i having degree d_i.

Random graphs with a given degree sequence

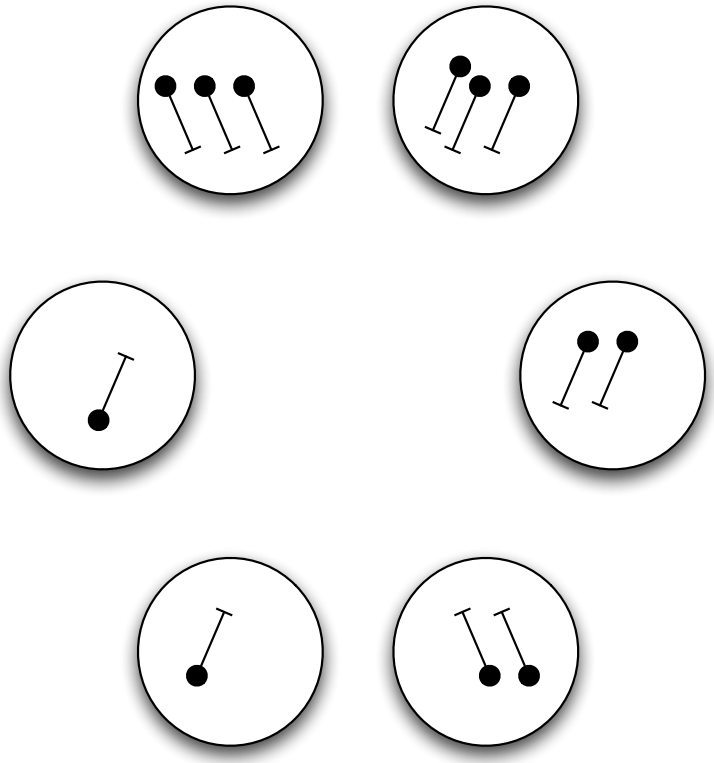
- Fix a degree sequence $\mathbf{d} = (d_1, \ldots, d_n)$.
- $G_{\mathbf{d}}$ uniformly random among simple graphs on $\{1, \ldots, n\}$ with node i having degree d_i.
- How to generate such a graph?

Configuration model

- Example: degrees $(3, 3, 2, 2, 1, 1)$.

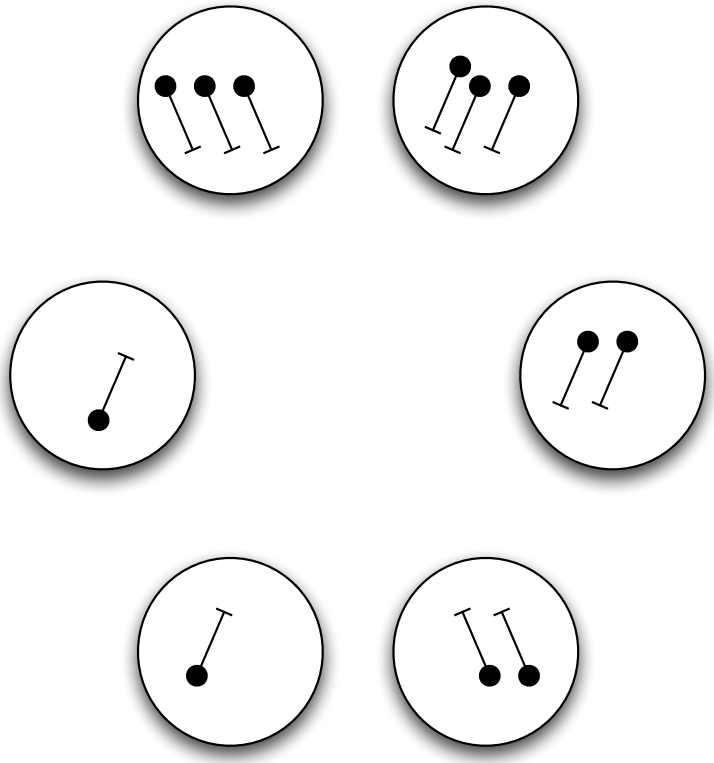
Configuration model

- Example: degrees $(3, 3, 2, 2, 1, 1)$.
- For each vertex i, make a bag with d_i half-edges.



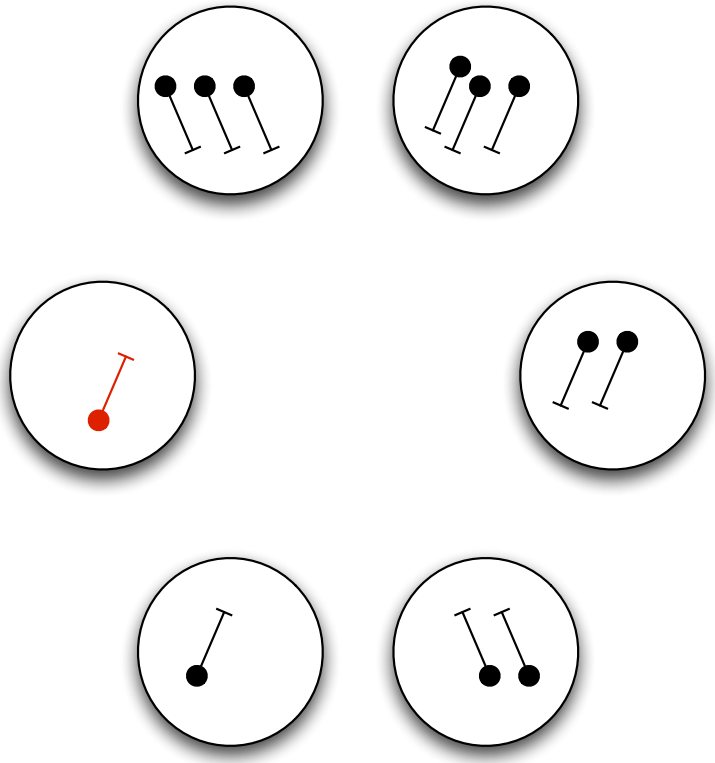
Configuration model

- Example: degrees $(3, 3, 2, 2, 1, 1)$.
- For each vertex i, make a bag with d_i half-edges.
- Pair half-edges uniformly at random.



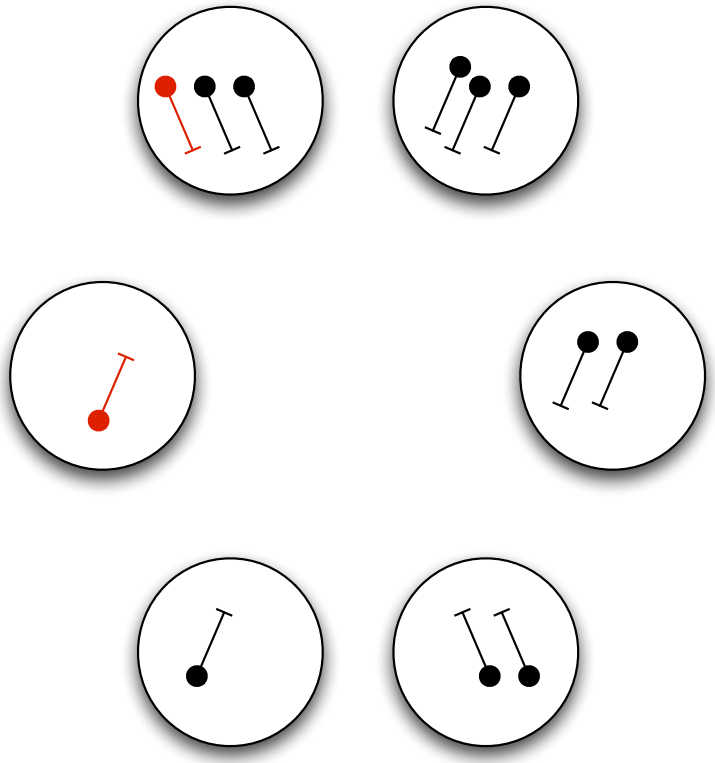
Configuration model

- Example: degrees $(3, 3, 2, 2, 1, 1)$.
- For each vertex i, make a bag with d_i half-edges.
- Pair half-edges uniformly at random.



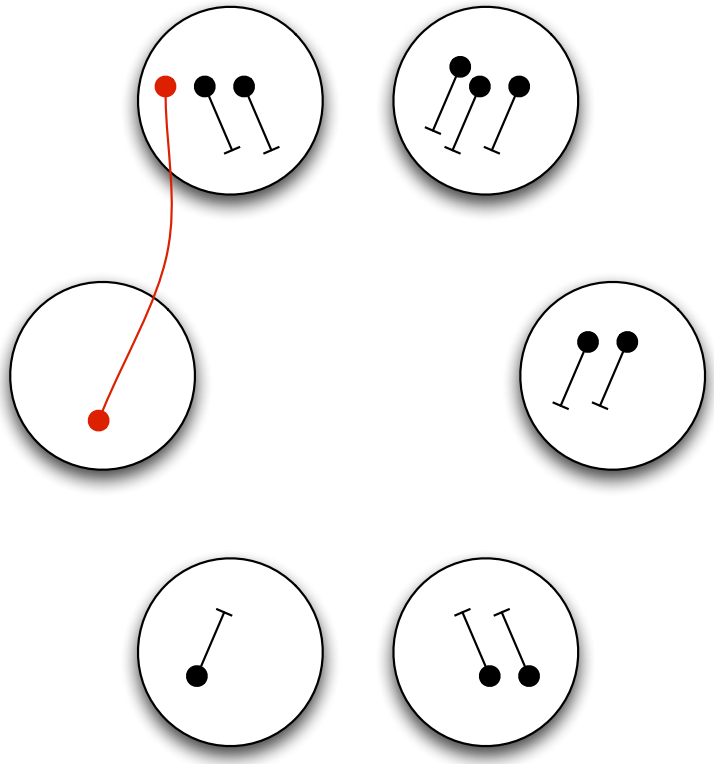
Configuration model

- Example: degrees $(3, 3, 2, 2, 1, 1)$.
- For each vertex i, make a bag with d_i half-edges.
- Pair half-edges uniformly at random.



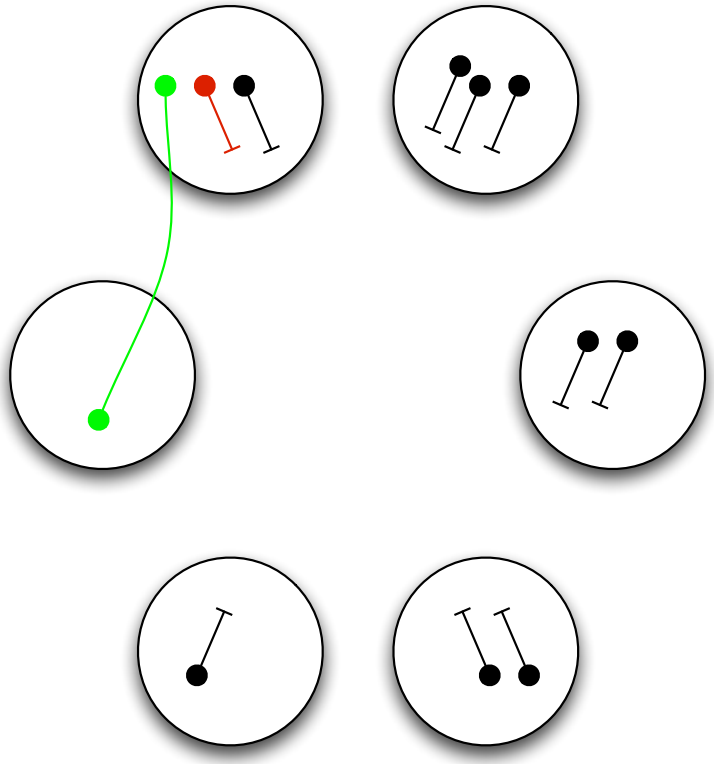
Configuration model

- Example: degrees $(3, 3, 2, 2, 1, 1)$.
- For each vertex i, make a bag with d_i half-edges.
- Pair half-edges uniformly at random.



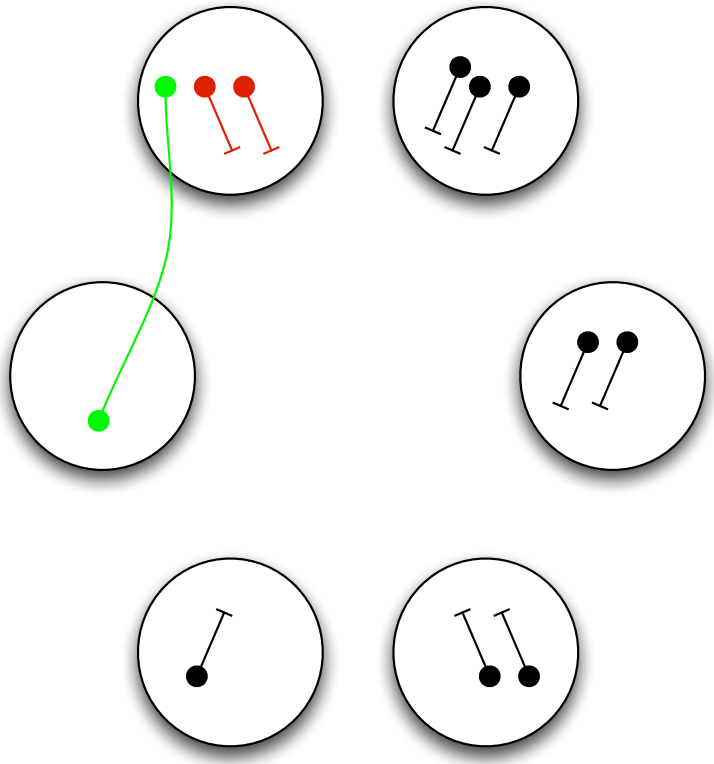
Configuration model

- Example: degrees $(3, 3, 2, 2, 1, 1)$.
- For each vertex i, make a bag with d_i half-edges.
- Pair half-edges uniformly at random.



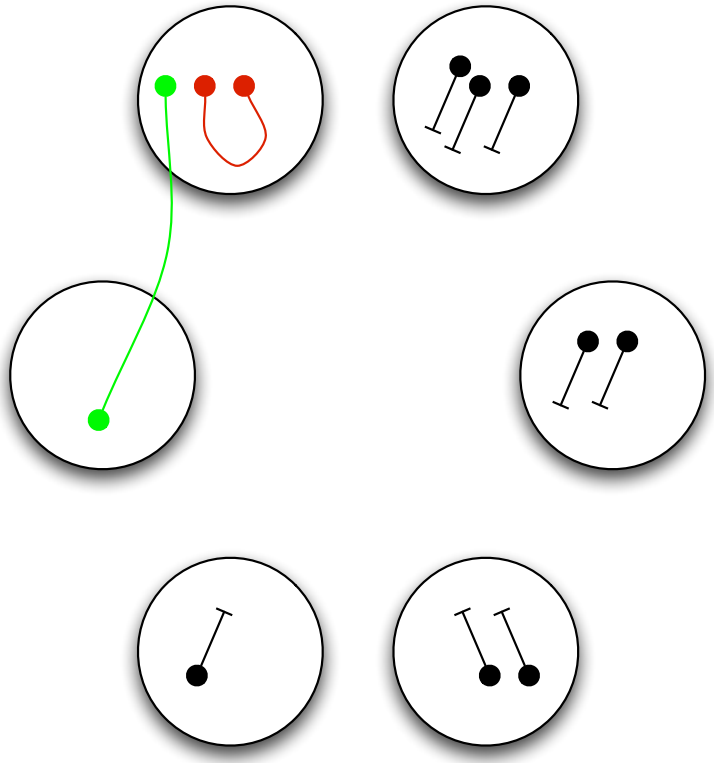
Configuration model

- Example: degrees $(3, 3, 2, 2, 1, 1)$.
- For each vertex i, make a bag with d_i half-edges.
- Pair half-edges uniformly at random.



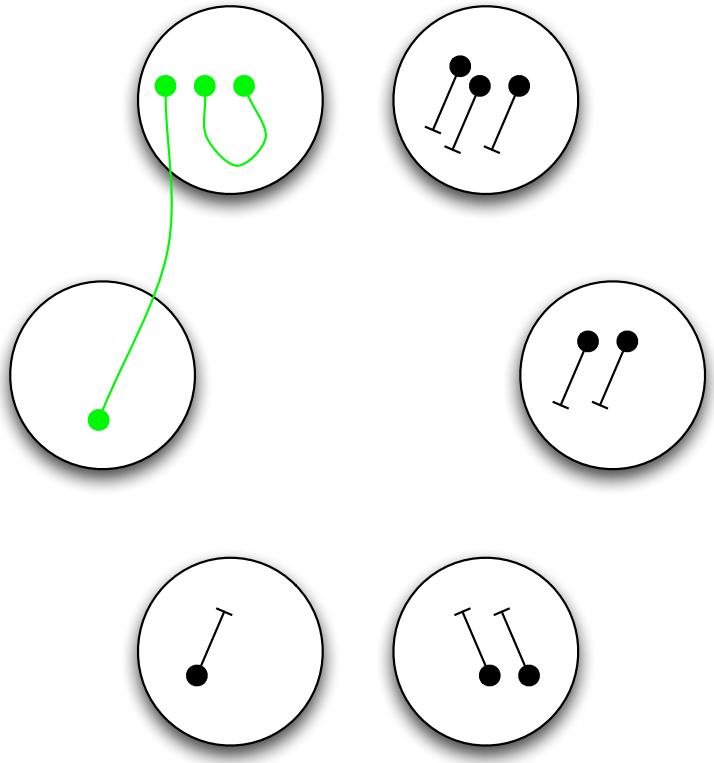
Configuration model

- Example: degrees $(3, 3, 2, 2, 1, 1)$.
- For each vertex i, make a bag with d_i half-edges.
- Pair half-edges uniformly at random.



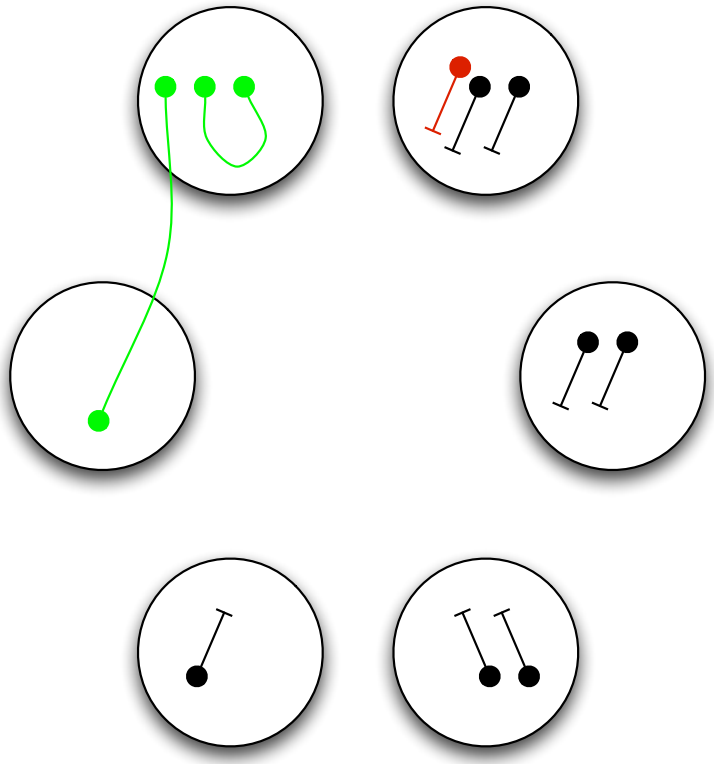
Configuration model

- Example: degrees $(3, 3, 2, 2, 1, 1)$.
- For each vertex i, make a bag with d_i half-edges.
- Pair half-edges uniformly at random.



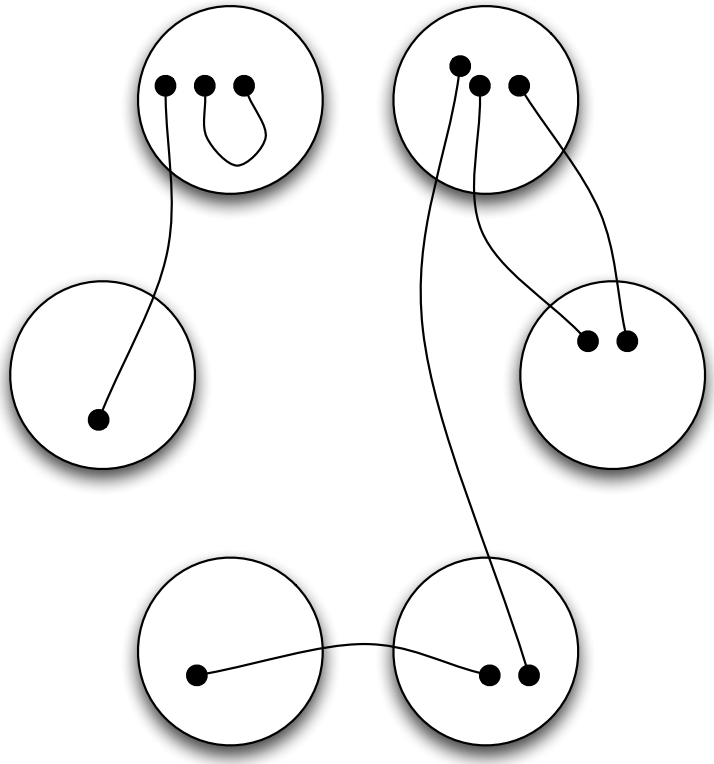
Configuration model

- Example: degrees $(3, 3, 2, 2, 1, 1)$.
- For each vertex i, make a bag with d_i half-edges.
- Pair half-edges uniformly at random.



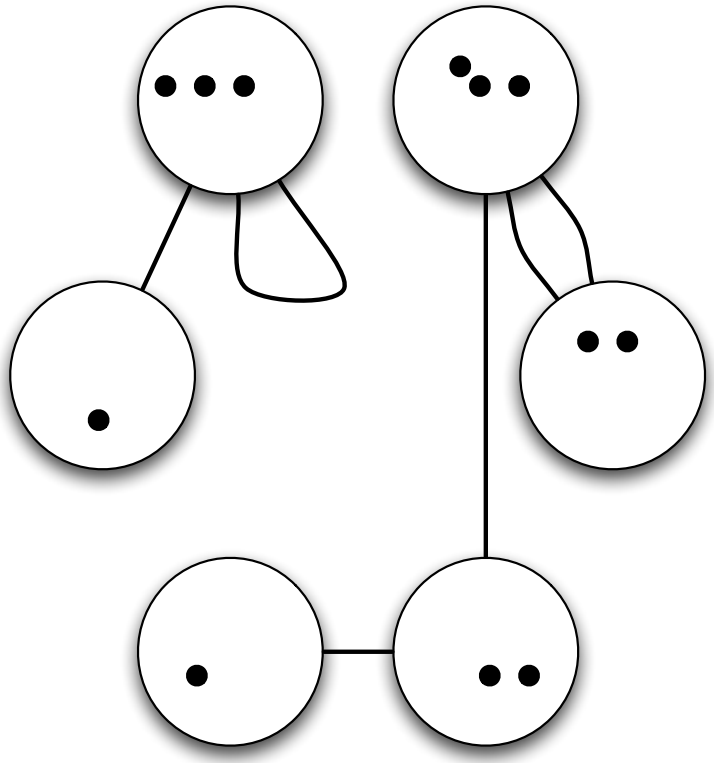
Configuration model

- Example: degrees $(3, 3, 2, 2, 1, 1)$.
- For each vertex i, make a bag with d_i half-edges.
- Pair half-edges uniformly at random.



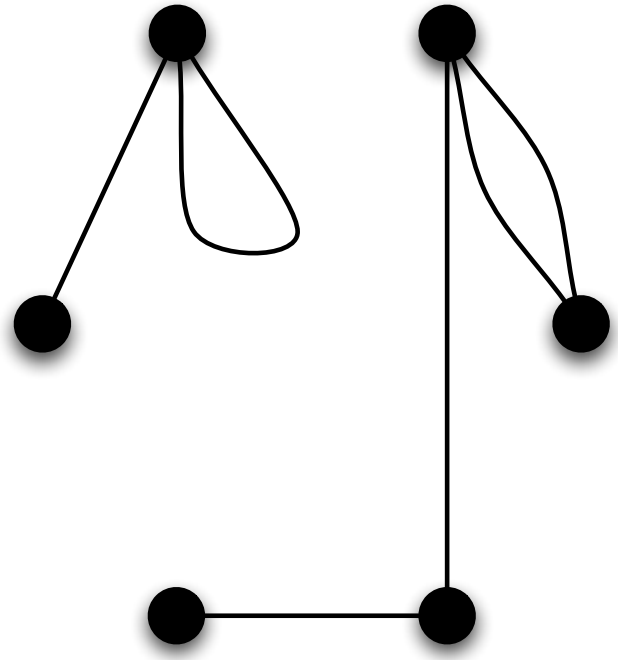
Configuration model

- Example: degrees $(3, 3, 2, 2, 1, 1)$.
- For each vertex i, make a bag with d_i half-edges.
- Pair half-edges uniformly at random.



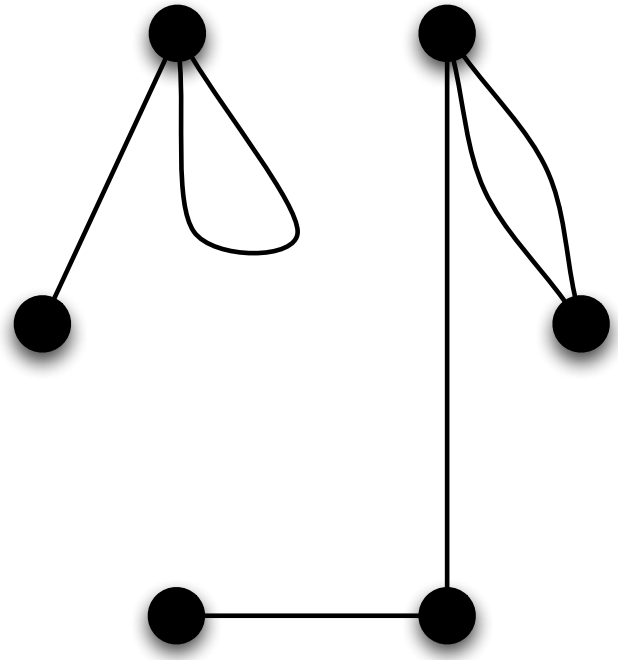
Configuration model

- Example: degrees $(3, 3, 2, 2, 1, 1)$.
- For each vertex i, make a bag with d_i half-edges.
- Pair half-edges uniformly at random.
- Collapse bags down to points.



Configuration model

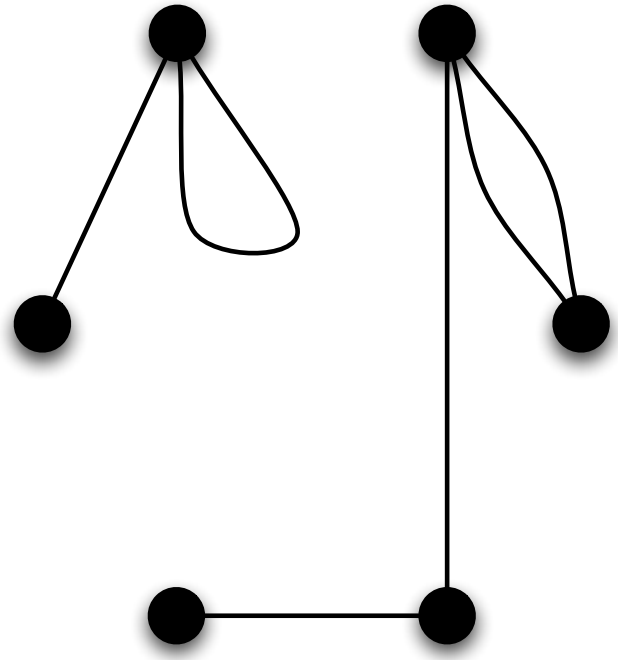
- Example: degrees $(3, 3, 2, 2, 1, 1)$.
- For each vertex i, make a bag with d_i half-edges.
- Pair half-edges uniformly at random.
- Collapse bags down to points.



Conditional on obtaining a simple graph, result has distribution $G_{\mathbf{d}}$.

Configuration model

- Example: degrees $(3, 3, 2, 2, 1, 1)$.
- For each vertex i, make a bag with d_i half-edges.
- Pair half-edges uniformly at random.
- Collapse bags down to points.



Conditional on obtaining a simple graph, result has distribution $G_{\mathbf{d}}$.
Note: in exploration process, a bag with i remaining half-edges is chosen with probability proportional to i. (Size-biasing.)

The universality class

Note: in exploration process, a bag with i remaining half-edges is chosen with probability proportional to i. (Size-biasing.)

The universality class

Note: in exploration process, a bag with i remaining half-edges is chosen with probability proportional to i. (Size-biasing.)

- Because of this, to obtain "normal" limiting behaviour, require degree distribution to have *finite third moments*.

The universality class

Note: in exploration process, a bag with i remaining half-edges is chosen with probability proportional to i. (Size-biasing.)

- Because of this, to obtain "normal" limiting behaviour, require degree distribution to have *finite third moments*.
- This means that $\sum_{i=1}^{n} d_i^3 = O(n)$ and $\max_{1 \le i \le n} d_i = o(n^{1/3})$.

The universality class

Note: in exploration process, a bag with i remaining half-edges is chosen with probability proportional to i. (Size-biasing.)

- Because of this, to obtain "normal" limiting behaviour, require degree distribution to have *finite third moments*.
- This means that $\sum_{i=1}^{n} d_i^3 = O(n)$ and $\max_{1 \le i \le n} d_i = o(n^{1/3})$.
- Without finite third moments, the limit is genuinely different.

The universality class

Note: in exploration process, a bag with i remaining half-edges is chosen with probability proportional to i. (Size-biasing.)

- Because of this, to obtain "normal" limiting behaviour, require degree distribution to have *finite third moments*.
- This means that $\sum_{i=1}^{n} d_i^3 = O(n)$ and $\max_{1 \le i \le n} d_i = o(n^{1/3})$.
- Without finite third moments, the limit is genuinely different. Reflected Brownian motion with drift becomes a thinned Lévy process. (Bhamidi, van der Hofstad, van Leeuwaarden, 2010.)

Work in progress

We believe we can prove that critical random graphs with a given degree sequence and finite third moments have the same scaling limit as $G_{n,1/n}$.

Work in progress

We believe we can prove that critical random graphs with a given degree sequence and finite third moments have the same scaling limit as $G_{n,1/n}$.
Some ingredients of the proof:

Work in progress

We believe we can prove that critical random graphs with a given degree sequence and finite third moments have the same scaling limit as $G_{n,1/n}$.
Some ingredients of the proof:

- The exploration process of such $G_{\mathbf{d}}$ has "reflected Brownian motion with quadratic drift, plus Poisson marks" as its scaling limit (Joseph 2010, Riordan 2011).

Work in progress

We believe we can prove that critical random graphs with a given degree sequence and finite third moments have the same scaling limit as $G_{n,1/n}$.
Some ingredients of the proof:

- The exploration process of such $G_{\mathbf{d}}$ has "reflected Brownian motion with quadratic drift, plus Poisson marks" as its scaling limit (Joseph 2010, Riordan 2011).
- A random *tree* with a given degree sequence and finite second moments, has scaling limit the Brownian CRT. (Broutin, Marckert, in preparation).

Work in progress

We believe we can prove that critical random graphs with a given degree sequence and finite third moments have the same scaling limit as $G_{n,1/n}$.
Some ingredients of the proof:

- The exploration process of such $G_{\mathbf{d}}$ has "reflected Brownian motion with quadratic drift, plus Poisson marks" as its scaling limit (Joseph 2010, Riordan 2011).
- A random *tree* with a given degree sequence and finite second moments, has scaling limit the Brownian CRT. (Broutin, Marckert, in preparation).
- A random tree with a given degree sequence and finite second moments, has sub-Gaussian tails for the height (Addario-Berry, Devroye, Janson 2010 and in preparation).

Random interlacements are in the same universality class

Random interlacements are in the same universality class

- Suppose G_d is a random d-regular graph (degree sequence $(d, d, \ldots, d)$).

Random interlacements are in the same universality class

- Suppose G_d is a random d-regular graph (degree sequence $(d, d, \ldots, d)$).
- Perform a random walk on G_d and consider the *vacant set after k steps*.

Random interlacements are in the same universality class

- Suppose G_d is a random d-regular graph (degree sequence $(d, d, \ldots, d)$).
- Perform a random walk on G_d and consider the *vacant set after k steps*.
- There is a critical k at which the large connected components of the vacant set have size $n^{2/3}$. (Černý and Teixeira, 2011.)

Random interlacements are in the same universality class

- Suppose G_d is a random d-regular graph (degree sequence $(d, d, \ldots, d)$).
- Perform a random walk on G_d and consider the *vacant set after k steps*.
- There is a critical k at which the large connected components of the vacant set have size $n^{2/3}$. (Černý and Teixeira, 2011.)
- Uses that, *given* that the vacant set has degree sequence $\mathbf{d} = (d_1, \ldots, d_m)$, it is distributed as $G_{\mathbf{d}}$ (Cooper, Frieze 2010).

Random interlacements are in the same universality class

- Suppose G_d is a random d-regular graph (degree sequence $(d, d, \ldots, d)$).
- Perform a random walk on G_d and consider the *vacant set after k steps*.
- There is a critical k at which the large connected components of the vacant set have size $n^{2/3}$. (Černý and Teixeira, 2011.)
- Uses that, *given* that the vacant set has degree sequence $\mathbf{d} = (d_1, \ldots, d_m)$, it is distributed as $G_{\mathbf{d}}$ (Cooper, Frieze 2010).
- At the critical k, the vacant set precisely has a critical degree sequence, and so will have the same scaling limit as $G_{n,1/n}$.

The last slide, I promise

The last slide, I promise

- I have many more open problems I find personally humiliating . . .

The last slide, I promise

- I have many more open problems I find personally humiliating . . .
- . . . but this one I don't.

The last slide, I promise

- I have many more open problems I find personally humiliating . . .
- . . . but this one I don't.

Question

Fix d large and let T be the MST of a d-dimensional torus (square lattice) with side length n and iid edge weights.

The last slide, I promise

- I have many more open problems I find personally humiliating . . .
- . . . but this one I don't.

Question

Fix d large and let T be the MST of a d-dimensional torus (square lattice) with side length n and iid edge weights.
Are two uniformly random lattice points at expected distance $n^{d/3}$ in T?

The last slide, I promise

Thanks again to Ben, James, and Matthias, and to all of you.