Sampling graphs in at least two ways

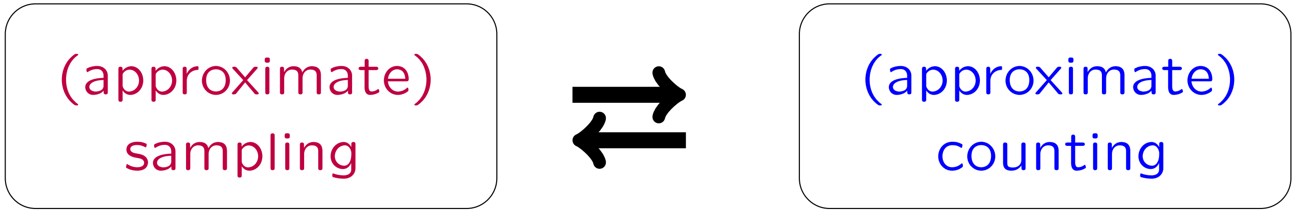
Catherine Greenhill

School of Mathematics and Statistics
UNSW Sydney

Martin Dyer Day, 16 July 2018

(approximate) sampling

(approximate) counting


(approximate)
sampling
(approximate)
counting

(approximate) sampling ⇄ (approximate) counting

Markov chains:

(approximate) sampling ⇄ (approximate) counting

Markov chains: path coupling,

(approximate) sampling ⇄ (approximate) counting

Markov chains: path coupling, canonical paths (approximate) sampling $\rightleftarrows$ (approximate) counting

Markov chains: path coupling, canonical paths

Cooper, Dyer & Greenhill (2007): The switch chain is rapidly mixing for regular graphs

(approximate) sampling ⇄ (approximate) counting

Markov chains: path coupling, canonical paths

Cooper, Dyer & Greenhill (2007): The switch chain is rapidly mixing for regular graphs

Earlier work:

Jerrum & Sinclair (1990): A different chain, rapidly mixing for P-stable (irregular) degree sequences.

Kannan, Tetali & Vempala (1999): switch chain for bipartite graphs, irregular degrees.

Asymptotics are as $n \to \infty$.

Assumptions:

Asymptotics are as $n \to \infty$.

Assumptions:

- Algorithms for sampling graphs with given degrees are useful.

Asymptotics are as $n \to \infty$.

Assumptions:

- Algorithms for sampling graphs with given degrees are useful.

- Performance guarantees are desirable.

Asymptotics are as $n \to \infty$.

Assumptions:

- Algorithms for sampling graphs with given degrees are useful.
- Performance guarantees are desirable.
- Any polynomial bound on the running time is good.

Asymptotics are as $n \to \infty$.

Assumptions:

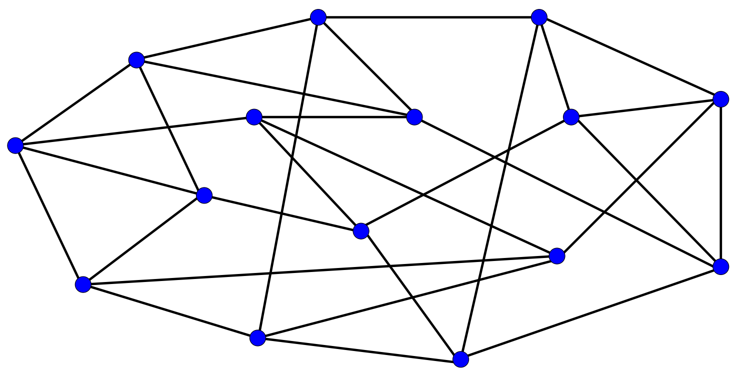
- Algorithms for sampling graphs with given degrees are useful.

- Performance guarantees are desirable.

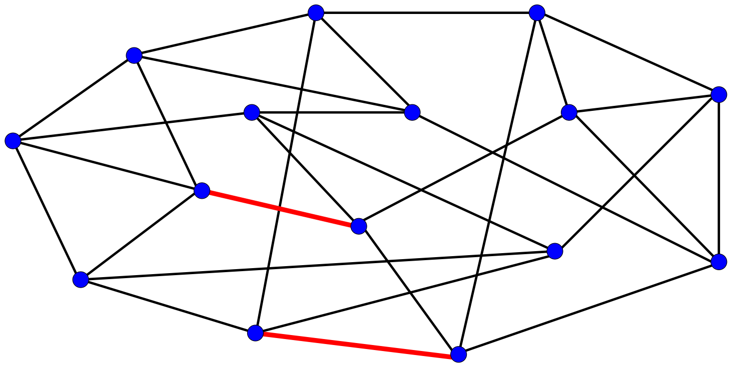
- Any polynomial bound on the running time is good.

Rapidly mixing Markov chains give approximately uniform sampling in deterministic polynomial time, with a user-specifed tolerance on the distance from uniform.

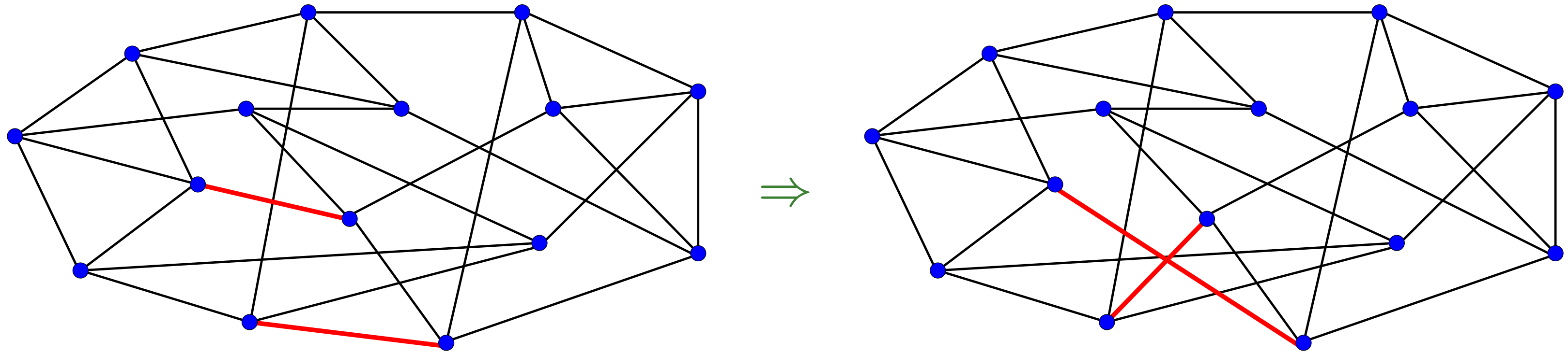
The switch chain for sampling graphs:



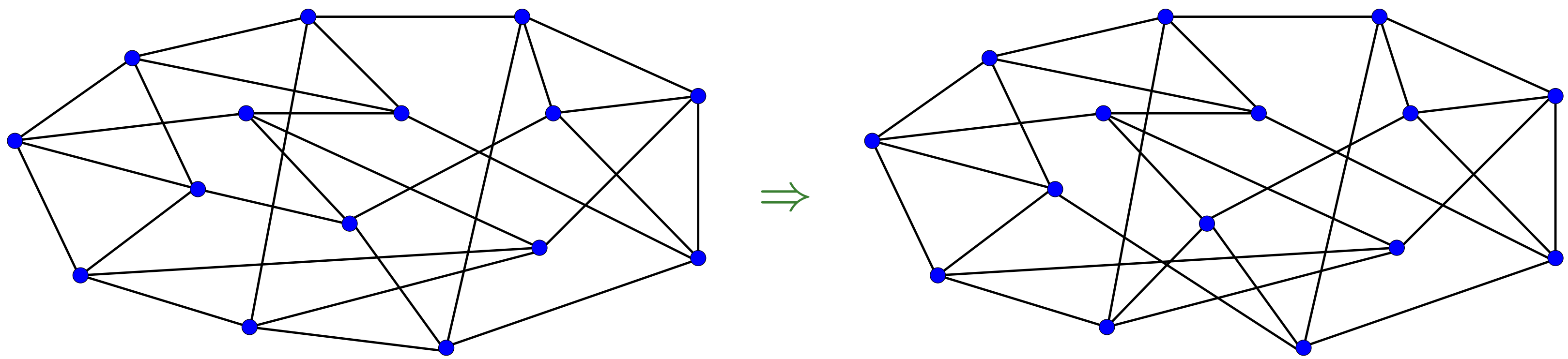
The switch chain for sampling graphs:



The switch chain for sampling graphs:

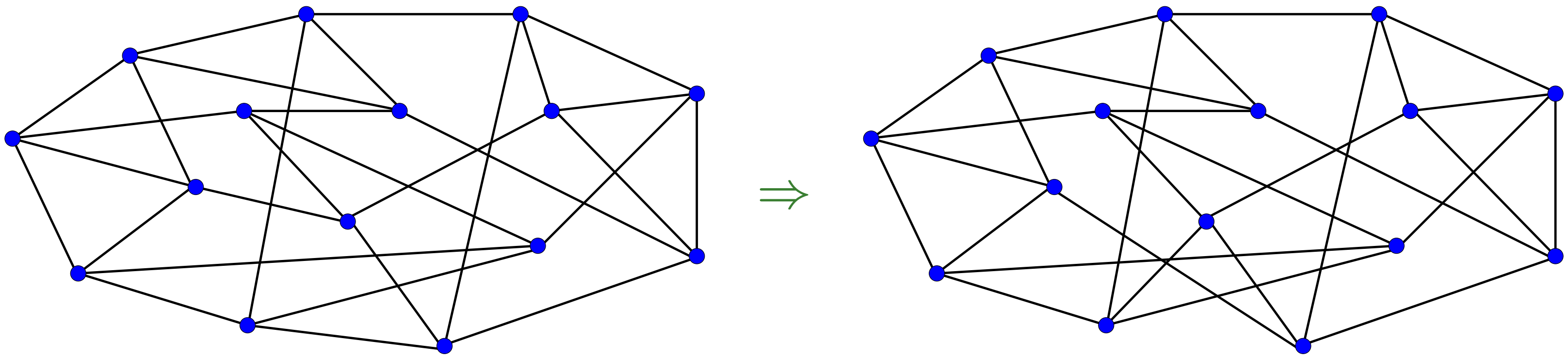


The switch chain for sampling graphs:



After many steps, the current graph is very close to a random regular graph (4-regular, in this case).

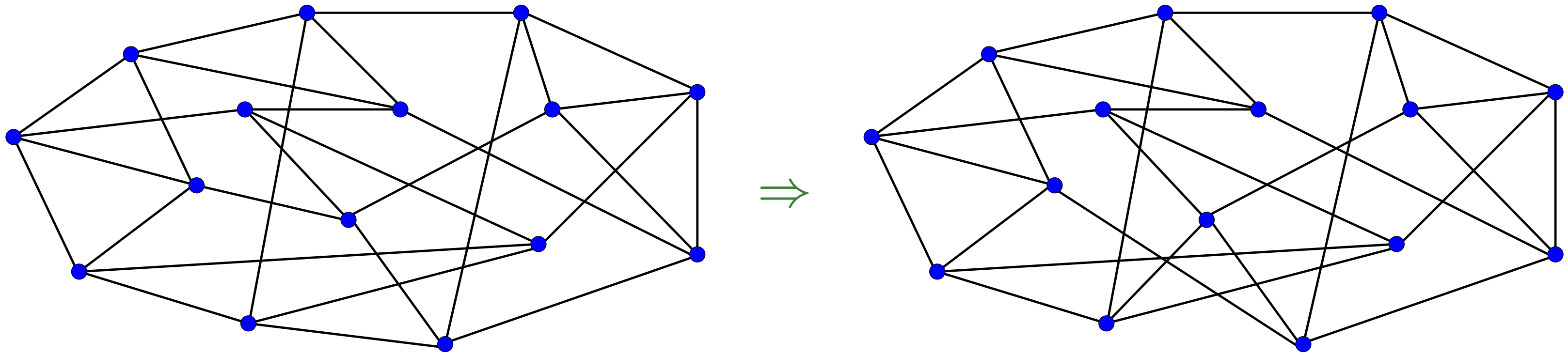
The switch chain for sampling graphs:



After many steps, the current graph is very close to a random regular graph (4-regular, in this case).

Cooper, Dyer, Greenhill (2007):
$d^{23}n^{8}(dn \log(dn) + \log(\varepsilon^{-1}))$ steps suffice to get within ε of uniform in total variation distance. (Any $d = d(n)$.)

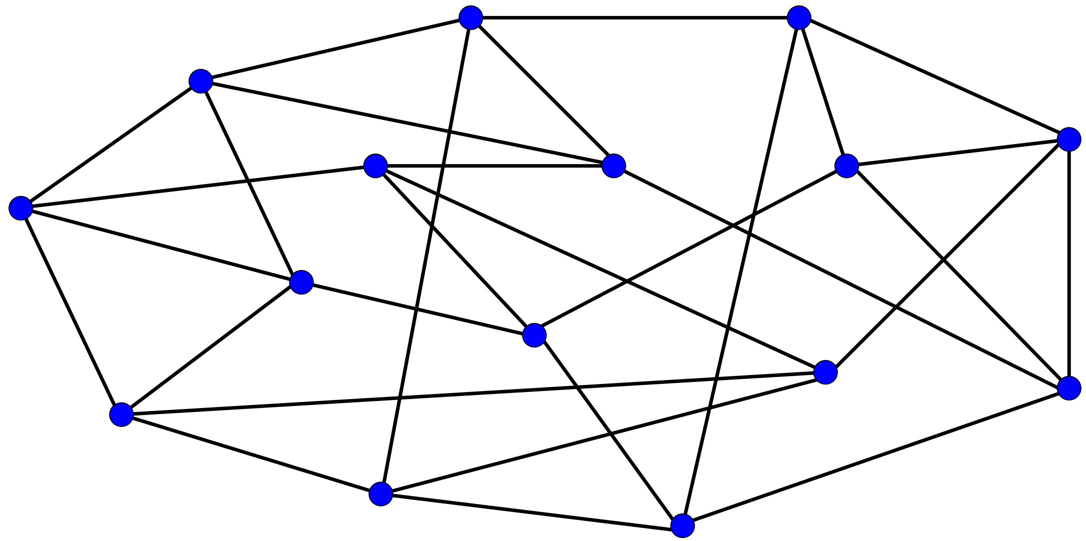
The switch chain for sampling graphs:



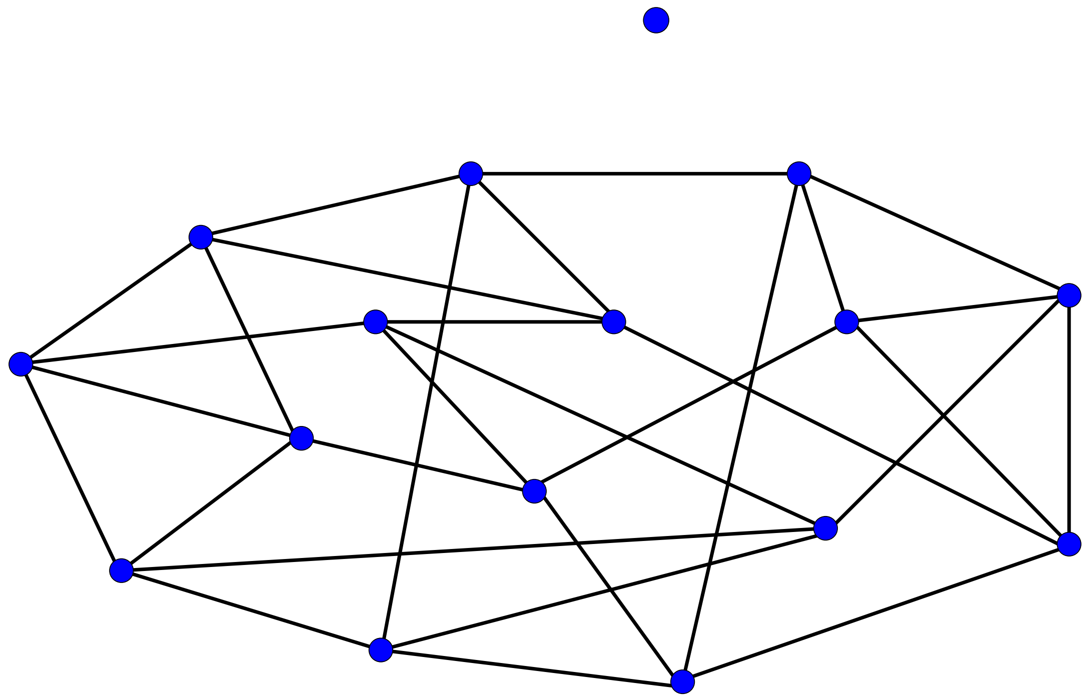
After many steps, the current graph is very close to a random regular graph (4-regular, in this case).

Cooper, Dyer, Greenhill (2007):
$d^{23}n^8(dn\log(dn) + \log(\varepsilon^{-1}))$ steps suffice to get within ε of uniform in total variation distance. (Any $d = d(n)$.)
This bound is probably way too high.

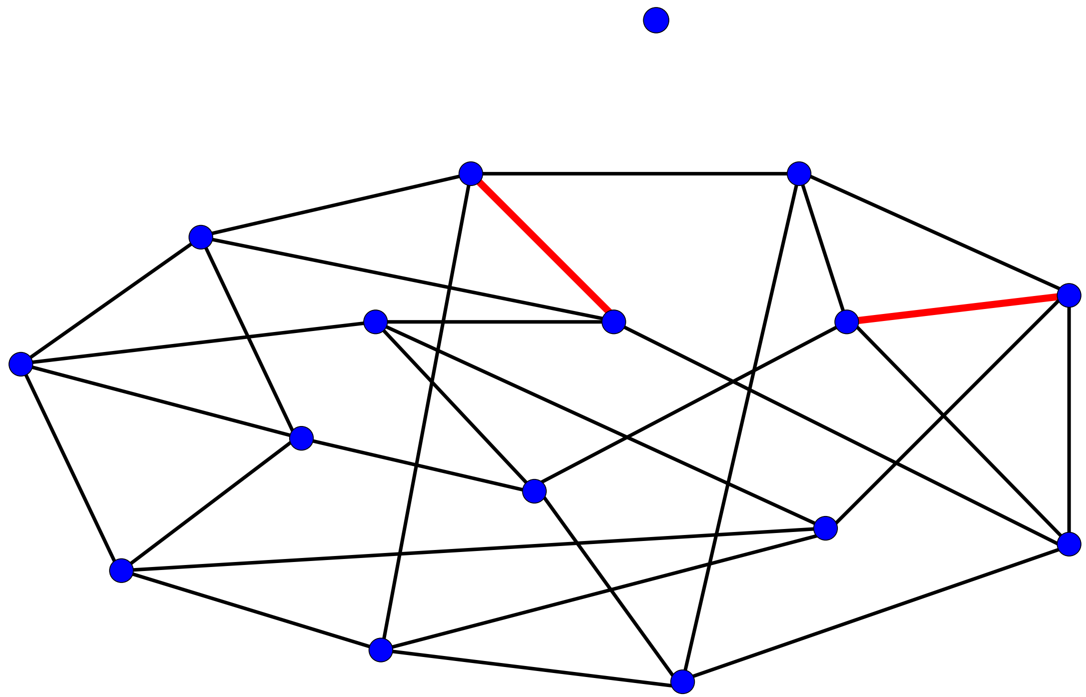
Small-World Wide-Area Network (Bourassa & Holt, 2003): A decentralised communication network based on random regular graphs.



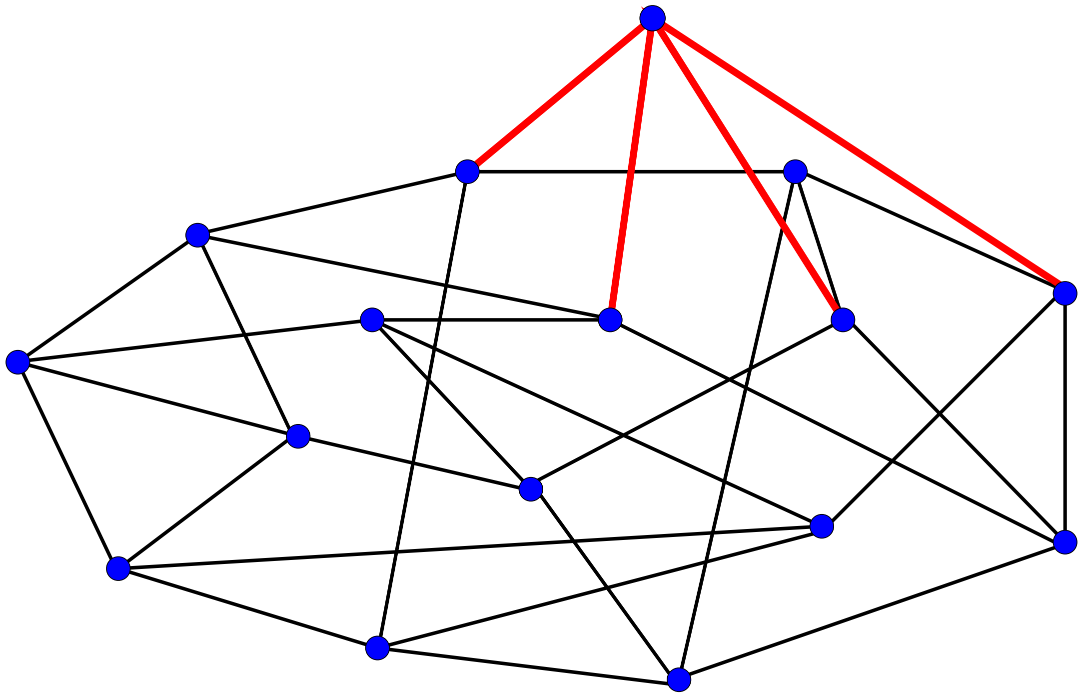
Small-World Wide-Area Network (Bourassa & Holt, 2003): A decentralised communication network based on random regular graphs.



Small-World Wide-Area Network (Bourassa & Holt, 2003): A decentralised communication network based on random regular graphs.



Small-World Wide-Area Network (Bourassa & Holt, 2003):
A decentralised communication network based on random regular graphs.



Asymptotic enumeration: want approximate formula for size of a set, with $o(1)$ relative error.

Asymptotic enumeration: want approximate formula for size of a set, with $o(1)$ relative error.

Number of permutations of $[n]$ (Stirling's formula):

$$n! = \sqrt{2\pi n}\left(\frac{n}{e}\right)^n (1 + O(1/n)).$$

Asymptotic enumeration: want approximate formula for size of a set, with $o(1)$ relative error.

Number of permutations of $[n]$ (Stirling's formula):

$$n! = \sqrt{2\pi n}\left(\frac{n}{e}\right)^n (1 + O(1/n)).$$

Number of d-regular graphs on $[n]$:

$$(1 + o(1))\,\sqrt{2}e^{1/4}\left(\lambda^\lambda (1-\lambda)^{1-\lambda}\right)^{\binom{n}{2}}\binom{n-1}{d}^n$$

where $\lambda = d/(n-1)$.

Number of d-regular graphs on $[n]$:

$$(1 + o(1))\,\sqrt{2}e^{1/4}\left(\lambda^{\lambda}\,(1-\lambda)^{1-\lambda}\right)^{\binom{n}{2}}\binom{n-1}{d}^{n}$$

where $\lambda = d/(n-1)$.

McKay & Wormald (1990, 1991):

Number of d-regular graphs on $[n]$:

$$(1 + o(1))\sqrt{2}e^{1/4}\left(\lambda^{\lambda}(1-\lambda)^{1-\lambda}\right)^{\binom{n}{2}}\binom{n-1}{d}^{n}$$

where $\lambda = d/(n-1)$.

McKay & Wormald (1990, 1991):

- sparse $[d = o(n^{1/2})]$;

Number of d-regular graphs on $[n]$:

$$(1+o(1))\,\sqrt{2}e^{1/4}\left(\lambda^{\lambda}(1-\lambda)^{1-\lambda}\right)^{\binom{n}{2}}\binom{n-1}{d}^{n}$$

where $\lambda = d/(n-1)$.

McKay & Wormald (1990, 1991):

- sparse $[d = o(n^{1/2})]$;
- quite dense $[\min\{d, n-d-1\} > cn/(\log n)$ for some $c > 2/3]$;

Number of d-regular graphs on $[n]$:

$$(1+o(1))\,\sqrt{2}e^{1/4}\,\left(\lambda^{\lambda}\,(1-\lambda)^{1-\lambda}\right)^{\binom{n}{2}}\binom{n-1}{d}^{n}$$

where $\lambda = d/(n-1)$.

McKay & Wormald (1990, 1991):

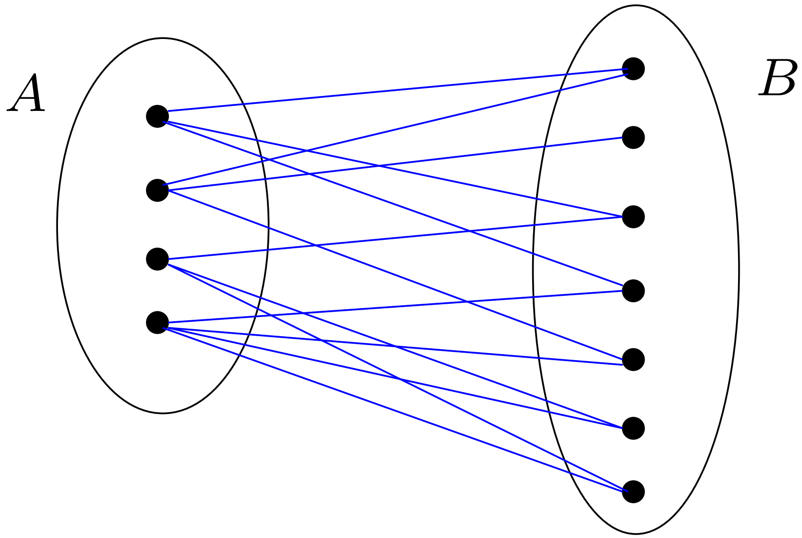
• sparse [$d = o(n^{1/2})$];

• quite dense [$\min\{d, n-d-1\} > cn/(\log n)$ for some $c > 2/3$];

Liebenau & Wormald (2017): filled the gap.

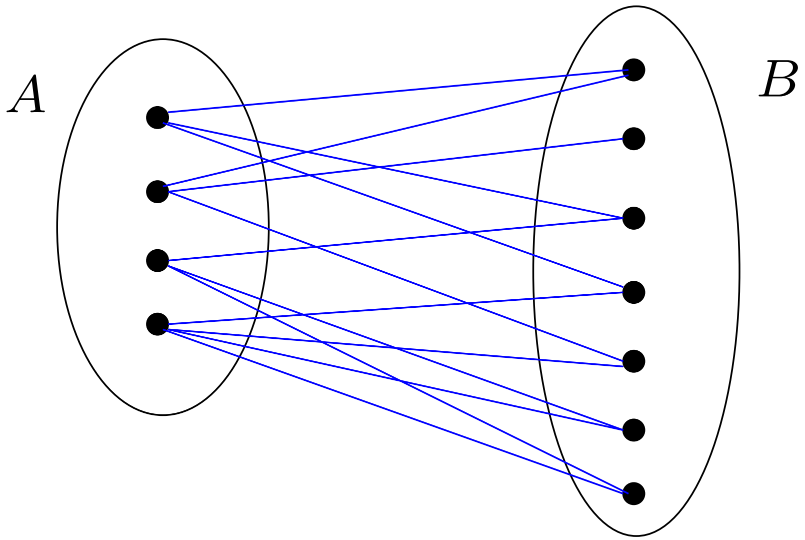
Sparse enumeration: switching method (McKay, 1981)

Basic idea: repeated double counting.



Sparse enumeration: switching method (McKay, 1981)

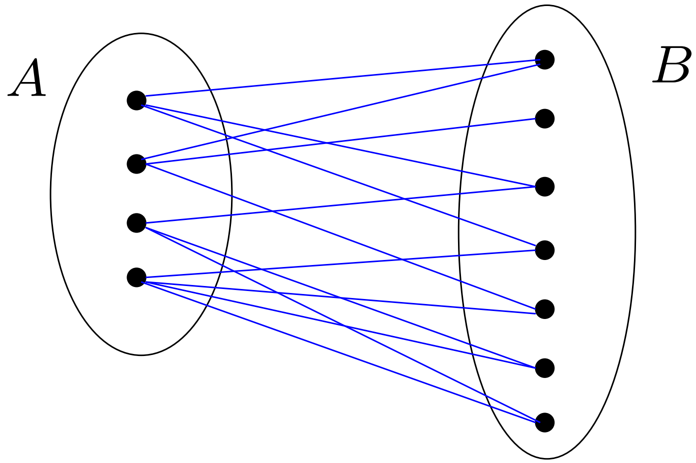
Basic idea: repeated double counting.



In the "big bipartite graph",

$$3|A| \leq \#\text{ edges} \leq 2|B|, \qquad \text{so} \qquad \frac{|A|}{|B|} \leq \frac{2}{3}.$$

Sparse enumeration: switching method (McKay, 1981)
Basic idea: repeated double counting.

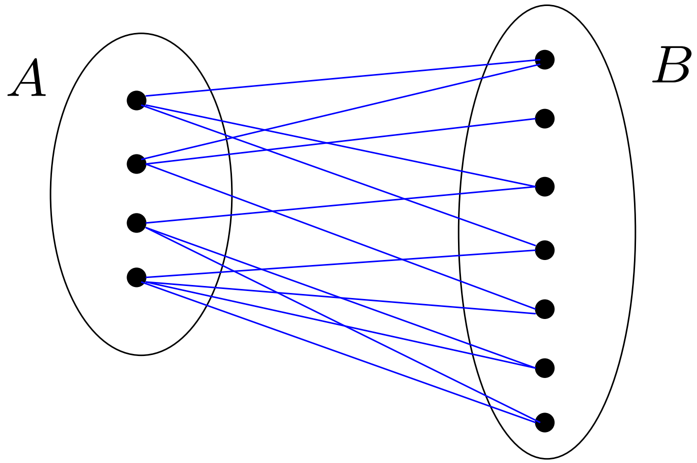


In the "big bipartite graph", if

$$|A|\,N_A(1+O(\varepsilon_A)) = \#\text{ edges} = |B|\,N_B(1+O(\varepsilon_B))$$

with $\varepsilon_A + \varepsilon_B = o(1)$

Sparse enumeration: switching method (McKay, 1981)
Basic idea: repeated double counting.

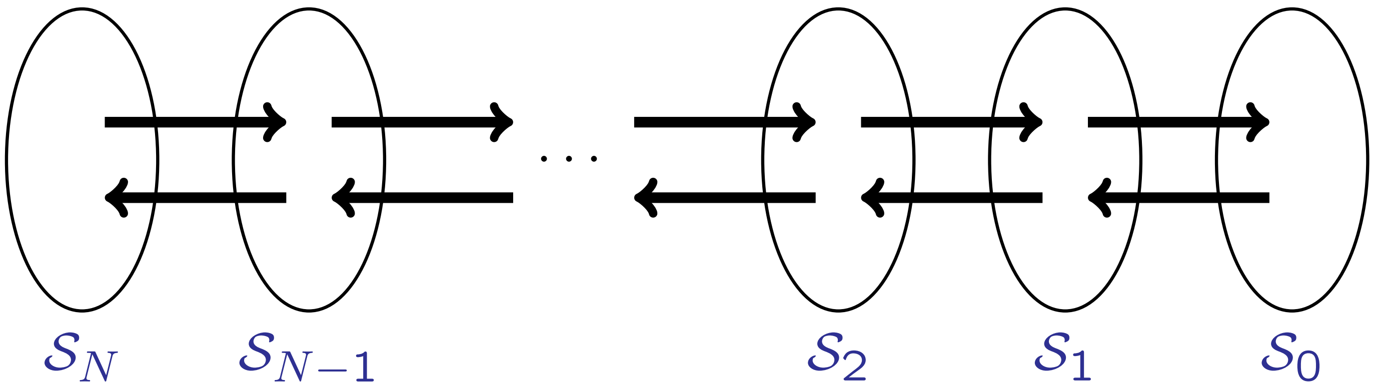


In the “big bipartite graph”, if

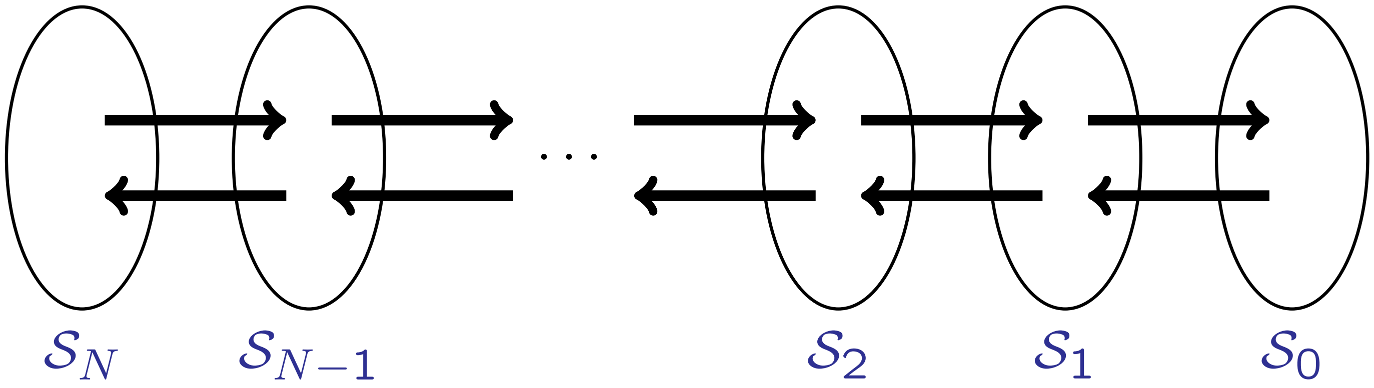
$$|A|\, N_A(1 + O(\varepsilon_A)) = \#\text{ edges} = |B|\, N_B(1 + O(\varepsilon_B))$$

with $\varepsilon_A + \varepsilon_B = o(1)$ then

$$\frac{|A|}{|B|} = \frac{N_B}{N_A}\,(1 + O(\varepsilon_A + \varepsilon_B)).$$



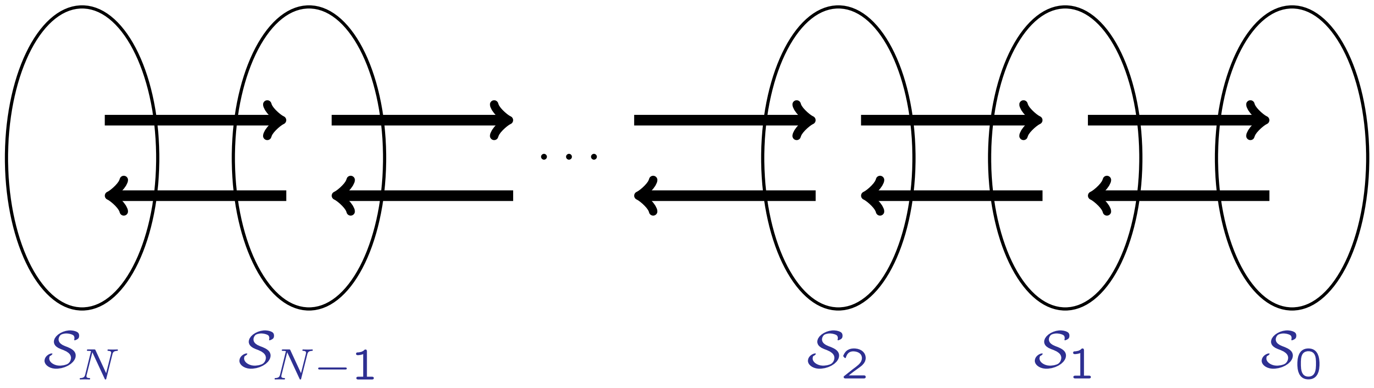
$\mathcal{S}_N$
$\mathcal{S}_{N-1}$
$\mathcal{S}_2$
$\mathcal{S}_1$
$\mathcal{S}_0$



Say we know $|\mathcal{S}|$ and we want to know $|\mathcal{S}_0|$ where

$$\mathcal{S} = \mathcal{S}_{\text{bad}} \cup \bigcup_{j=0}^{N} \mathcal{S}_j$$

and $|\mathcal{S}_{\text{bad}}|/|\mathcal{S}| = o(1)$ (all unions disjoint).



Say we know $|\mathcal{S}|$ and we want to know $|\mathcal{S}_0|$ where

$$\mathcal{S} = \mathcal{S}_{\text{bad}} \cup \bigcup_{j=0}^{N} \mathcal{S}_j$$

and $|\mathcal{S}_{\text{bad}}|/|\mathcal{S}| = o(1)$ (all unions disjoint). Then

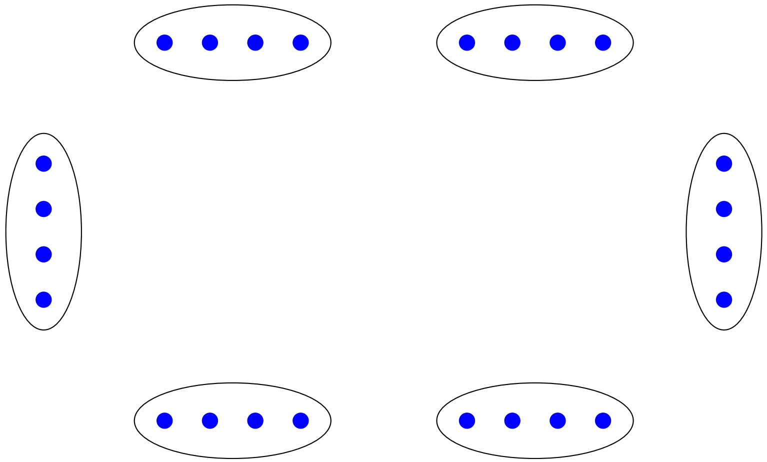
$$\frac{|\mathcal{S}|}{|\mathcal{S}_0|}(1 - o(1)) = \sum_{j=1}^{N} \frac{|\mathcal{S}_j|}{|\mathcal{S}_0|} = \sum_{j=1}^{N} \prod_{i=0}^{j-1} \frac{|\mathcal{S}_{i+1}|}{|\mathcal{S}_i|}.$$

McKay & Wormald (1991), sparse d-regular graphs

Work with the configuration model (Bollobás, 1980).

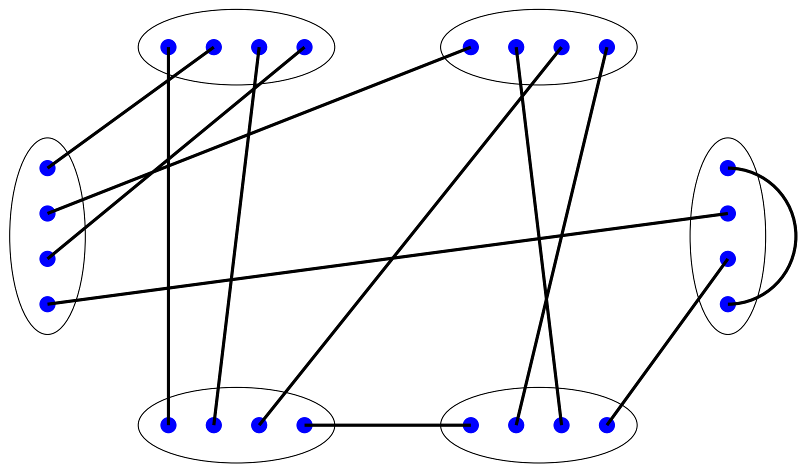
Configuration model (Bollobás, 1980)

Start with n cells, each containing d points. Take a uniformly random perfect matching of dn points into $dn/2$ pairs.



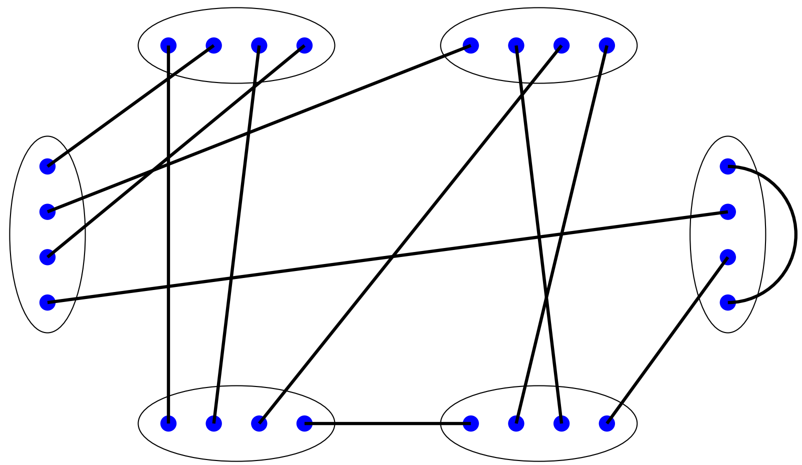
Configuration model (Bollobás, 1980)

Start with n cells, each containing d points. Take a uniformly random perfect matching of dn points into $dn/2$ pairs.



Configuration model (Bollobás, 1980)

Start with n cells, each containing d points. Take a uniformly random perfect matching of dn points into $dn/2$ pairs.



Shrink each cell to a vertex to get a d-regular multigraph.
If the result is not simple, just try again.
Expected polynomial time sampling if $d = O(\sqrt{\log n})$.

McKay & Wormald (1991), sparse d-regular graphs

Work with the configuration model (Bollobás, 1980)

$$\begin{aligned} \mathcal{S} &= \text{all configurations,} \\ \mathcal{S}_{0,0,0} &= \text{simple configurations,} \\ \mathcal{S}_{\ell,b,t} &= \text{set of configurations with } \ell \text{ loops, } b \text{ double pairs,} \\ &\qquad t \text{ triple pairs and no pairs with multiplicity} \geq 4. \end{aligned}$$

McKay & Wormald (1991), sparse d-regular graphs

Work with the configuration model (Bollobás, 1980)

$$\begin{aligned}\mathcal{S} &= \text{all configurations,}\\ \mathcal{S}_{0,0,0} &= \text{simple configurations,}\\ \mathcal{S}_{\ell,b,t} &= \text{set of configurations with } \ell \text{ loops, } b \text{ double pairs,}\\ &\qquad t \text{ triple pairs and no pairs with multiplicity} \geq 4.\end{aligned}$$

Here $\mathcal{S}_{\text{bad}}$ = set with "too many" loops, doubles or triples, or any pair of multiplicity ≥ 4.

From an element of $\mathcal{S}_{\ell,b,t}$:

From an element of $\mathcal{S}_{\ell,b,t}$:

- First, apply a switching to remove loops (one at a time);

From an element of $\mathcal{S}_{\ell,b,t}$:

- First, apply a switching to remove loops (one at a time);
- Then, apply a switching to remove triple pairs;

From an element of $\mathcal{S}_{\ell,b,t}$:

- First, apply a switching to remove loops (one at a time);
- Then, apply a switching to remove triple pairs;
- Finally, apply a switching to remove double pairs.

From an element of $\mathcal{S}_{\ell,b,t}$:

- First, apply a switching to remove loops (one at a time);
- Then, apply a switching to remove triple pairs;
- Finally, apply a switching to remove double pairs.

$\Rightarrow$ asymptotic enumeration formula

From an element of $\mathcal{S}_{\ell,b,t}$:

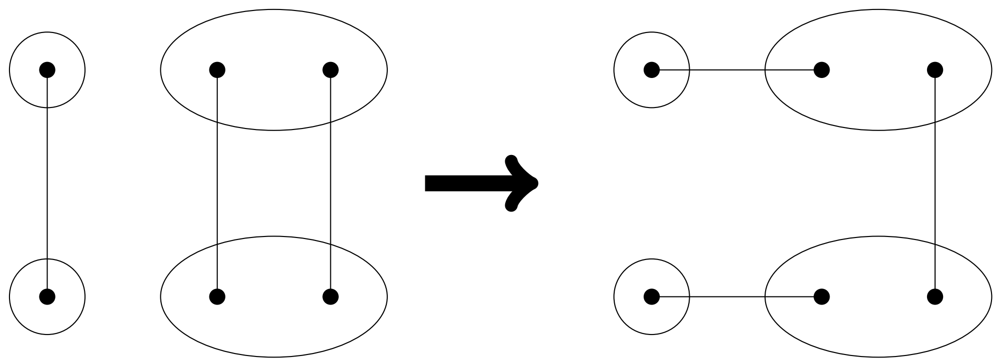
- First, apply a switching to remove loops (one at a time);
- Then, apply a switching to remove triple pairs;
- Finally, apply a switching to remove double pairs.

$\Rightarrow$ asymptotic enumeration formula

Also $\Rightarrow$ exactly uniform sampling algorithm!

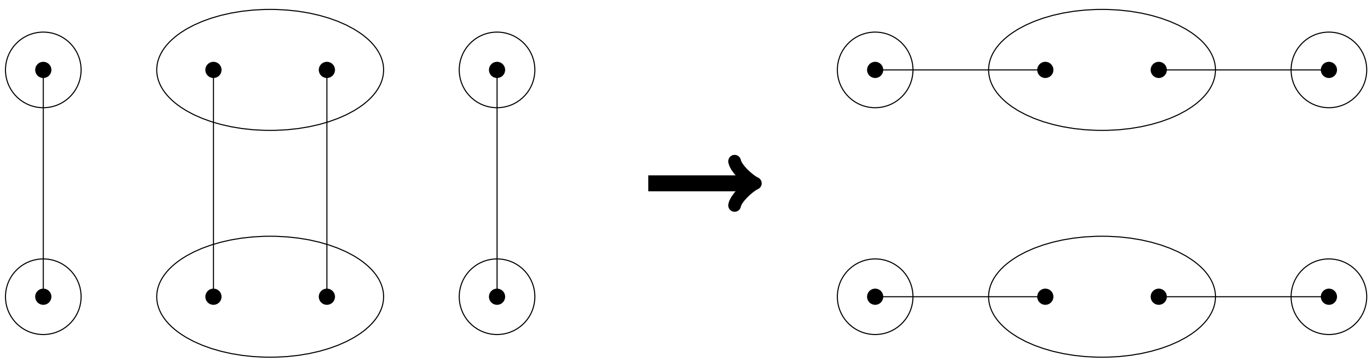
McKay & Wormald (1991) improved on an earlier formula of McKay (1985) by using more complicated switchings.

OLD:

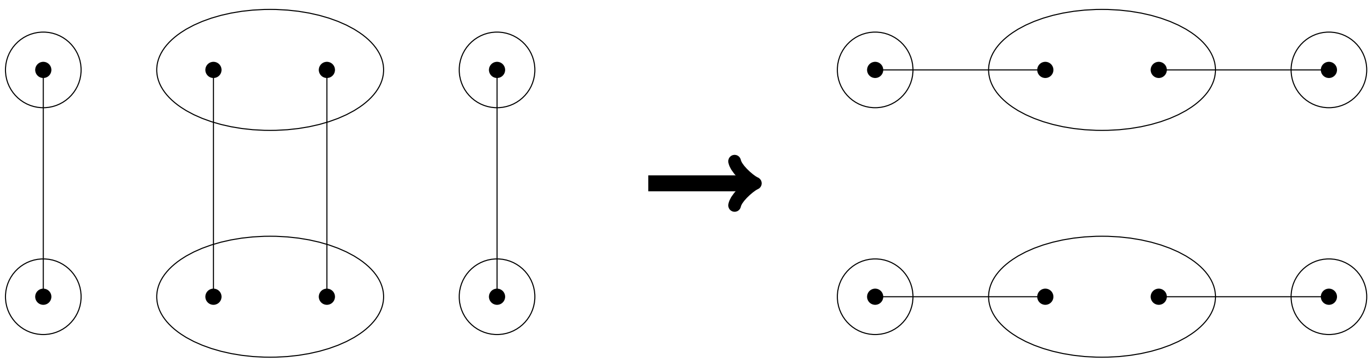


McKay & Wormald (1991) improved on an earlier formula of McKay (1985) by using more complicated switchings.

NEW:



McKay & Wormald (1991) improved on an earlier formula of McKay (1985) by using more complicated switchings.
NEW:



This inspired my adaptation of Cooper, Dyer, Greenhill (2007) to irregular degree sequences which are not too dense (SODA 2015).

Markov chains, recent directions:

Markov chains, recent directions:

- Curveball algorithm: introduced by Verhelst (2008).
  Carstens & Kleer (2017):
  Spectral gap comparision with switch chain.

Markov chains, recent directions:

- Curveball algorithm: introduced by Verhelst (2008). Carstens & Kleer (2017): Spectral gap comparision with switch chain.

- Amanatidis & Kleer (2018), rapid mixing of switch chain for strong stable degree sequences. Builds on chain of Jerrum & Sinclair (1990).

Markov chains, recent directions:

- Curveball algorithm: introduced by Verhelst (2008). Carstens & Kleer (2017): Spectral gap comparision with switch chain.

- Amanatidis & Kleer (2018), rapid mixing of switch chain for strong stable degree sequences. Builds on chain of Jerrum & Sinclair (1990).

- Erdős, Miklós & Torozckai (2016), new families of rapidly mixing switch degree sequences from old, using Tyshkevich deompositions.

Other kinds of sampling algorithms?

Other kinds of sampling algorithms?

Yes! Exactly uniform in expected polynomial time.

Other kinds of sampling algorithms?

Yes! Exactly uniform in expected polynomial time.

- Configuration model, Bollobás (1979).
  Expected polynomial time if $d_{\mathsf{max}} = O(\sqrt{\log n})$.

Other kinds of sampling algorithms?

Yes! Exactly uniform in expected polynomial time.

- Configuration model, Bollobás (1979).
  Expected polynomial time if $d_{\mathsf{max}} = O(\sqrt{\log n})$.

- McKay & Wormald (1990).
  Expected runtime $O(d_{\mathsf{max}}^4 n^2)$ if $d_{\mathsf{max}} = O(m^{1/4})$.

Other kinds of sampling algorithms?

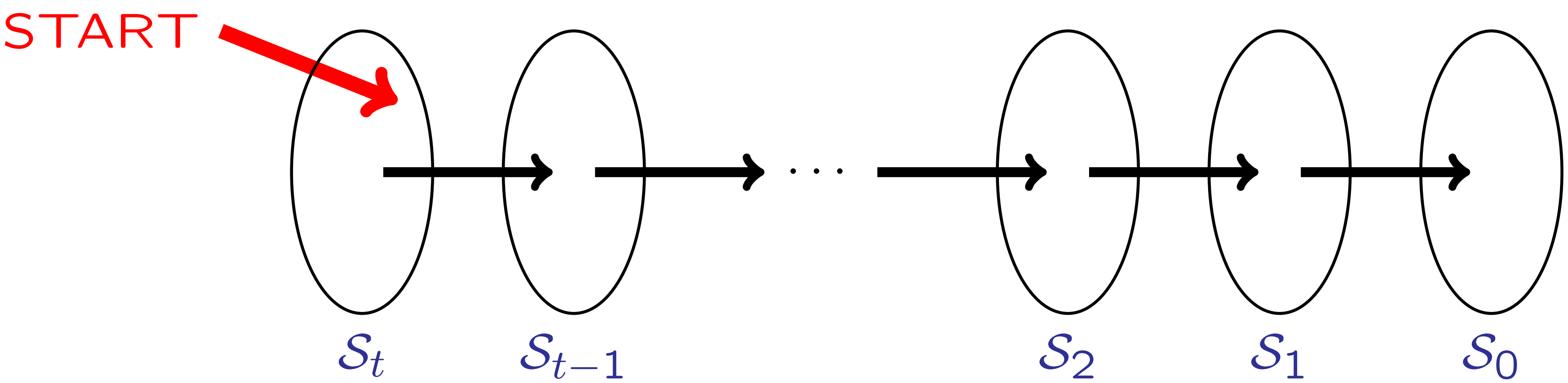
Yes! Exactly uniform in expected polynomial time.

- Configuration model, Bollobás (1979).
  Expected polynomial time if $d_{\text{max}} = O(\sqrt{\log n})$.

- McKay & Wormald (1990).
  Expected runtime $O(d_{\text{max}}^4 n^2)$ if $d_{\text{max}} = O(m^{1/4})$.

  If d-regular then this condition is $d = O(n^{1/3})$ and the expected runtime can be improved to $O(d^3 n)$.

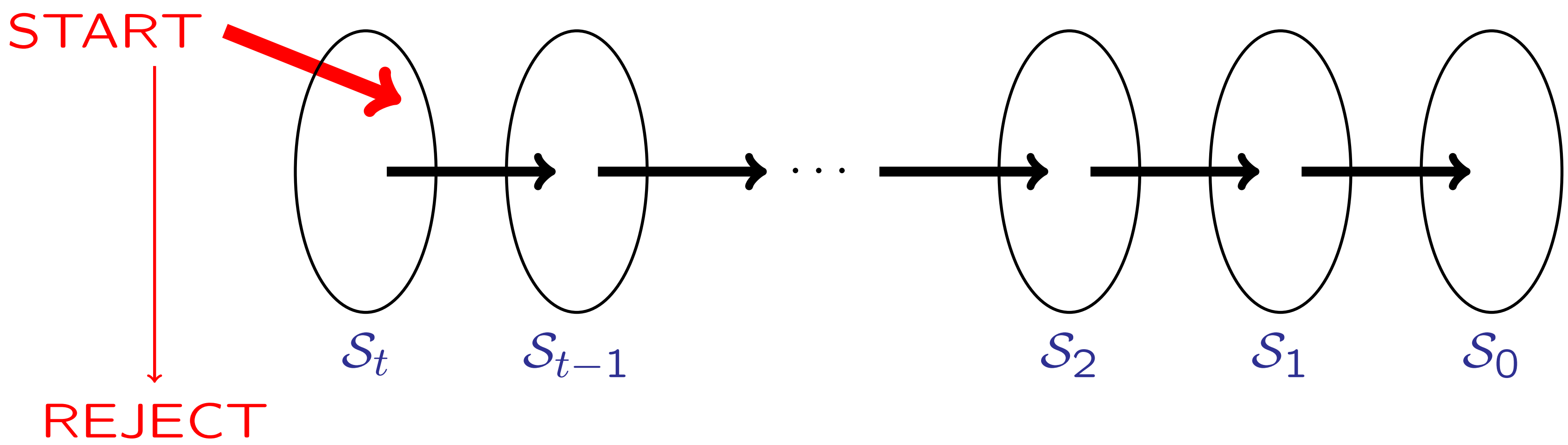
McKay-Wormald algorithm (1990).



.

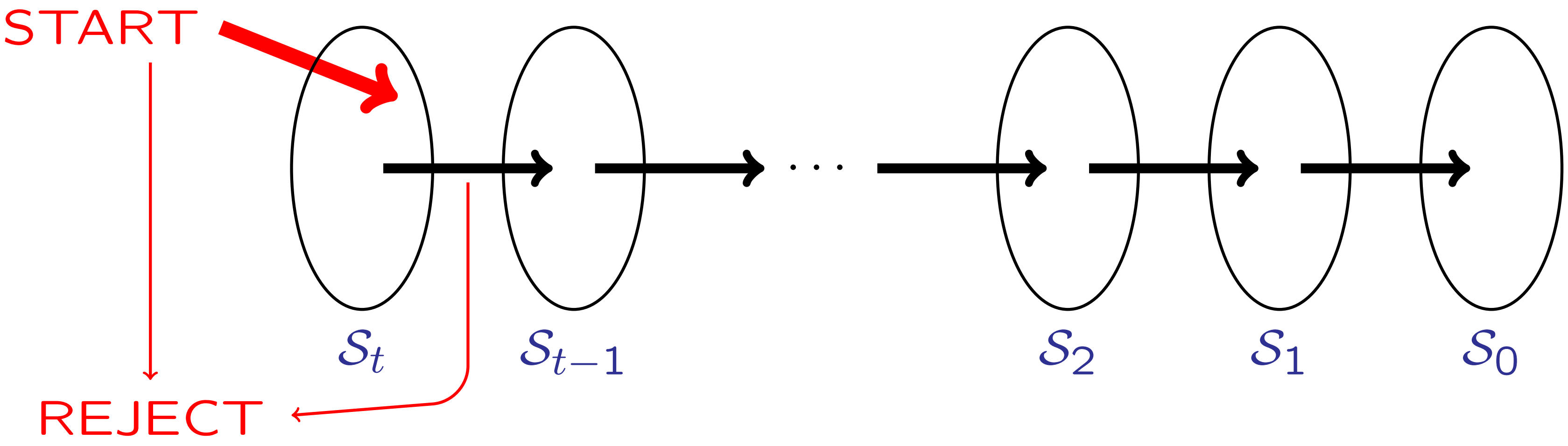
Generate uniformly random configuration.

McKay-Wormald algorithm (1990).



Generate uniformly random configuration. Reject if “bad”.

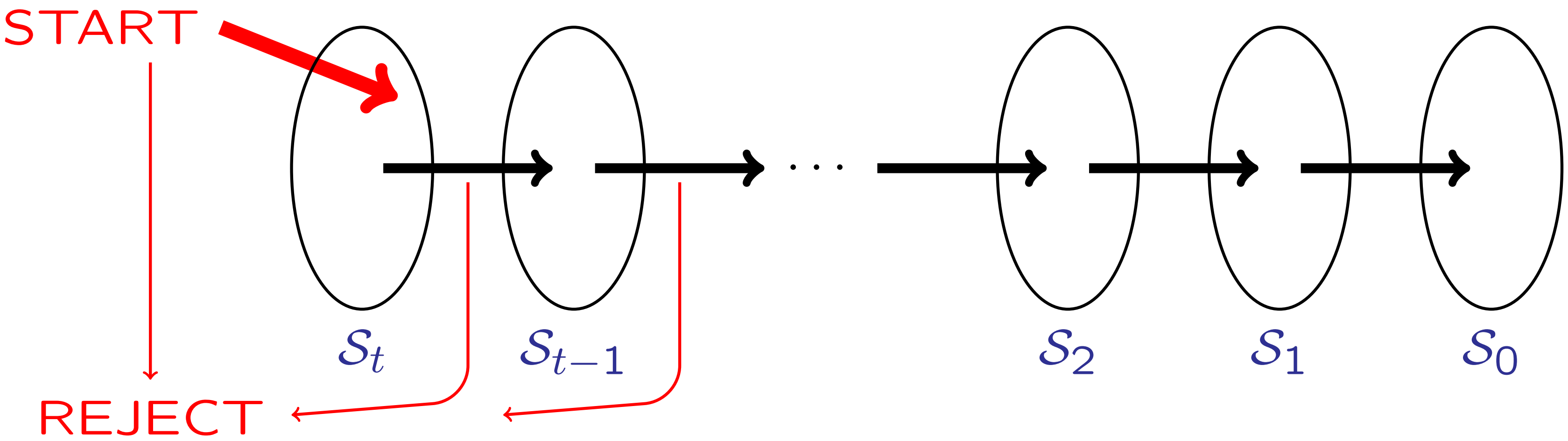
McKay-Wormald algorithm (1990).



Generate uniformly random configuration. Reject if "bad".

Repeatedly apply switchings with rejection.
Rejection probability chosen so that uniformity is preserved with each switching.

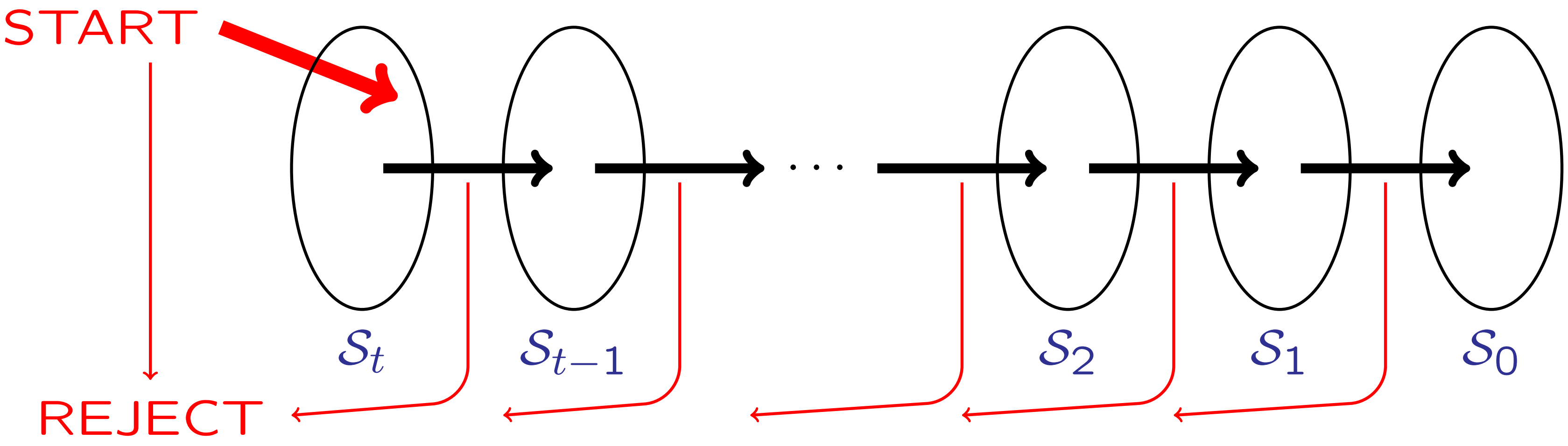
McKay-Wormald algorithm (1990).



Generate uniformly random configuration. Reject if "bad".

Repeatedly apply switchings with rejection.
Rejection probability chosen so that uniformity is preserved with each switching.

McKay-Wormald algorithm (1990).



Generate uniformly random configuration. Reject if “bad”.

Repeatedly apply switchings with rejection.
Rejection probability chosen so that uniformity is preserved with each switching.

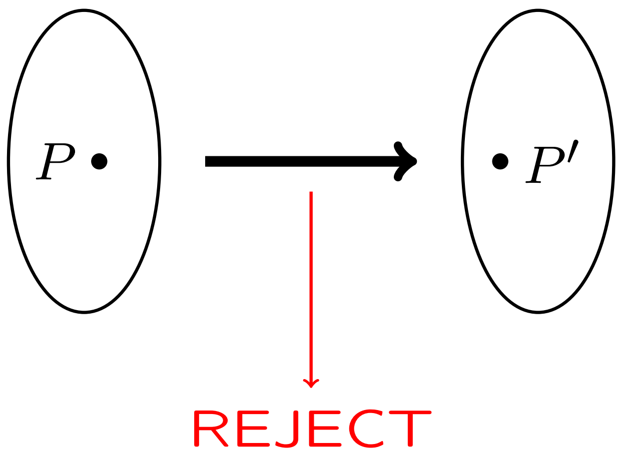
Gao & Wormald, 2015:
Extended this algorithm to give expected polynomial-time exactly uniform sampling in the regular case.
When $d = o(n^{1/2})$ the expected runtime is $O(d^3 n)$.

Gao & Wormald, 2015:

Extended this algorithm to give expected polynomial-time exactly uniform sampling in the regular case.

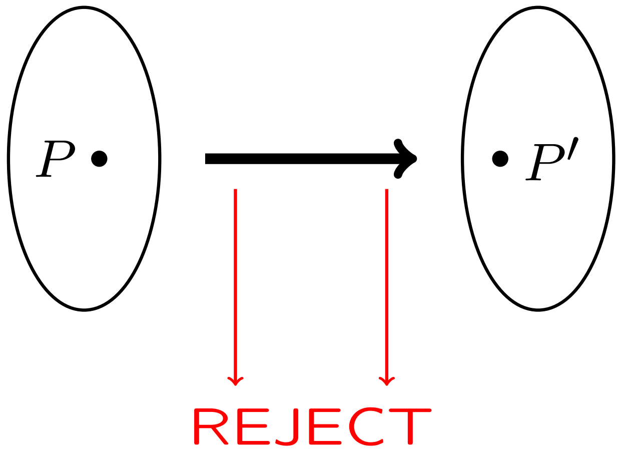
When $d = o(n^{1/2})$ the expected runtime is $O(d^3 n)$.

First idea: split rejection probability into two parts:

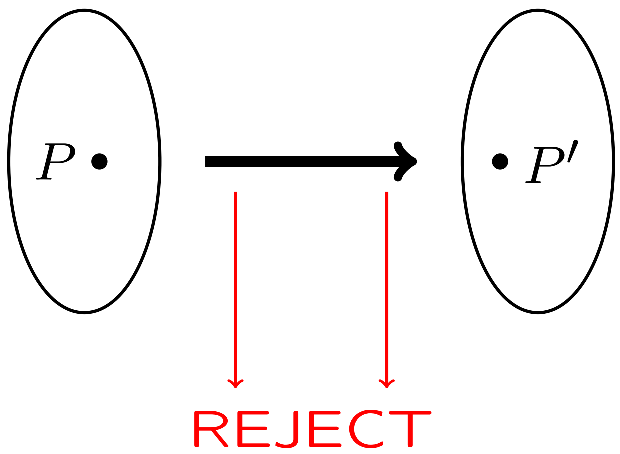


Gao & Wormald, 2015:
Extended this algorithm to give expected polynomial-time exactly uniform sampling in the regular case.
When $d = o(n^{1/2})$ the expected runtime is $O(d^3 n)$.

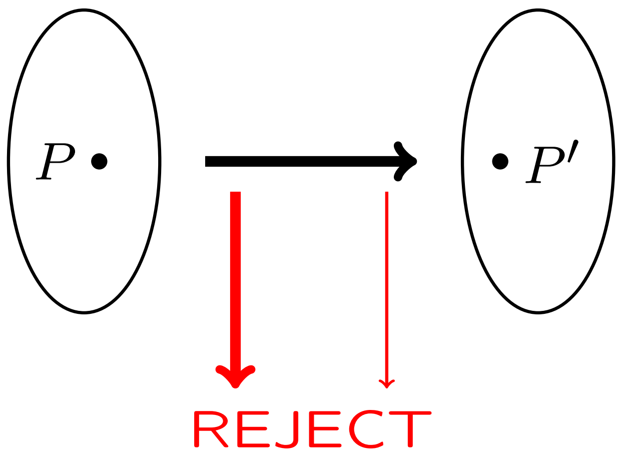
First idea: split rejection probability into two parts:



First idea: split rejection probability into two parts:

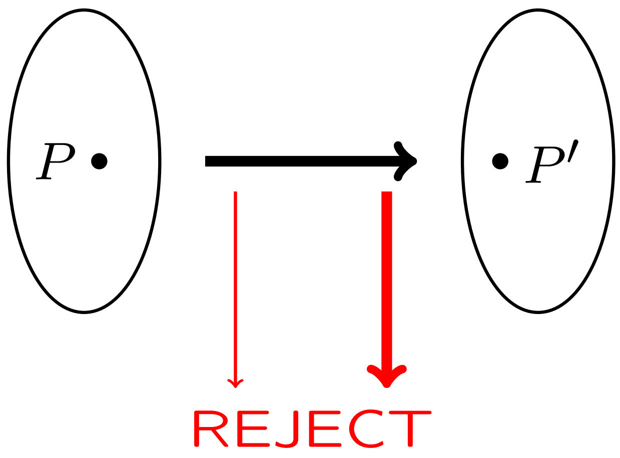


First idea: split rejection probability into two parts:



- f-rejection, depending only on P, and

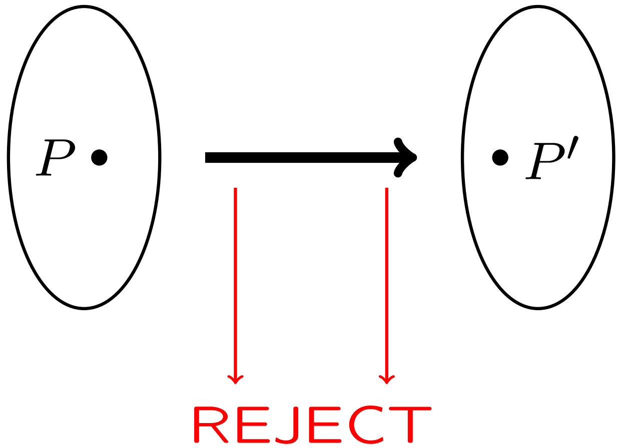
First idea: split rejection probability into two parts:



- f-rejection, depending only on P, and

- b-rejection, depending only on P'.

First idea: split rejection probability into two parts:



- f-rejection, depending only on P, and

- b-rejection, depending only on P'.

Then they provide strategies to reduce each of these.

As usual, double pairs cause all the problems. Get rid of loops and triple pairs first.

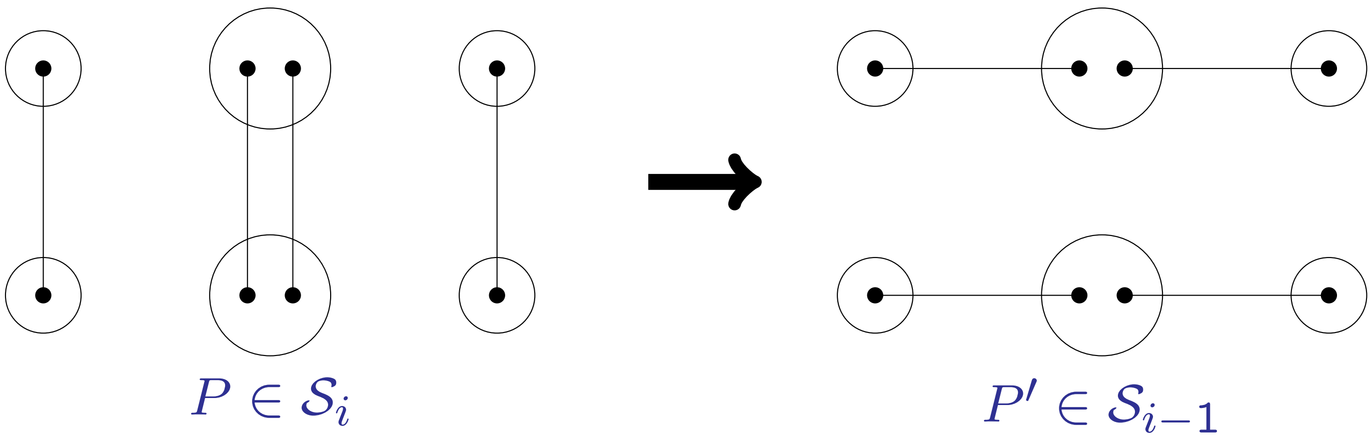
As usual, double pairs cause all the problems. Get rid of loops and triple pairs first.

Let $\mathcal{S}_i$ be the set of all configurations with precisely i double pairs, no loops and no pairs with multiplicity > 2.

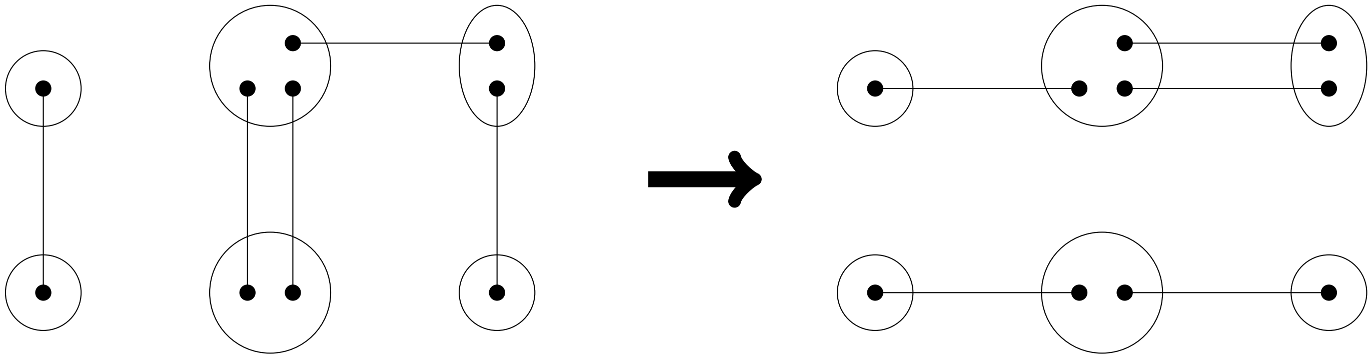
As usual, double pairs cause all the problems. Get rid of loops and triple pairs first.

Let $\mathcal{S}_i$ be the set of all configurations with precisely i double pairs, no loops and no pairs with multiplicity > 2.

Type I, Class A switching:



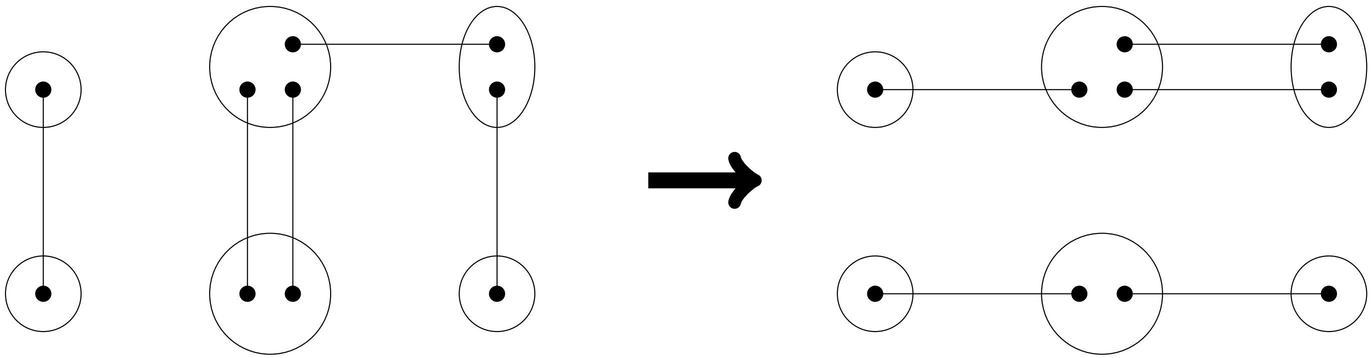
Introduce Type I, Class B switchings which do not change the number of double pairs.



$P \in \mathcal{S}_i$ $\qquad$ $P' \in \mathcal{S}_i$

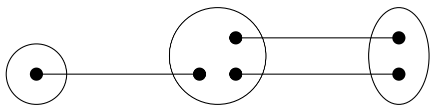
This reduces the probability of f-rejection...

Introduce Type I, Class B switchings which do not change the number of double pairs.

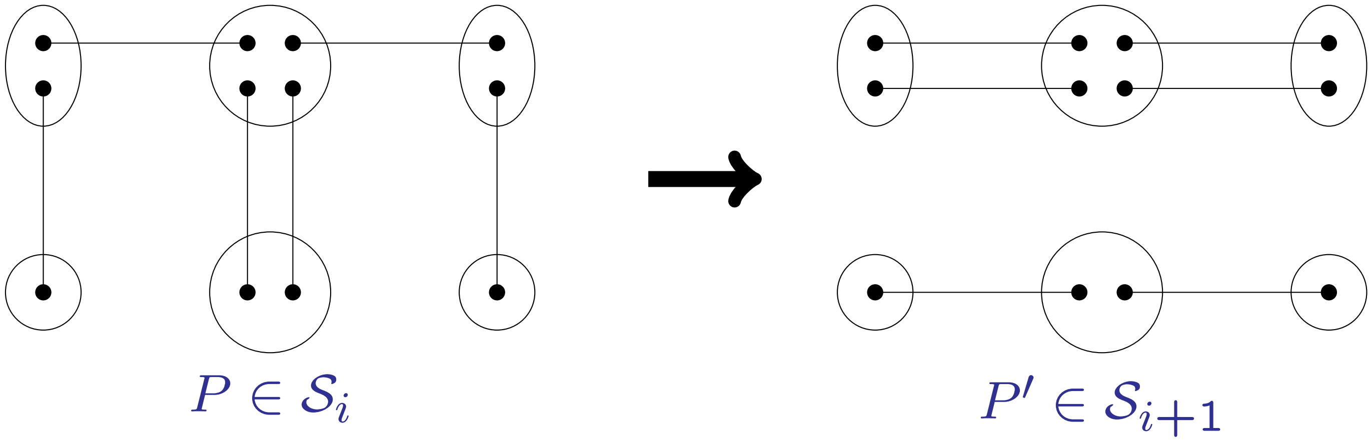


$P \in \mathcal{S}_i$ $\qquad$ $P' \in \mathcal{S}_i$

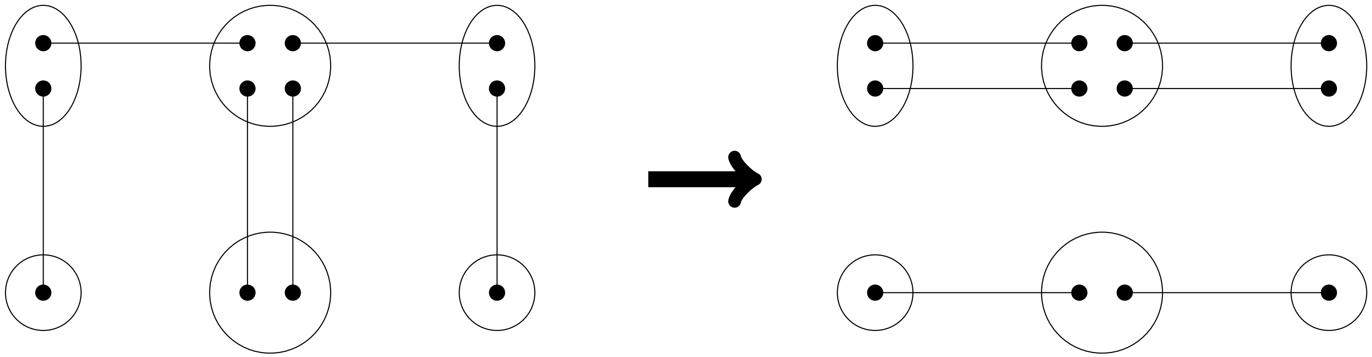
This reduces the probability of f-rejection, BUT also causes a new problem:

The number of  varies a lot among $P' \in \mathcal{S}_i$, leading to high probability of b-rejection.

Introduce Type II, Class B switchings which increase the number of of double pairs by one.

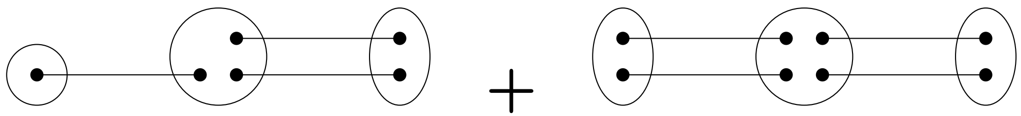


Introduce Type II, Class B switchings which increase the number of of double pairs by one.



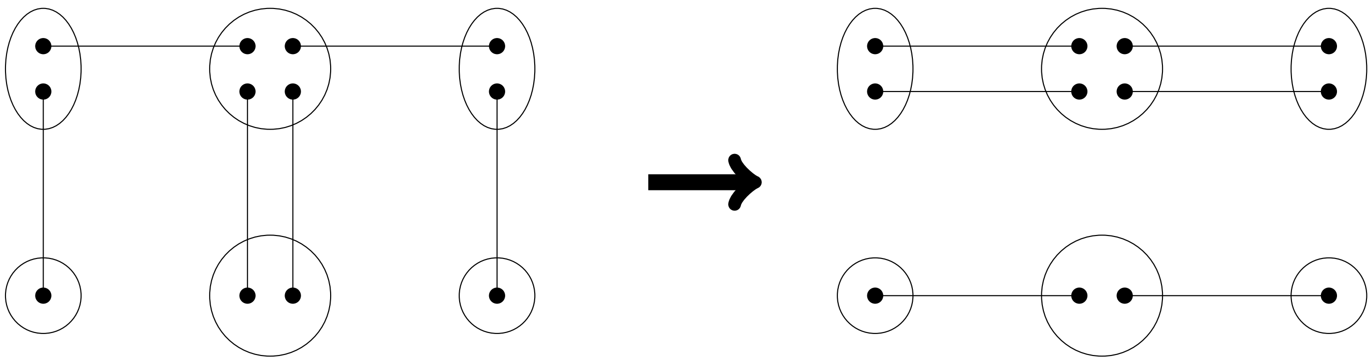
$P \in \mathcal{S}_i$ $\qquad$ $P' \in \mathcal{S}_{i+1}$

Why? Because the number of



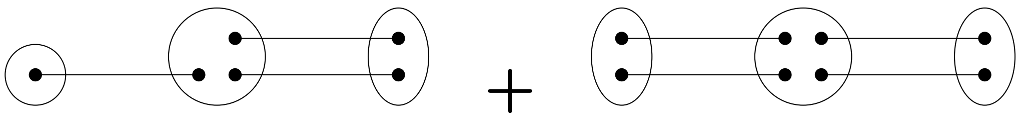
doesn't vary too much over a given $\mathcal{S}_j$

Introduce Type II, Class B switchings which increase the number of of double pairs by one.

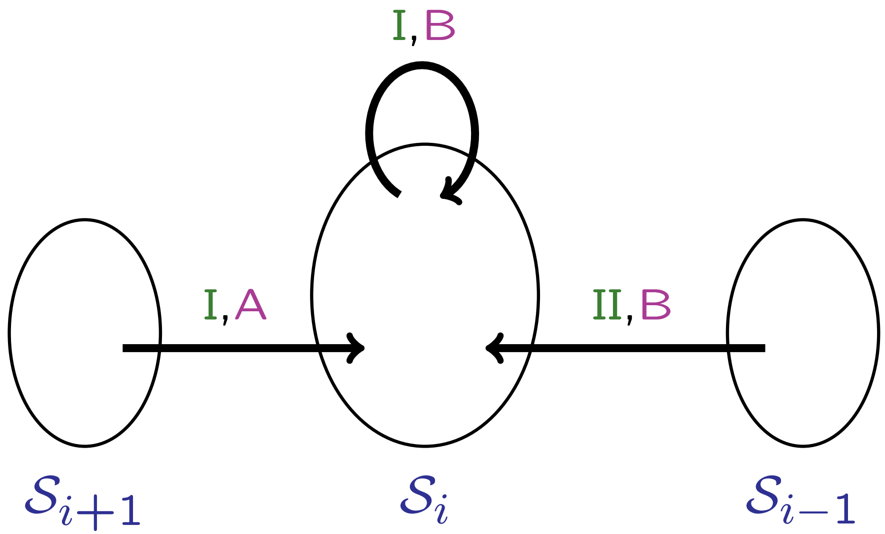


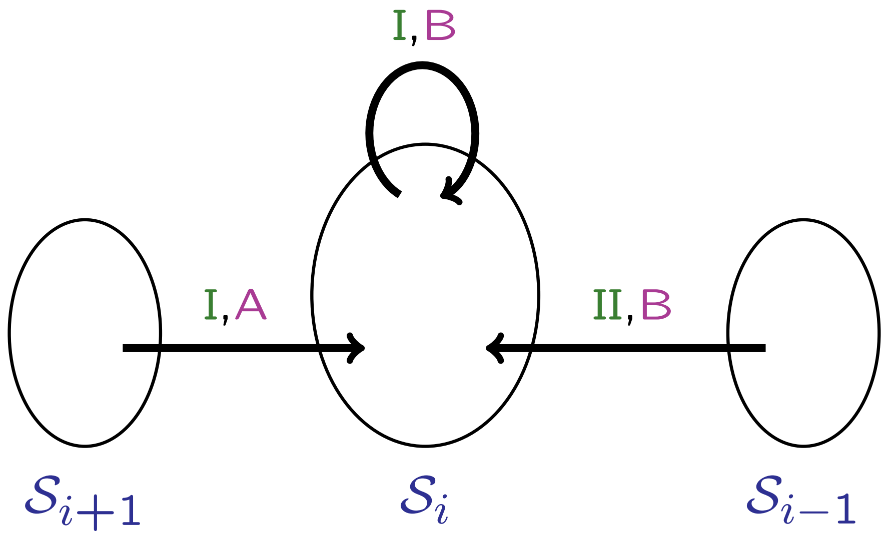
$P \in \mathcal{S}_i$ $\quad$ $P' \in \mathcal{S}_{i+1}$

Why? Because the number of

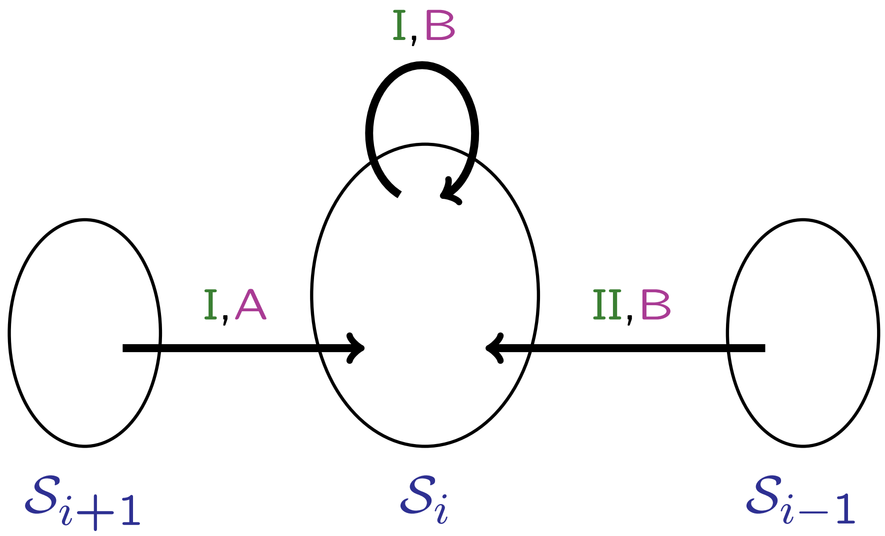


doesn't vary too much over a given $\mathcal{S}_j \Rightarrow$ less b-rejection.


I,B
I,A
II,B
$\mathcal{S}_{i+1}$
$\mathcal{S}_i$
$\mathcal{S}_{i-1}$



If the proposed switching $P \mapsto P'$ has type τ and class α then



If the proposed switching $P \mapsto P'$ has type τ and class α then

- f-rejection probability depends only on (P, τ),
- b-rejection probability depends only on (P', α).

Must also define $\rho_T(i) \geq 0$ such that $\sum_T \rho_T(i) \leq 1$ for all $i \leq i_1$, satisfying certain conditions.

Must also define $\rho_{\mathcal{T}}(i) \geq 0$ such that $\sum_{\mathcal{T}} \rho_{\mathcal{T}}(i) \leq 1$ for all $i \leq i_1$, satisfying certain conditions.

Algorithm: Given $P \in \mathcal{S}_i$,

Must also define $\rho_T(i) \geq 0$ such that $\sum_T \rho_T(i) \leq 1$ for all $i \leq i_1$, satisfying certain conditions.

Algorithm: Given $P \in \mathcal{S}_i$,

- If $i = 0$ then output the graph corresponding to P;

Must also define $\rho_\tau(i) \geq 0$ such that $\sum_\tau \rho_\tau(i) \leq 1$ for all $i \leq i_1$, satisfying certain conditions.

Algorithm: Given $P \in \mathcal{S}_i$,

- If $i = 0$ then output the graph corresponding to P;
- Choose type τ with probability $\rho_\tau(i)$ and u.a.r. choose a type τ switching $P \mapsto P'$. This decides the class α.

Must also define $\rho_\tau(i) \geq 0$ such that $\sum_\tau \rho_\tau(i) \leq 1$ for all $i \leq i_1$, satisfying certain conditions.

Algorithm: Given $P \in \mathcal{S}_i$,

- If $i = 0$ then output the graph corresponding to P;
- Choose type τ with probability $\rho_\tau(i)$ and u.a.r. choose a type τ switching $P \mapsto P'$. This decides the class α.
- Perform f-rejection and b-rejection: if neither occurs then move to P' and repeat.

Gao & Wormald, SODA 2018:

Extension to power-law degree sequences with exponent slightly below 3, with expected runtime $O(n^{2.107})$ with high probability.

Gao & Wormald, SODA 2018:

Extension to power-law degree sequences with exponent slightly below 3, with expected runtime $O(n^{2.107})$ with high probability.

Much more complicated: several phases; many types and classes of switching.

Gao & Wormald, SODA 2018:
Extension to power-law degree sequences with exponent slightly below 3, with expected runtime $O(n^{2.107})$ with high probability.

Much more complicated: several phases; many types and classes of switching.

Also a new kind of rejection, called pre-b-rejection.

Gao & Greenhill (arXiv, early July 2018):

Used same approach to exactly uniformly sample d-factors of a given regular host graph H.

Or: sampling d-regular graphs which avoid all edges of $\overline{H}$.

Gao & Greenhill (arXiv, early July 2018):
Used same approach to exactly uniformly sample d-factors of a given regular host graph H.
Or: sampling d-regular graphs which avoid all edges of $\overline{H}$.

If H is $(n - \Delta - 1)$-regular and $d^2 + \Delta^2 = o(n)$ then the expected runtime is $O((d + \Delta)^4 n + d^3 n \log n)$.

Gao & Greenhill (arXiv, early July 2018):
Used same approach to exactly uniformly sample d-factors of a given regular host graph H.
Or: sampling d-regular graphs which avoid all edges of $\overline{H}$.

If H is $(n - \Delta - 1)$-regular and $d^2 + \Delta^2 = o(n)$ then the expected runtime is $O((d + \Delta)^4 n + d^3 n \log n)$.

Our $\mathcal{S}_i$ is the set of d-regular graphs with exactly i "forbidden" edges (edges of $\overline{H}$).

Gao & Greenhill (arXiv, early July 2018):
Used same approach to exactly uniformly sample d-factors of a given regular host graph H.
Or: sampling d-regular graphs which avoid all edges of $\overline{H}$.

If H is $(n - \Delta - 1)$-regular and $d^2 + \Delta^2 = o(n)$ then the expected runtime is $O((d + \Delta)^4 n + d^3 n \log n)$.

Our $\mathcal{S}_i$ is the set of d-regular graphs with exactly i "forbidden" edges (edges of $\overline{H}$).

Ended up with switching types I, IIa±, IIb±, IIc±, III and switching classes A, B1±, B2±, C±.

Ended up with switching types **I**, **II**a±, **II**b±, **II**c±, **III** and switching classes A, B1±, B2±, C±.

Ended up with switching types I, IIa±, IIb±, IIc±, III and switching classes A, B1±, B2±, C±.