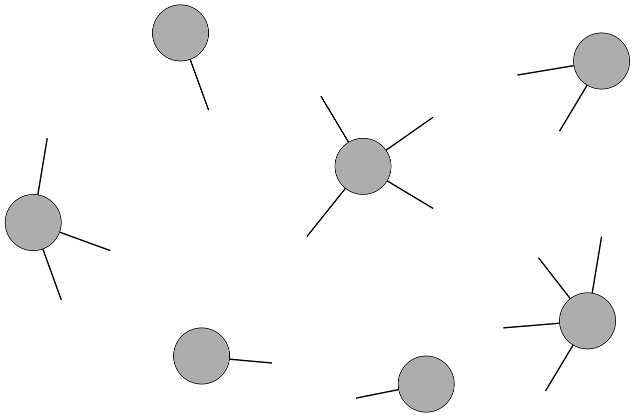
# Fast generation of random connected graphs with prescribed degrees

Fabien Viger[*,†], Matthieu Latapy[†]

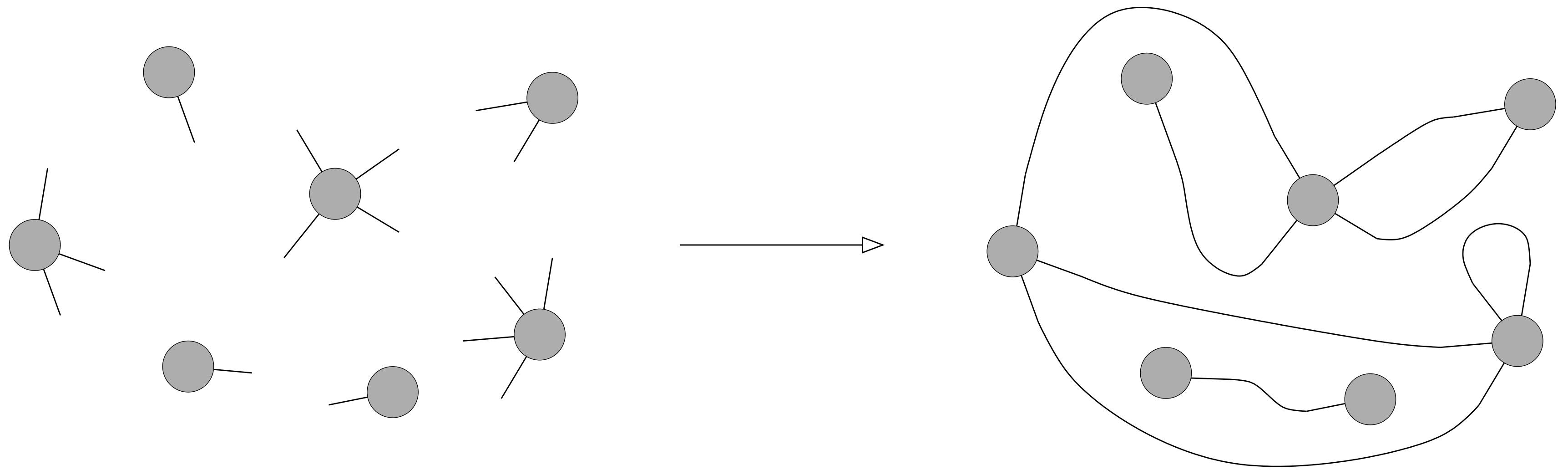
[*] LIP6, CNRS and University Pierre and Marie Curie, Paris, France

[†] LIAFA, CNRS and University Denis Diderot, Paris, France

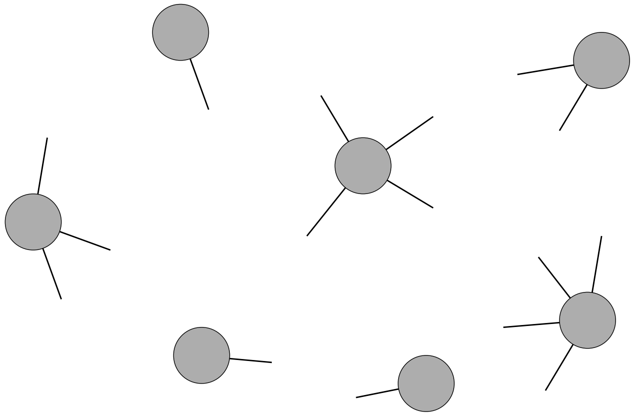
August 17th, 2005



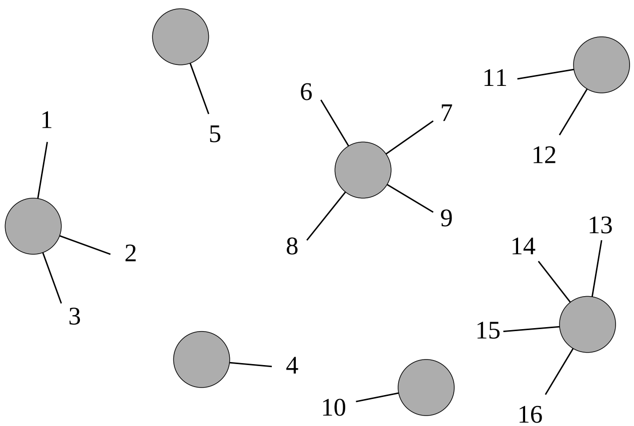
Random element in the set of all multigraphs with these degrees



Random element in the set of all multigraphs with these degrees



Random element in the set of all multigraphs with these degrees




$1 \leftrightarrow 6$

$2 \leftrightarrow 15$

$3 \leftrightarrow 16$

$4 \leftrightarrow 10$

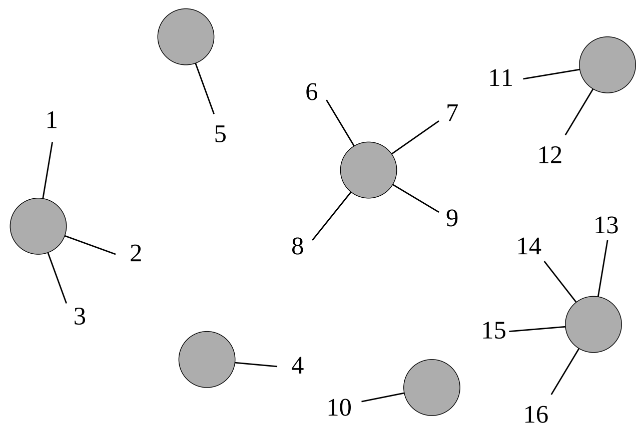
$5 \leftrightarrow 8$

$7 \leftrightarrow 11$

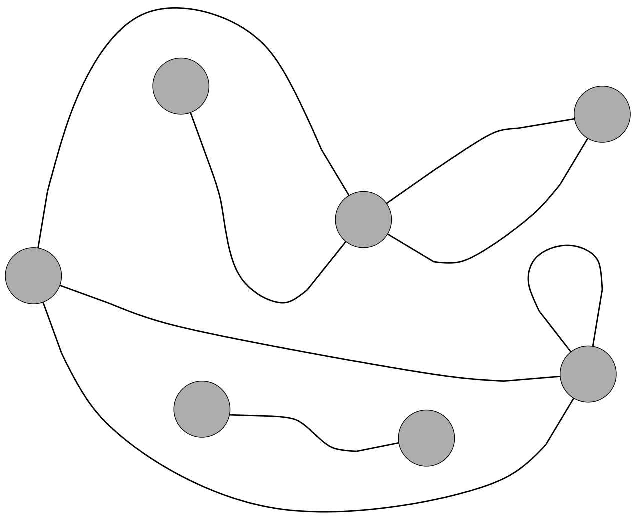
$9 \leftrightarrow 12$

$13 \leftrightarrow 14$

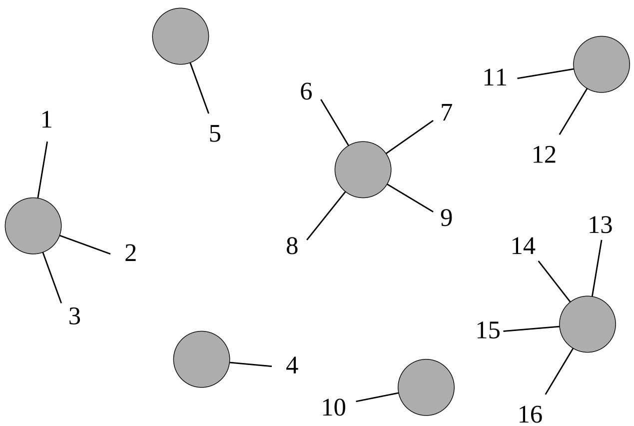
Random element in the set of all multigraphs with these degrees



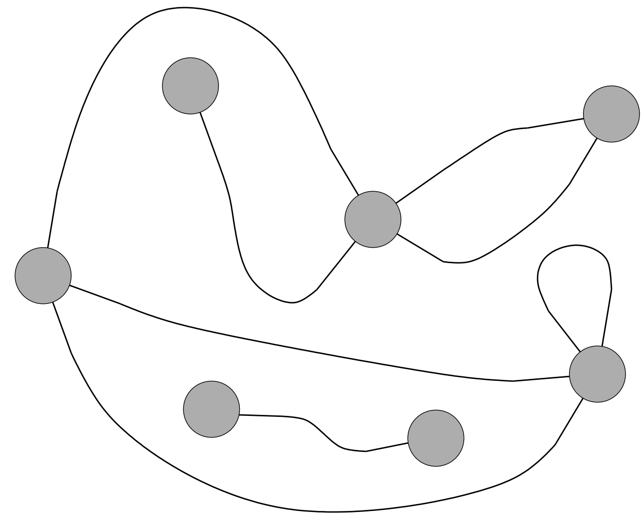

1 ⟷ 6
2 ⟷ 15
3 ⟷ 16
4 ⟷ 10
5 ⟷ 8
7 ⟷ 11
9 ⟷ 12
13 ⟷ 14



Random element in the set of all multigraphs with these degrees

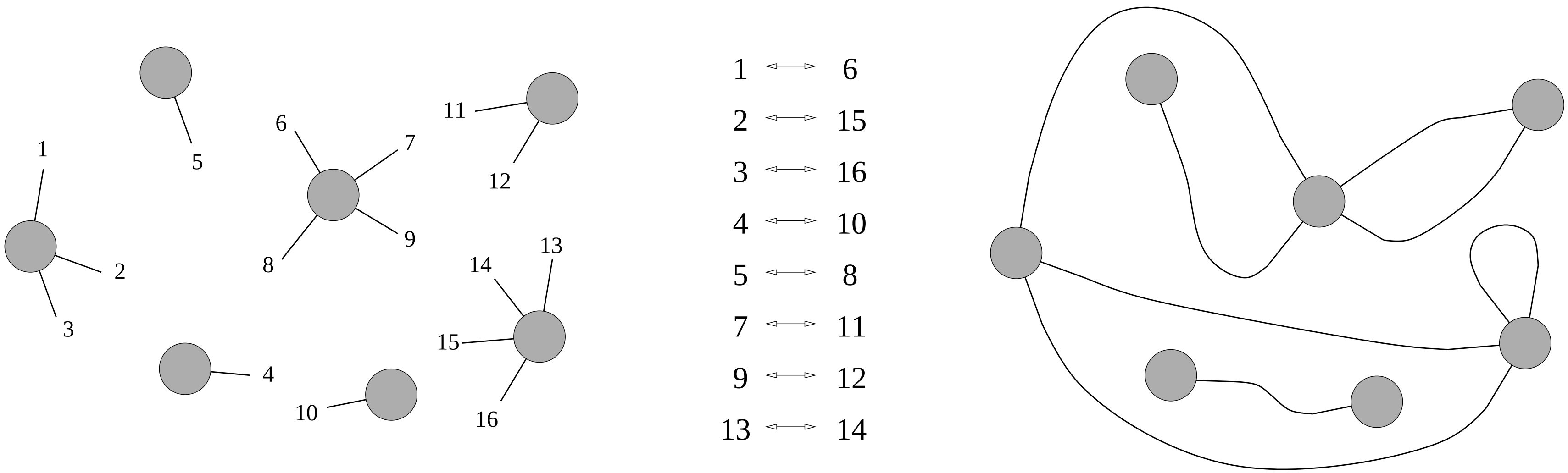



1 ⟷ 6
2 ⟷ 15
3 ⟷ 16
4 ⟷ 10
5 ⟷ 8
7 ⟷ 11
9 ⟷ 12
13 ⟷ 14



Rigorous randomness and linear complexity. . .

Random element in the set of all multigraphs with these degrees




Rigorous randomness and linear complexity. . .

. . .But the graph isn't always simple and/or connected

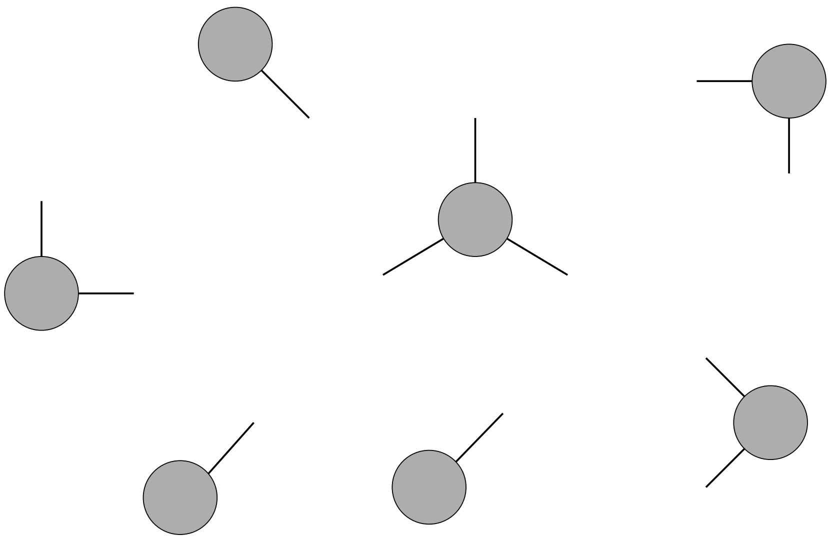
## Plan

- State of the art
- Towards optimal heuristics
- Prevent the disconnection

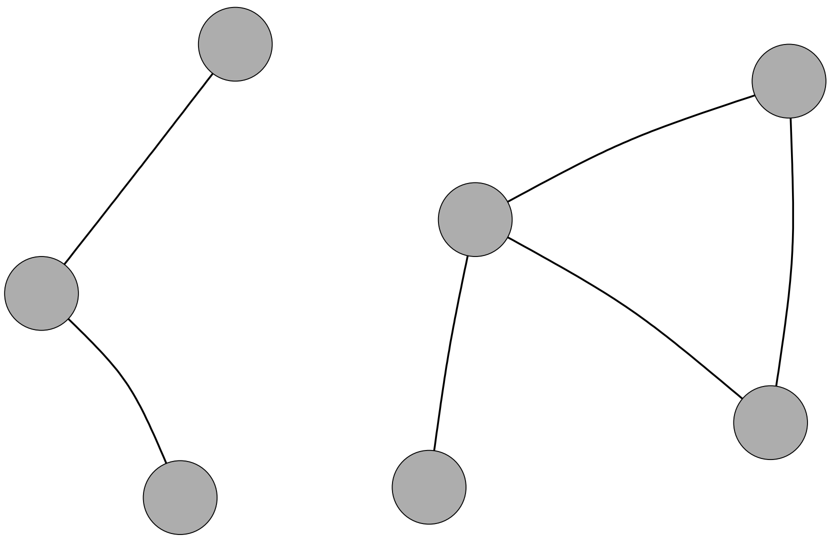
# Part I

# State of the art

## Generation of random simple connected graphs with prescribed degrees

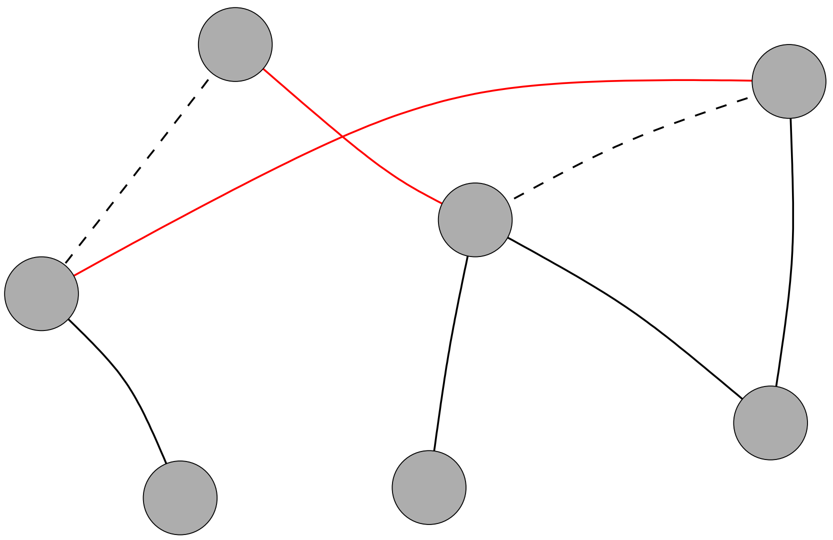


Simple



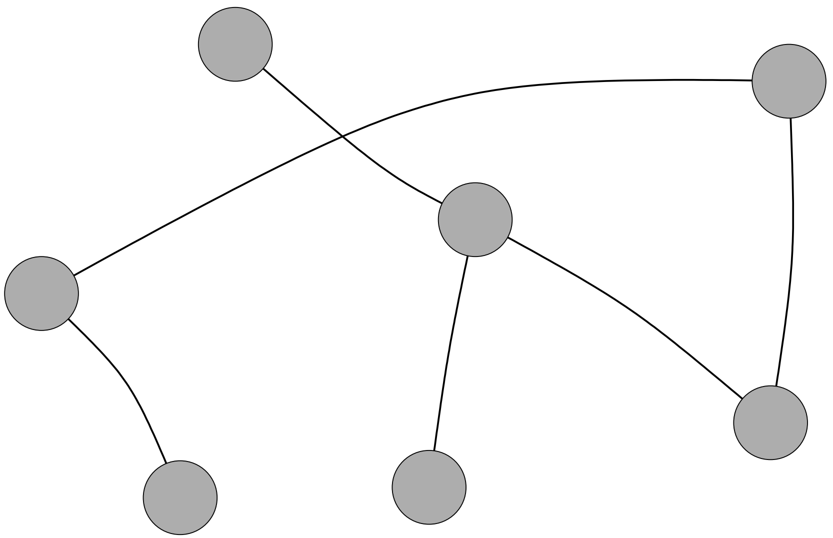
◇ **Realize** the degree sequence : **linear** (Havel-Hakimi 1955)

Simple



- **Realize** the degree sequence : **linear** (Havel-Hakimi 1955)
- **Connection** : **linear** number of edge swaps

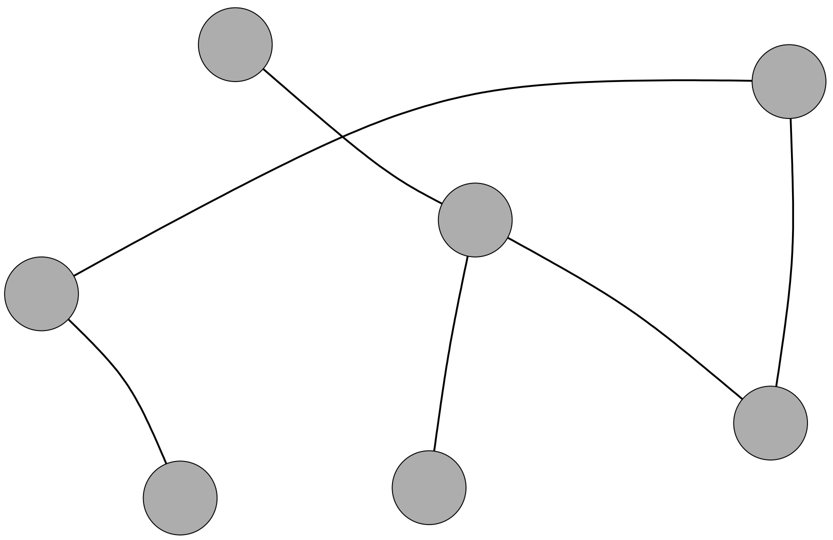
Simple
Connected



- **Realize** the degree sequence : **linear** (Havel-Hakimi 1955)
- **Connection** : **linear** number of edge swaps
  - ▷ At this point, the graph is highly **biased**

Simple

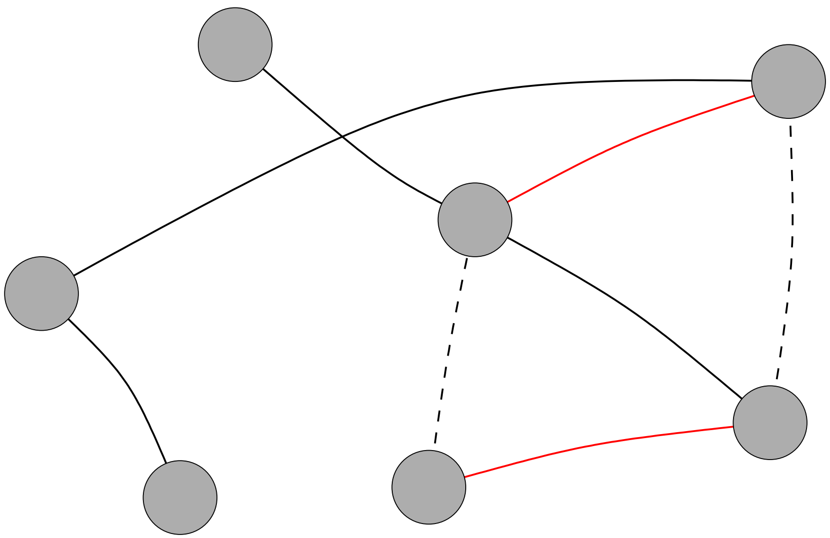
Connected



- **Realize** the degree sequence : **linear** (Havel-Hakimi 1955)
- **Connection** : **linear** number of edge swaps
- **Shuffle** : perform a certain number of random edge swaps that **keep the graph simple and connected**

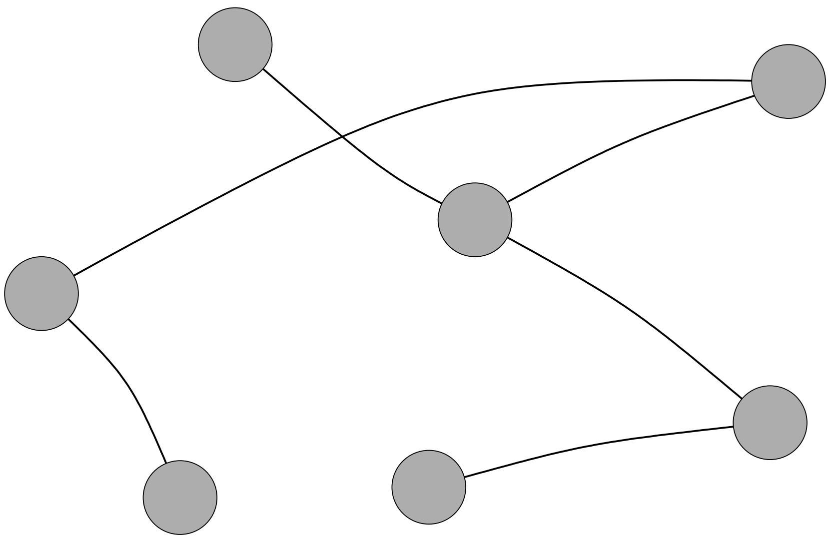
Simple

Connected



- **Realize** the degree sequence : **linear** (Havel-Hakimi 1955)
- **Connection** : **linear** number of edge swaps
- **Shuffle** : perform a certain number of random edge swaps that **keep the graph simple and connected**

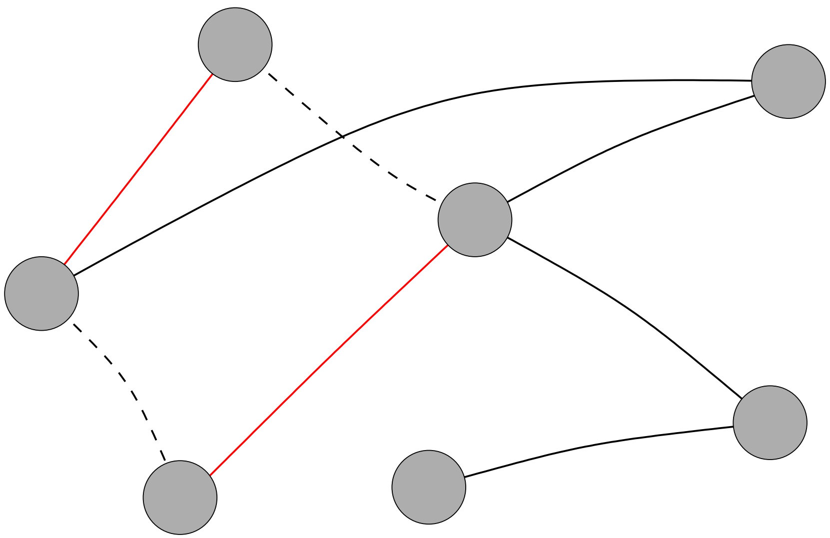
Simple
Connected



- **Realize** the degree sequence : **linear** (Havel-Hakimi 1955)
- **Connection** : **linear** number of edge swaps
- **Shuffle** : perform a certain number of random edge swaps that **keep the graph simple and connected**

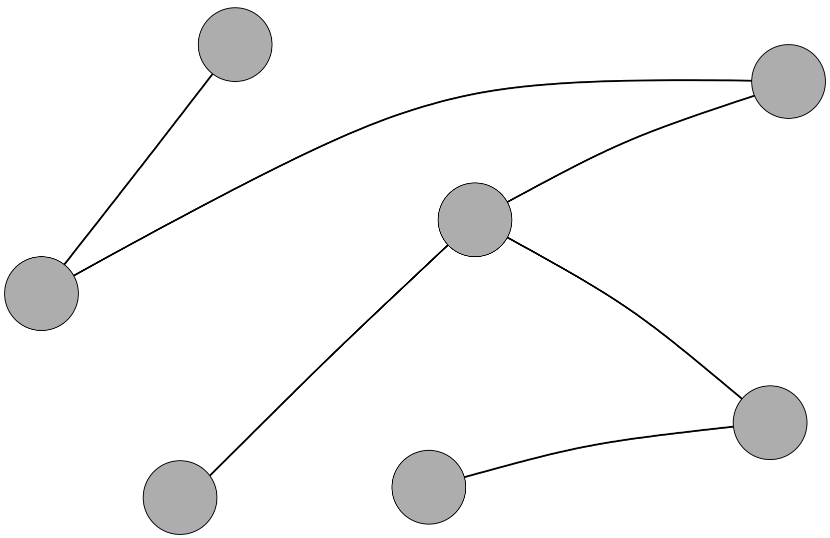
Simple

Connected

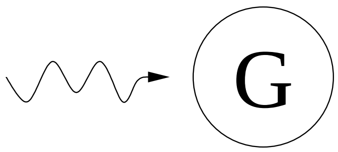


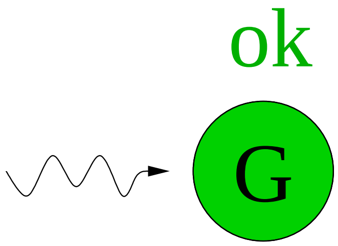
- **Realize** the degree sequence : **linear** (Havel-Hakimi 1955)
- **Connection** : **linear** number of edge swaps
- **Shuffle** : perform a certain number of random edge swaps that **keep the graph simple and connected**

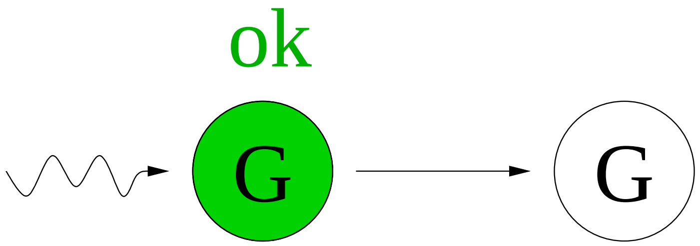
Simple
Connected
Random

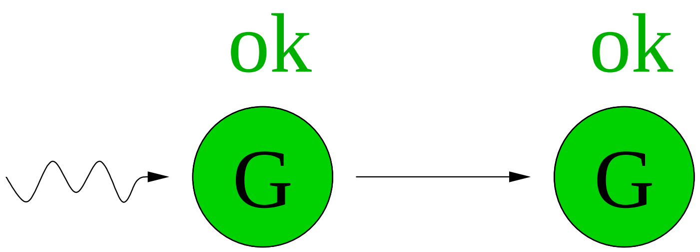


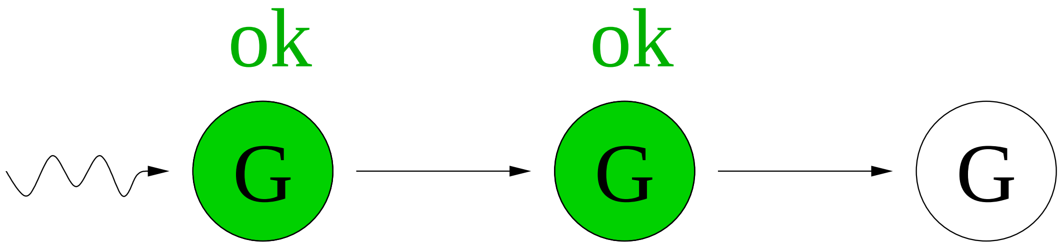
- **Realize** the degree sequence : **linear** (Havel-Hakimi 1955)
- **Connection** : **linear** number of edge swaps
- **Shuffle** : perform a certain number of random edge swaps that **keep the graph simple and connected**

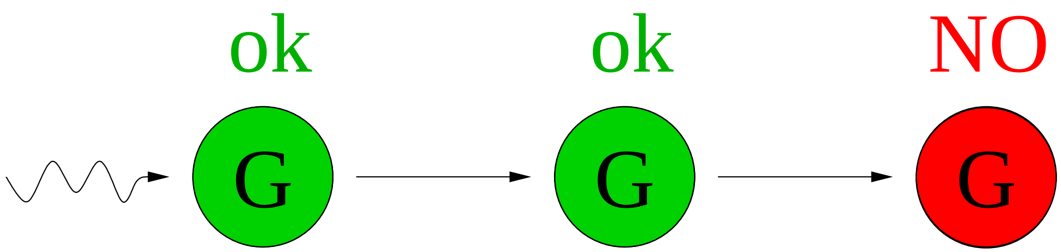

G

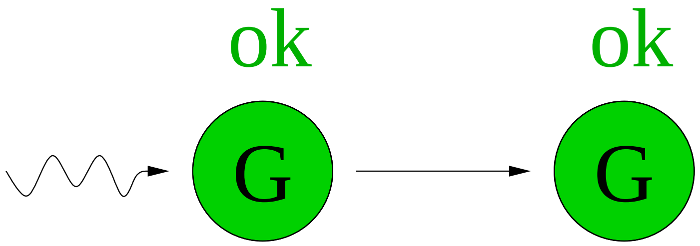

ok
G

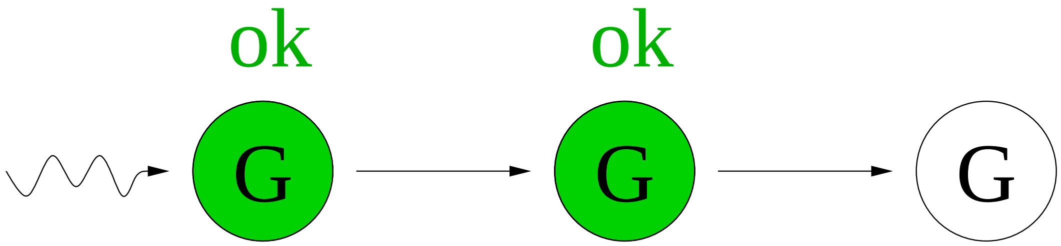

ok
G
G

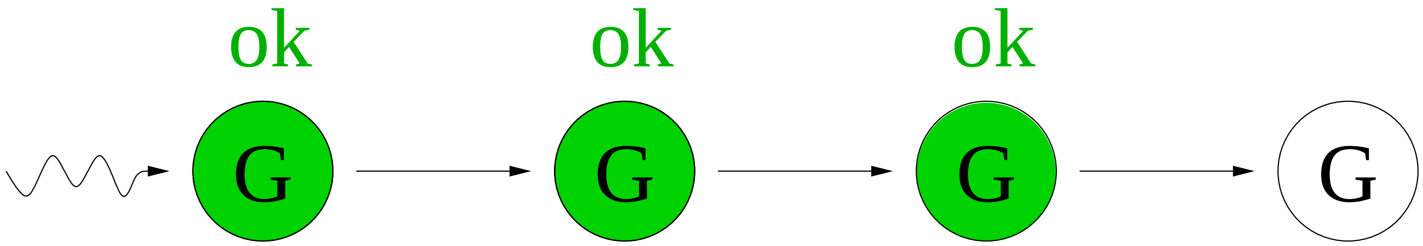

ok
ok
G
G


ok
ok
G
G
G

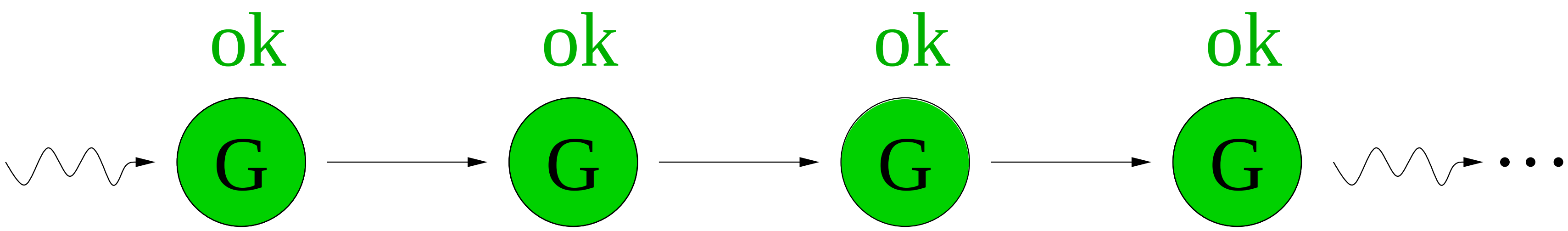

ok
ok
NO
G
G
G


ok
ok
G
G


ok
ok
G
G
G



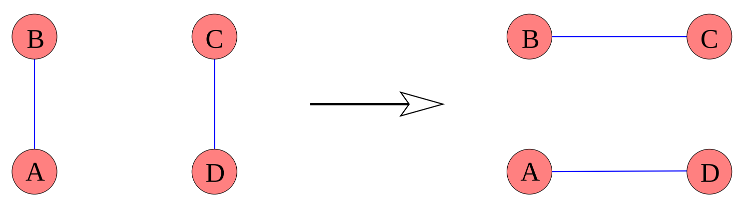
ok
ok
ok
G
G
G
G


ok
ok
ok
ok
G
G
G
G
...

# The shuffle seen as a Markov chain

## Generation of simple connected graphs

- State space : all simple connected graphs with the right degrees
- Initial state : graph obtained after the first two steps
- Transitions : *valid* edges swaps




▷ **Theorem** (Taylor 1982) :

This Markov chain is ergodic and symmetric. It converges towards the **uniform** distribution over all states

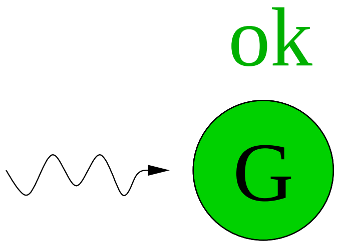
▷ Empirical result (Milo 2001, Gkantsidis 2003) :

After $O(|G|)$ transitions, no difference can be made between the graphs obtained at this point and the graphs obtained with further iterations.

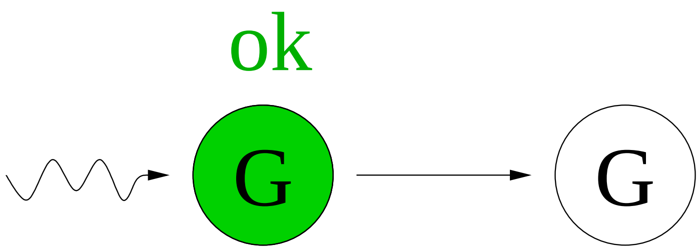
▷ But each transition takes $O(|G|)$ time (connectivity test)

▷ **Quadratic** complexity

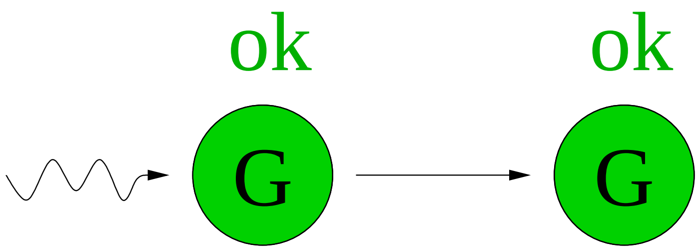
▷ Naive : One connectivity test for each transition



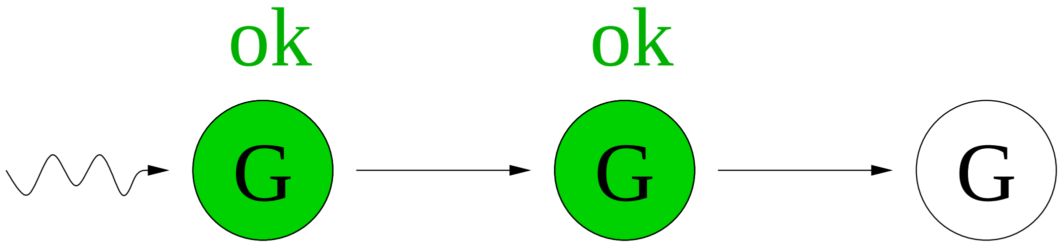
▷ Naive : One connectivity test for each transition



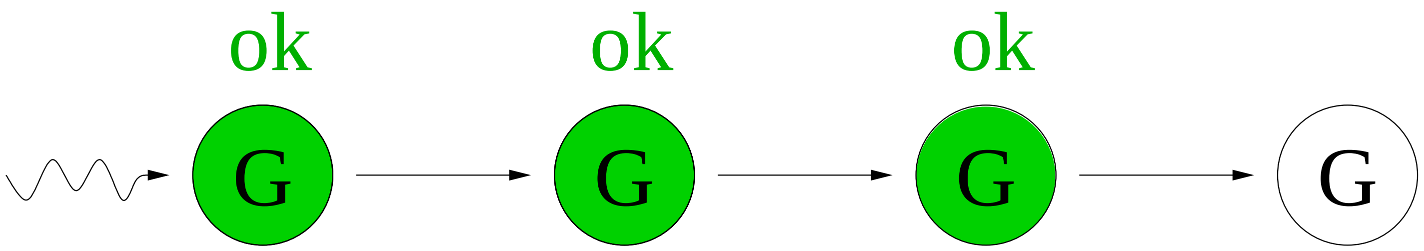
▷ Naive : One connectivity test for each transition



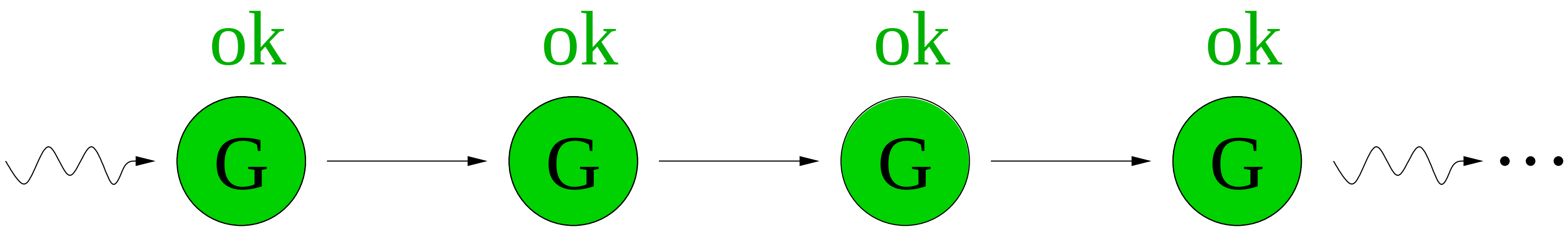
▷ Naive : One connectivity test for each transition



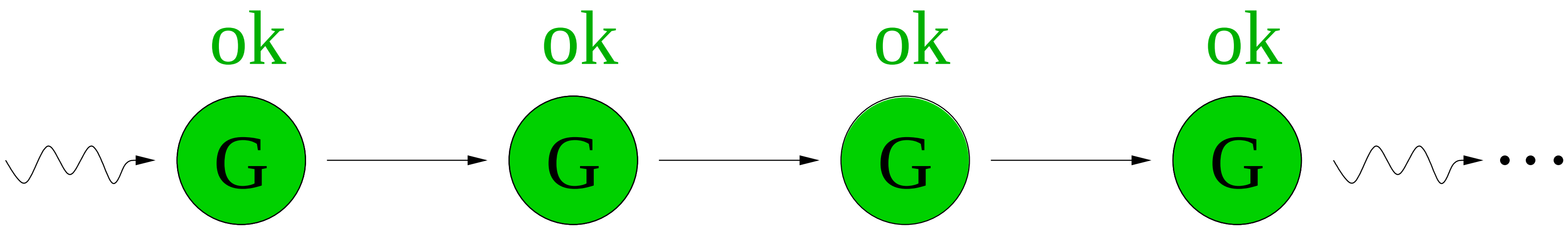
▷ Naive : One connectivity test for each transition



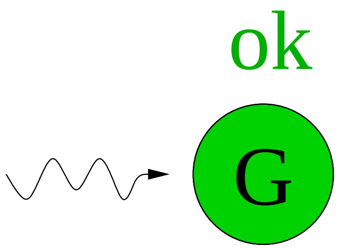
- Naive : One connectivity test for each transition



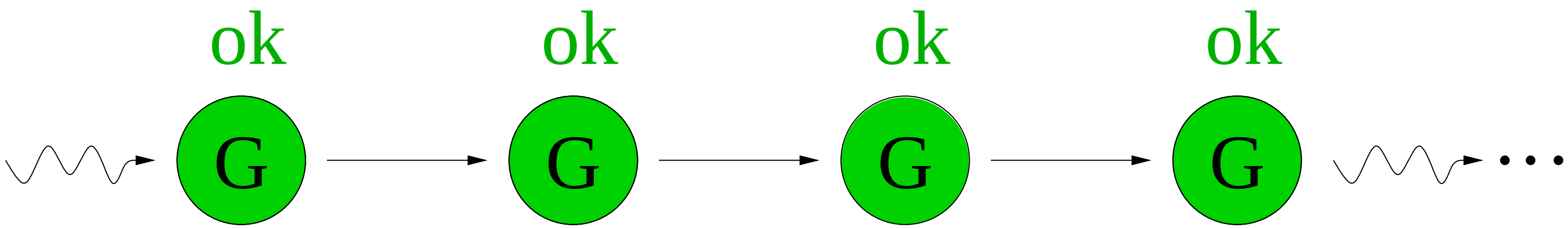
▷ Naive : One connectivity test for each transition



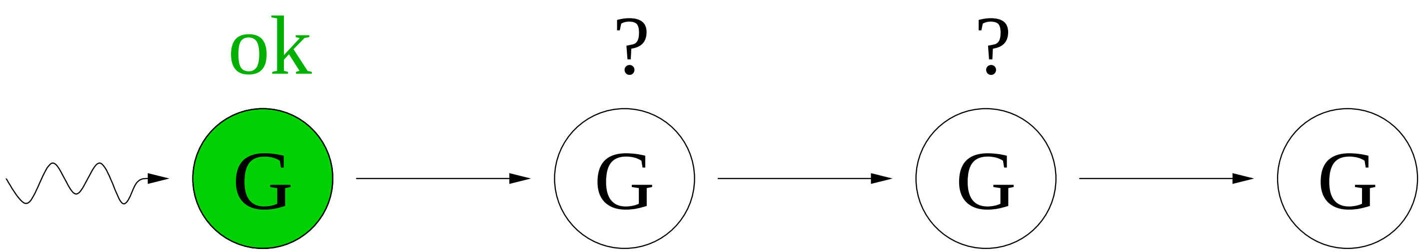

▷ Speed-up : One connectivity test every $T$ edge swaps



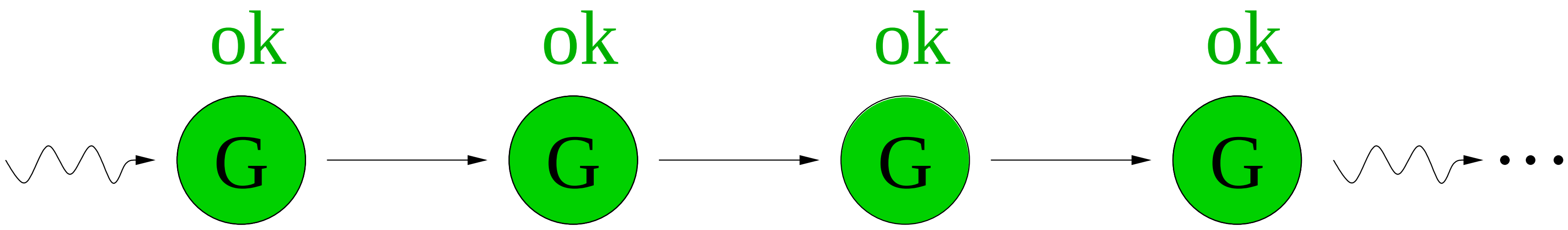
▷ Naive : One connectivity test for each transition



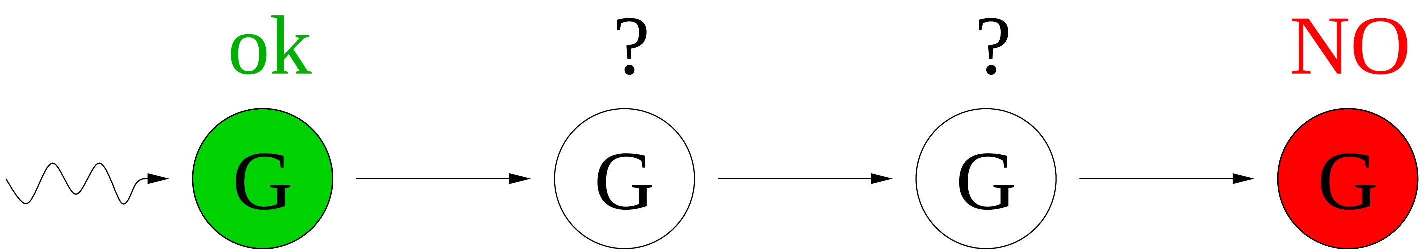

▷ Speed-up : One connectivity test every $T$ edge swaps



▷ Naive : One connectivity test for each transition



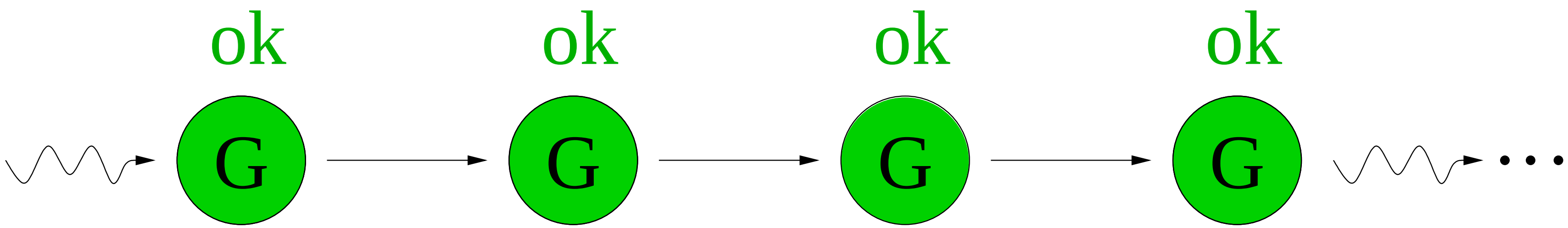

▷ Speed-up : One connectivity test every $T$ edge swaps



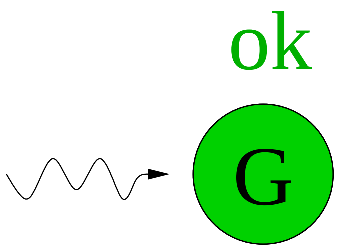
# Speed-up (Gkantsidis et al. 2003)

## Generation of simple connected graphs

▷ Naive : One connectivity test for each transition



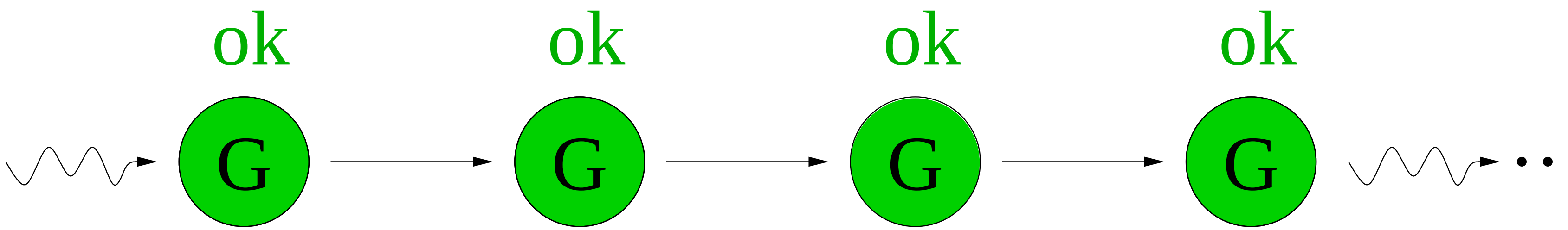

▷ Speed-up : One connectivity test every $T$ edge swaps



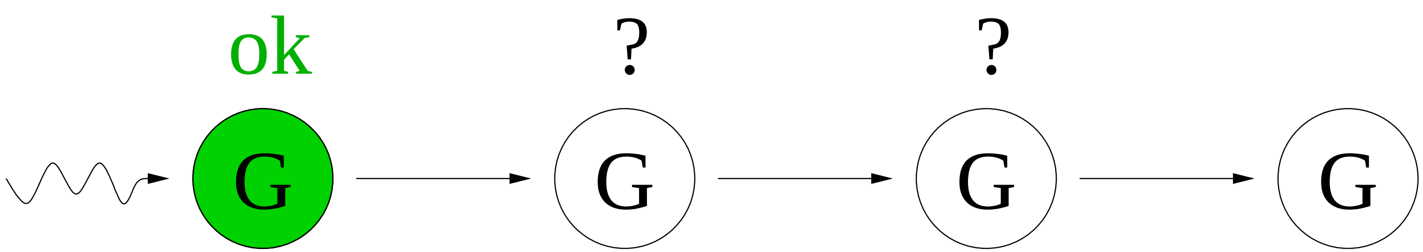
# Speed-up (Gkantsidis et al. 2003)

## Generation of simple connected graphs

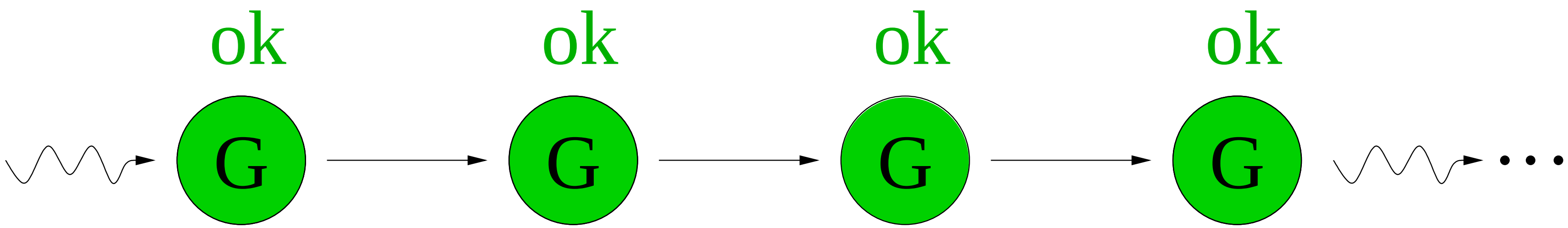
▷ Naive : One connectivity test for each transition



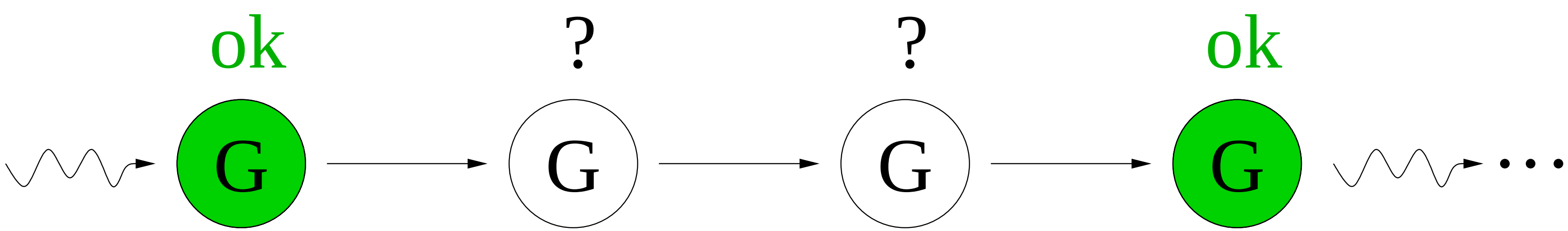

▷ Speed-up : One connectivity test every $T$ edge swaps



▷ Naive : One connectivity test for each transition



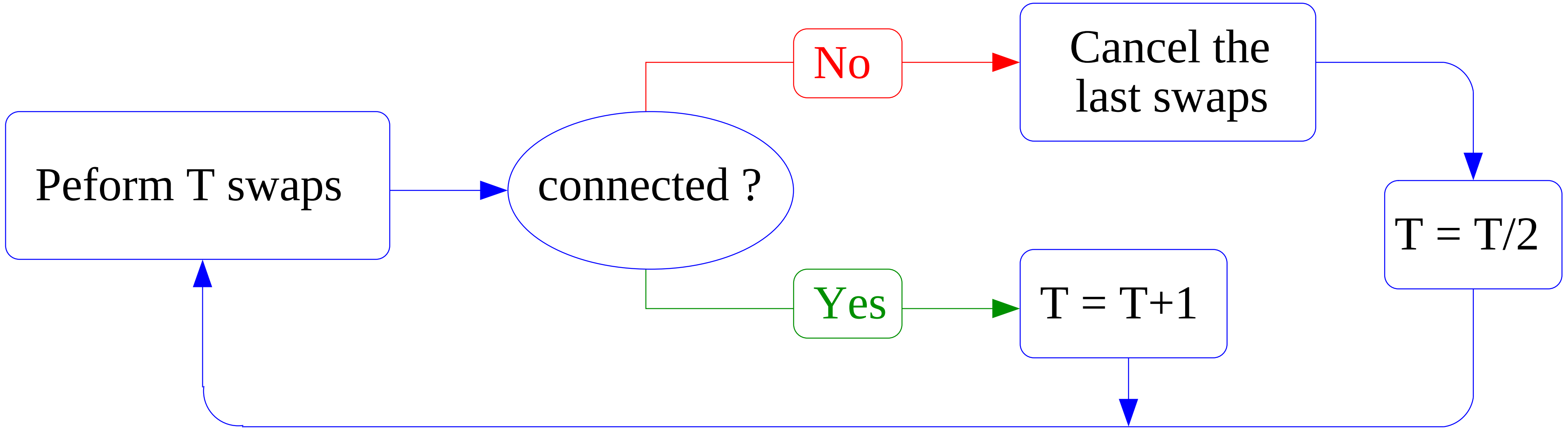

▷ Speed-up : One connectivity test every $T$ edge swaps



# Choice of the speed-up window $T$ : heuristics

## Speed-up the shuffle process

▷ Gkantsidis et al. (2003) : auto-adjust




▷ Efficiency ?

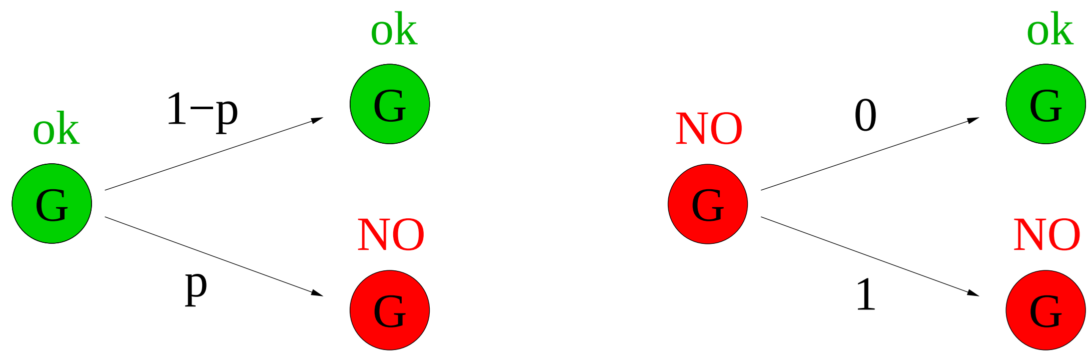
| Size | Naive | Gkan. |
|---|---:|---:|
| $1000$ | 2.9 s | 7.2 |
| $10^4$ | 6 min | 13.3 |
| $10^5$ | $\approx$10 hours | 5 |
| $10^6$ | $\approx$40 days | 2.6 |

# Part II

# Towards optimal heuristics

Formal analysis
Proposal of new heuristics

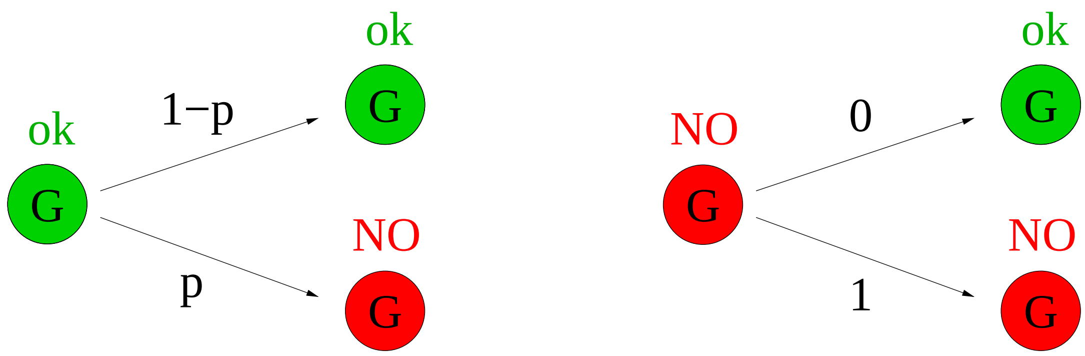
▷ Disconnection probability $p$



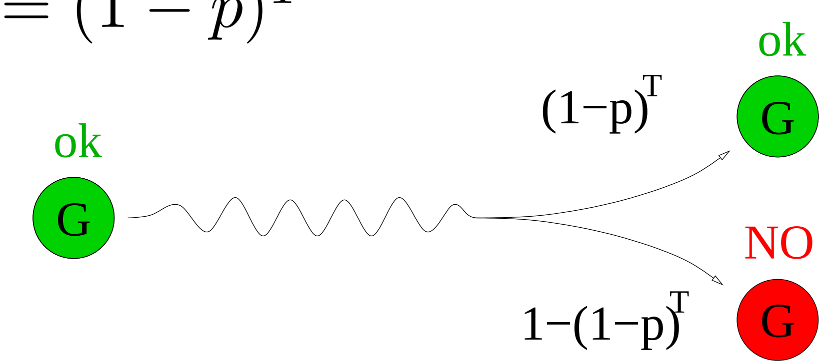
# Formal analysis

## Towards optimal heuristics

▷ Disconnection probability $p$

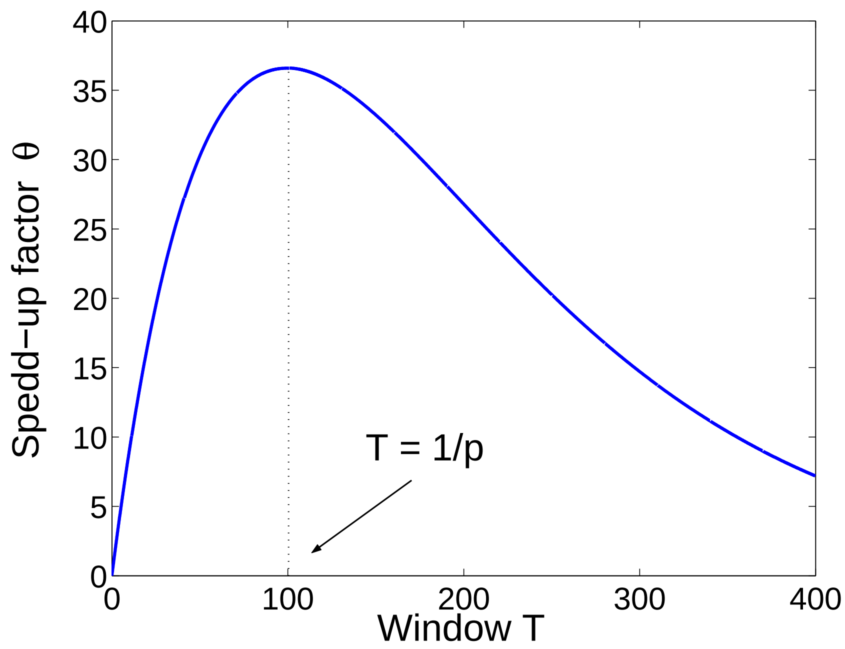


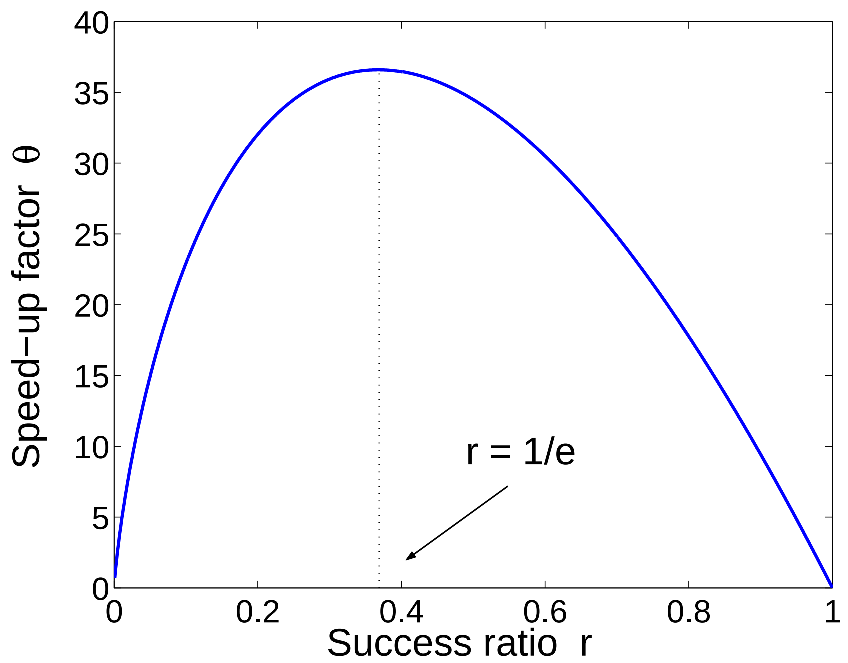

▷ Success ratio $r = (1-p)^T$




▷ Speed-up factor $\theta = r \cdot T = T \cdot (1-p)^T$

▷ **Speed-up factor** $\theta = T \cdot (1-p)^T$




$\theta$ is maximal when $T = 1/p$ i.e. $r = 1/e$ and $\theta_{max} = \frac{1}{p \cdot e}$

- Auto-stabilisation of the window $T$ towards a steady state

$$r \cdot (T+1) + (1-r) \cdot \frac{T}{2} \;=\; T$$

- The steady-state success rate is very close to $1$

- Speed-up ratio obtained : $\theta \sim \sqrt{\theta_{max}}$

**Success** $\Rightarrow$ $T = T * (1 + q^+)$ instead of $T = T + 1$

**Failure** $\Rightarrow$ $T = T * (1 - q^-)$ instead of $T = T/2$

- ▷ Steady-state window only depends on the ratio $q^+/q^-$
- ▷ Optimality condition $T_{steady} = \frac{1}{p}$ is satisfied $\iff \frac{q^+}{q^-} = e - 1$
- ▷ Speed-up factor close to $\theta_{max}$

- Definition of the optimal heuristics

- Comparison of the speed-up factors

| $n$ | $z$ | $\theta_{Gk}$ | $\theta$ | $\theta_{opt}$ |
|---|---|---|---|---|
| $10^4$ | 2.1 | 0.79 | 0.88 | 0.90 |
| $10^4$ | 3 | 3.00 | 5.00 | 5.19 |
| $10^4$ | 6 | 20.9 | 112 | 117 |
| $10^4$ | 20 | 341 | 35800 | 37000 |

- 90% close to the optimal

| Size | Naive | Gkan. | Opt. Heur. |
|---|---|---|---|
| $1000$ | 2.9 s | 7.2 | 11.4 |
| $10^4$ | 6 min | 13.3 | 50 |
| $10^5$ | ≈10 hours | 5 | 11.8 |
| $10^6$ | ≈40 days | 2.6 | 5 |

Part III

# Prevent the disconnection

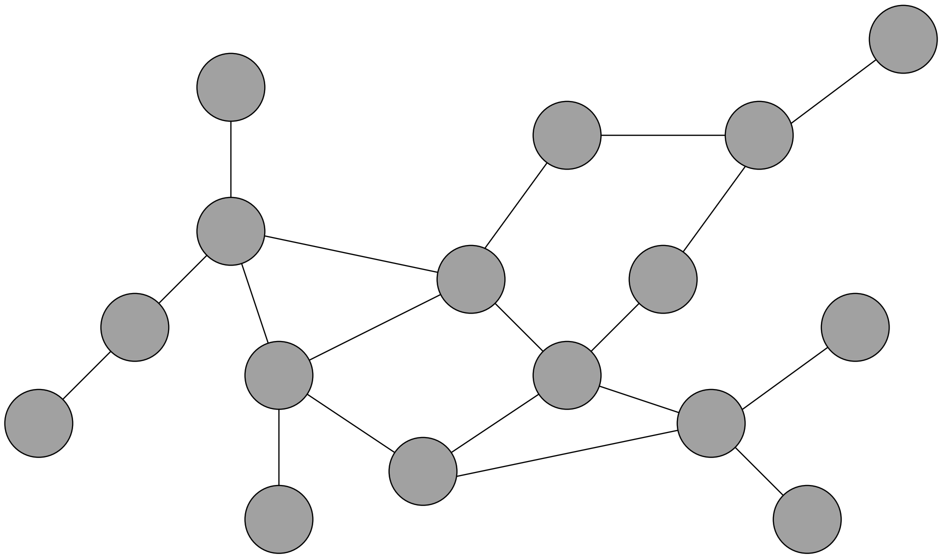
**Decrease the disconnection probability $p$**

▷ Idea

*decrease* $p$ to raise the speed-up factor $\theta$

▷ How ?

Avoid the formation of *isolated pairs*

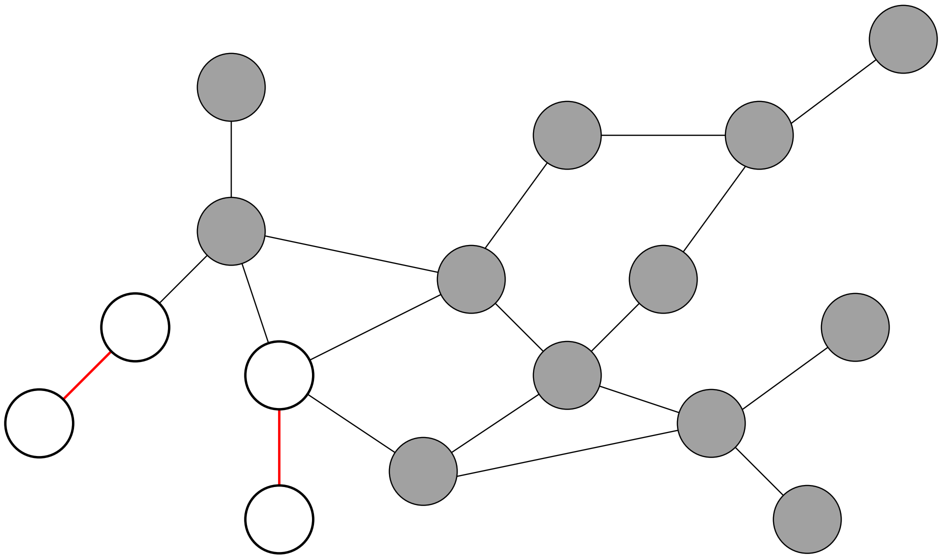


▷ Idea

*decrease* $p$ to raise the speed-up factor $\theta$

▷ How ?

Avoid the formation of *isolated pairs*

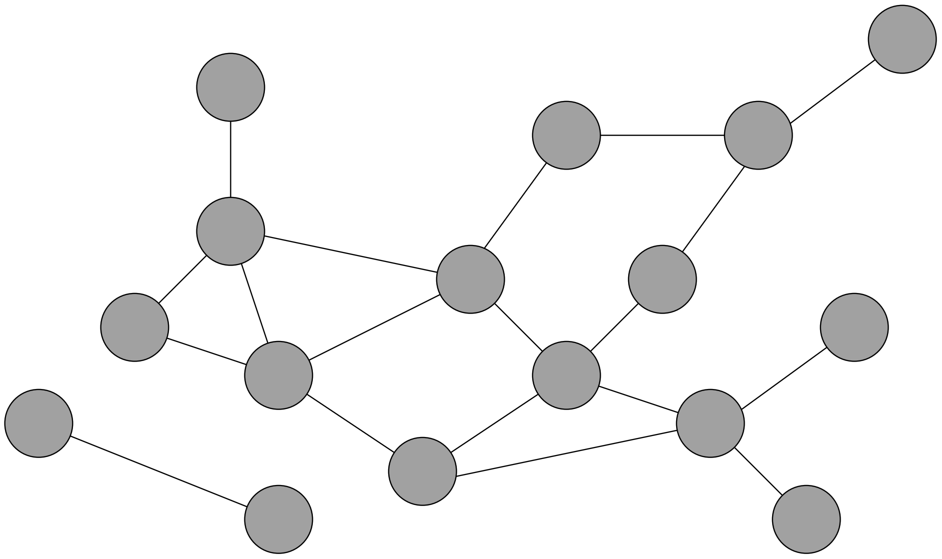


▷ Idea

*decrease* $p$ to raise the speed-up factor $\theta$

▷ How ?

Avoid the formation of *isolated pairs*

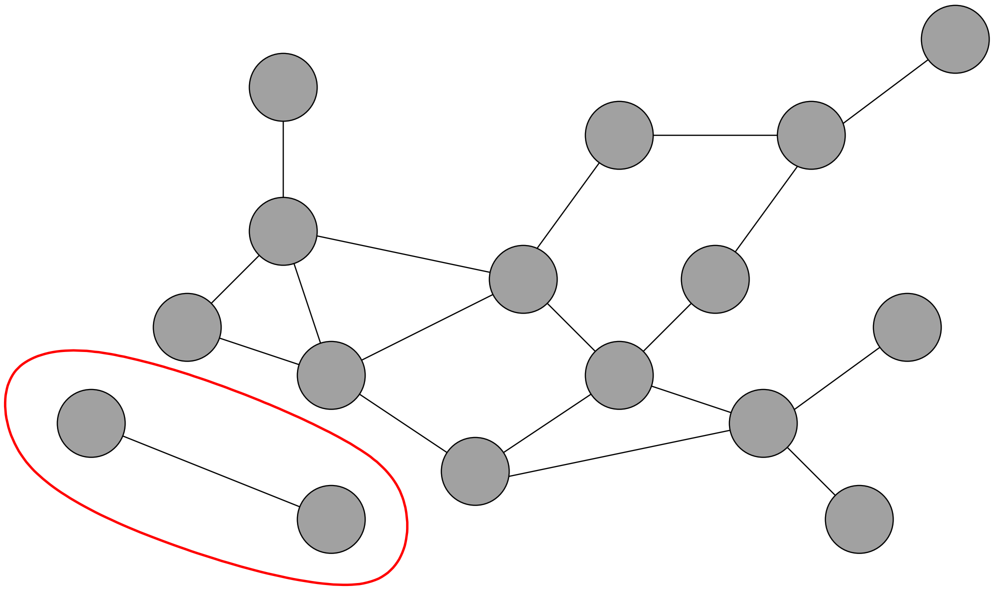


▷ Idea

*decrease* $p$ to raise the speed-up factor $\theta$

▷ How ?

Avoid the formation of *isolated pairs*



# Prevent the disconnection

$\triangleright$ Idea

*decrease* $p$ to raise the speed-up factor $\theta$

$\triangleright$ How ?

Avoid the formation of *isolated pairs*

In practice, reduction factor from $1/2$ to $1/20$

▷ Detect and avoid the formation of *small* isolated components

## Going further : $K$-isolation tests

### Prevent the disconnection

- Detect and avoid the formation of *small* isolated components
  - For every edge swap, perform two $K$-limited breadth- or depth-first search from the vertices that might have been disconnected

## Going further : $K$-isolation tests

### Prevent the disconnection

- Detect and avoid the formation of *small* isolated components
  - For every edge swap, perform two $K$-limited breadth- or depth-first search from the vertices that might have been disconnected
  - If a small component is detected, cancel the swap rightaway

# Going further : $K$-isolation tests

## Prevent the disconnection

- Detect and avoid the formation of *small* isolated components
  - For every edge swap, perform two $K$-limited breadth- or depth-first search from the vertices that might have been disconnected
  - If a small component is detected, cancel the swap rightaway
  - If not, validate the swap

- Detect and avoid the formation of *small* isolated components
  - For every edge swap, perform two $K$-limited breadth- or depth-first search from the vertices that might have been disconnected
  - If a small component is detected, cancel the swap rightaway
  - If not, validate the swap

- Time complexity $O(K)$ per edge swap, instead of $O(1)$

# Going further : $K$-isolation tests

## Prevent the disconnection

- Detect and avoid the formation of *small* isolated components
  - For every edge swap, perform two $K$-limited breadth- or depth-first search from the vertices that might have been disconnected
  - If a small component is detected, cancel the swap rightaway
  - If not, validate the swap

- Time complexity $O(K)$ per edge swap, instead of $O(1)$
- But the lower probability $p$ causes a raise of the speed-up factor $\theta$

- **Detect** and **avoid** the formation of *small* isolated components
  - For every edge swap, perform two $K$-limited breadth- or depth-first search from the vertices that might have been disconnected
  - If a small component is detected, cancel the swap rightaway
  - If not, validate the swap

- Time complexity $O(K)$ per edge swap, instead of $O(1)$
- But the lower probability $p$ causes a raise of the speed-up factor $\theta$
  - How much will $p$ decrease ?

# Going further : $K$-isolation tests

## Prevent the disconnection

- ▷ Detect and avoid the formation of *small* isolated components
  - ◇ For every edge swap, perform two $K$-limited breadth- or depth-first search from the vertices that might have been disconnected
  - ◇ If a small component is detected, cancel the swap rightaway
  - ◇ If not, validate the swap

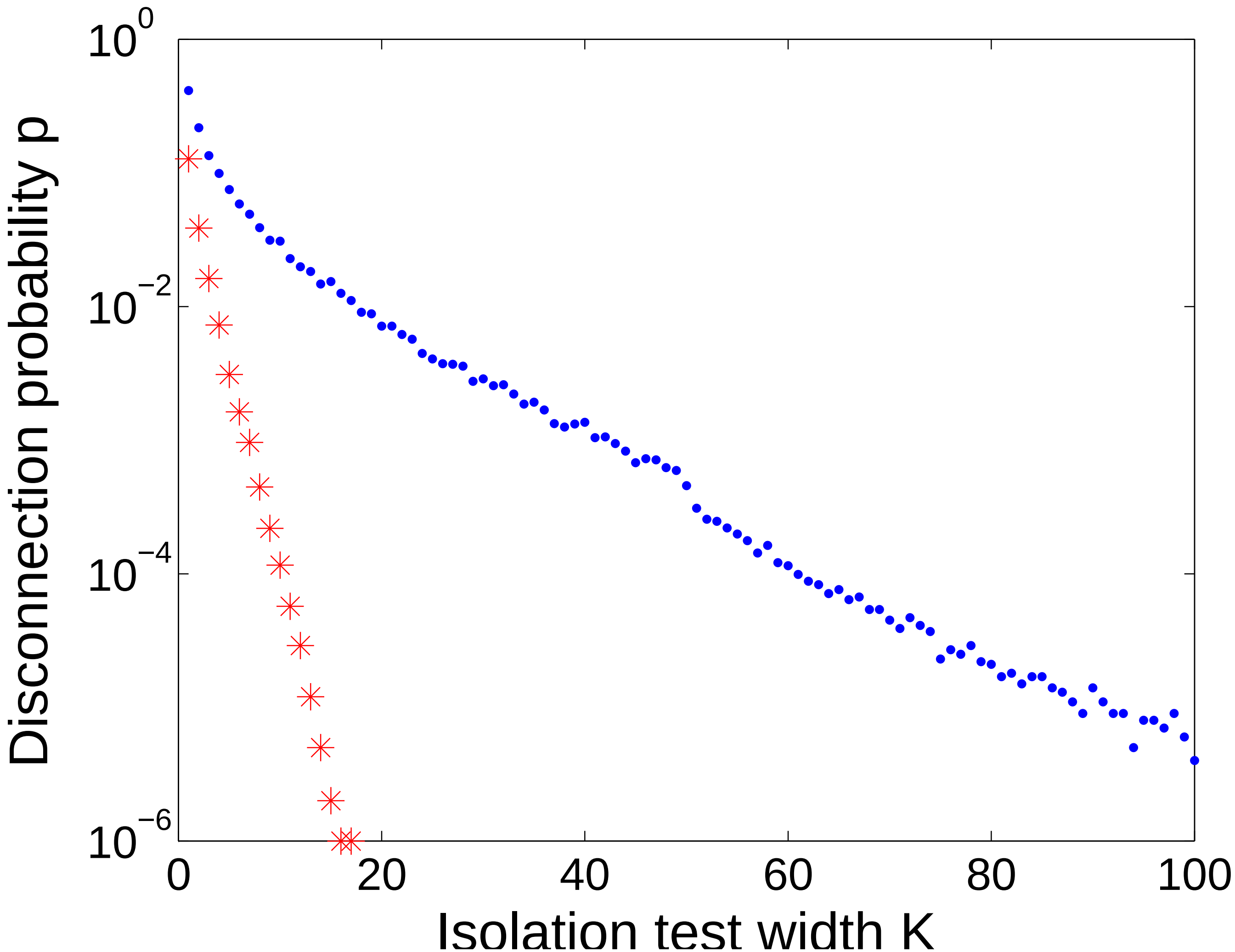
- ▷ Time complexity $O(K)$ per edge swap, instead of $O(1)$
- ▷ But the lower probability $p$ causes a raise of the speed-up factor $\theta$
  - ◇ How much will $p$ decrease ?
  - ◇ Intuition : $K$ vertices are $K$-exponentially unlikely to be isolated

- Detect and avoid the formation of *small* isolated components
  - For every edge swap, perform two $K$-limited breadth- or depth-first search from the vertices that might have been disconnected
  - If a small component is detected, cancel the swap rightaway
  - If not, validate the swap

- Time complexity $O(K)$ per edge swap, instead of $O(1)$
- But the lower probability $p$ causes a raise of the speed-up factor $\theta$
  - How much will $p$ decrease ?
  - Intuition : $K$ vertices are $K$-exponentially unlikely to be isolated
  - Consequence : $p$ would decrease exponentially with $K$ ?

# Effect on the disconnection probability

## $K$-Isolation tests



# Adjusting the isolation test width $K$



Empirically : $p \sim e^{-\lambda K}$

$$\theta_{max} = \frac{1}{p \cdot e} \Rightarrow \theta_{max} \sim e^{\lambda K}$$

▷ Exponential decrease of $C_{tests}$ (connectivity tests complexity)

▷ Linear increase of $C_{swaps}$ (complexity of edge swaps)

# Adjusting the isolation test width $K$

Empirically : $p \sim e^{-\lambda K}$

$$\theta_{max} = \frac{1}{p \cdot e} \;\Rightarrow\; \theta_{max} \sim e^{\lambda K}$$

- ▷ Exponential decrease of $C_{tests}$ (connectivity tests complexity)
- ▷ Linear increase of $C_{swaps}$ (complexity of edge swaps)
- ▷ The tradeoff consists in balancing both $C_{swaps}$ and $C_{tests}$

$$\left.\begin{array}{r} C_{swaps} = O(K \cdot |G|) \\ C_{tests} = O\left(\frac{|G|^2}{e^{\lambda K}}\right) \end{array}\right\} \;\Rightarrow\; K = O(\log |G|)$$

Empirically : $p \sim e^{-\lambda K}$

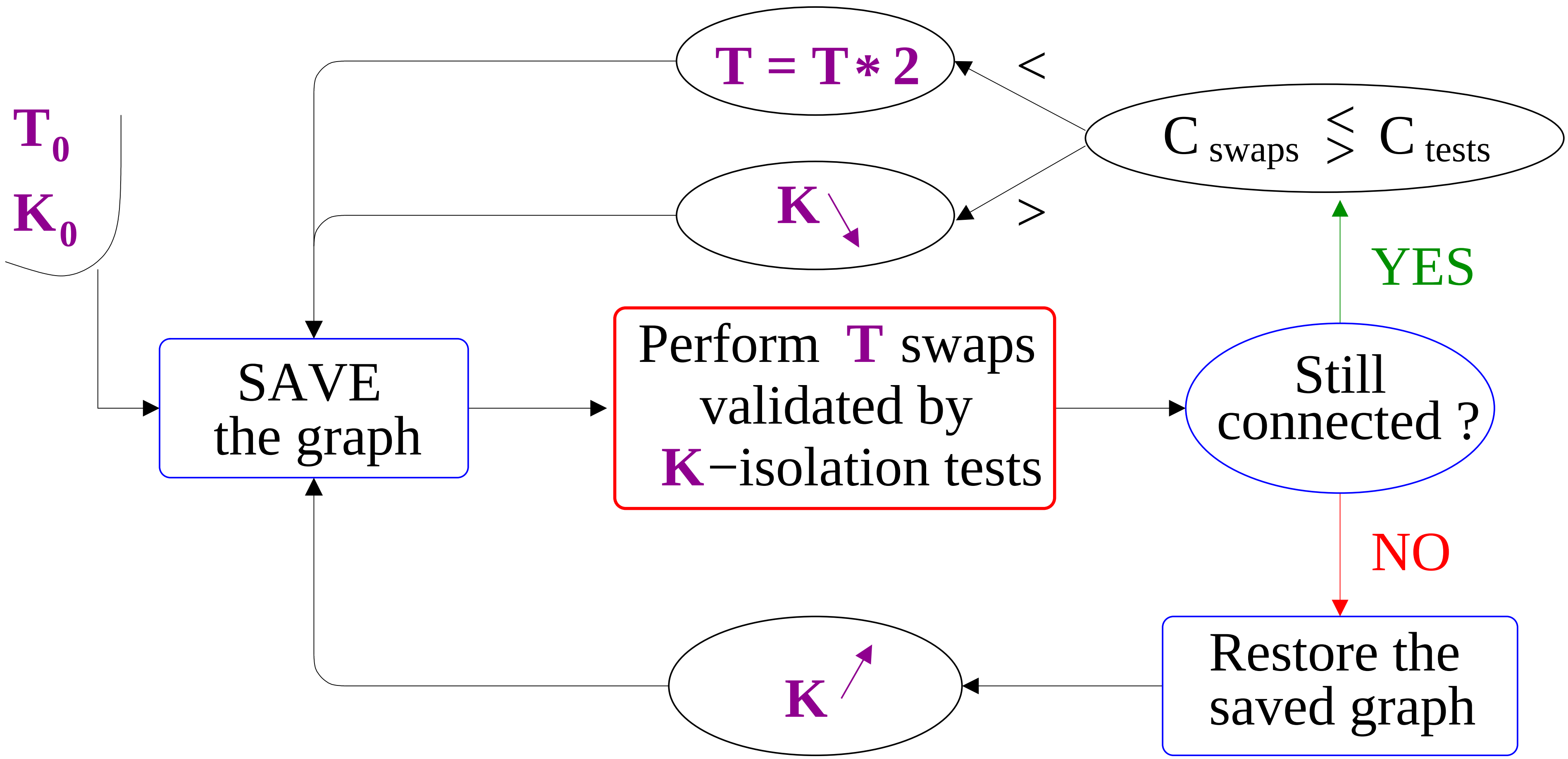
$$\theta_{max} = \frac{1}{p \cdot e} \Rightarrow \theta_{max} \sim e^{\lambda K}$$

- ▷ Exponential decrease of $C_{tests}$ (connectivity tests complexity)
- ▷ Linear increase of $C_{swaps}$ (complexity of edge swaps)
- ▷ The tradeoff consists in balancing both $C_{swaps}$ and $C_{tests}$

$$\left.\begin{array}{r} C_{swaps} = O(K \cdot |G|) \\ C_{tests} = O\left(\frac{|G|^2}{e^{\lambda K}}\right) \end{array}\right\} \Rightarrow K = O(\log |G|)$$

- ▷ Final complexity is $O(|G| \log |G|)$ instead of $O(|G|^2)$

# Adjusting the isolation test width $K$




Maybe not optimal, but works fine

# Benchmark

| Size | Naive | Gkan. | Opt. Heur. | Final |
|---|---|---|---|---|
| $1000$ | 2.9 s | 7.2 | 11.4 | 22.3 |
| $10^4$ | 6 min | 13.3 | 50 | 510 |
| $10^5$ | ≈10 hours | 5 | 11.8 | 2180 |
| $10^6$ | ≈40 days | 2.6 | 5 | 7780 |

# Part IV

# Conclusion

# Contributions

- ▷ Analysis of Gkantsidis et al. heuristics
- ▷ New heuristics, designed to reach the optimal
- ▷ Validation, benchmarks
- ▷ New idea to prevent the disconnection during the shuffle
- ▷ Log-linear algorithm. Implementation, benchmarks

# Future work

- ▷ More formal proofs
- ▷ Extension to directed graphs
- ▷ Application to some dynamic connectivity algorithms

Thank you