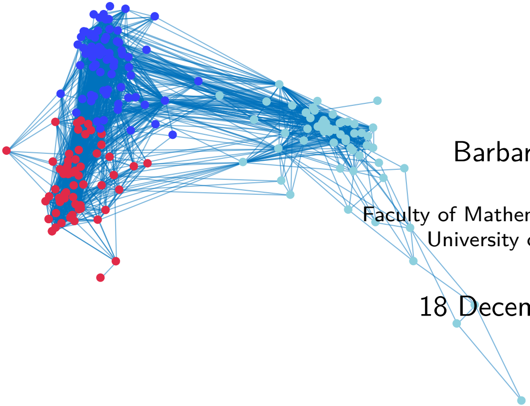
Evolutionary Dynamics on Evolving Graphs

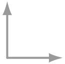


Barbara Ikica

Faculty of Mathematics and Physics
University of Ljubljana

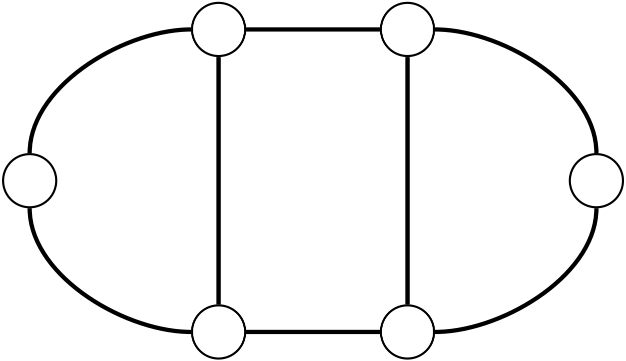
18 December 2019

Evolutionary Dynamics on Evolving Graphs

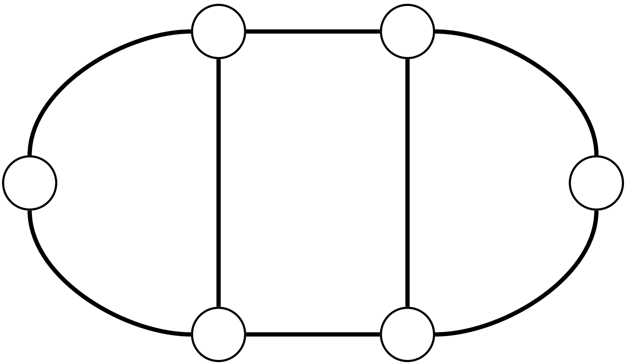


Evolutionary Dynamics on Evolving Graphs

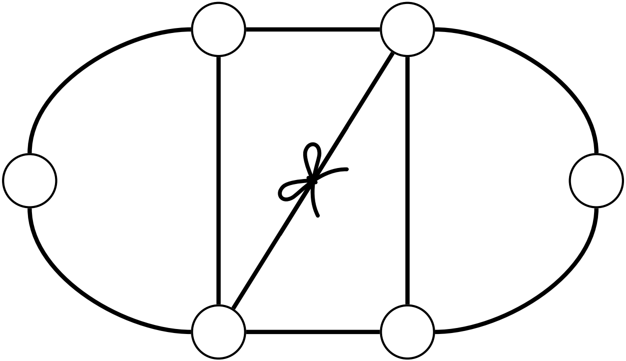
Evolutionary Dynamics on Evolving Graphs



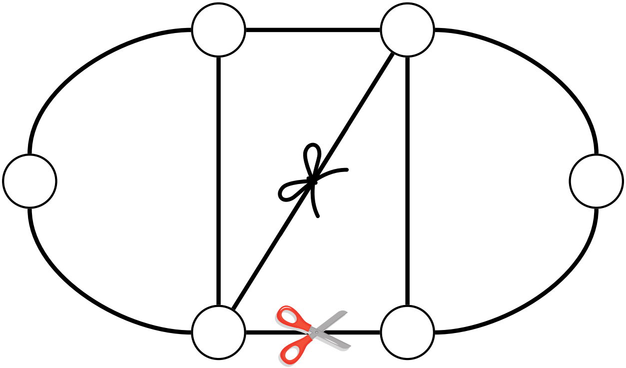
Evolutionary Dynamics on Evolving Graphs



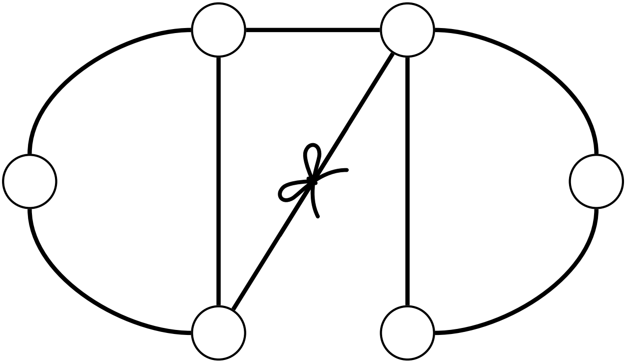
Evolutionary Dynamics on Evolving Graphs



Evolutionary Dynamics on Evolving Graphs



Evolutionary Dynamics on Evolving Graphs



Evolutionary Dynamics on Evolving Graphs

Evolutionary Dynamics on Evolving Graphs

Players/agents $\mathcal{P} =$ 

Evolutionary Dynamics on Evolving Graphs

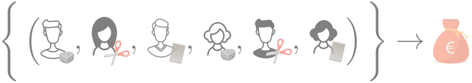
Players/agents $\mathcal{P} = \{$  $\}$

Strategies/states $S_i = \{$  $\}$

Evolutionary Dynamics on Evolving Graphs

Players/agents $\mathcal{P} =$ 

Strategies/states $S_i =$ 

Payoff $\pi_i :$ 

Evolutionary Dynamics on Evolving Graphs

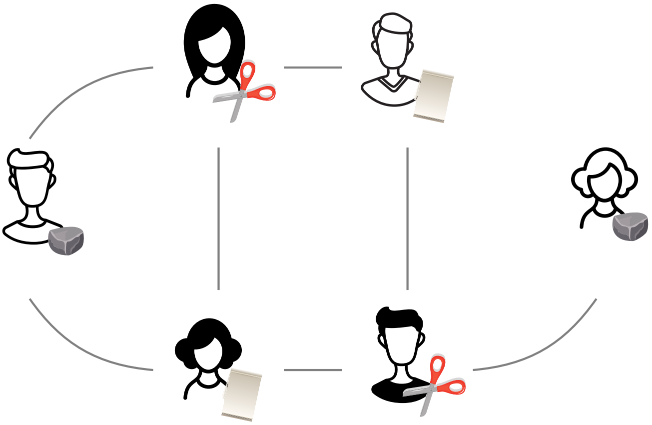
Players/agents $\mathcal{P} = \{$  $\}$

Strategies/states $S_i = \{$  $\}$

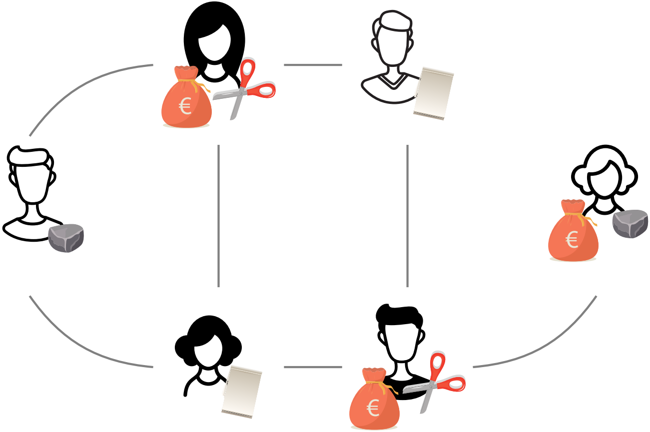
Payoff $\pi_i : \{$  $\} \to$ 

Update rule $p_{i \to j} = \left(1 + e^{-\omega[\pi_i - \pi_j]}\right)^{-1}$ (Fermi rule)

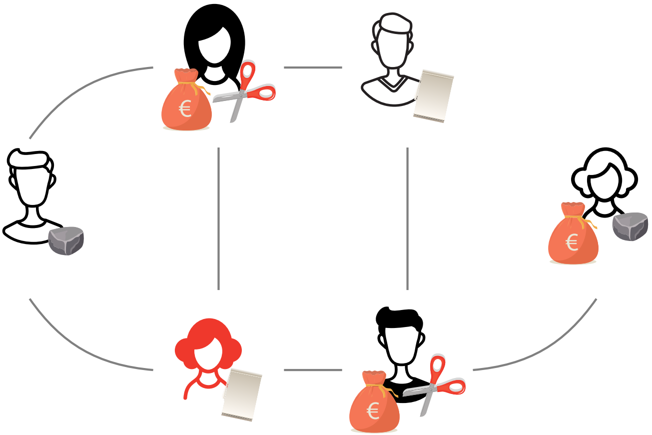
Evolutionary Dynamics on Evolving Graphs



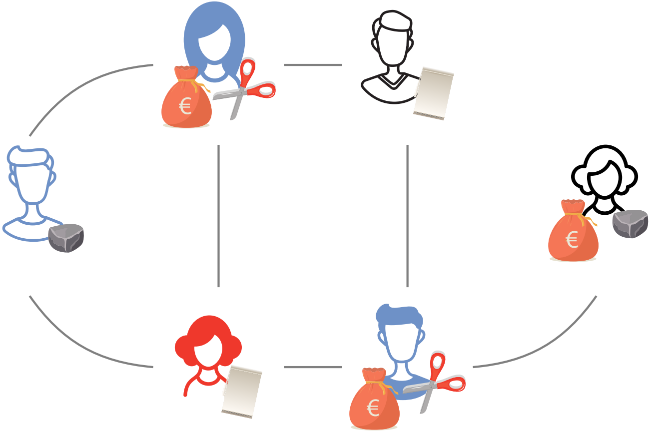
Evolutionary Dynamics on Evolving Graphs



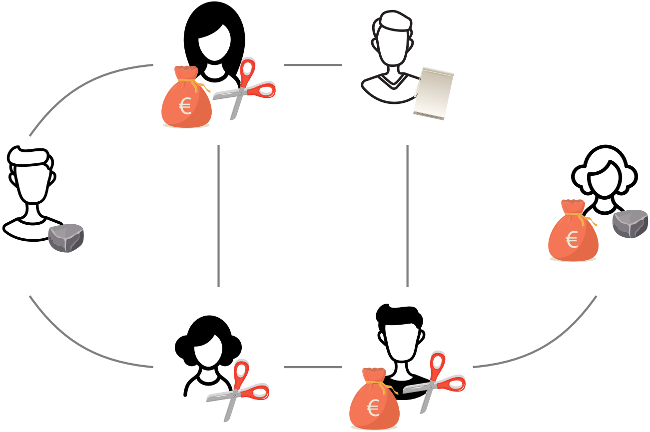
Evolutionary Dynamics on Evolving Graphs



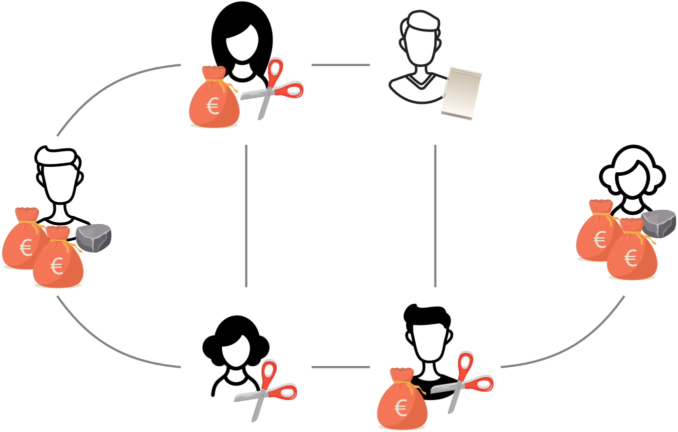
Evolutionary Dynamics on Evolving Graphs



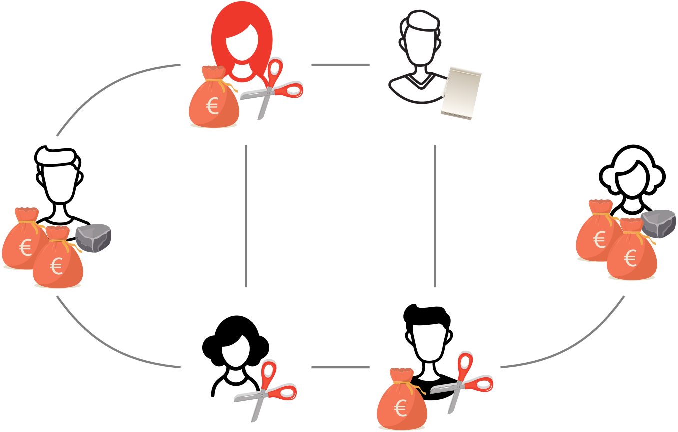
Evolutionary Dynamics on Evolving Graphs



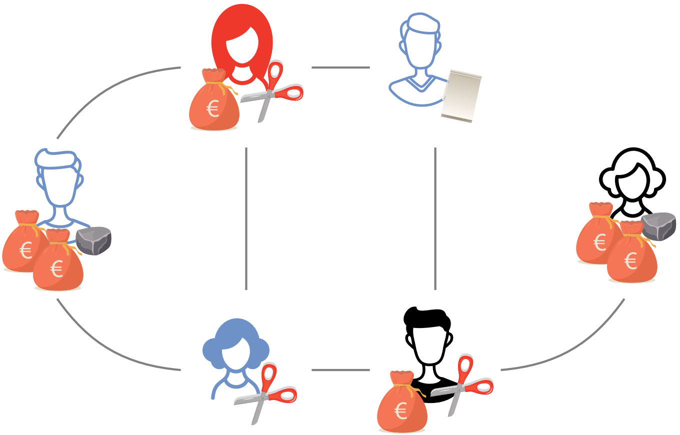
Evolutionary Dynamics on Evolving Graphs



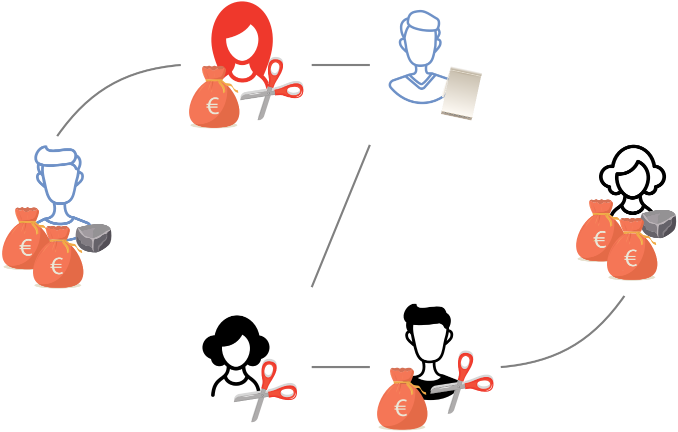
Evolutionary Dynamics on Evolving Graphs



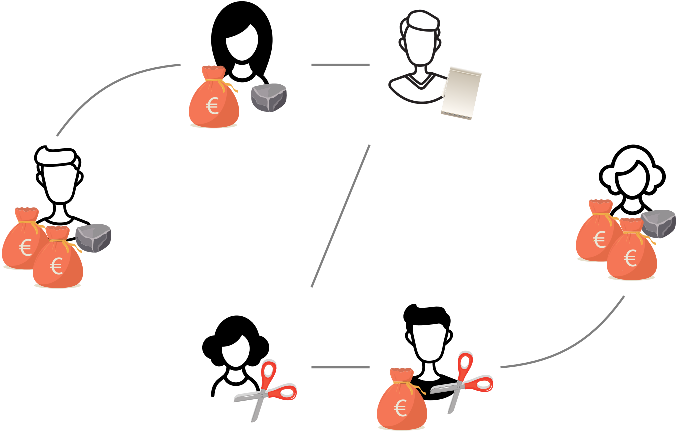
Evolutionary Dynamics on Evolving Graphs



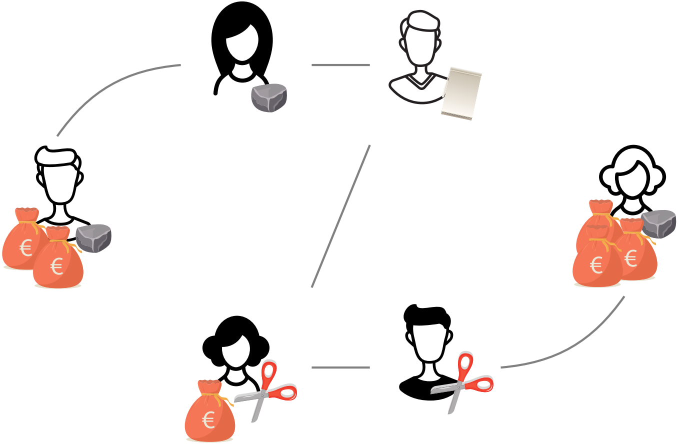
Evolutionary Dynamics on Evolving Graphs



Evolutionary Dynamics on Evolving Graphs



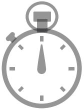
Evolutionary Dynamics on Evolving Graphs



Evolutionary Dynamics on *Evolving* Graphs

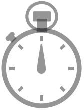
Evolutionary Dynamics on *Evolving* Graphs

$$\left.\begin{array}{l} t_D \\ t_G \end{array}\right\} := \text{time scale of the} \left\{\begin{array}{l}\text{dynamics}\\ \text{graph}\end{array}\right.$$



Evolutionary Dynamics on *Evolving* Graphs

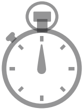
$$\left.\begin{array}{l} t_D \\ t_G \end{array}\right\} := \text{time scale of the} \left\{\begin{array}{l} \text{dynamics} \\ \text{graph} \end{array}\right.$$



$$t_D \gg t_G$$

Evolutionary Dynamics on *Evolving* Graphs

$$\left.\begin{array}{l} t_D \\ t_G \end{array}\right\} := \text{time scale of the} \left\{\begin{array}{l}\text{dynamics}\\ \text{graph}\end{array}\right.$$

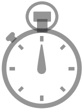


$$t_D \gg t_G$$

$$t_D \ll t_G$$

Evolutionary Dynamics on *Evolving* Graphs

$$\left.\begin{array}{l} t_D \\ t_G \end{array}\right\} := \text{time scale of the} \left\{\begin{array}{l}\text{dynamics}\\ \text{graph}\end{array}\right.$$

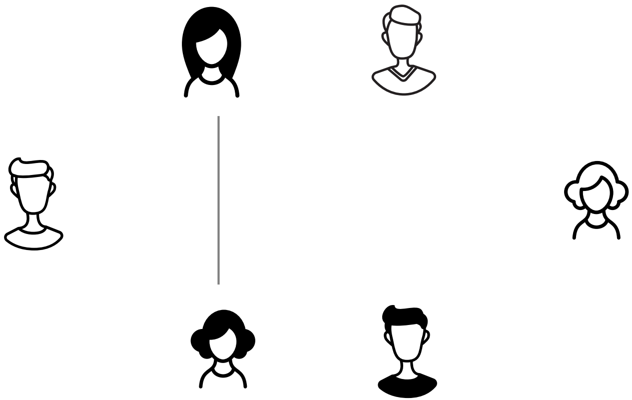


$$t_D \gg t_G$$

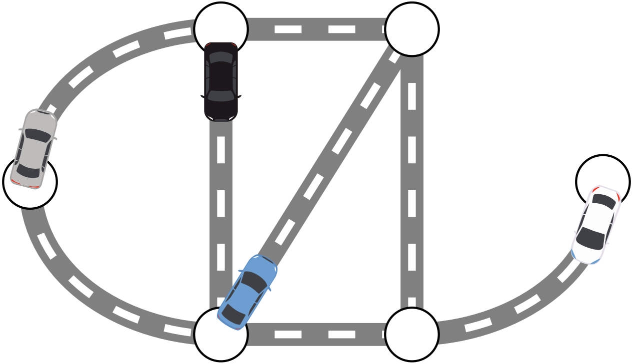
$$t_D \ll t_G$$

$$t_D \sim t_G$$

$t_D \gg t_G$



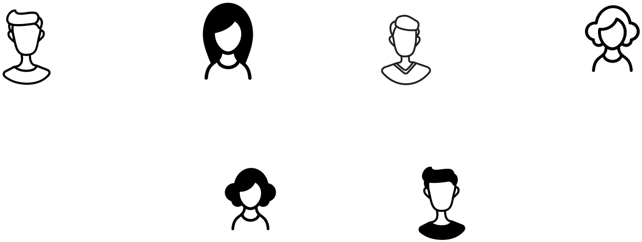
$t_D \ll t_G$



$t_D \sim t_G$

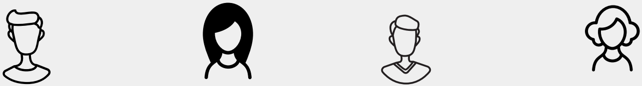
$t_D \sim t_G$

News providers

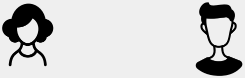


$t_D \sim t_G$

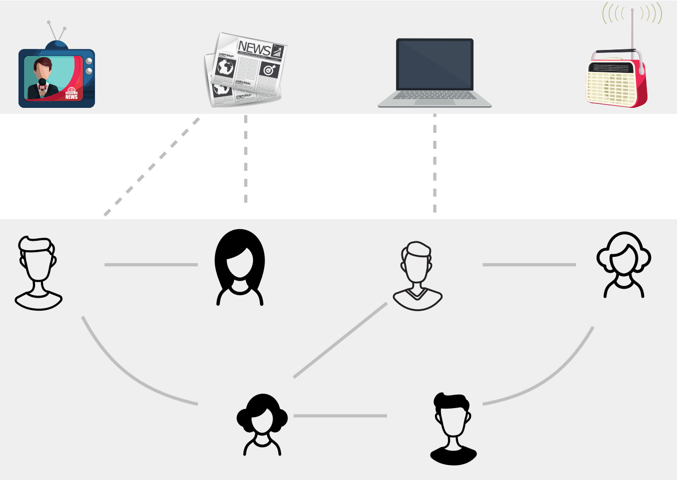
News providers



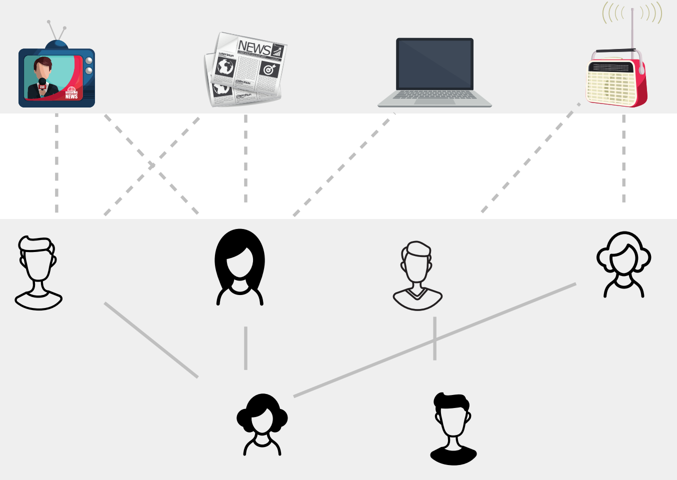
News consumers



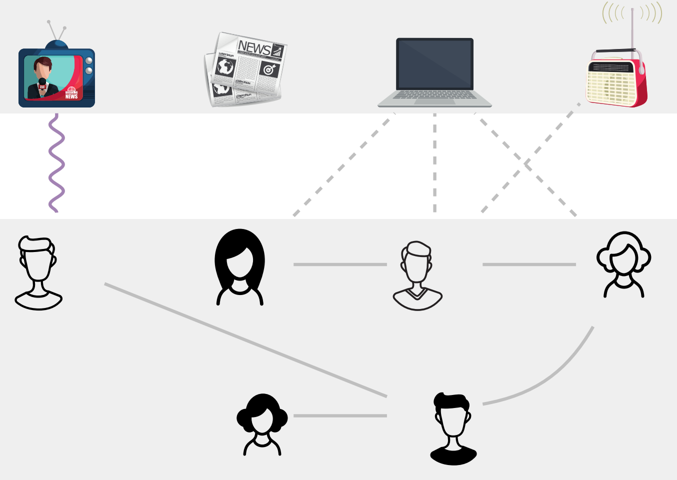
$t_D \sim t_G$



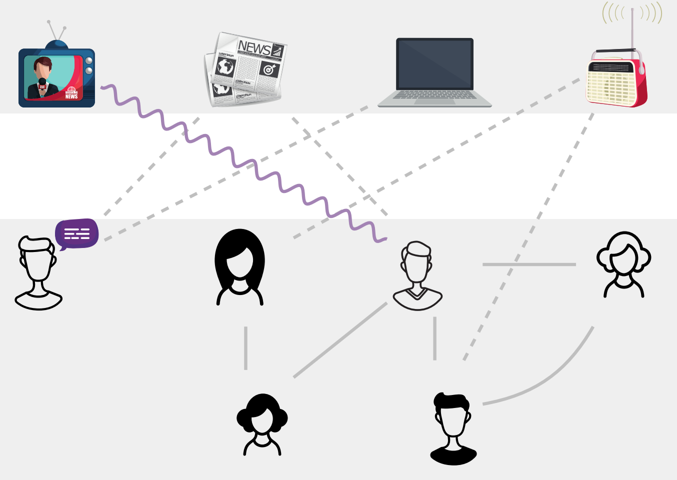
$t_D \sim t_G$



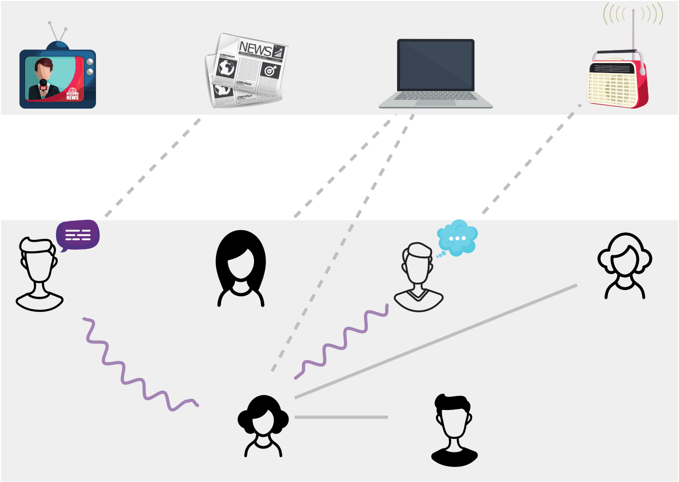
$t_D \sim t_G$



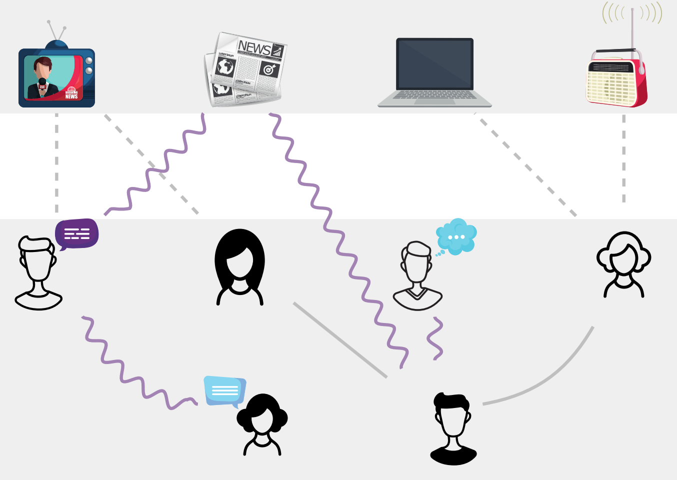
$t_D \sim t_G$



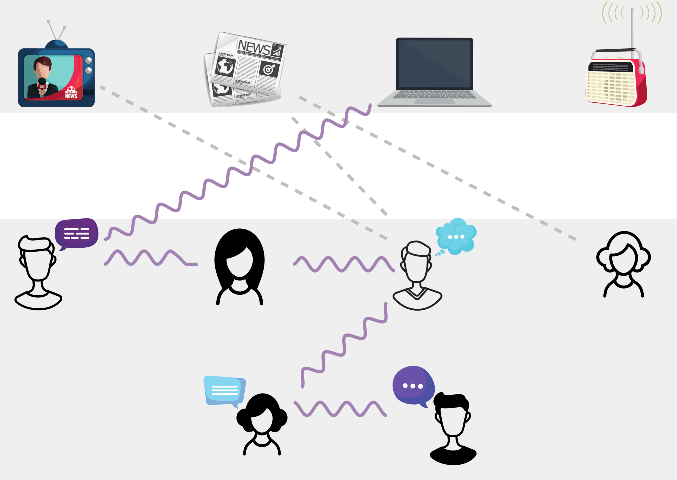
$t_D \sim t_G$



$t_D \sim t_G$

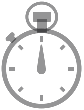


$t_D \sim t_G$



Evolutionary Dynamics on *Evolving* Graphs

$$\left.\begin{matrix} t_D \\ t_G \end{matrix}\right\} := \text{time scale of the} \left\{\begin{matrix} \text{dynamics} \\ \text{graph} \end{matrix}\right.$$



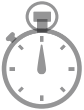
$t_D \gg t_G$

$t_D \ll t_G$

$t_D \sim t_G$

Evolutionary Dynamics on *Evolving* Graphs

$$\left.\begin{matrix} t_D \\ t_G \end{matrix}\right\} := \text{time scale of the} \left\{\begin{matrix} \text{dynamics} \\ \text{graph} \end{matrix}\right.$$



$$t_D \gg t_G$$

$$\boxed{t_D \ll t_G}$$

$$t_D \sim t_G$$

Evolutionary Dynamics on *Evolving* Graphs

$$\left.\begin{matrix} t_D \\ t_G \end{matrix}\right\} := \text{time scale of the} \left\{\begin{matrix} \text{dynamics} \\ \text{graph} \end{matrix}\right.$$

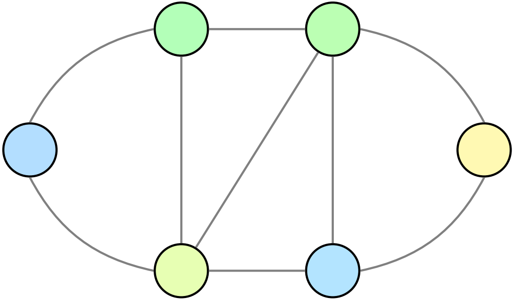
$$t_D \gg t_G$$

$$\boxed{t_D \ll t_G}$$

$$\boxed{t_D \sim t_G}$$

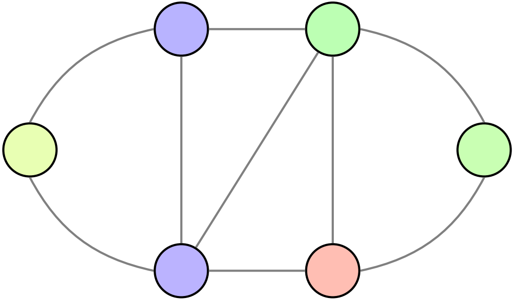
Outline

- **Evolutionary Dynamics on Graphs** $(t_D \ll t_G)$



Outline

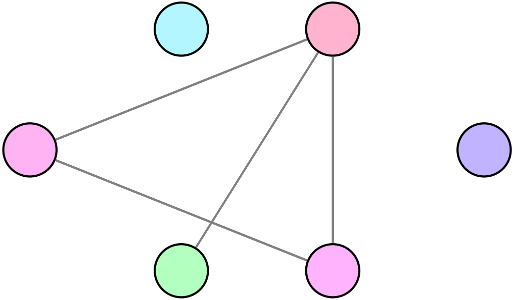
- **Evolutionary Dynamics on Graphs** $(t_D \ll t_G)$
  - *The Modified Petford–Welsh Algorithm*



Outline

I **Evolutionary Dynamics on Graphs** $(t_D \ll t_G)$
- *The Modified Petford–Welsh Algorithm*

II **Evolving Graphs** $(t_D \sim t_G)$



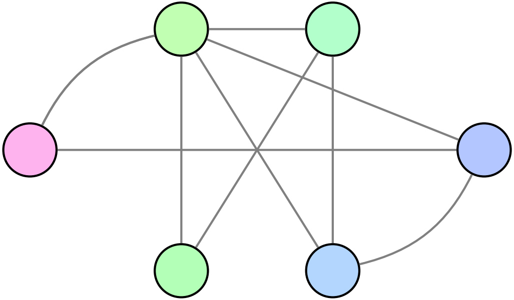
Outline

I **Evolutionary Dynamics on Graphs** $(t_D \ll t_G)$

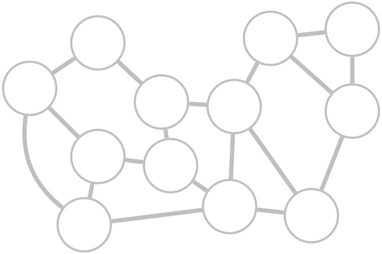
- *The Modified Petford–Welsh Algorithm*

II **Evolving Graphs** $(t_D \sim t_G)$

- *Co-evolution of the Multilayer News Flow*



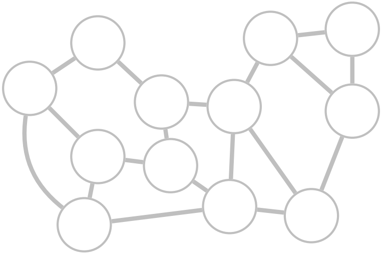
The Modified Petford–Welsh Algorithm



The Modified Petford–Welsh Algorithm

Petford–Welsh algorithm

Petford, A. D. & Welsh, D. J. A. A Randomised 3-Colouring Algorithm, *Discrete Math.* **74** (1989), 253–261.

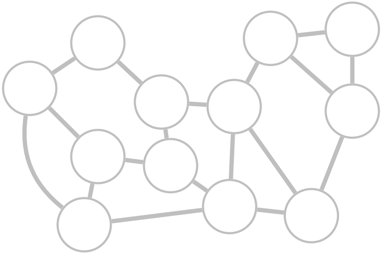


The Modified Petford–Welsh Algorithm

Petford–Welsh algorithm

Petford, A. D. & Welsh, D. J. A. A Randomised 3-Colouring Algorithm, *Discrete Math.* **74** (1989), 253–261.

Žerovnik, J. A Randomized Algorithm for k-Colorability, *Discrete Math.* **131** (1994), 379–393.

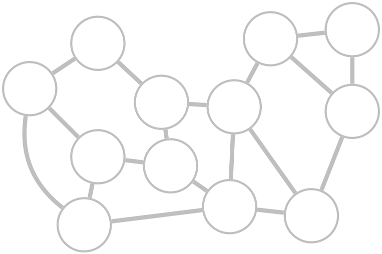


The Modified Petford–Welsh Algorithm

Petford–Welsh algorithm

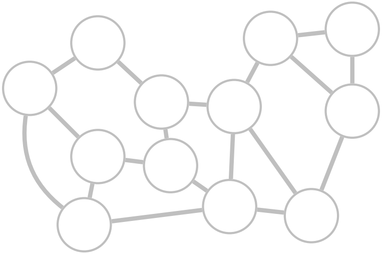
Petford, A. D. & Welsh, D. J. A. A Randomised 3-Colouring Algorithm, *Discrete Math.* **74** (1989), 253–261.

Žerovnik, J. A Randomized Algorithm for k-Colorability, *Discrete Math.* **131** (1994), 379–393.



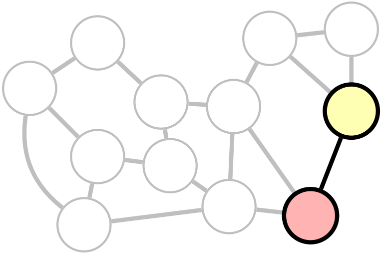
The Modified Petford–Welsh Algorithm

A **proper colouring** is an assignment of colours to the vertices of a graph so that no two adjacent vertices have the same colour, i.e., $c : V \to \{1, 2, \ldots, k\}$ s.t. $c(i) \neq c(j)$ for all $ij \in E$.



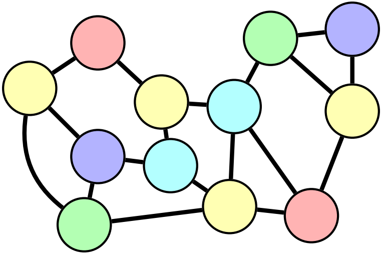
The Modified Petford–Welsh Algorithm

A **proper colouring** is an assignment of colours to the vertices of a graph so that no two adjacent vertices have the same colour, i.e., $c : V \to \{1, 2, \ldots, k\}$ s.t. $c(i) \neq c(j)$ for all $ij \in E$.



The Modified Petford–Welsh Algorithm

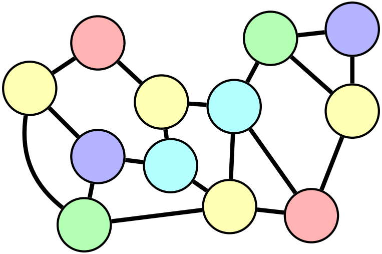
A **proper colouring** is an assignment of colours to the vertices of a graph so that no two adjacent vertices have the same colour, i.e., $c : V \to \{1, 2, \ldots, k\}$ s.t. $c(i) \neq c(j)$ for all $ij \in E$.



The Modified Petford–Welsh Algorithm

A **proper colouring** is an assignment of colours to the vertices of a graph so that no two adjacent vertices have the same colour, i.e., $c : V \to \{1, 2, \ldots, k\}$ s.t. $c(i) \neq c(j)$ for all $ij \in E$.

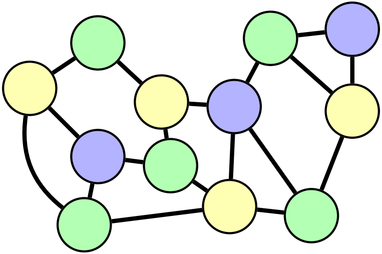
A graph that has a k-colouring is said to be k-**colourable**.



The Modified Petford–Welsh Algorithm

A **proper colouring** is an assignment of colours to the vertices of a graph so that no two adjacent vertices have the same colour, i.e., $c : V \to \{1, 2, \ldots, k\}$ s.t. $c(i) \neq c(j)$ for all $ij \in E$.

A graph that has a k-colouring is said to be k-**colourable**.

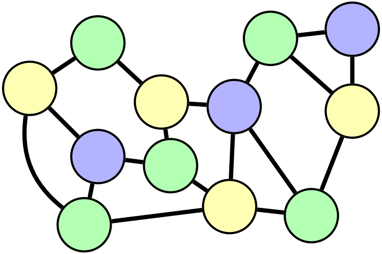


The Modified Petford–Welsh Algorithm

Petford–Welsh algorithm

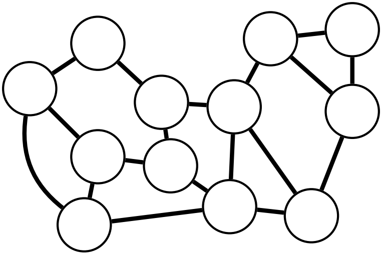
Petford, A. D. & Welsh, D. J. A. A Randomised 3-Colouring Algorithm, *Discrete Math.* **74** (1989), 253–261.

Žerovnik, J. A Randomized Algorithm for k-Colorability, *Discrete Math.* **131** (1994), 379–393.



The Modified Petford–Welsh Algorithm

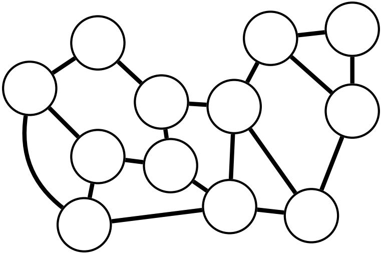
A randomised k-colouring algorithm



The Modified Petford–Welsh Algorithm

A randomised k-colouring algorithm

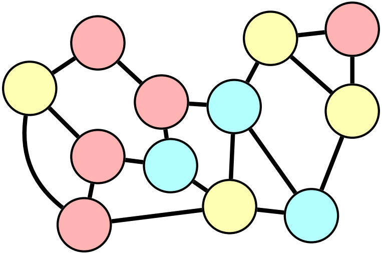
1. generate an initial k-colouring



The Modified Petford–Welsh Algorithm

A randomised k-colouring algorithm

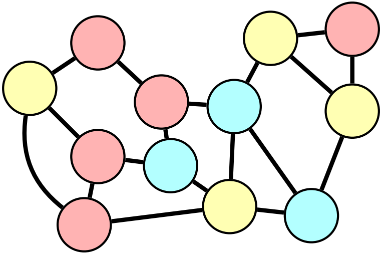
1. generate an initial k-colouring



The Modified Petford–Welsh Algorithm

A randomised k-colouring algorithm

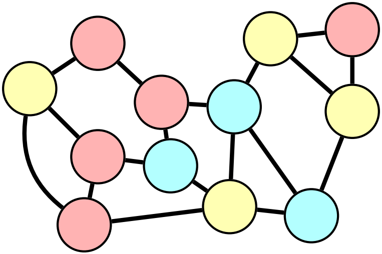
1. generate an initial k-colouring
2. **while** (there is a *bad vertex*) **and** (not too many steps) **repeat**



The Modified Petford–Welsh Algorithm

A randomised k-colouring algorithm

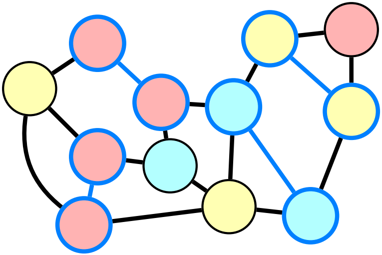
1. generate an initial k-colouring
2. **while** (there is a *bad vertex*) **and** (not too many steps) **repeat**



The Modified Petford–Welsh Algorithm

A randomised k-colouring algorithm

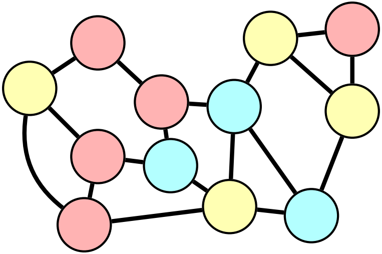
1. generate an initial k-colouring
2. **while** (there is a *bad vertex*) **and** (not too many steps) **repeat**



The Modified Petford–Welsh Algorithm

A randomised k-colouring algorithm

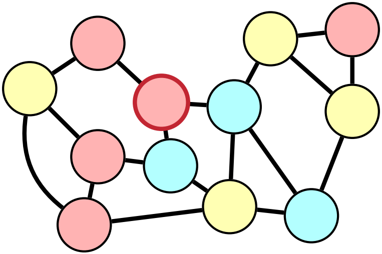
1. generate an initial k-colouring
2. **while** (there is a *bad vertex*) **and** (not too many steps) **repeat**
   2.1 choose a bad vertex v uniformly at random



The Modified Petford–Welsh Algorithm

A randomised k-colouring algorithm

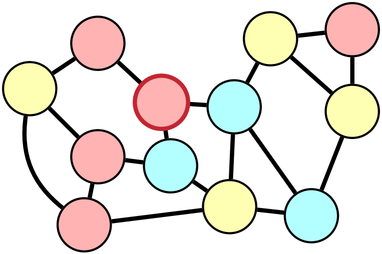
1. generate an initial k-colouring
2. **while** (there is a *bad vertex*) **and** (not too many steps) **repeat**
   2.1 choose a bad vertex v uniformly at random



The Modified Petford–Welsh Algorithm

A randomised k-colouring algorithm

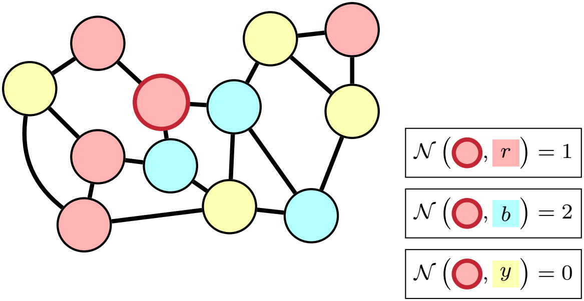
1. generate an initial k-colouring
2. **while** (there is a *bad vertex*) **and** (not too many steps) **repeat**
   - 2.1 choose a bad vertex v uniformly at random
   - 2.2 choose a new colour i for v proportionally to $\omega^{-\mathcal{N}(v,i)}$, $\omega > 1$



The Modified Petford–Welsh Algorithm

A randomised k-colouring algorithm

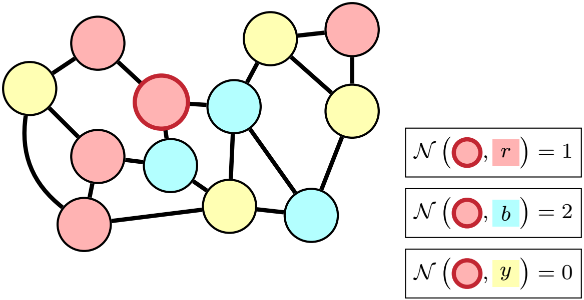
1. generate an initial k-colouring
2. **while** (there is a *bad vertex*) **and** (not too many steps) **repeat**
   - 2.1 choose a bad vertex v uniformly at random
   - 2.2 choose a new colour i for v proportionally to $\omega^{-\mathcal{N}(v,i)}$, $\omega > 1$



The Modified Petford–Welsh Algorithm

A randomised k-colouring algorithm

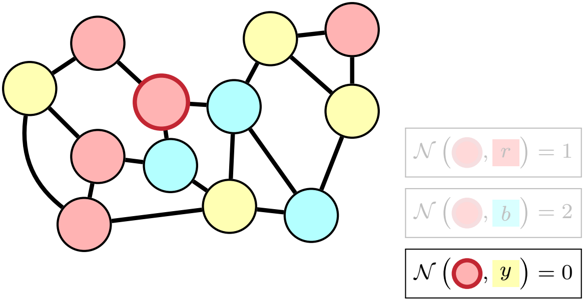
1. generate an initial k-colouring
2. **while** (there is a *bad vertex*) **and** (not too many steps) **repeat**
   - 2.1 choose a bad vertex v uniformly at random
   - 2.2 choose a new colour i for v proportionally to $\omega^{-\mathcal{N}(v,i)}$, $\omega > 1$



The Modified Petford–Welsh Algorithm

A randomised k-colouring algorithm

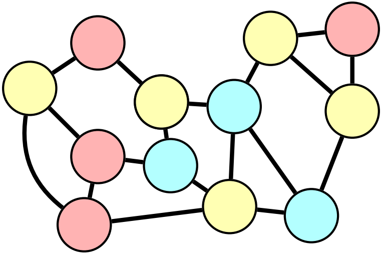
1. generate an initial k-colouring
2. **while** (there is a *bad vertex*) **and** (not too many steps) **repeat**
   2.1 choose a bad vertex v uniformly at random
   2.2 choose a new colour i for v proportionally to $\omega^{-\mathcal{N}(v,i)}$, $\omega > 1$



The Modified Petford–Welsh Algorithm

A randomised k-colouring algorithm

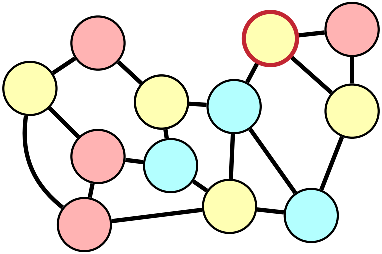
1. generate an initial k-colouring
2. **while** (there is a *bad vertex*) **and** (not too many steps) **repeat**
   - 2.1 choose a bad vertex v uniformly at random
   - 2.2 choose a new colour i for v proportionally to $\omega^{-\mathcal{N}(v,i)}$, $\omega > 1$



The Modified Petford–Welsh Algorithm

A randomised k-colouring algorithm

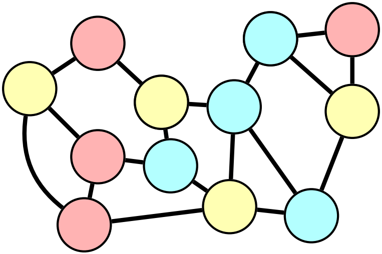
1. generate an initial k-colouring
2. **while** (there is a *bad vertex*) **and** (not too many steps) **repeat**
   2.1 choose a bad vertex v uniformly at random
   2.2 choose a new colour i for v proportionally to $\omega^{-\mathcal{N}(v,i)}$, $\omega > 1$



The Modified Petford–Welsh Algorithm

A randomised k-colouring algorithm

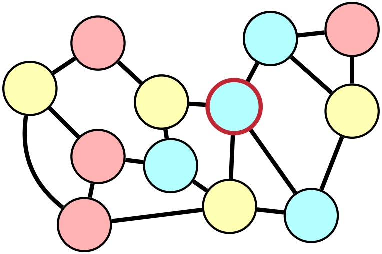
1. generate an initial k-colouring
2. **while** (there is a *bad vertex*) **and** (not too many steps) **repeat**
   2.1 choose a bad vertex v uniformly at random
   2.2 choose a new colour i for v proportionally to $\omega^{-\mathcal{N}(v,i)}$, $\omega > 1$



The Modified Petford–Welsh Algorithm

A randomised k-colouring algorithm

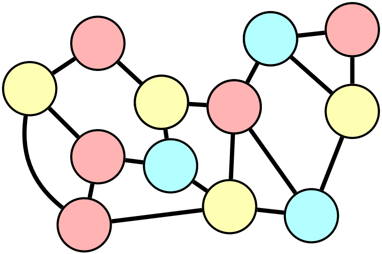
1. generate an initial k-colouring
2. **while** (there is a *bad vertex*) **and** (not too many steps) **repeat**
   - 2.1 choose a bad vertex v uniformly at random
   - 2.2 choose a new colour i for v proportionally to $\omega^{-\mathcal{N}(v,i)}$, $\omega > 1$



The Modified Petford–Welsh Algorithm

A randomised k-colouring algorithm

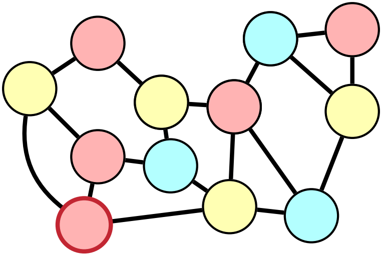
1. generate an initial k-colouring
2. **while** (there is a *bad vertex*) **and** (not too many steps) **repeat**
   - 2.1 choose a bad vertex v uniformly at random
   - 2.2 choose a new colour i for v proportionally to $\omega^{-\mathcal{N}(v,i)}$, $\omega > 1$



The Modified Petford–Welsh Algorithm

A randomised k-colouring algorithm

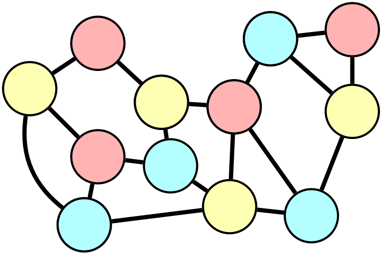
1. generate an initial k-colouring
2. **while** (there is a *bad vertex*) **and** (not too many steps) **repeat**
   - 2.1 choose a bad vertex v uniformly at random
   - 2.2 choose a new colour i for v proportionally to $\omega^{-\mathcal{N}(v,i)}$, $\omega > 1$



The Modified Petford–Welsh Algorithm

A randomised k-colouring algorithm

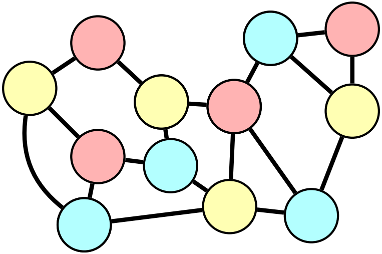
1. generate an initial k-colouring
2. **while** (there is a *bad vertex*) **and** (not too many steps) **repeat**
   - 2.1 choose a bad vertex v uniformly at random
   - 2.2 choose a new colour i for v proportionally to $\omega^{-\mathcal{N}(v,i)}$, $\omega > 1$



The Modified Petford–Welsh Algorithm

The Petford–Welsh algorithm ...

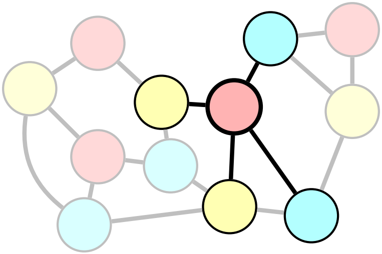
- ... acts locally; thus, it is highly parallelisable.

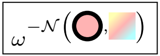


The Modified Petford–Welsh Algorithm

The Petford–Welsh algorithm ...

- ... acts locally; thus, it is highly parallelisable.

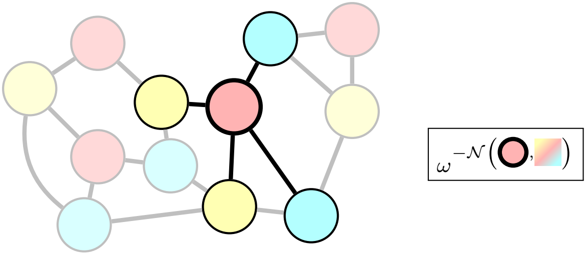




The Modified Petford–Welsh Algorithm

The Petford–Welsh algorithm …

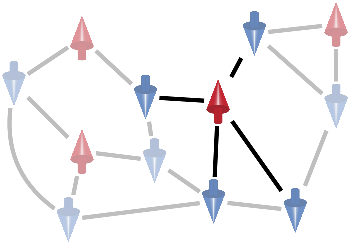
- … acts locally; thus, it is highly parallelisable.
- … is akin to the *Glauber dynamics* of the Ising model.



The Modified Petford–Welsh Algorithm

The Petford–Welsh algorithm …

- … acts locally; thus, it is highly parallelisable.
- … is akin to the *Glauber dynamics* of the Ising model.

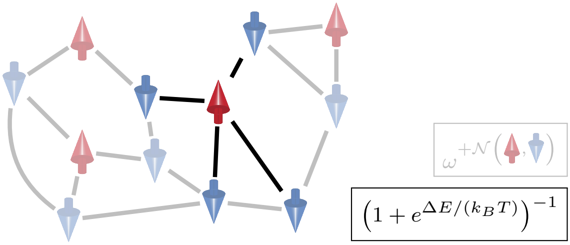


$\omega^{+\mathcal{N}(\uparrow,\downarrow)}$

The Modified Petford–Welsh Algorithm

The Petford–Welsh algorithm ...

- ... acts locally; thus, it is highly parallelisable.
- ... is akin to the *Glauber dynamics* of the Ising model.

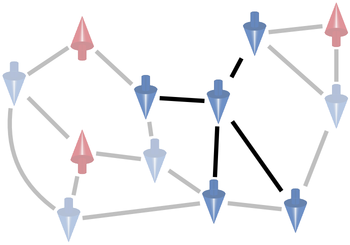


$$\left(1 + e^{\Delta E/(k_B T)}\right)^{-1}$$

The Modified Petford–Welsh Algorithm

The Petford–Welsh algorithm ...

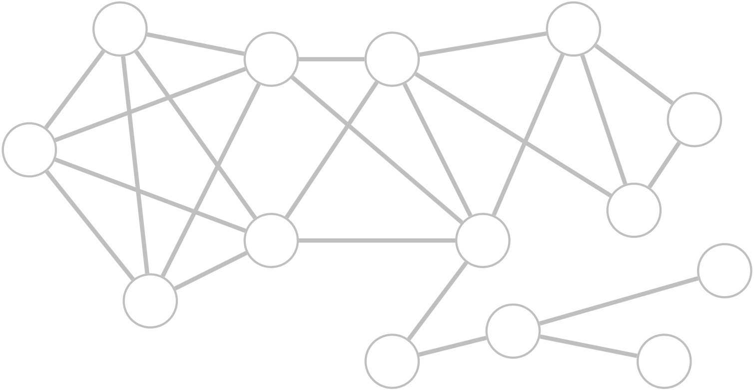
- ... acts locally; thus, it is highly parallelisable.
- ... is akin to the *Glauber dynamics* of the Ising model.



$$\omega^{+\mathcal{N}(\uparrow,\downarrow)}$$

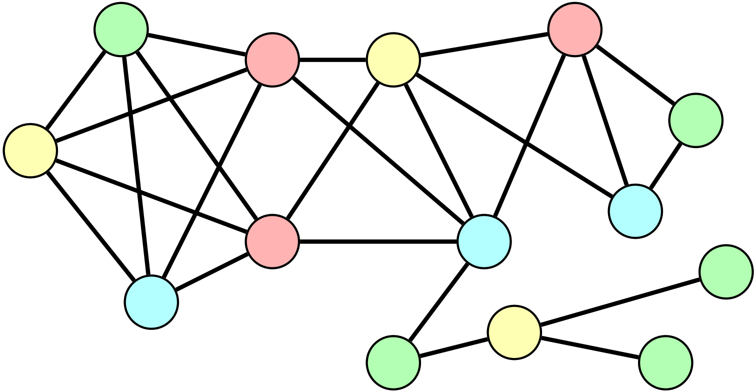
$$\boxed{\left(1+e^{\Delta E/(k_B T)}\right)^{-1}}$$

The Modified Petford–Welsh Algorithm

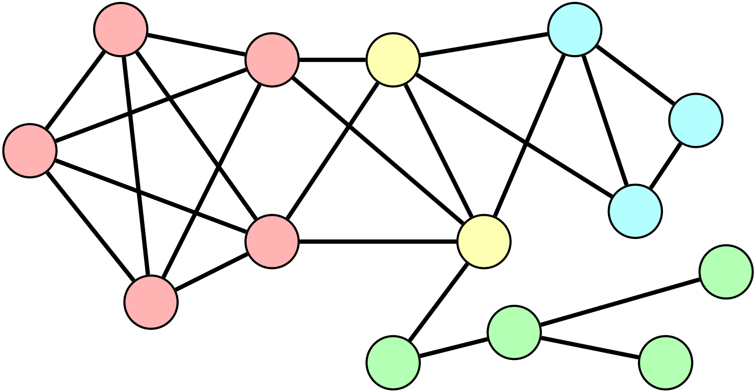


The Modified Petford–Welsh Algorithm

Graph **colouring**

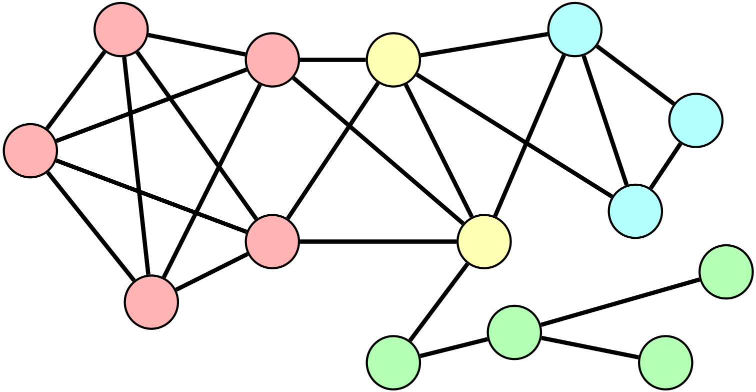


The Modified Petford–Welsh Algorithm



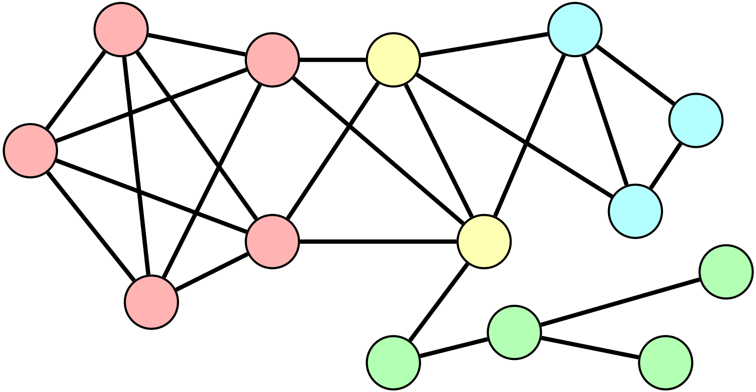
The Modified Petford–Welsh Algorithm

Graph **clustering**



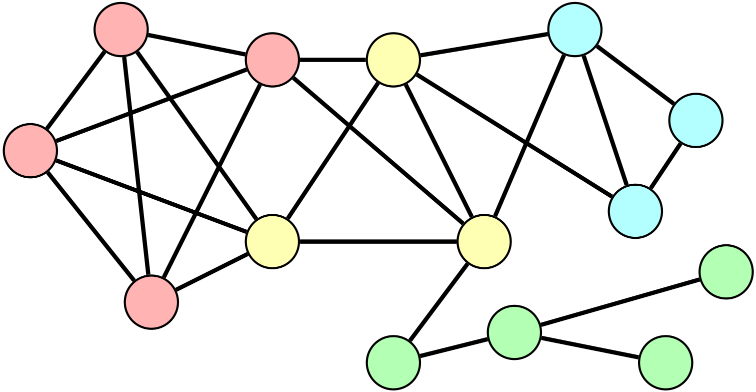
The Modified Petford–Welsh Algorithm

clustering [partitioning or grouping data into *similar* subsets]



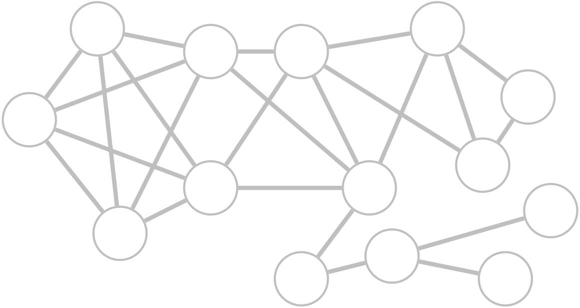
The Modified Petford–Welsh Algorithm

clustering [partitioning or grouping data into *similar* subsets]



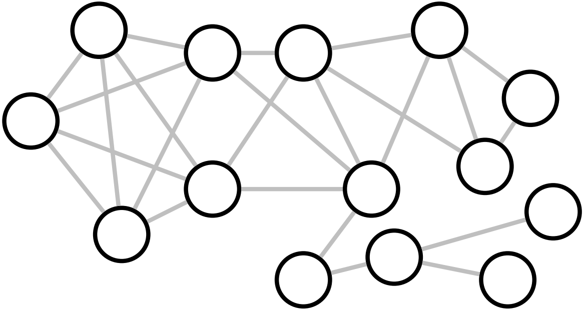
The Modified Petford–Welsh Algorithm

The goal of **clustering** is to separate a given set of objects $X = \{x_1, x_2, \ldots, x_n\}$ into non-overlapping groups/*clusters* $\mathcal{C} = \{C_1, C_2, \ldots, C_m\}$ that satisfy



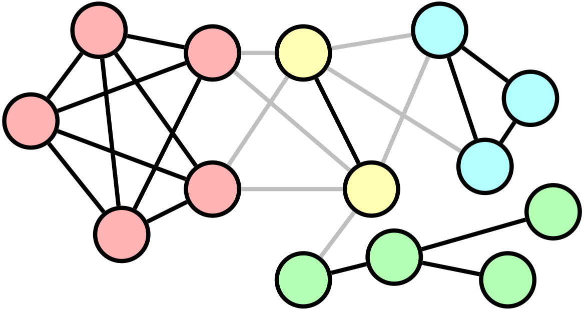
The Modified Petford–Welsh Algorithm

The goal of **clustering** is to separate a given set of objects $X = \{x_1, x_2, \ldots, x_n\}$ into non-overlapping groups/*clusters* $\mathcal{C} = \{C_1, C_2, \ldots, C_m\}$ that satisfy



The Modified Petford–Welsh Algorithm

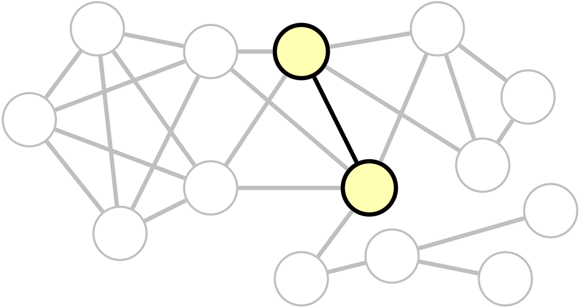
The goal of **clustering** is to separate a given set of objects $X = \{x_1, x_2, \ldots, x_n\}$ into non-overlapping groups/*clusters* $\mathcal{C} = \{C_1, C_2, \ldots, C_m\}$ that satisfy



The Modified Petford–Welsh Algorithm

The goal of **clustering** is to separate a given set of objects $X = \{x_1, x_2, \ldots, x_n\}$ into non-overlapping groups/*clusters* $\mathcal{C} = \{C_1, C_2, \ldots, C_m\}$ that satisfy

$$\boxed{C_i \neq \emptyset} \quad \text{for all} \quad 1 \leq i \leq m,$$

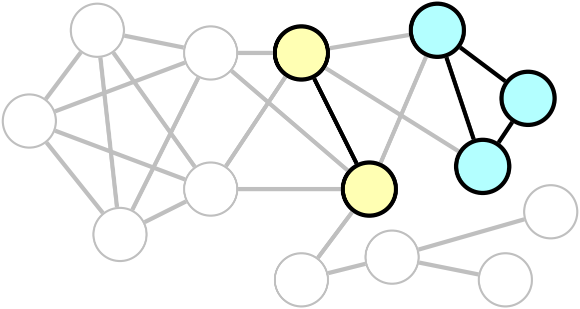


The Modified Petford–Welsh Algorithm

The goal of **clustering** is to separate a given set of objects $X = \{x_1, x_2, \ldots, x_n\}$ into non-overlapping groups/*clusters* $\mathcal{C} = \{C_1, C_2, \ldots, C_m\}$ that satisfy

$$\boxed{C_i \neq \emptyset} \quad \text{for all} \quad 1 \leq i \leq m,$$

$$\boxed{C_i \cap C_j = \emptyset} \quad \text{for all} \quad 1 \leq i < j \leq m,$$



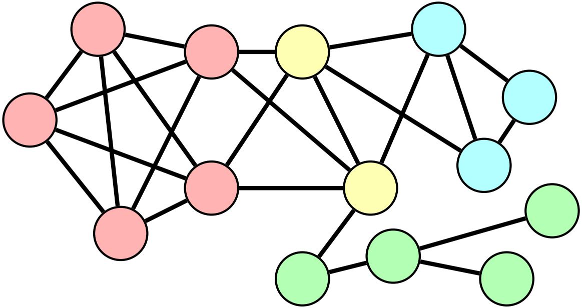
The Modified Petford–Welsh Algorithm

The goal of **clustering** is to separate a given set of objects $X = \{x_1, x_2, \ldots, x_n\}$ into non-overlapping groups/*clusters* $\mathcal{C} = \{C_1, C_2, \ldots, C_m\}$ that satisfy

$$\boxed{C_i \neq \emptyset} \quad \text{for all} \quad 1 \leq i \leq m,$$

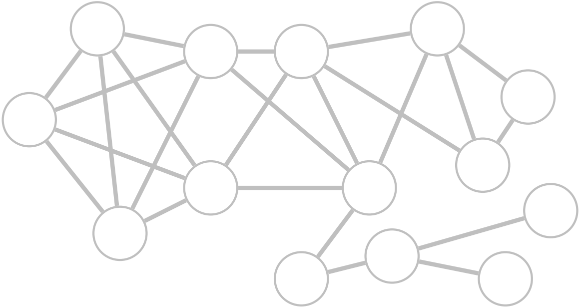
$$\boxed{C_i \cap C_j = \emptyset} \quad \text{for all} \quad 1 \leq i < j \leq m,$$

$$\boxed{\bigcup_{i=1}^{m} C_i = X}.$$



The Modified Petford–Welsh Algorithm

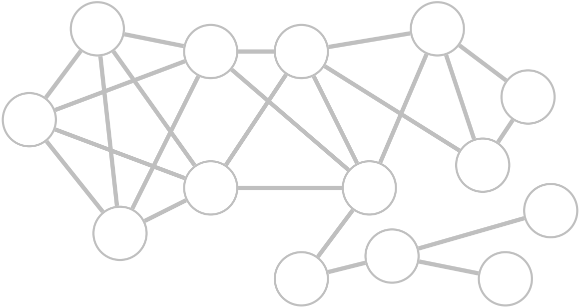
A randomised *clustering* algorithm [`https://github.com/ikicab/mPW`]



The Modified Petford–Welsh Algorithm

A randomised *clustering* algorithm [`https://github.com/ikicab/mPW`]

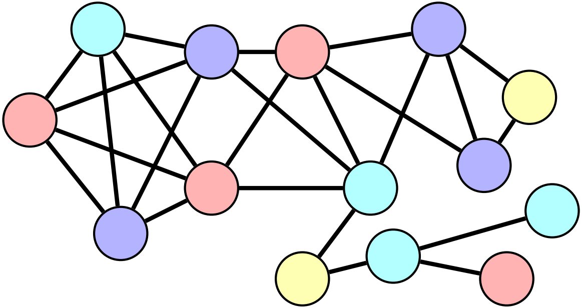
1. generate an initial k-*clustering*



The Modified Petford–Welsh Algorithm

A randomised *clustering* algorithm [`https://github.com/ikicab/mPW`]

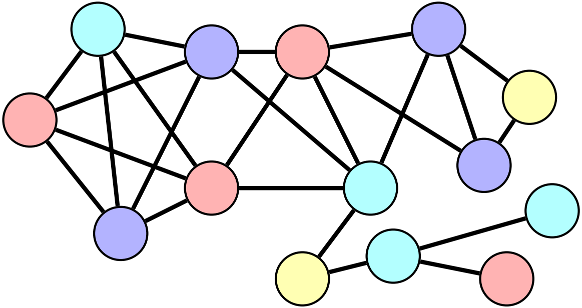
1. generate an initial k-*clustering*



The Modified Petford–Welsh Algorithm

A randomised *clustering* algorithm [`https://github.com/ikicab/mPW`]

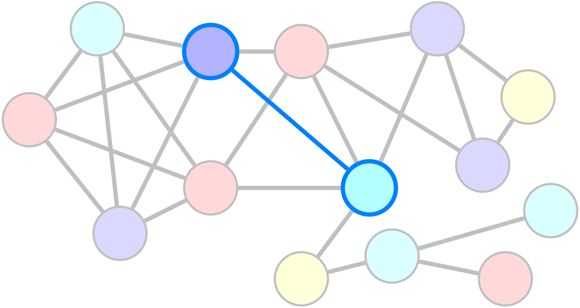
1. generate an initial k-*clustering*
2. **while** (there is a *bad vertex*) **and** ($\mathrm{Var}\left[\textit{bad edges}\right] \geq \mathtt{tol}$) **repeat**



The Modified Petford–Welsh Algorithm

A randomised *clustering* algorithm [`https://github.com/ikicab/mPW`]

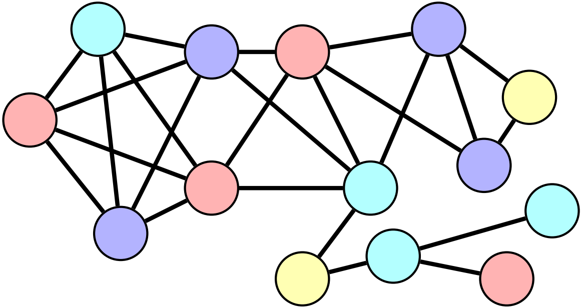
1. generate an initial k-*clustering*
2. **while** (there is a *bad vertex*) **and** ($\mathrm{Var}\,[\textit{bad edges}] \geq \mathtt{tol}$) **repeat**



The Modified Petford–Welsh Algorithm

A randomised *clustering* algorithm [`https://github.com/ikicab/mPW`]

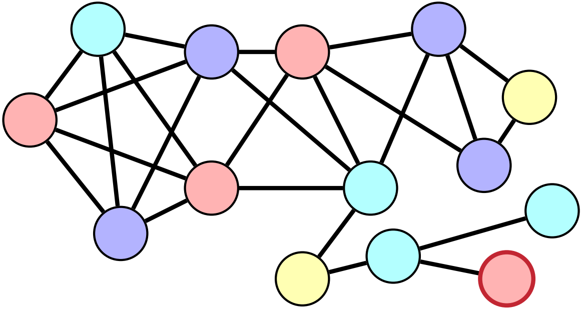
1. generate an initial k-*clustering*
2. **while** (there is a *bad vertex*) **and** ($\mathrm{Var}\left[\textit{bad edges}\right] \geq \mathtt{tol}$) **repeat**
   2.1 choose a bad vertex v uniformly at random



The Modified Petford–Welsh Algorithm

A randomised *clustering* algorithm [`https://github.com/ikicab/mPW`]

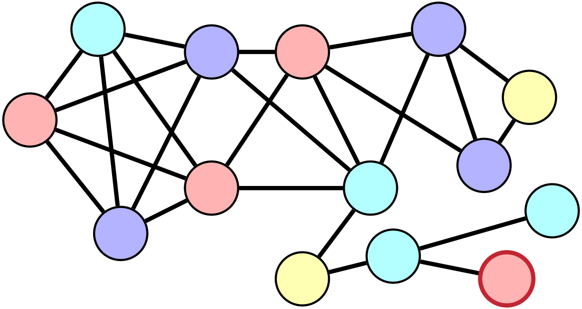
1. generate an initial k-*clustering*
2. **while** (there is a *bad vertex*) **and** $(\mathrm{Var}\left[\textit{bad edges}\right] \geq \texttt{tol})$ **repeat**
   2.1 choose a bad vertex v uniformly at random



The Modified Petford–Welsh Algorithm

A randomised *clustering* algorithm [`https://github.com/ikicab/mPW`]

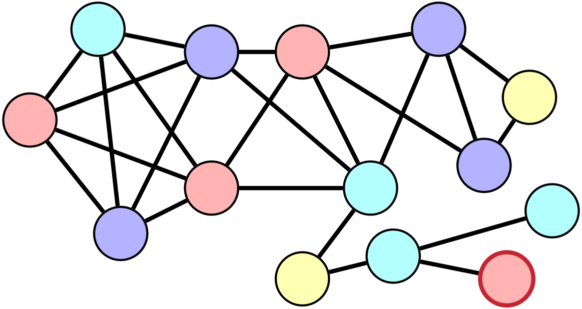
1. generate an initial k-*clustering*
2. **while** (there is a *bad vertex*) **and** ($\mathrm{Var}\left[\textit{bad edges}\right] \geq \texttt{tol}$) **repeat**
   2.1 choose a bad vertex v uniformly at random
   2.2 choose a new colour i for v proportionally to $\omega^{+\mathcal{N}(v,i)}$, $\omega > 1$



The Modified Petford–Welsh Algorithm

A randomised *clustering* algorithm [`https://github.com/ikicab/mPW`]

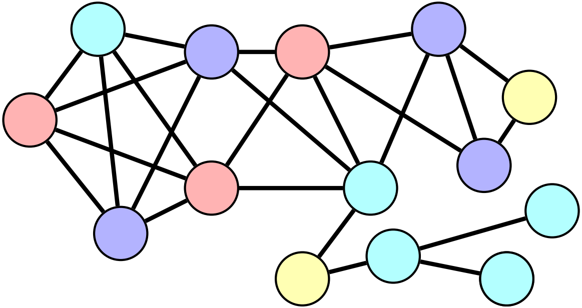
1. generate an initial k-*clustering*
2. **while** (there is a *bad vertex*) **and** ($\mathrm{Var}\left[\textit{bad edges}\right] \geq \texttt{tol}$) **repeat**
   - 2.1 choose a bad vertex v uniformly at random
   - 2.2 choose a new colour i for v proportionally to $\omega^{+\mathcal{N}(v,i)}$, $\omega > 1$



The Modified Petford–Welsh Algorithm

A randomised *clustering* algorithm [`https://github.com/ikicab/mPW`]

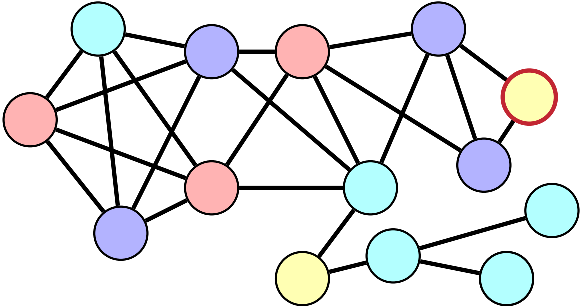
1. generate an initial k-*clustering*
2. **while** (there is a *bad vertex*) **and** ($\mathrm{Var}\,[\textit{bad edges}] \geq \texttt{tol}$) **repeat**
   - 2.1 choose a bad vertex v uniformly at random
   - 2.2 choose a new colour i for v proportionally to $\omega^{+\mathcal{N}(v,i)}$, $\omega > 1$



The Modified Petford–Welsh Algorithm

A randomised *clustering* algorithm [`https://github.com/ikicab/mPW`]

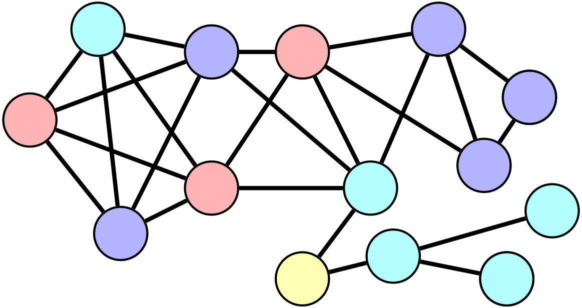
1. generate an initial k-*clustering*
2. **while** (there is a *bad vertex*) **and** ($\mathrm{Var}\left[\textit{bad edges}\right] \geq \texttt{tol}$) **repeat**
   - 2.1 choose a bad vertex v uniformly at random
   - 2.2 choose a new colour i for v proportionally to $\omega^{+\mathcal{N}(v,i)}$, $\omega > 1$



The Modified Petford–Welsh Algorithm

A randomised *clustering* algorithm [https://github.com/ikicab/mPW]

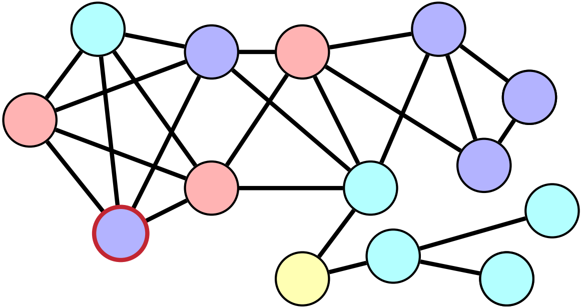
1. generate an initial k-*clustering*
2. **while** (there is a *bad vertex*) **and** $(\mathrm{Var}\,[\textit{bad edges}] \geq \texttt{tol})$ **repeat**
   - 2.1 choose a bad vertex v uniformly at random
   - 2.2 choose a new colour i for v proportionally to $\omega^{+\mathcal{N}(v,i)}$, $\omega > 1$



The Modified Petford–Welsh Algorithm

A randomised *clustering* algorithm [`https://github.com/ikicab/mPW`]

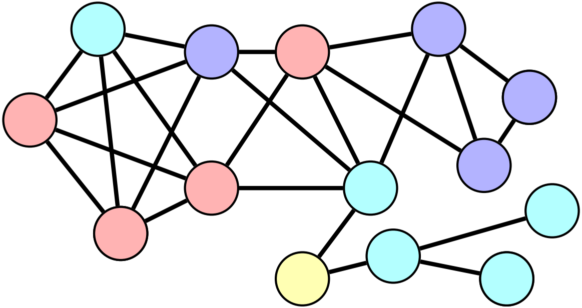
1. generate an initial k-*clustering*
2. **while** (there is a *bad vertex*) **and** ($\mathrm{Var}\,[\textit{bad edges}] \geq \texttt{tol}$) **repeat**
   - 2.1 choose a bad vertex v uniformly at random
   - 2.2 choose a new colour i for v proportionally to $\omega^{+\mathcal{N}(v,i)}$, $\omega > 1$



The Modified Petford–Welsh Algorithm

A randomised *clustering* algorithm [`https://github.com/ikicab/mPW`]

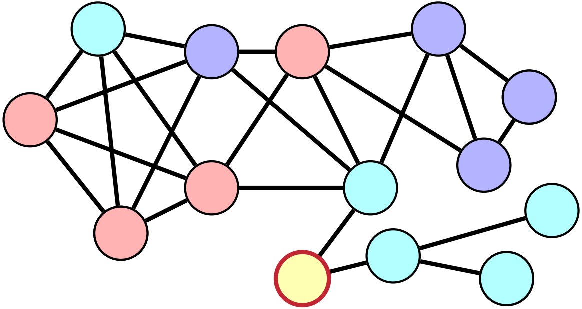
1. generate an initial k-*clustering*
2. **while** (there is a *bad vertex*) **and** ($\mathrm{Var}\,[\textit{bad edges}] \geq \mathtt{tol}$) **repeat**
   - 2.1 choose a bad vertex v uniformly at random
   - 2.2 choose a new colour i for v proportionally to $\omega^{+\mathcal{N}(v,i)}$, $\omega > 1$



The Modified Petford–Welsh Algorithm

A randomised *clustering* algorithm [`https://github.com/ikicab/mPW`]

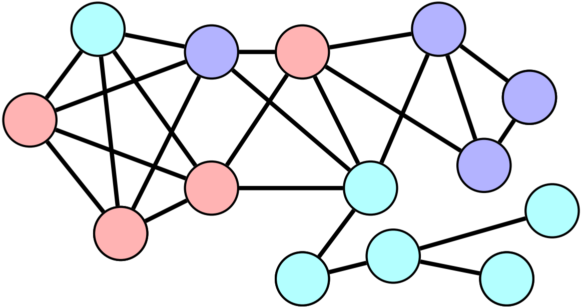
1. generate an initial k-*clustering*
2. **while** (there is a *bad vertex*) **and** ($\mathrm{Var}\,[\textit{bad edges}] \geq \mathtt{tol}$) **repeat**
   2.1 choose a bad vertex v uniformly at random
   2.2 choose a new colour i for v proportionally to $\omega^{+\mathcal{N}(v,i)}$, $\omega > 1$



The Modified Petford–Welsh Algorithm

A randomised *clustering* algorithm [`https://github.com/ikicab/mPW`]

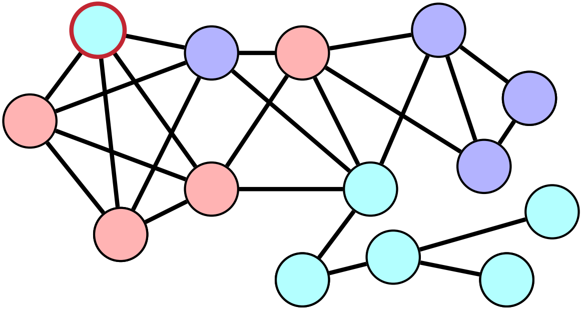
1. generate an initial k-*clustering*
2. **while** (there is a *bad vertex*) **and** ($\mathrm{Var}\left[\textit{bad edges}\right] \geq \texttt{tol}$) **repeat**
   2.1 choose a bad vertex v uniformly at random
   2.2 choose a new colour i for v proportionally to $\omega^{+\mathcal{N}(v,i)}$, $\omega > 1$



The Modified Petford–Welsh Algorithm

A randomised *clustering* algorithm [`https://github.com/ikicab/mPW`]

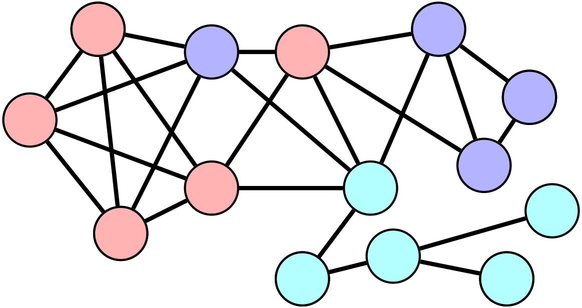
1. generate an initial k-*clustering*
2. **while** (there is a *bad vertex*) **and** ($\mathrm{Var}\,[\textit{bad edges}] \geq \texttt{tol}$) **repeat**
   - 2.1 choose a bad vertex v uniformly at random
   - 2.2 choose a new colour i for v proportionally to $\omega^{+\mathcal{N}(v,i)}$, $\omega > 1$



The Modified Petford–Welsh Algorithm

A randomised *clustering* algorithm [`https://github.com/ikicab/mPW`]

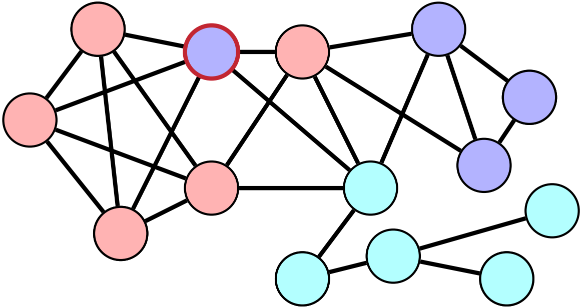
1. generate an initial k-*clustering*
2. **while** (there is a *bad vertex*) **and** ($\mathrm{Var}\left[\textit{bad edges}\right] \geq \texttt{tol}$) **repeat**
   - 2.1 choose a bad vertex v uniformly at random
   - 2.2 choose a new colour i for v proportionally to $\omega^{+\mathcal{N}(v,i)}$, $\omega > 1$



The Modified Petford–Welsh Algorithm

A randomised *clustering* algorithm [`https://github.com/ikicab/mPW`]

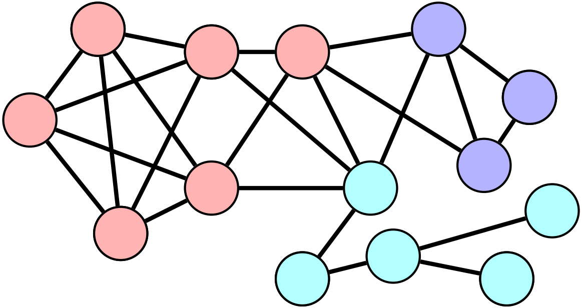
1. generate an initial k-*clustering*
2. **while** (there is a *bad vertex*) **and** ($\mathrm{Var}\left[\textit{bad edges}\right] \geq \texttt{tol}$) **repeat**
   - 2.1 choose a bad vertex v uniformly at random
   - 2.2 choose a new colour i for v proportionally to $\omega^{+\mathcal{N}(v,i)}$, $\omega > 1$



The Modified Petford–Welsh Algorithm

A randomised *clustering* algorithm [`https://github.com/ikicab/mPW`]

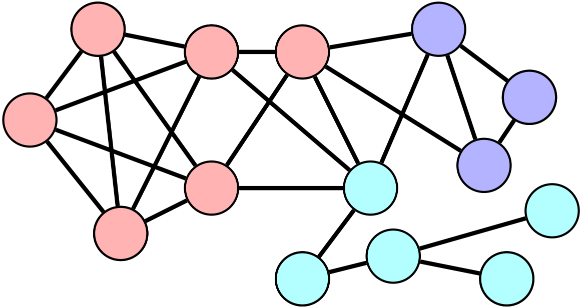
1. generate an initial k-*clustering*
2. **while** (there is a *bad vertex*) **and** ($\mathrm{Var}\,[\textit{bad edges}] \geq \texttt{tol}$) **repeat**
   - 2.1 choose a bad vertex v uniformly at random
   - 2.2 choose a new colour i for v proportionally to $\omega^{+\mathcal{N}(v,i)}$, $\omega > 1$



The Modified Petford–Welsh Algorithm

A randomised *clustering* algorithm [`https://github.com/ikicab/mPW`]

1. generate an initial k-*clustering*
2. **while** (there is a *bad vertex*) **and** ($\mathrm{Var}\left[\textit{bad edges}\right] \geq \texttt{tol}$) **repeat**
   - 2.1 choose a bad vertex v uniformly at random
   - 2.2 choose a new colour i for v proportionally to $\omega^{+\mathcal{N}(v,i)}$, $\omega > 1$



The Modified Petford–Welsh Algorithm

Stopping Condition

$$\mathrm{Var}_{\texttt{step}} = \mathrm{Var}\left(\texttt{bad edges}\left[1 : \texttt{step}\right]\right) < \texttt{tol}$$

The Modified Petford–Welsh Algorithm

Stopping Condition

$$\mathrm{Var}_{\texttt{step}} = \mathrm{Var}\left(\texttt{bad edges}[\texttt{step} - L + 1 : \texttt{step}]\right) < \texttt{tol}$$

The Modified Petford–Welsh Algorithm

Stopping Condition

$$\texttt{bad edges} = []$$

$$\boxed{\mu_L = \frac{1}{L}(b_1 + b_2 + \ldots + b_L)}$$

$$\boxed{\mathrm{Var}_L = \frac{1}{L-1}\left(b_1^2 + b_2^2 + \ldots + b_L^2\right) - \frac{L}{L-1}\mu_L^2}$$

The Modified Petford–Welsh Algorithm

Stopping Condition

$\texttt{step} = 1 :$ $\texttt{bad edges} = [b_1]$

$$\boxed{\mu_L = \frac{1}{L}(b_1 + b_2 + \ldots + b_L)}$$

$$\boxed{\mathrm{Var}_L = \frac{1}{L-1}\left(b_1^2 + b_2^2 + \ldots + b_L^2\right) - \frac{L}{L-1}\mu_L^2}$$

The Modified Petford–Welsh Algorithm

Stopping Condition

$\texttt{step} = 1 :$ $\texttt{bad edges} = [b_1]$

$$\boxed{\mu_L = \frac{1}{L}(b_1 + b_2 + \ldots + b_L)}$$

$$\boxed{\mathrm{Var}_L = \frac{1}{L-1}\left(b_1^2 + b_2^2 + \ldots + b_L^2\right) - \frac{L}{L-1}\mu_L^2}$$

The Modified Petford–Welsh Algorithm

Stopping Condition

$\texttt{step} = 2:$ $\texttt{bad edges} = [b_1, b_2]$

$$\boxed{\mu_L = \frac{1}{L}(b_1 + b_2 + \ldots + b_L)}$$

$$\boxed{\mathrm{Var}_L = \frac{1}{L-1}\left(b_1^2 + b_2^2 + \ldots + b_L^2\right) - \frac{L}{L-1}\mu_L^2}$$

The Modified Petford–Welsh Algorithm

Stopping Condition

$\texttt{step} = L :$ $\texttt{bad edges} = [b_1, b_2, \ldots, b_L]$

$$\boxed{\mu_L = \frac{1}{L}(b_1 + b_2 + \ldots + b_L)}$$

$$\boxed{\mathrm{Var}_L = \frac{1}{L-1}\left(b_1^2 + b_2^2 + \ldots + b_L^2\right) - \frac{L}{L-1}\mu_L^2}$$

The Modified Petford–Welsh Algorithm

Stopping Condition

$\texttt{step} = L: \qquad \texttt{bad edges} = [b_1, b_2, \ldots, b_L]$

$$\boxed{\mu_L = \frac{1}{L}(b_1 + b_2 + \ldots + b_L)}$$

$$\boxed{\mathrm{Var}_L = \frac{1}{L-1}\left(b_1^2 + b_2^2 + \ldots + b_L^2\right) - \frac{L}{L-1}\mu_L^2}$$

The Modified Petford–Welsh Algorithm

Stopping Condition

$\texttt{step} = L: \qquad \texttt{bad edges} = [b_1, b_2, \ldots, b_L]$

$$\boxed{\mu_L = \frac{1}{L}(b_1 + b_2 + \ldots + b_L)}$$

$$\boxed{\mathrm{Var}_L = \frac{1}{L-1}\left(b_1^2 + b_2^2 + \ldots + b_L^2\right) - \frac{L}{L-1}\mu_L^2}$$

The Modified Petford–Welsh Algorithm

Stopping Condition

$$\texttt{step} = L + 1: \qquad \texttt{bad edges} = [b_1, b_2, \ldots, b_L, b_{L+1}]$$

$$\boxed{\mu_L = \frac{1}{L}(b_1 + b_2 + \ldots + b_L)}$$

$$\boxed{\mathrm{Var}_L = \frac{1}{L-1}\left(b_1^2 + b_2^2 + \ldots + b_L^2\right) - \frac{L}{L-1}\mu_L^2}$$

The Modified Petford–Welsh Algorithm

Stopping Condition

$\texttt{step} = L + 1: \quad \texttt{bad edges} = [b_1, b_2, \ldots, b_L, b_{L+1}]$

$$\boxed{\mu_L = \frac{1}{L}(b_1 + b_2 + \ldots + b_L)}$$

$$\boxed{\mathrm{Var}_L = \frac{1}{L-1}\left(b_1^2 + b_2^2 + \ldots + b_L^2\right) - \frac{L}{L-1}\mu_L^2}$$

$$\mu_{L+1} = \mu_L + \frac{1}{L}(b_{L+1} - b_1)$$

The Modified Petford–Welsh Algorithm

Stopping Condition

$$\texttt{step} = L + 1 : \qquad \texttt{bad edges} = [b_1, b_2, \ldots, b_L, b_{L+1}]$$

$$\boxed{\mu_L = \frac{1}{L}(b_1 + b_2 + \ldots + b_L)}$$

$$\boxed{\mathrm{Var}_L = \frac{1}{L-1}\left(b_1^2 + b_2^2 + \ldots + b_L^2\right) - \frac{L}{L-1}\mu_L^2}$$

$$\mu_{L+1} = \mu_L + \frac{1}{L}(b_{L+1} - b_1)$$

$$\mathrm{Var}_{L+1} = \mathrm{Var}_L + \frac{1}{L-1}(b_{L+1} - b_1)(b_{L+1} + b_1 - \mu_{L+1} - \mu_L)$$

The Modified Petford–Welsh Algorithm

Stopping Condition

$$\texttt{step} = L + 2: \qquad \texttt{bad edges} = [b_1, b_2, \ldots, b_L, b_{L+1}, b_{L+2}]$$

$$\boxed{\mu_L = \frac{1}{L}(b_1 + b_2 + \ldots + b_L)}$$

$$\boxed{\mathrm{Var}_L = \frac{1}{L-1}\left(b_1^2 + b_2^2 + \ldots + b_L^2\right) - \frac{L}{L-1}\mu_L^2}$$

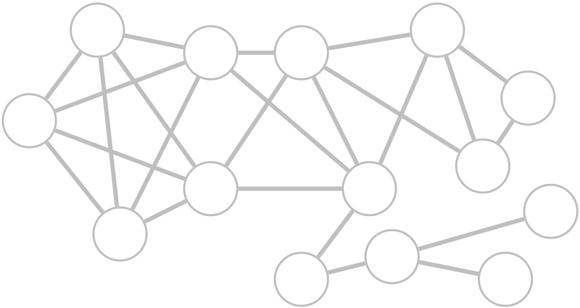
$$\mu_{L+2} = \mu_{L+1} + \frac{1}{L}(b_{L+2} - b_2)$$

$$\mathrm{Var}_{L+2} = \mathrm{Var}_{L+1} + \frac{1}{L-1}(b_{L+2} - b_2)(b_{L+2} + b_2 - \mu_{L+2} - \mu_{L+1})$$

The Modified Petford–Welsh Algorithm

Fine-tuning

Problems

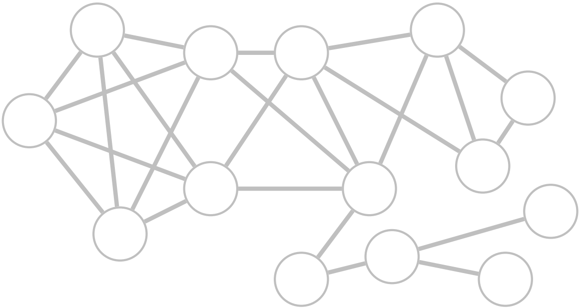


The Modified Petford–Welsh Algorithm

Fine-tuning

Problems

- different clusters get assigned the same colour due to random seeds

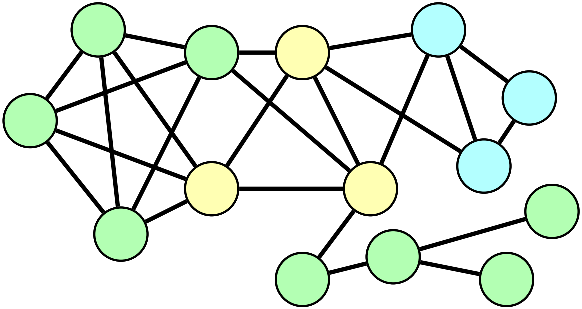


The Modified Petford–Welsh Algorithm

Fine-tuning

Problems

- different clusters get assigned the same colour due to random seeds

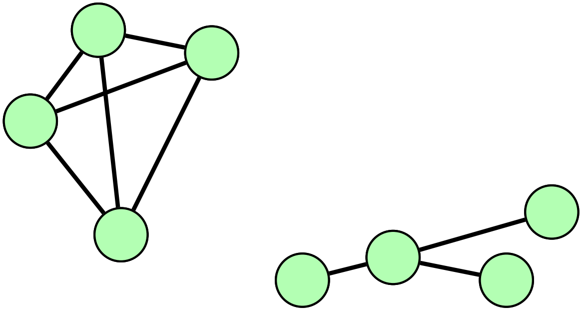


The Modified Petford–Welsh Algorithm

Fine-tuning

Problems

- different clusters get assigned the same colour due to random seeds

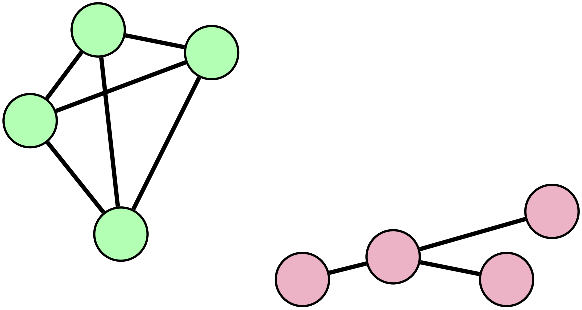


The Modified Petford–Welsh Algorithm

Fine-tuning

Problems

- different clusters get assigned the same colour due to random seeds

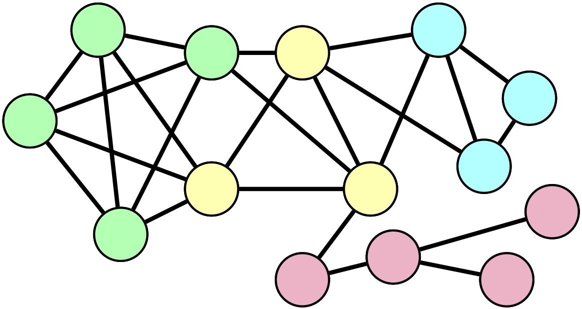


The Modified Petford–Welsh Algorithm

Fine-tuning

Problems

- different clusters get assigned the same colour due to random seeds

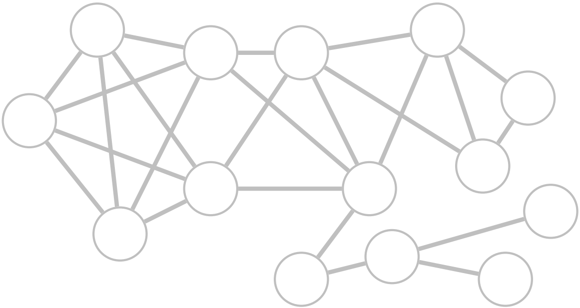


The Modified Petford–Welsh Algorithm

Fine-tuning

Problems

- different clusters get assigned the same colour due to random seeds

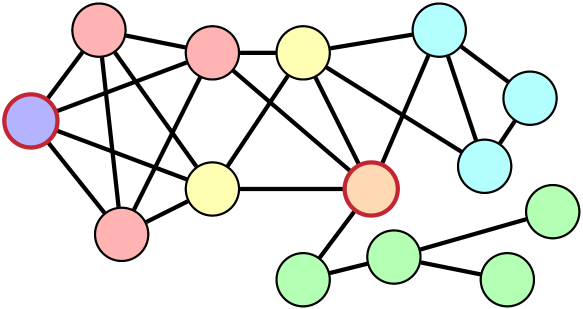


The Modified Petford–Welsh Algorithm

Fine-tuning

Problems

- different clusters get assigned the same colour due to random seeds
- outliers (singleton clusters)

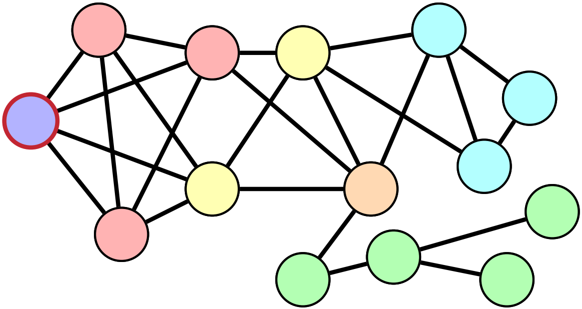


The Modified Petford–Welsh Algorithm

Fine-tuning

Problems

- different clusters get assigned the same colour due to random seeds
- outliers (singleton clusters)

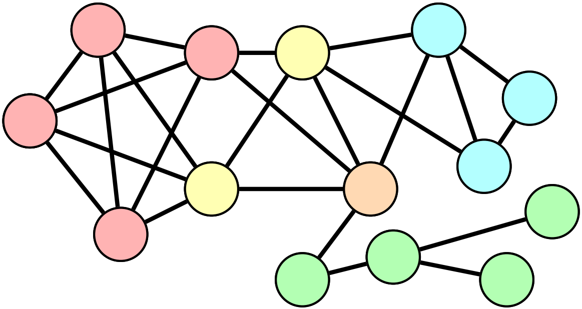


The Modified Petford–Welsh Algorithm

Fine-tuning

Problems

- different clusters get assigned the same colour due to random seeds
- outliers (singleton clusters)

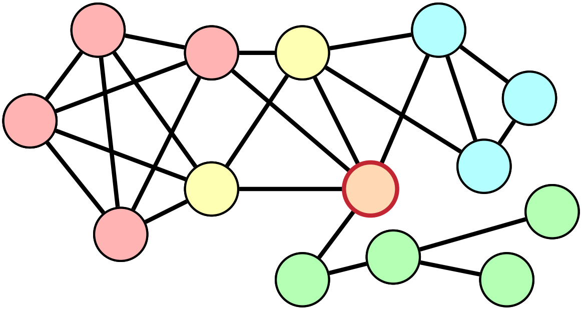


The Modified Petford–Welsh Algorithm

Fine-tuning

Problems

- different clusters get assigned the same colour due to random seeds
- outliers (singleton clusters)

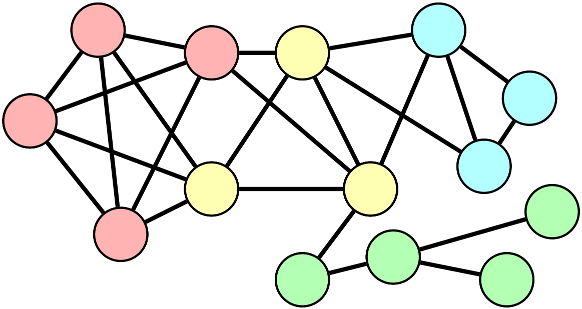


The Modified Petford–Welsh Algorithm

Fine-tuning

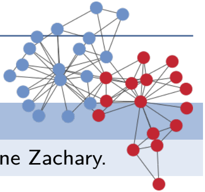
Problems

- different clusters get assigned the same colour due to random seeds
- outliers (singleton clusters)



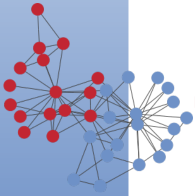
The Modified Petford–Welsh Algorithm

Experiments



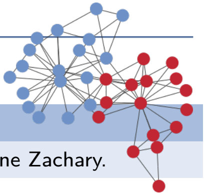
Zachary ($|V| = 34, |E| = 78$)

Ties among the members of a university karate club by Wayne Zachary.



The Modified Petford–Welsh Algorithm

Experiments



Zachary ($|V| = 34, |E| = 78$)

Ties among the members of a university karate club by Wayne Zachary.

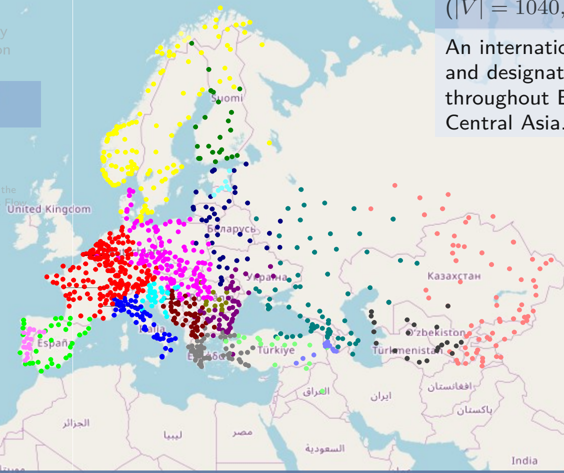
Method	NMI	ARI	ϕ	γ	Q	$\lvert\mathcal{C}\rvert$
Edge betweenness	0.517	0.392	0.424	0.692	0.401	5
Fastgreedy	0.576	0.568	0.574	0.756	0.381	3
Infomap	0.578	0.591	0.668	0.821	0.402	3
Label propagation	0.865	0.882	**0.773**	**0.949**	0.415	3
Leading eigenvector	0.612	0.435	0.487	0.667	0.393	4
Multilevel	0.516	0.392	0.558	0.731	0.419	4
Spinglass	0.627	0.509	0.563	0.756	**0.420**	4
Walktrap	0.531	0.321	0.434	0.590	0.353	5
mPW	**1.000**	**1.000**	**0.773**	**0.949**	0.403	2

The Modified Petford–Welsh Algorithm

Experiments

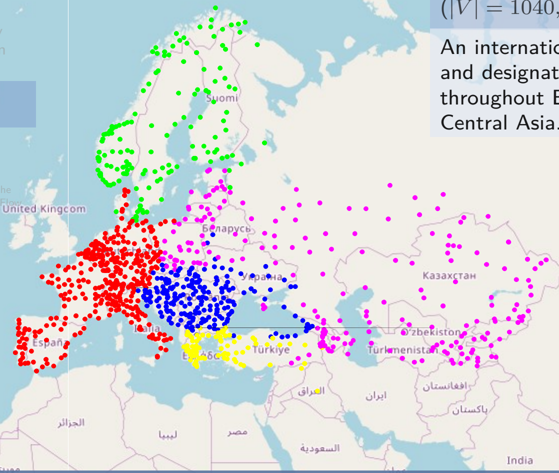
International E-road network ($|V| = 1040, |E| = 1305$) $\boxed{\texttt{tol} = 0.001}$

An international system for numbering and designating roads stretching throughout Europe and some parts of Central Asia.



The Modified Petford–Welsh Algorithm

Experiments

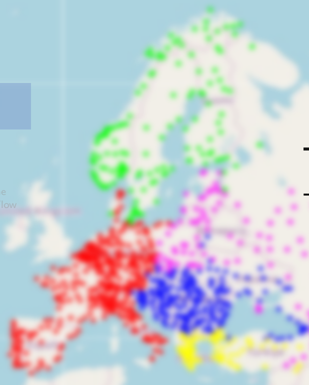


International E-road network
$(|V| = 1040, |E| = 1305)$ $\boxed{\texttt{tol} = 0.0007}$

An international system for numbering and designating roads stretching throughout Europe and some parts of Central Asia.

The Modified Petford–Welsh Algorithm

Experiments



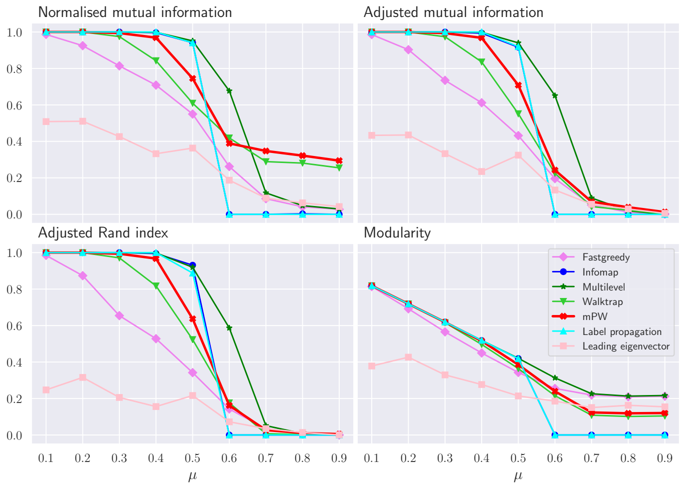
International E-road network
$(|V| = 1040, |E| = 1305)$

An international system for numbering and designating roads stretching throughout Europe and some parts of Central Asia.

Method	ϕ	γ	Q	$\lvert\mathcal{C}\rvert$
Edge betweenness	–	–	–	–
Fastgreedy	0.860	0.917	0.861	24
Infomap	0.663	0.787	0.777	126
Label propagation	0.731	0.856	0.828	82
Leading eigenvector	0.794	0.887	0.835	26
Multilevel	0.873	0.921	0.867	24
Spinglass	0.866	0.924	**0.872**	25
Walktrap	0.757	0.886	0.828	67
mPW	**0.945**	**0.979**	0.845	17

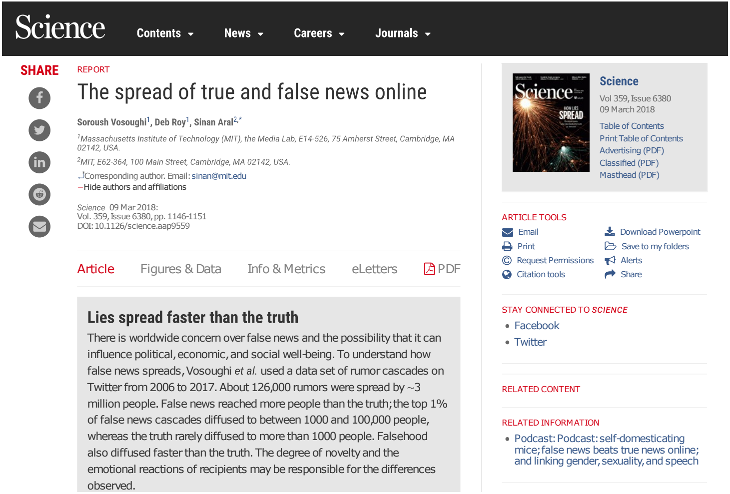
The Modified Petford–Welsh Algorithm

Experiments



$$\mathrm{LFR}(|V| = 1000, \gamma = 2, \beta = 1, \texttt{k\_avg} = 15, \texttt{k\_max} = 100, \texttt{c\_min} = 50, \texttt{c\_max} = 100)$$

Co-evolution of the Multilayer News Flow


Science Contents News Careers Journals

SHARE

REPORT

The spread of true and false news online

Soroush Vosoughi[1], Deb Roy[1], Sinan Aral[2,*]

[1]Massachusetts Institute of Technology (MIT), the Media Lab, E14-526, 75 Amherst Street, Cambridge, MA 02142, USA.

[2]MIT, E62-364, 100 Main Street, Cambridge, MA 02142, USA.

*Corresponding author. Email: sinan@mit.edu

–Hide authors and affiliations

Science 09 Mar 2018:
Vol. 359, Issue 6380, pp. 1146-1151
DOI: 10.1126/science.aap9559

Article Figures & Data Info & Metrics eLetters PDF

Lies spread faster than the truth

There is worldwide concern over false news and the possibility that it can influence political, economic, and social well-being. To understand how false news spreads, Vosoughi *et al.* used a data set of rumor cascades on Twitter from 2006 to 2017. About 126,000 rumors were spread by ~3 million people. False news reached more people than the truth; the top 1% of false news cascades diffused to between 1000 and 100,000 people, whereas the truth rarely diffused to more than 1000 people. Falsehood also diffused faster than the truth. The degree of novelty and the emotional reactions of recipients may be responsible for the differences observed.

Science
Vol 359, Issue 6380
09 March 2018
Table of Contents
Print Table of Contents
Advertising (PDF)
Classified (PDF)
Masthead (PDF)

ARTICLE TOOLS
Email
Download Powerpoint
Print
Save to my folders
Request Permissions
Alerts
Citation tools
Share

STAY CONNECTED TO *SCIENCE*
- Facebook
- Twitter

RELATED CONTENT

RELATED INFORMATION
- Podcast: Podcast: self-domesticating mice; false news beats true news online; and linking gender, sexuality, and speech

Vosoughi, S., Roy, D. & Aral, S. The spread of true and false news online, *Science* **359**(6380) (2018), 1146–1151.

Co-evolution of the Multilayer News Flow

news any asserted claim

Vosoughi, S., Roy, D. & Aral, S. The spread of true and false news online, *Science* **359**(6380) (2018), 1146–1151.

Co-evolution of the Multilayer News Flow

news any asserted claim

rumour

the social phenomena of a news diffusing through a network

Vosoughi, S., Roy, D. & Aral, S. The spread of true and false news online, *Science* **359**(6380) (2018), 1146–1151.

Co-evolution of the Multilayer News Flow

news any asserted claim

rumour

the social phenomena of a news diffusing through a network

rumour cascade an uninterrupted chain of posts stemming from a single common assertion about a news story

Vosoughi, S., Roy, D. & Aral, S. The spread of true and false news online, *Science* **359**(6380) (2018), 1146–1151.

Co-evolution of the Multilayer News Flow

news any asserted claim

rumour the social phenomena of a news diffusing through a network

rumour cascade an uninterrupted chain of posts stemming from a single common assertion about a news story

true/false news a news story that can be objectively verified/disproved

Vosoughi, S., Roy, D. & Aral, S. The spread of true and false news online, *Science* **359**(6380) (2018), 1146–1151.

Co-evolution of the Multilayer News Flow

RESEARCH

SOCIAL SCIENCE

The spread of true and false news online

Soroush Vosoughi,[1] Deb Roy,[1] Sinan Aral[2]*

We investigated the differential diffusion of all of the verified true and false news stories distributed on Twitter from 2006 to 2017. The data comprise ~126,000 stories tweeted by ~3 million people more than 4.5 million times. We classified news as true or false using information from six independent fact-checking organizations that exhibited 95 to 98% agreement on the classifications. Falsehood diffused significantly farther, faster, deeper, and more broadly than the truth in all categories of information, and the effects were more pronounced for false political news than for false news about terrorism, natural disasters, science, urban legends, or financial information. We found that false news was more novel than true news, which suggests that people were more likely to share novel information. Whereas false stories inspired fear, disgust, and surprise in replies, true stories inspired anticipation, sadness, joy, and trust. Contrary to conventional wisdom, robots accelerated the spread of true and false news at the same rate, implying that false news spreads more than the truth because humans, not robots, are more likely to spread it.

Foundational theories of decision-making (*1*–*3*), cooperation (*4*), communication (*5*), and markets (*6*) all view some conceptualization of truth or accuracy as central to the functioning of nearly every human endeavor. Yet, both true and false information

Current work analyzes the spread of single rumors, like the discovery of the Higgs boson (*13*) or the Haitian earthquake of 2010 (*14*), and multiple rumors from a single disaster event, like the Boston Marathon bombing of 2013 (*10*), or it develops theoretical models of rumor diffusion

support their positions as unreliable or fake news, whereas sources that support their positions are labeled reliable or not fake, the term has lost all connection to the actual veracity of the information presented, rendering it meaningless for use in academic classification. We have therefore explicitly avoided the term fake news throughout this paper and instead use the more objectively verifiable terms "true" or "false" news. Although the terms fake news and misinformation also imply a willful distortion of the truth, we do not make any claims about the intent of the purveyors of the information in our analyses. We instead focus our attention on veracity and stories that have been verified as true or false.

We also purposefully adopt a broad definition of the term news. Rather than defining what constitutes news on the basis of the institutional source of the assertions in a story, we refer to any asserted claim made on Twitter as news (we defend this decision in the supplementary materials section on "reliable sources," section S1.2). We define news as any story or claim with an assertion in it and a rumor as the social phenomena of a news story or claim spreading or diffusing through the Twitter network. That is, rumors are inherently social and involve the sharing of claims between people. News, on the other hand, is an assertion with claims, whether it is shared or not.

A rumor cascade begins on Twitter when a

Vosoughi, S., Roy, D. & Aral, S. The spread of true and false news online, *Science* **359**(6380) (2018), 1146–1151.

Co-evolution of the Multilayer News Flow

Falsehood diffused significantly farther, faster, deeper, and more broadly than the truth in all categories of information.

Vosoughi, S., Roy, D. & Aral, S. The spread of true and false news online, *Science* **359**(6380) (2018), 1146–1151.

Co-evolution of the Multilayer News Flow

> Falsehood diffused significantly farther, faster, deeper, and more broadly than the truth in all categories of information.

> Whereas the truth rarely diffused to more than 1000 people, the top 1% of false-news cascades routinely diffused to between 1000 and 100,000 people. [...] It took the truth about six times as long as falsehood to reach 1500 people and 20 times as long as falsehood to reach a cascade depth of 10. [...] falsehoods were 70% more likely to be retweeted than the truth, even when controlling for [user characteristics]

Vosoughi, S., Roy, D. & Aral, S. The spread of true and false news online, *Science* **359**(6380) (2018), 1146–1151.

Co-evolution of the Multilayer News Flow

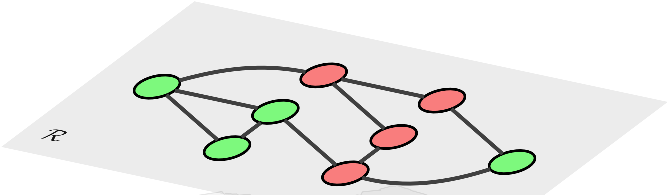
Falsehood diffused significantly farther, faster, deeper, and more broadly than the truth in all categories of information.

Whereas the truth rarely diffused to more than 1000 people, the top 1% of false-news cascades routinely diffused to between 1000 and 100,000 people. [...] It took the truth about six times as long as falsehood to reach 1500 people and 20 times as long as falsehood to reach a cascade depth of 10. [...] falsehoods were 70% more likely to be retweeted than the truth, even when controlling for [user characteristics]

[...] falsehood did not simply spread through broadcast dynamics but rather through peer-to-peer diffusion characterized by a viral branching process.

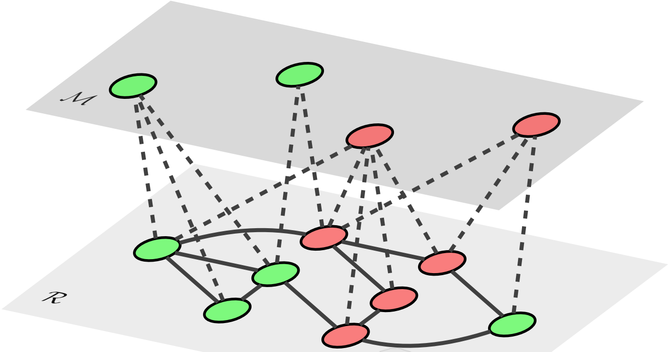
Vosoughi, S., Roy, D. & Aral, S. The spread of true and false news online, *Science* **359**(6380) (2018), 1146–1151.

Co-evolution of the Multilayer News Flow



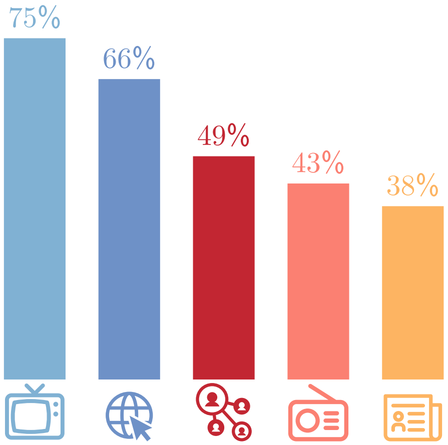
[...] falsehood did not simply spread through broadcast dynamics but rather through peer-to-peer diffusion characterized by a viral branching process.

Co-evolution of the Multilayer News Flow



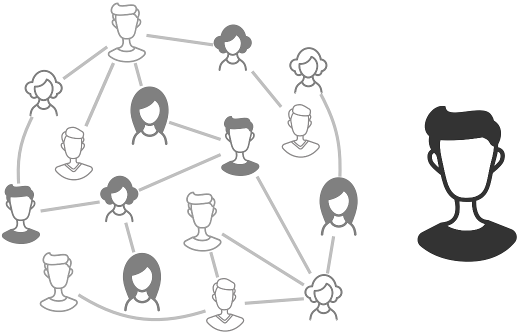
[...] falsehood did not simply spread through broadcast dynamics but rather through peer-to-peer diffusion characterized by a viral branching process.

Co-evolution of the Multilayer News Flow

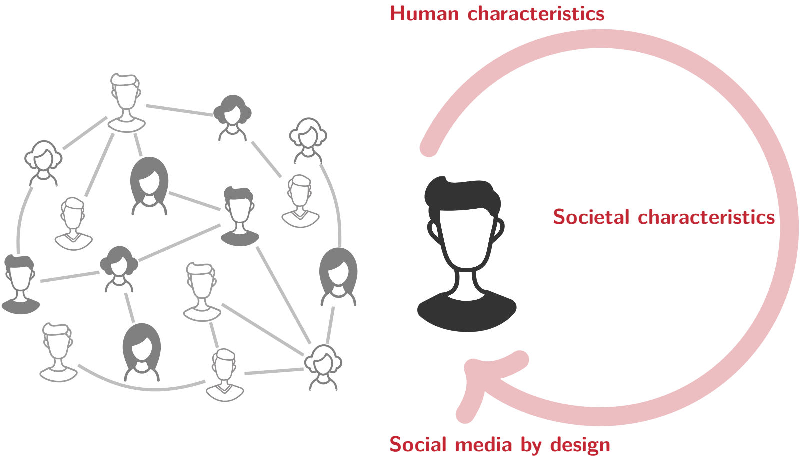


Ofcom, News consumption in the UK (July 2019).

Co-evolution of the Multilayer News Flow

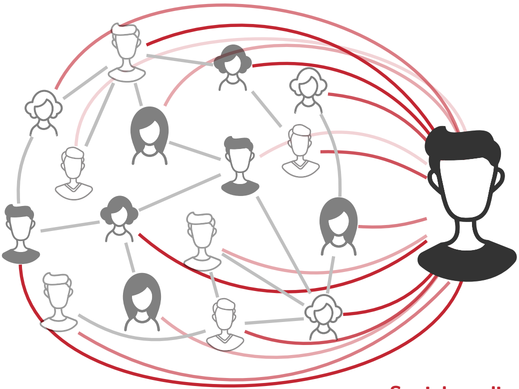


Co-evolution of the Multilayer News Flow



Co-evolution of the Multilayer News Flow

Human characteristics

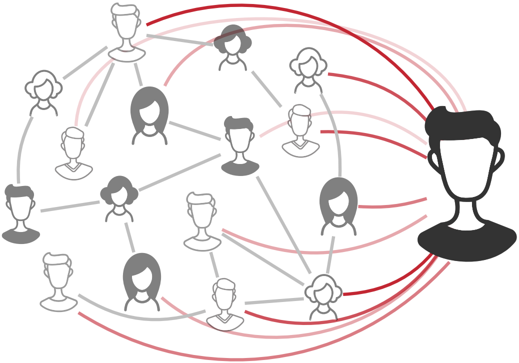


Societal characteristics

Social media by design

- fast-paced, massive influx of information

Co-evolution of the Multilayer News Flow



Human characteristics

Societal characteristics

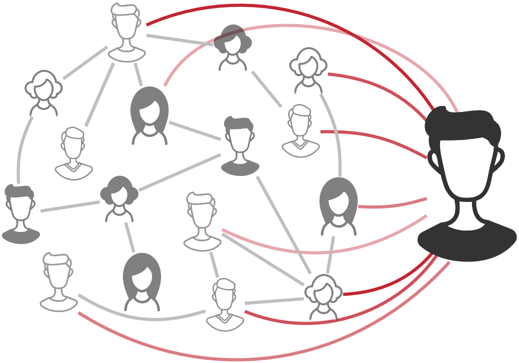
- Dunbar's number, homophily

Social media by design

- fast-paced, massive influx of information

Co-evolution of the Multilayer News Flow

Human characteristics



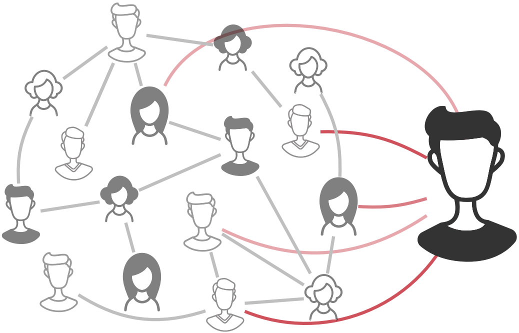
Societal characteristics

- Dunbar's number, homophily

Social media by design

- fast-paced, massive influx of information
- engagement-boosting business model

Co-evolution of the Multilayer News Flow



Human characteristics

- limited time, short attention span

Societal characteristics

- Dunbar's number, homophily

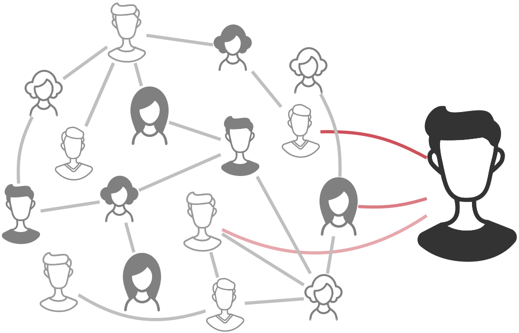
Social media by design

- fast-paced, massive influx of information
- engagement-boosting business model

Co-evolution of the Multilayer News Flow

Human characteristics

- limited time, short attention span
- confirmation bias, familiarity heuristics



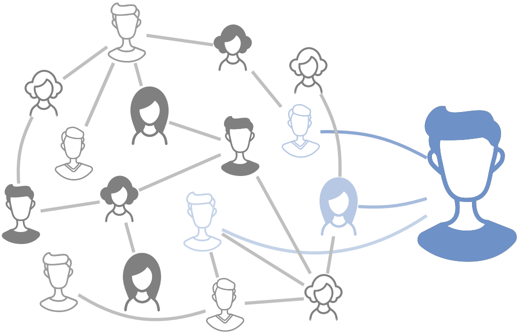
Societal characteristics

- Dunbar's number, homophily

Social media by design

- fast-paced, massive influx of information
- engagement-boosting business model

Co-evolution of the Multilayer News Flow



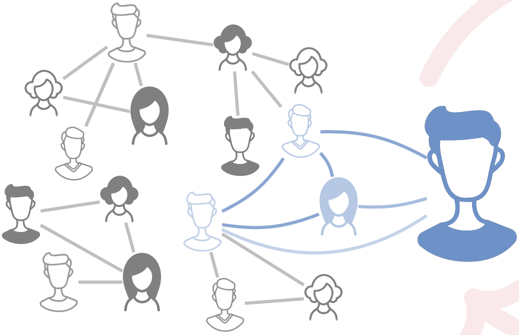
Human characteristics
- limited time, short attention span
- confirmation bias, familiarity heuristics

Societal characteristics
- Dunbar's number, homophily
- conformity bias

Social media by design
- fast-paced, massive influx of information
- engagement-boosting business model

Co-evolution of the Multilayer News Flow

Human characteristics
- limited time, short attention span
- confirmation bias, familiarity heuristics



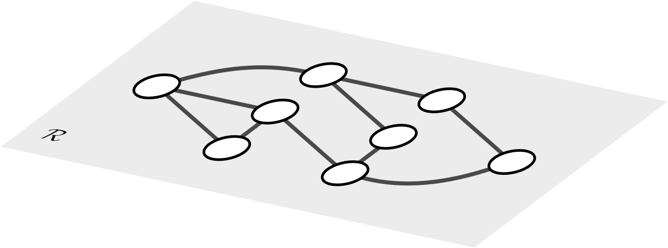
Societal characteristics
- Dunbar's number, homophily
- conformity bias

Social media by design
- fast-paced, massive influx of information
- engagement-boosting business model

Filter bubbles & echo chambers

Co-evolution of the Multilayer News Flow

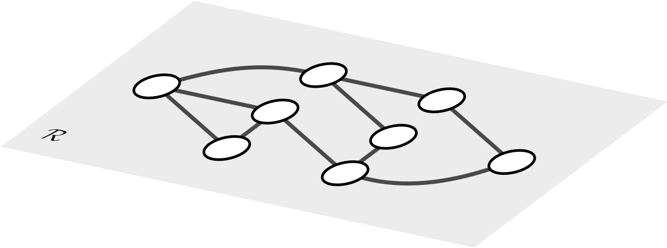
$G_R = (\mathcal{R}, E_R)$



Co-evolution of the Multilayer News Flow

$G_R = (\mathcal{R}, E_R)$

Barabási–Albert ($P(k) \sim k^{-\gamma}$, $\gamma = 2.5$)
$\Delta(G_R) \lessapprox$ Dunbar's number

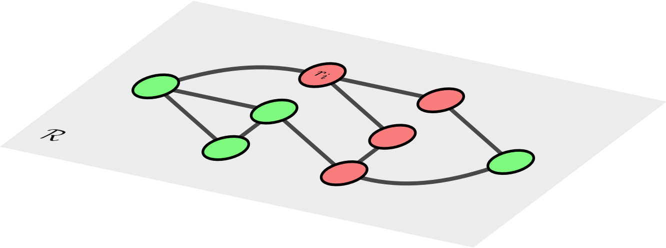


Co-evolution of the Multilayer News Flow

$G_R = (\mathcal{R}, E_R)$

$\boxed{r_i(t) \in \{0, 1\}}$

Barabási–Albert ($P(k) \sim k^{-\gamma}$, $\gamma = 2.5$)
$\Delta(G_R) \lessapprox$ Dunbar's number



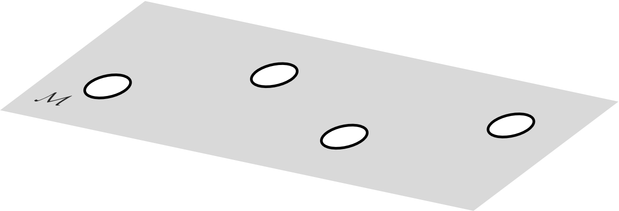
Co-evolution of the Multilayer News Flow

$G_R = (\mathcal{R}, E_R)$

$\boxed{r_i(t) \in \{0, 1\}}$

Barabási–Albert ($P(k) \sim k^{-\gamma}$, $\gamma = 2.5$)
$\Delta(G_R) \lessapprox$ Dunbar's number

$G_M = (\mathcal{M}, E_M)$



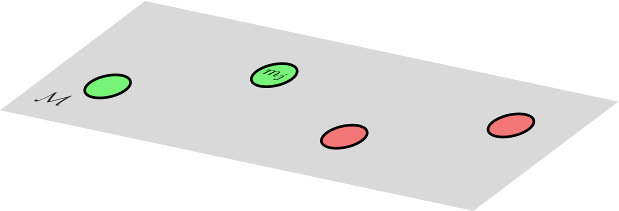
Co-evolution of the Multilayer News Flow

$G_R = (\mathcal{R}, E_R)$ $\boxed{r_i(t) \in \{0, 1\}}$

$G_M = (\mathcal{M}, E_M)$ $\boxed{m_j \in \{0, 1\}}$

Barabási–Albert ($P(k) \sim k^{-\gamma}$, $\gamma = 2.5$)

$\Delta(G_R) \lessapprox$ Dunbar's number

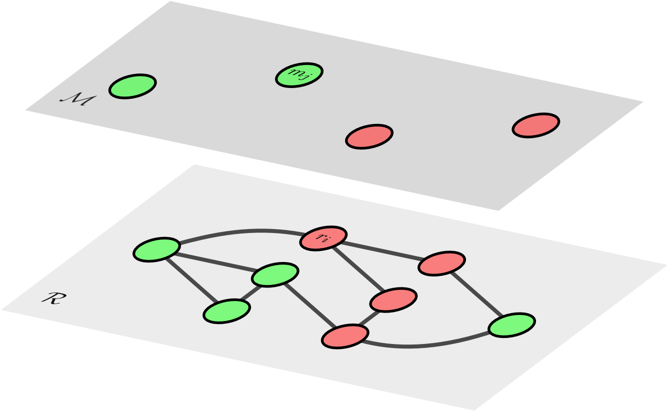


Co-evolution of the Multilayer News Flow

$G_R = (\mathcal{R}, E_R)$ $\boxed{r_i(t) \in \{0,1\}}$

$G_M = (\mathcal{M}, E_M)$ $\boxed{m_j \in \{0,1\}}$

Barabási–Albert ($P(k) \sim k^{-\gamma}$, $\gamma = 2.5$)
$\Delta(G_R) \lessapprox$ Dunbar's number

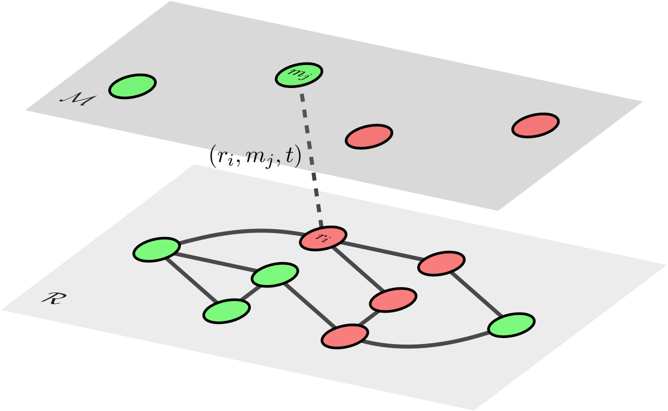


Co-evolution of the Multilayer News Flow

$G_R = (\mathcal{R}, E_R)$ $\boxed{r_i(t) \in \{0,1\}}$

$G_M = (\mathcal{M}, E_M)$ $\boxed{m_j \in \{0,1\}}$

Barabási–Albert $(P(k) \sim k^{-\gamma},\ \gamma = 2.5)$
$\Delta(G_R) \lessapprox$ Dunbar's number



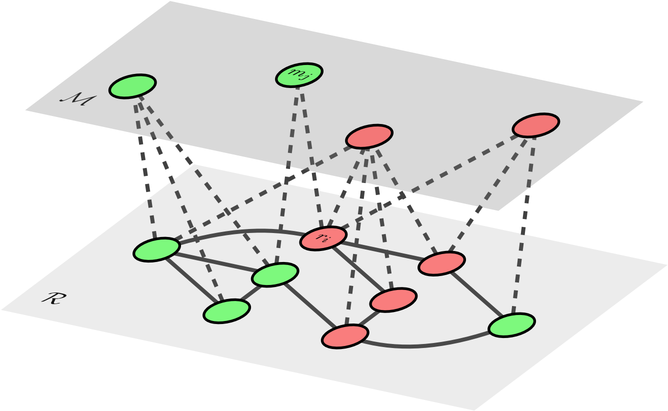
Co-evolution of the Multilayer News Flow

$G_R = (\mathcal{R}, E_R)$ $\boxed{r_i(t) \in \{0,1\}}$

$G_M = (\mathcal{M}, E_M)$ $\boxed{m_j \in \{0,1\}}$

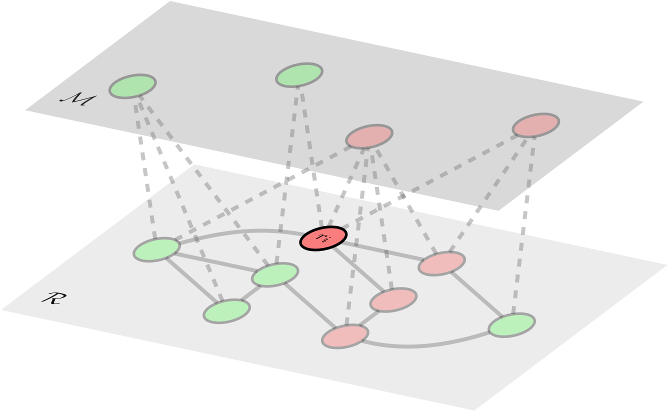
Barabási–Albert ($P(k) \sim k^{-\gamma}$, $\gamma = 2.5$)

$\Delta(G_R) \lessapprox$ Dunbar's number



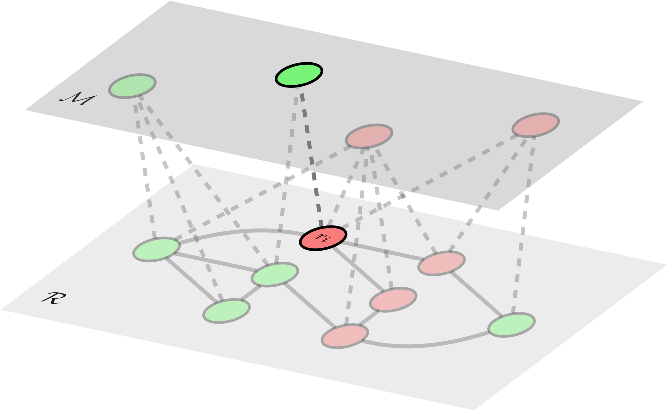
Co-evolution of the Multilayer News Flow

Media repertoire $\left(r_i^T(t), r_i^F(t)\right)$ $\qquad r_i^T(t) + r_i^F(t) \leq M \ll R$



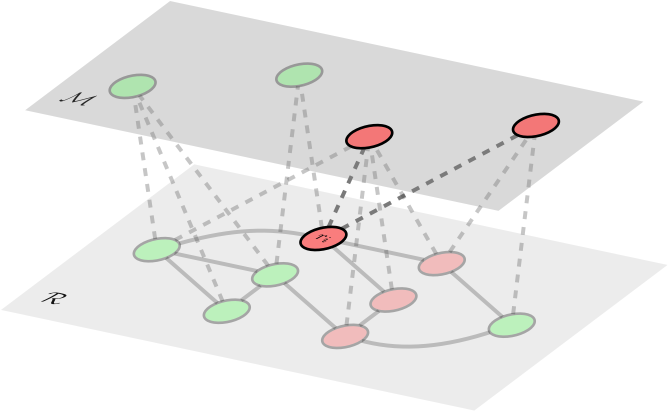
Co-evolution of the Multilayer News Flow

Media repertoire $\left(r_i^T(t), r_i^F(t)\right)$ $r_i^T(t) + r_i^F(t) \leq M \ll R$



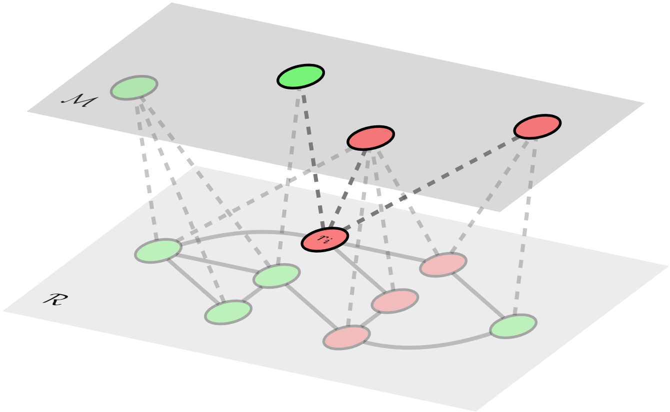
Co-evolution of the Multilayer News Flow

Media repertoire $\left(r_i^T(t), r_i^F(t)\right)$ $r_i^T(t) + r_i^F(t) \leq M \ll R$



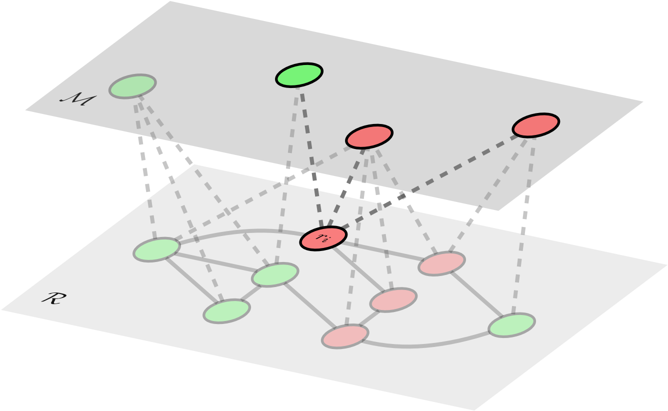
Co-evolution of the Multilayer News Flow

Media repertoire $\left(r_i^T(t), r_i^F(t)\right)$ $r_i^T(t) + r_i^F(t) \leq M \ll R$



Co-evolution of the Multilayer News Flow

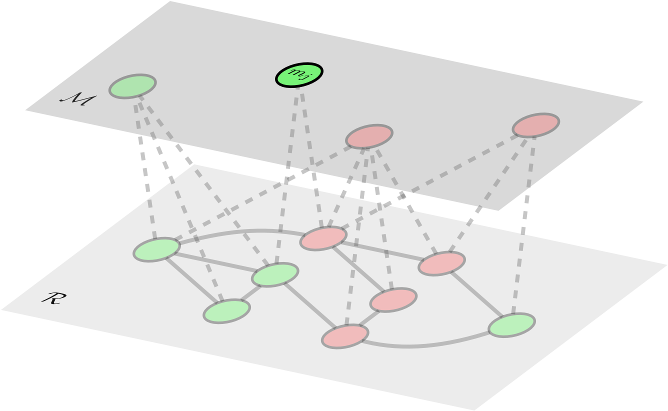
Media repertoire $\left(\rho_i^T(t), \rho_i^F(t)\right)$ $r_i^T(t) + r_i^F(t) \leq M \ll R$



Co-evolution of the Multilayer News Flow

Media repertoire $\left(r_i^T(t), r_i^F(t)\right)$ $r_i^T(t) + r_i^F(t) \leq M \ll R$

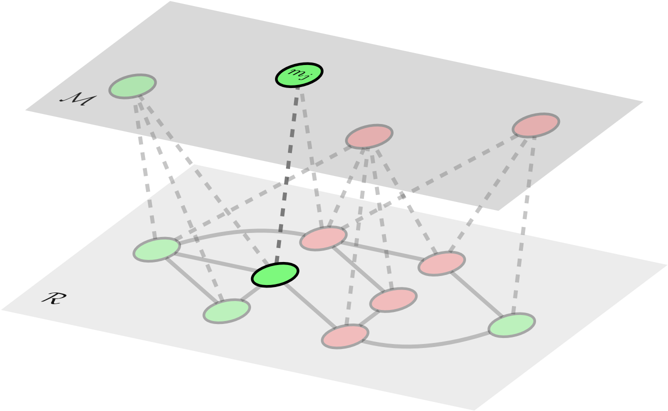
Media clientele $\left(m_j^T(t), m_j^F(t)\right)$



Co-evolution of the Multilayer News Flow

Media repertoire $\left(r_i^T(t), r_i^F(t)\right)$ $r_i^T(t) + r_i^F(t) \leq M \ll R$

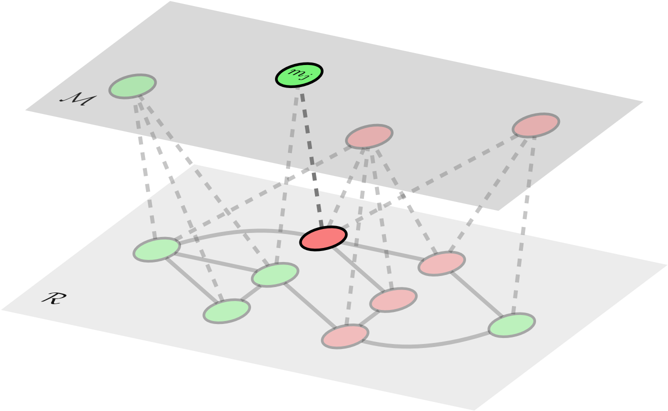
Media clientele $\left(m_j^T(t), m_j^F(t)\right)$



Co-evolution of the Multilayer News Flow

Media repertoire $\left(r_i^T(t), r_i^F(t)\right)$ $r_i^T(t) + r_i^F(t) \leq M \ll R$

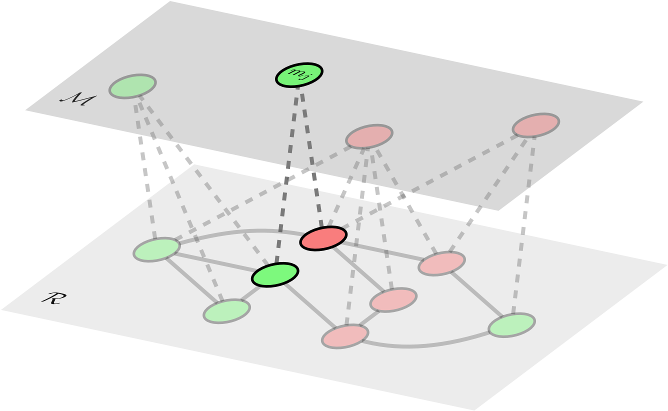
Media clientele $\left(m_j^T(t), m_j^F(t)\right)$



Co-evolution of the Multilayer News Flow

Media repertoire $\left(r_i^T(t), r_i^F(t)\right)$ $r_i^T(t) + r_i^F(t) \leq M \ll R$

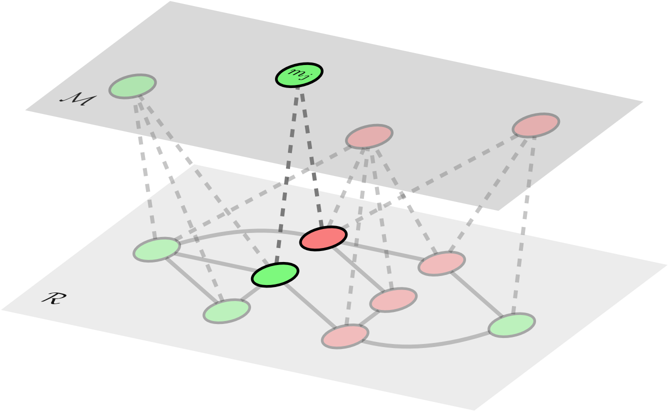
Media clientele $\left(m_j^T(t), m_j^F(t)\right)$



Co-evolution of the Multilayer News Flow

Media repertoire $\left(r_i^T(t), r_i^F(t)\right)$ $\quad r_i^T(t) + r_i^F(t) \leq M \ll R$

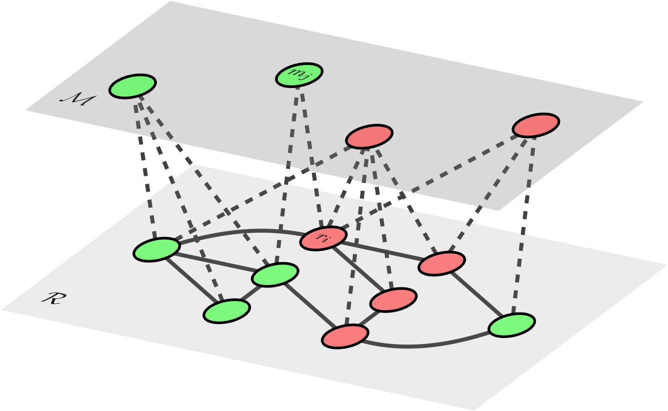
Media clientele $\left(\mu_j^T(t), \mu_j^F(t)\right)$



Co-evolution of the Multilayer News Flow

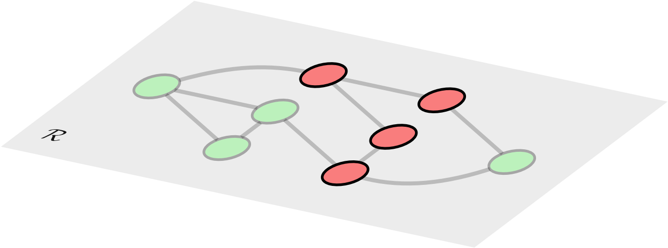
Media repertoire $\left(r_i^T(t), r_i^F(t)\right)$ $r_i^T(t) + r_i^F(t) \leq M \ll R$

Media clientele $\left(m_j^T(t), m_j^F(t)\right)$



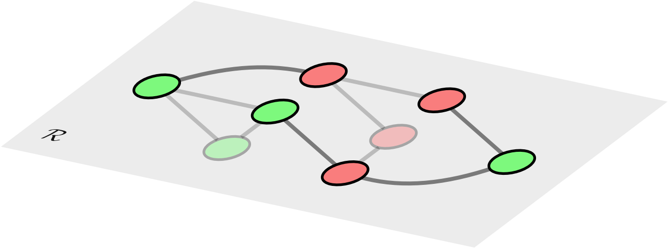
Co-evolution of the Multilayer News Flow

False-news followers $\mathcal{R}^F(t) = \{r_i \mid r_i(t) = 0\}$ $\rho^F(t)$



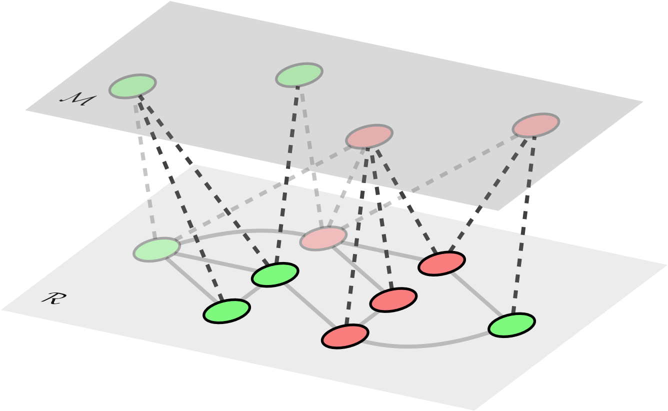
Co-evolution of the Multilayer News Flow

False-news followers	$\mathcal{R}^F(t) = \{r_i \mid r_i(t) = 0\}$	$\rho^F(t)$
Unbalanced edges	$E_R^{FT}(t) = \{r_i r_j \mid r_i(t) \neq r_j(t)\}$	$\rho^{FT}(t)$



Co-evolution of the Multilayer News Flow

False-news followers	$\mathcal{R}^F(t) = \{r_i \mid r_i(t) = 0\}$	$\rho^F(t)$
Unbalanced edges	$E_R^{FT}(t) = \{r_i r_j \mid r_i(t) \neq r_j(t)\}$	$\rho^{FT}(t)$
Polarised individuals	$\mathcal{P}(t) = \{r_i \mid r_i^T(t) \cdot r_i^F(t) = 0\}$	$\rho^P(t)$



Co-evolution of the Multilayer News Flow

Dynamics on the graph

edge-balancing

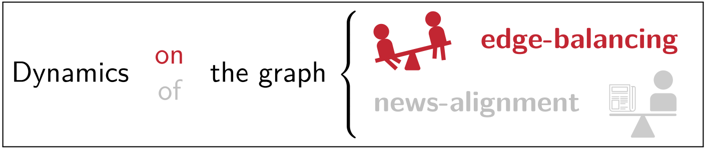
Co-evolution of the Multilayer News Flow

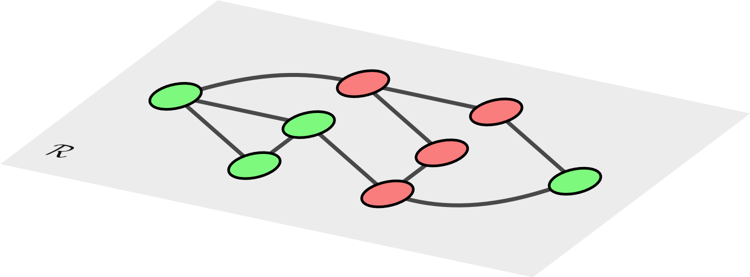
Dynamics on / of the graph

edge-balancing

news-alignment

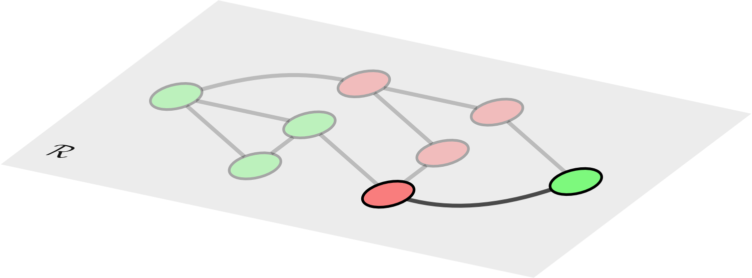
Co-evolution of the Multilayer News Flow





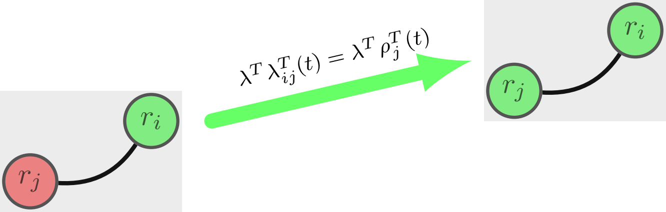
Co-evolution of the Multilayer News Flow

Dynamics on / of the graph



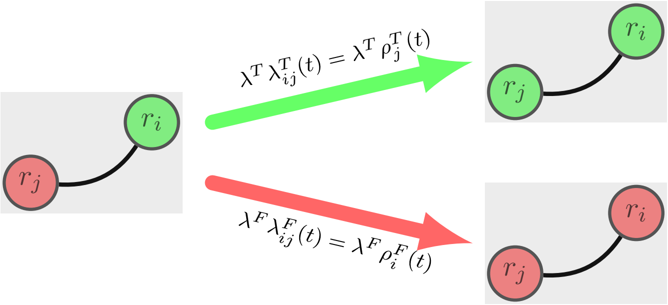
Co-evolution of the Multilayer News Flow

Dynamics on / of the graph {
- **edge-balancing**
- **news-alignment**

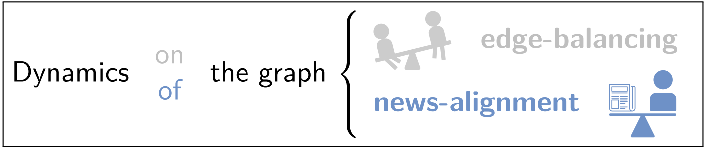


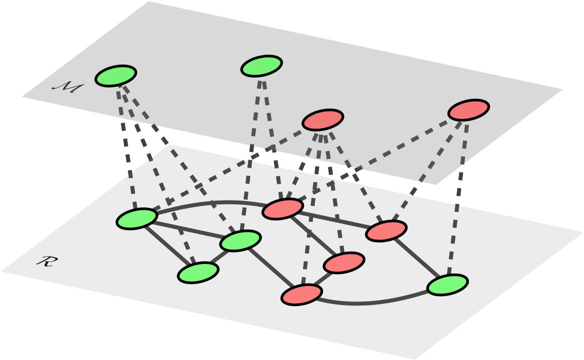
Co-evolution of the Multilayer News Flow

Dynamics on/of the graph { **edge-balancing** / news-alignment



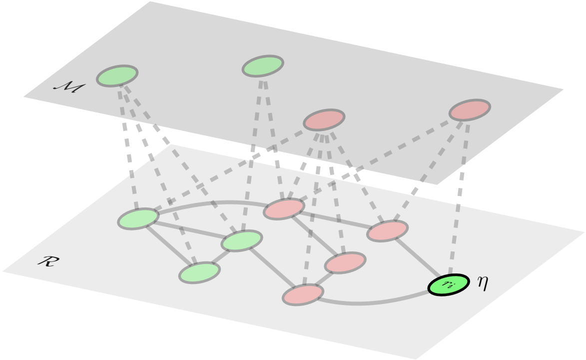
Co-evolution of the Multilayer News Flow





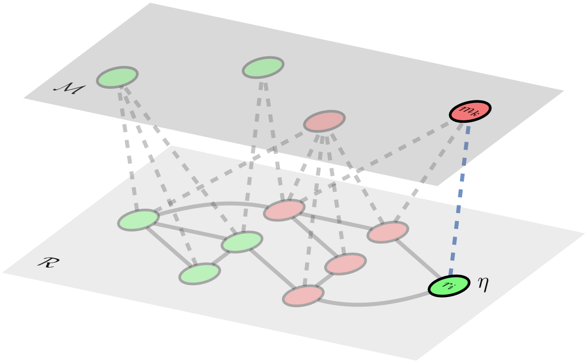
Co-evolution of the Multilayer News Flow

Dynamics on / of the graph {
- edge-balancing
- **news-alignment**



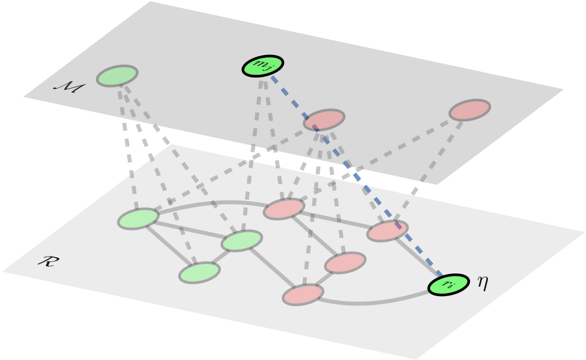
Co-evolution of the Multilayer News Flow

Dynamics on / of the graph $\left\{\begin{array}{l}\text{edge-balancing} \\ \text{news-alignment}\end{array}\right.$



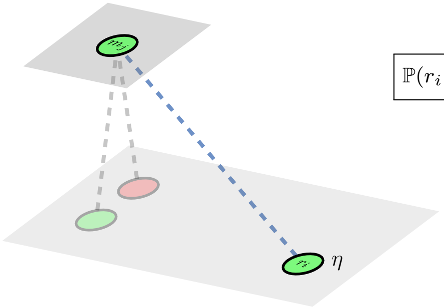
Co-evolution of the Multilayer News Flow

Dynamics on / of the graph
- edge-balancing
- **news-alignment**



Co-evolution of the Multilayer News Flow

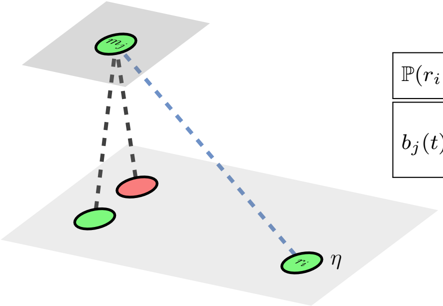
Dynamics on / of the graph $\left\{ \begin{array}{l} \text{edge-balancing} \\ \text{news-alignment} \end{array} \right.$



$$\boxed{\mathbb{P}(r_i \to m_j, t) \propto b_j(t) p_j^T(t)}$$

Co-evolution of the Multilayer News Flow

Dynamics on / of the graph $\left\{ \begin{array}{l} \text{edge-balancing} \\ \text{news-alignment} \end{array} \right.$

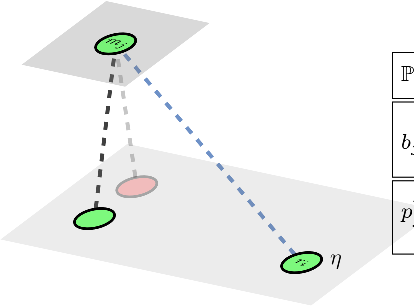


$$\boxed{\mathbb{P}(r_i \to m_j, t) \propto b_j(t) p_j^T(t)}$$

$$\boxed{b_j(t) := \frac{-\sum_S \mu_j^S(t) \log \mu_j^S(t)}{\log 2}}$$

Co-evolution of the Multilayer News Flow

Dynamics on / of the graph
- edge-balancing
- news-alignment



$$\boxed{\mathbb{P}(r_i \to m_j, t) \propto b_j(t) p_j^T(t)}$$

$$\boxed{b_j(t) := \frac{-\sum_S \mu_j^S(t) \log \mu_j^S(t)}{\log 2}}$$

$$\boxed{p_j^T(t) := \frac{1}{1 + e^{-m_j^T(t)}}}$$

Co-evolution of the Multilayer News Flow

[https://github.com/ikicab/NewsFlow]

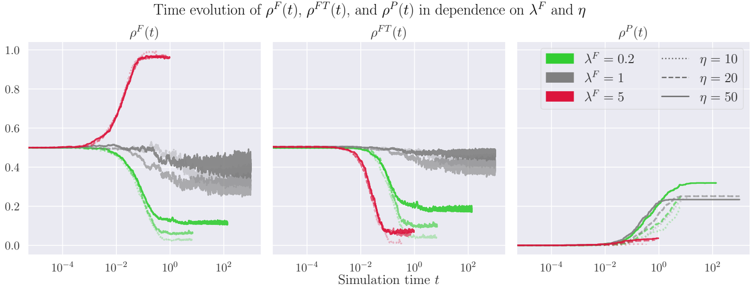
Gillespie algorithm

while $\left(\mathrm{Var}_n(P(t_n)) \geq \mathtt{tol}\right)$ **and** $\left(\mathrm{Var}_n(|E_R^{FT}(t_n)|) \geq \mathtt{tol}\right)$ **repeat**

1. $\lambda(t_n) = R\eta + \lambda^T \Lambda^T(t_n) + \lambda^F \Lambda^F(t_n)$
2. $\tau(t_n) = -\ln r / \lambda(t_n) = 1/\lambda(t_n)$
3. draw $\mathscr{A} \in \{\mathscr{A}_N, \mathscr{A}_B^T, \mathscr{A}_B^F\}$ at random:
   $$\mathbb{P}(\mathscr{A}_N) = \frac{R\eta}{\lambda(t_n)}, \quad \mathbb{P}(\mathscr{A}_B^T) = \frac{\lambda^T \Lambda^T(t_n)}{\lambda(t_n)}, \quad \mathbb{P}(\mathscr{A}_B^F) = \frac{\lambda^F \Lambda^F(t_n)}{\lambda(t_n)}$$
4. draw either $r_i \in \mathcal{R}$ or $r_i r_j \in E_R^{FT}(t_n)$ at random:
   $$\mathbb{P}(r_i \mid \mathscr{A}_N) = \frac{1}{R}, \quad \mathbb{P}(r_i r_j \mid \mathscr{A}_B^T) = \frac{\lambda_{ij}^T(t_n)}{\Lambda^T(t_n)}, \quad \mathbb{P}(r_i r_j \mid \mathscr{A}_B^F) = \frac{\lambda_{ij}^F(t_n)}{\Lambda^F(t_n)}$$
5. update the system; $t_n \leftarrow t_n + \tau(t_n)$

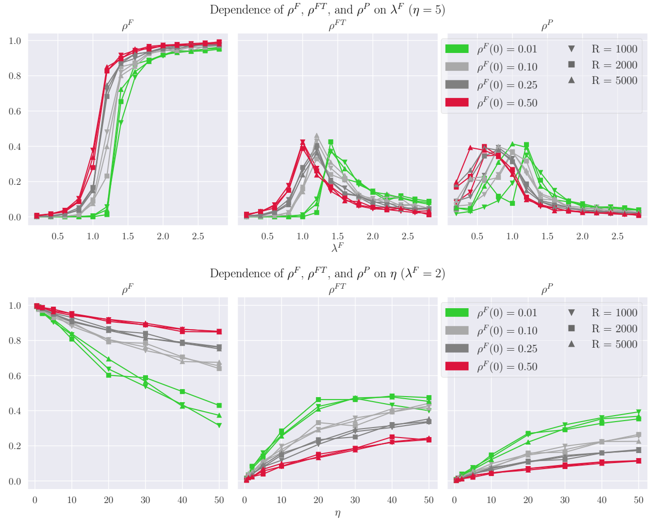
Co-evolution of the Multilayer News Flow

Experiments [macro-scale dynamics]



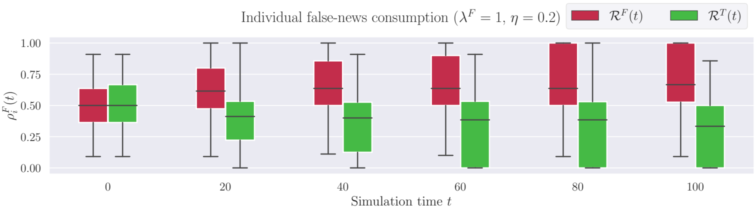
Co-evolution of the Multilayer News Flow

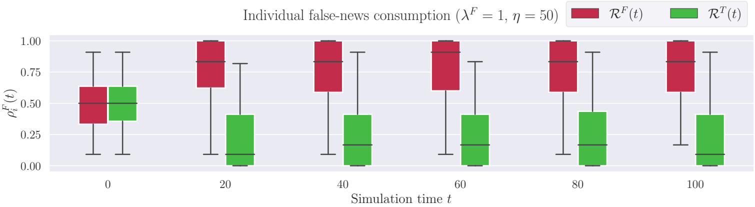
Experiments [macro-scale dynamics]



Co-evolution of the Multilayer News Flow

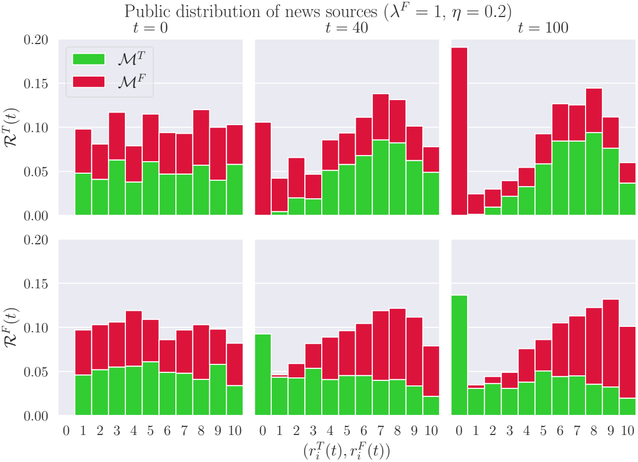
Experiments [micro-scale dynamics; news consumers]





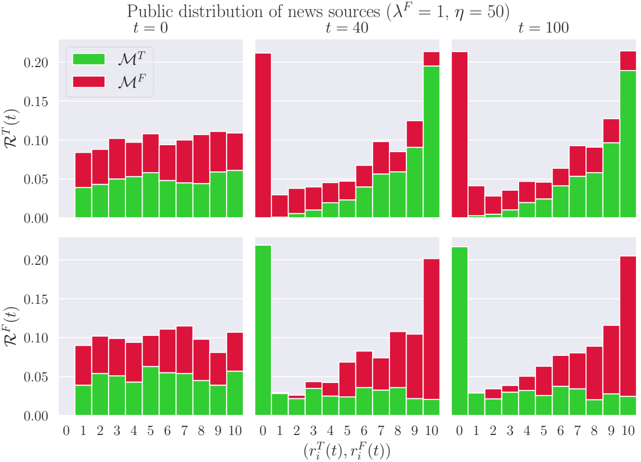
Co-evolution of the Multilayer News Flow

Experiments [micro-scale dynamics; news consumers]



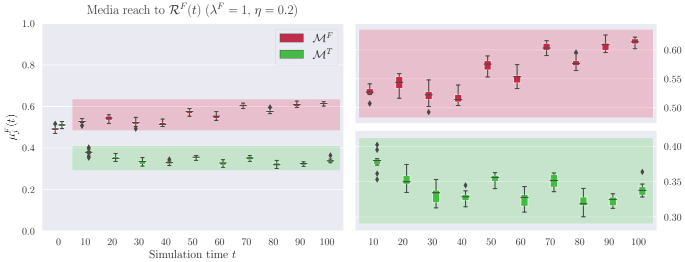
Co-evolution of the Multilayer News Flow

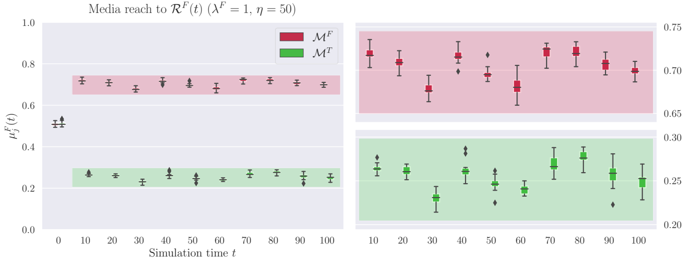
Experiments [micro-scale dynamics; news consumers]



Co-evolution of the Multilayer News Flow

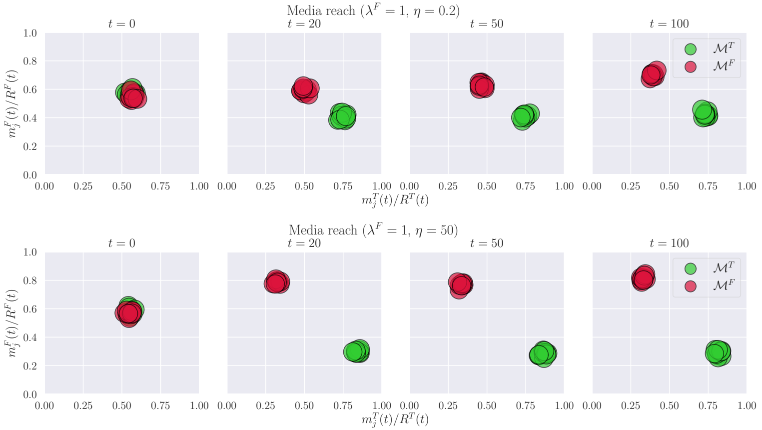
Experiments [micro-scale dynamics; news providers]





Co-evolution of the Multilayer News Flow

Experiments [micro-scale dynamics; news providers]




Thank you