Discrete logarithms

Tanja Lange

Eindhoven University of Technology

29 September 2020

Notation and target

Setting: cyclic group $G = \langle g \rangle$; group operation written multiiplicatively on these slides.
Given $h \in G$ as well as g and how to compute in G.
Target: Compute $a = \log_g h$, i.e. a such that $h = g^a$.
a is called the discrete logarithm of h to base g.
We typically take $0 \leq a < \ell$.

1. Pohlig-Hellman reduces the DLP in G to several DLPs in prime-order subgroups of G.
2. This part covers DLP in groups of prime order ℓ.

Baby-step giant-step algorithm

Let $0 < m < \ell$ then there exist $0 \leq a_0 < m$ and $0 \leq a_1 \leq \ell/m + 1$ with

$$a = a_0 + ma_1.$$

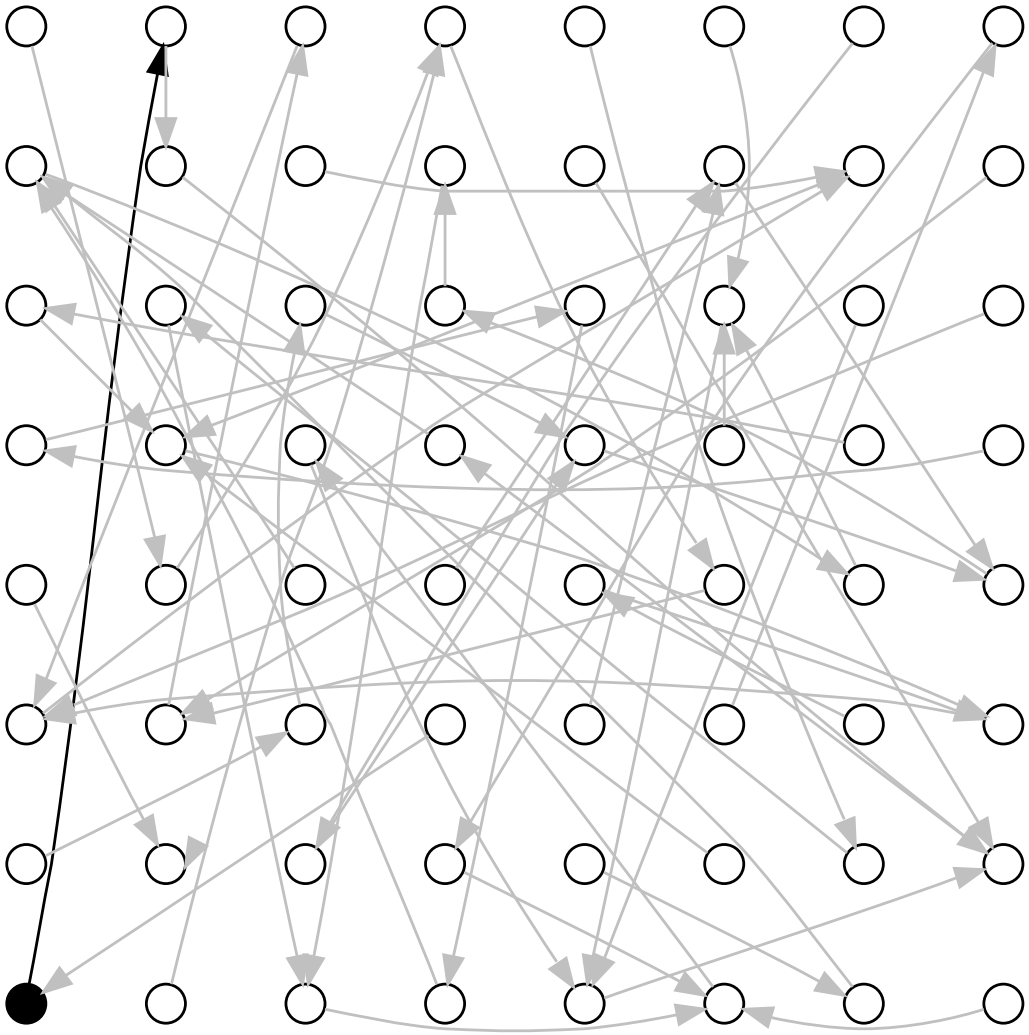
Recompute all possibilities for g^{a_0}, this costs m steps and m space.
$a = a_0 + ma_1$ means $h = g^a = g^{a_0 + ma_1}$, thus

$$g^{a_0} = h \cdot g^{-a_1 m}$$

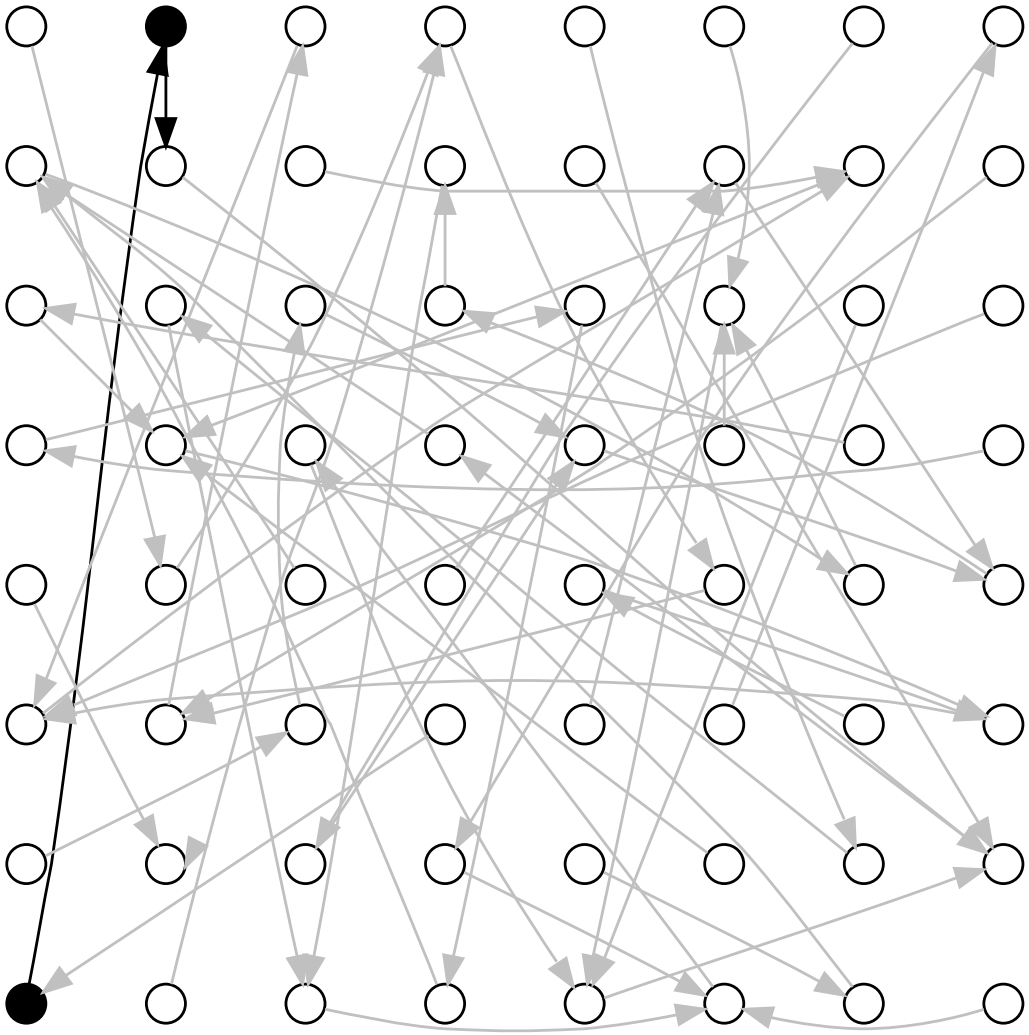
Finding this match takes at most $\ell/m + 1$ steps. Each step multiplies by g^{-m}, which we precompute from $g \cdot g^{m-1}$ and one XGCD.

We balance both for $m \sim \sqrt{\ell}$ to get to a total cost of $2\sqrt{\ell}+$ some small constant expenses, hence the cost of $O(\sqrt{\ell})$.
Note that this also costs $\sqrt{\ell}$ in storage.

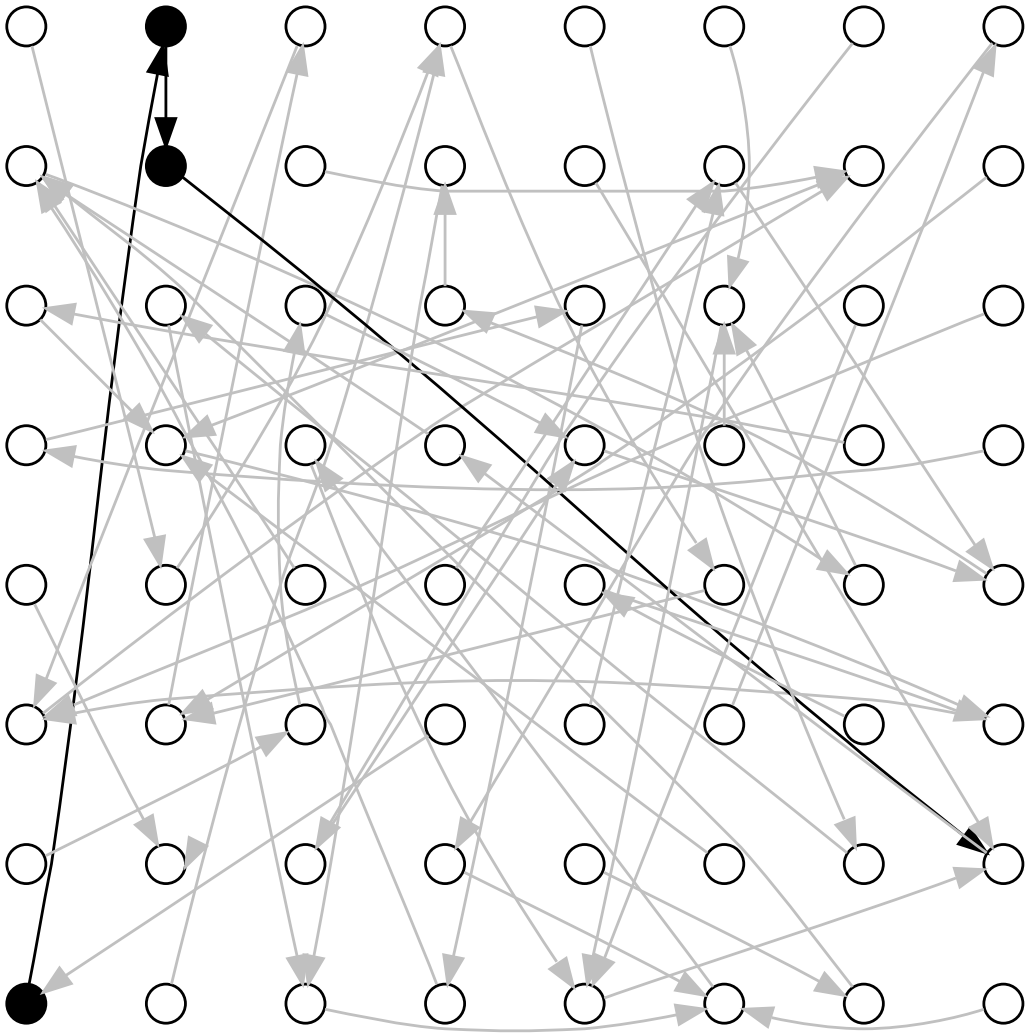
Pollard's rho method



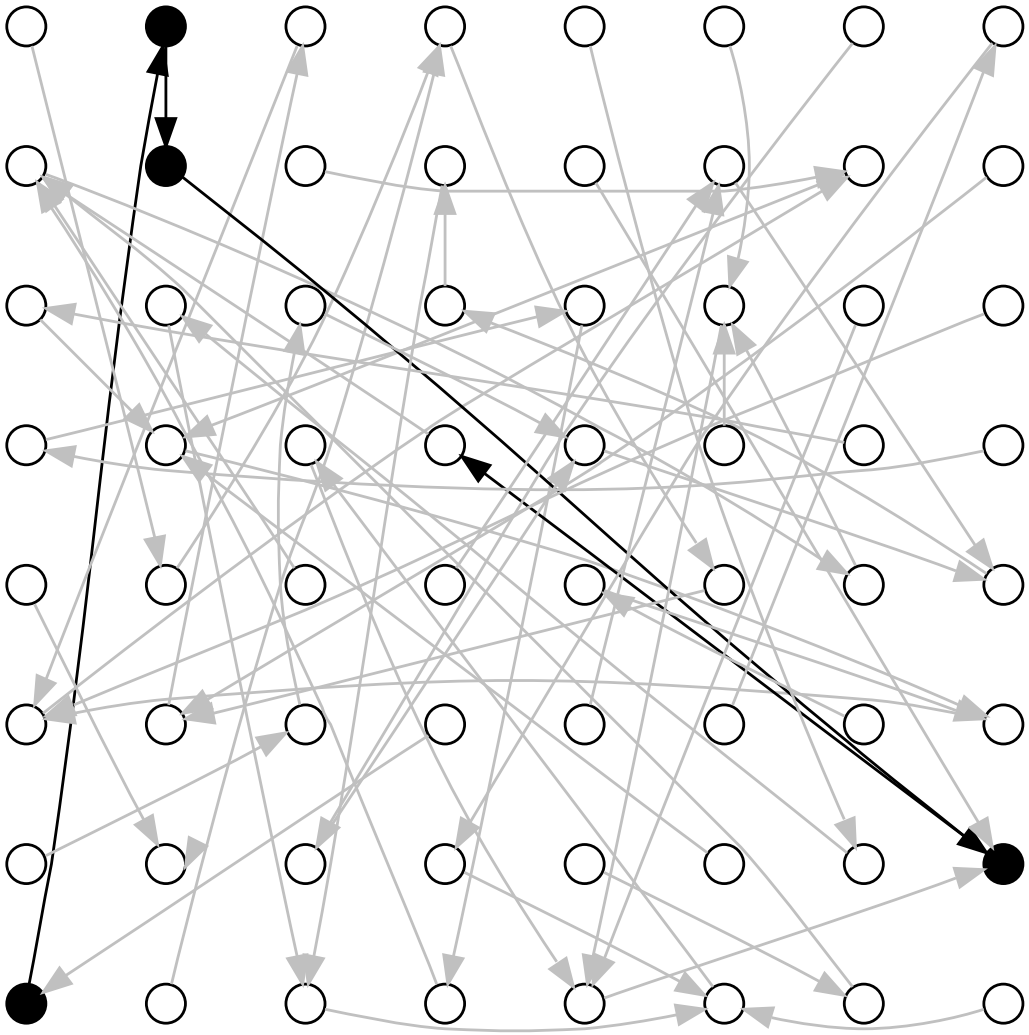
Pollard's rho method



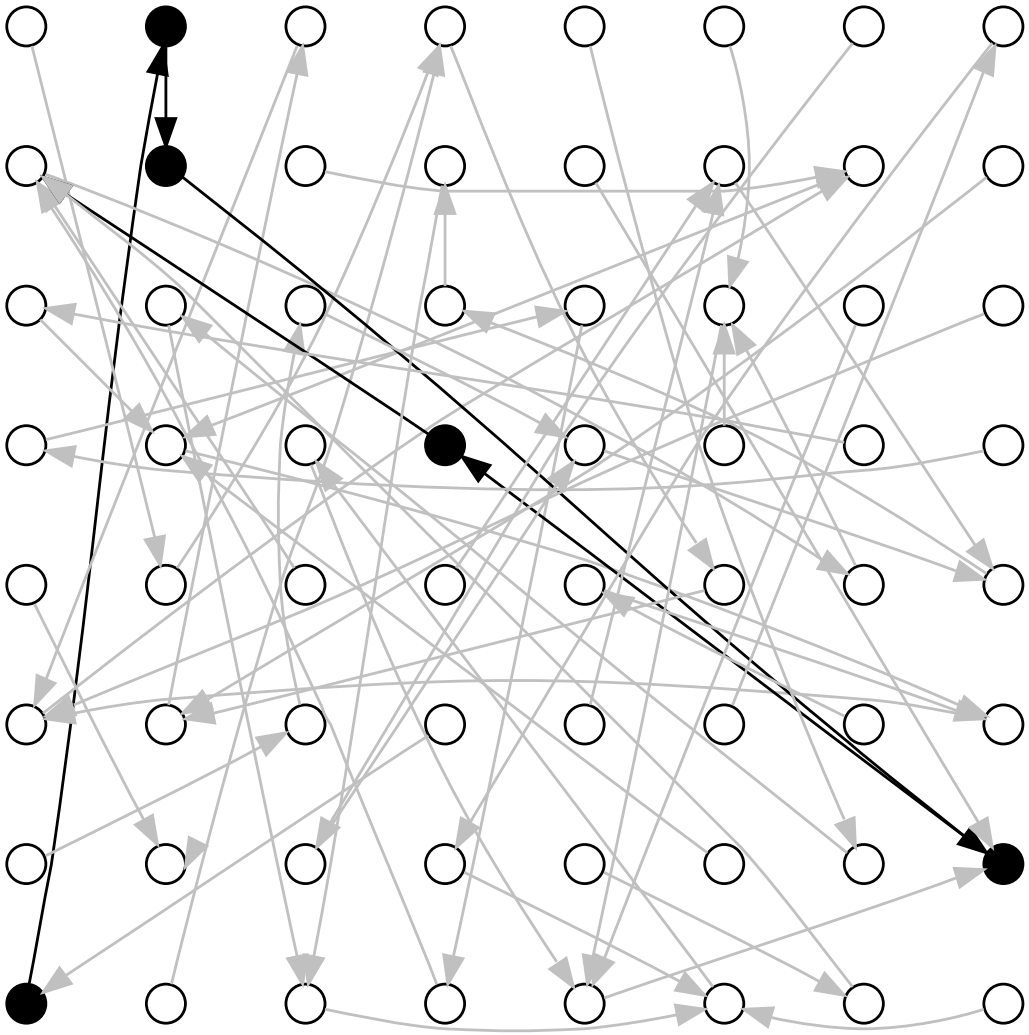
Pollard's rho method



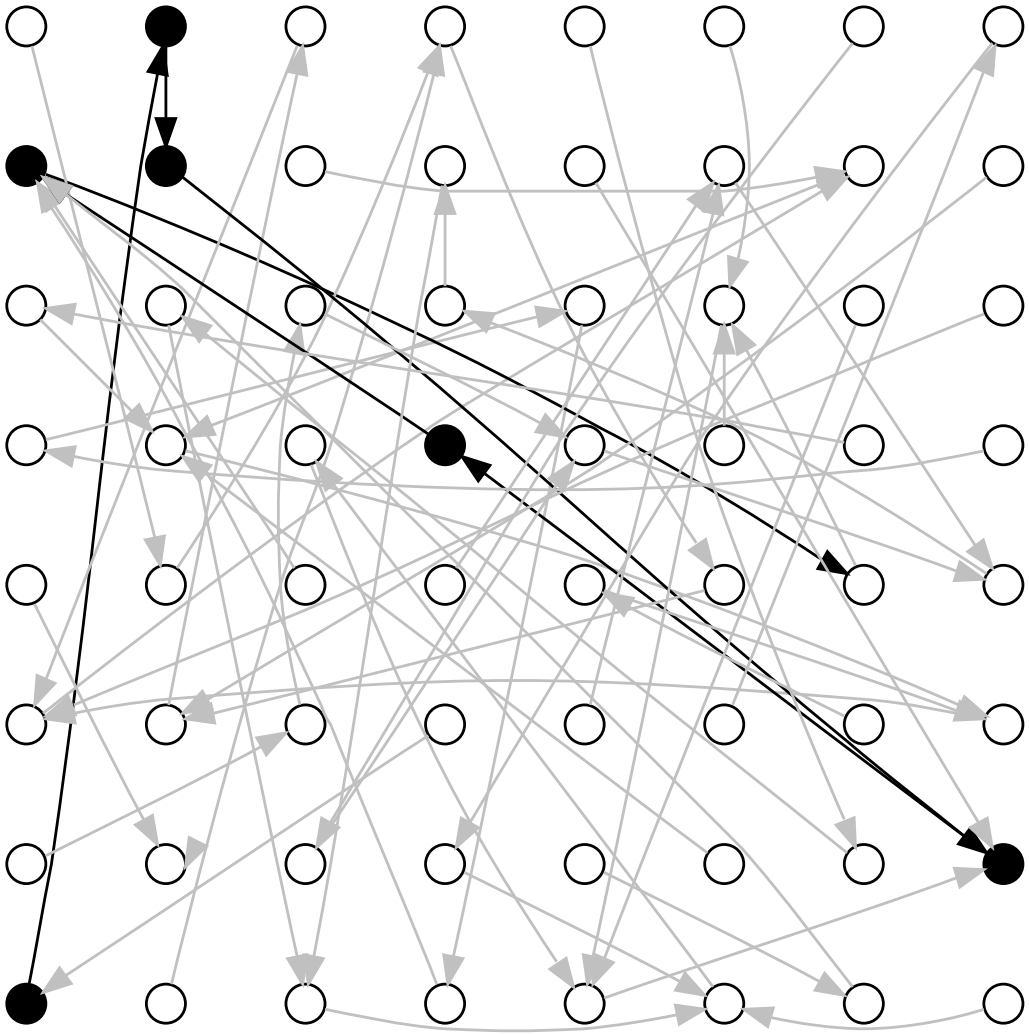
Pollard's rho method



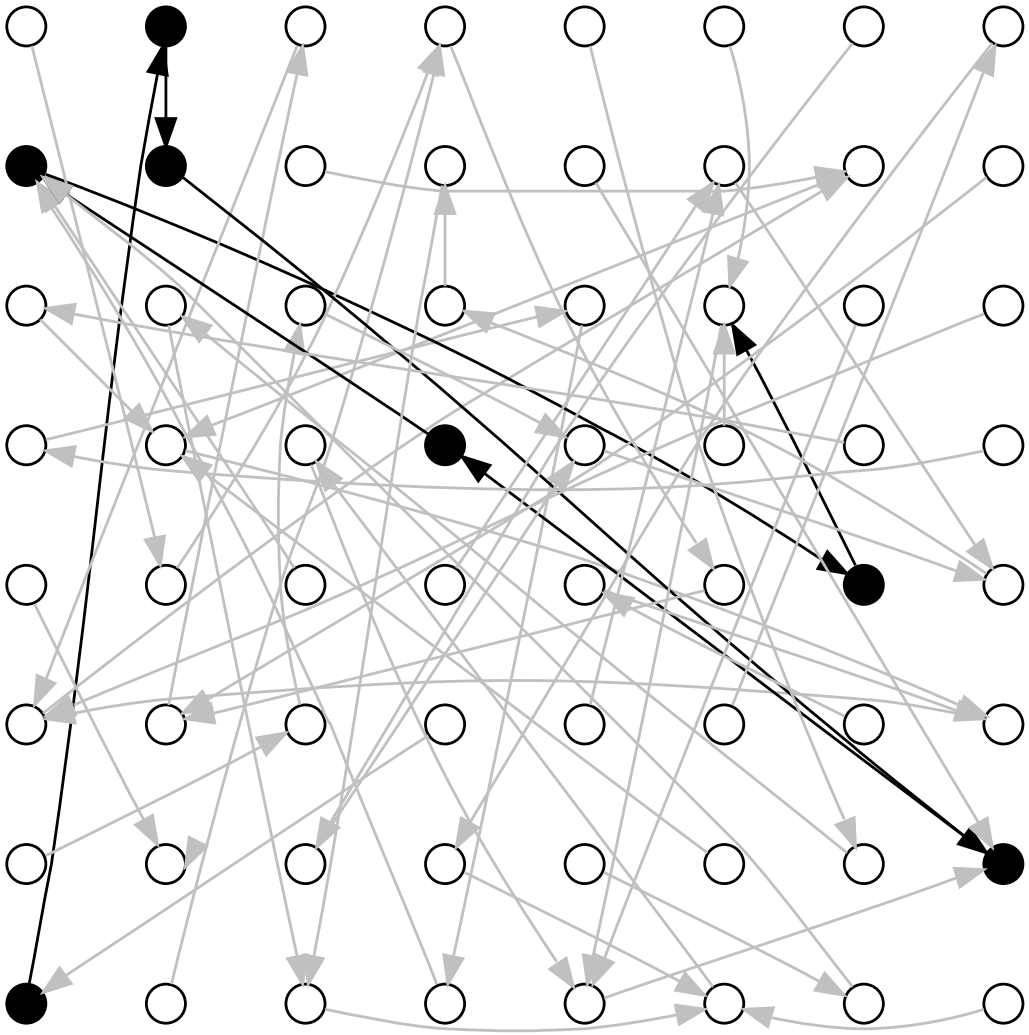
Pollard's rho method



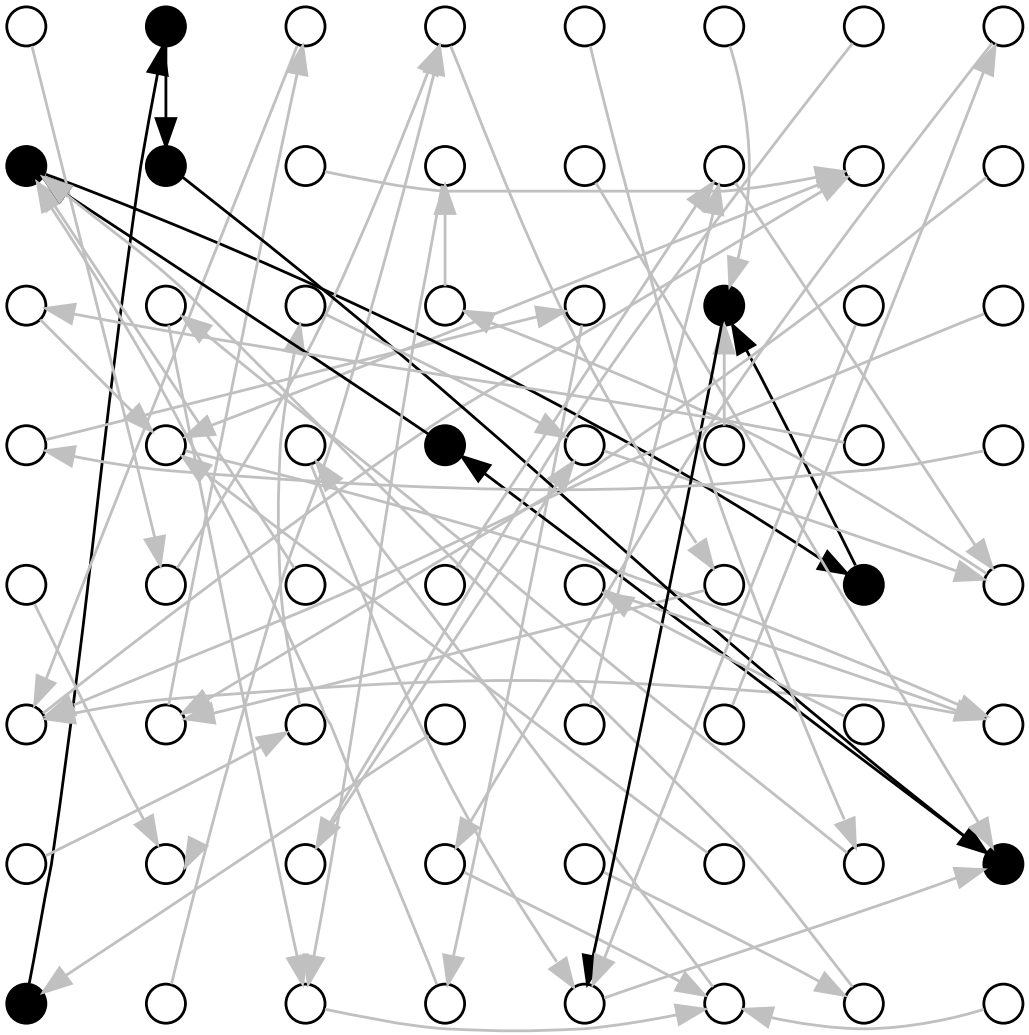
Pollard's rho method



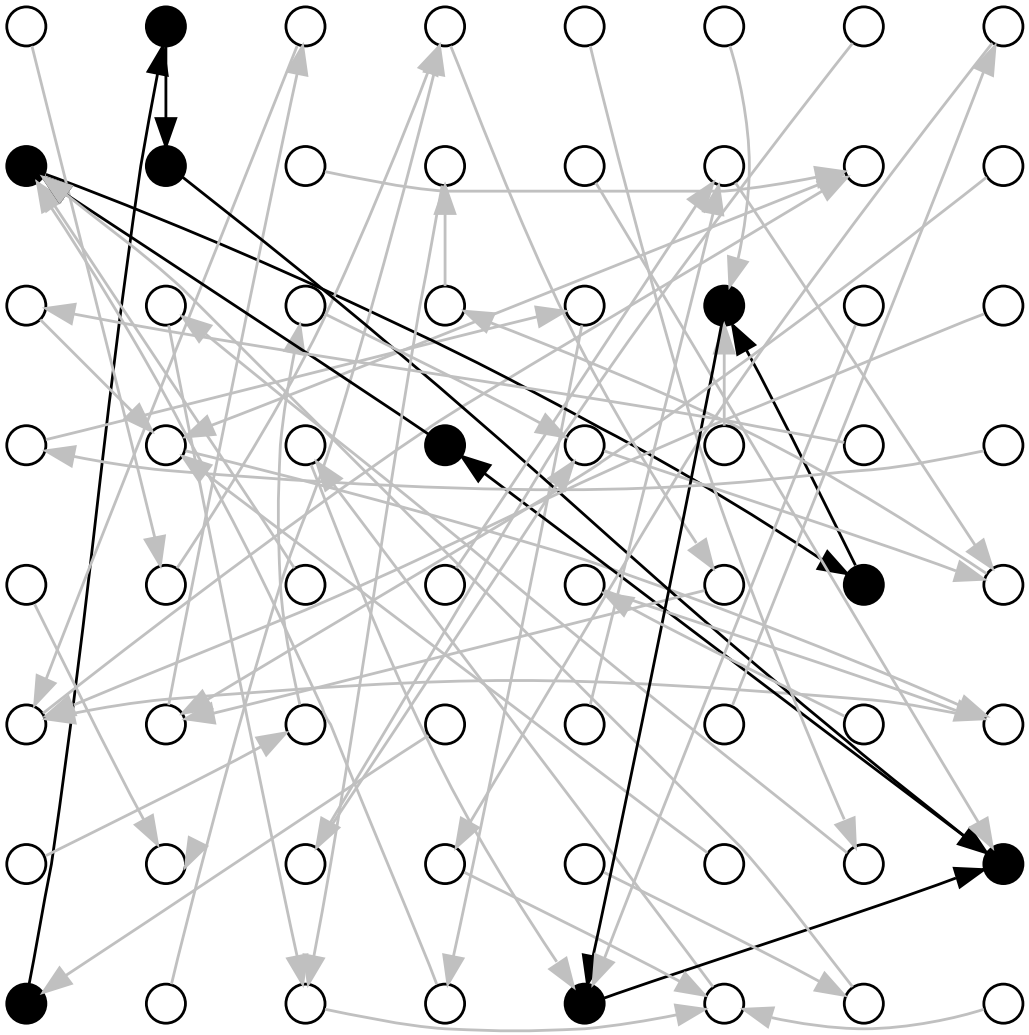
Pollard's rho method



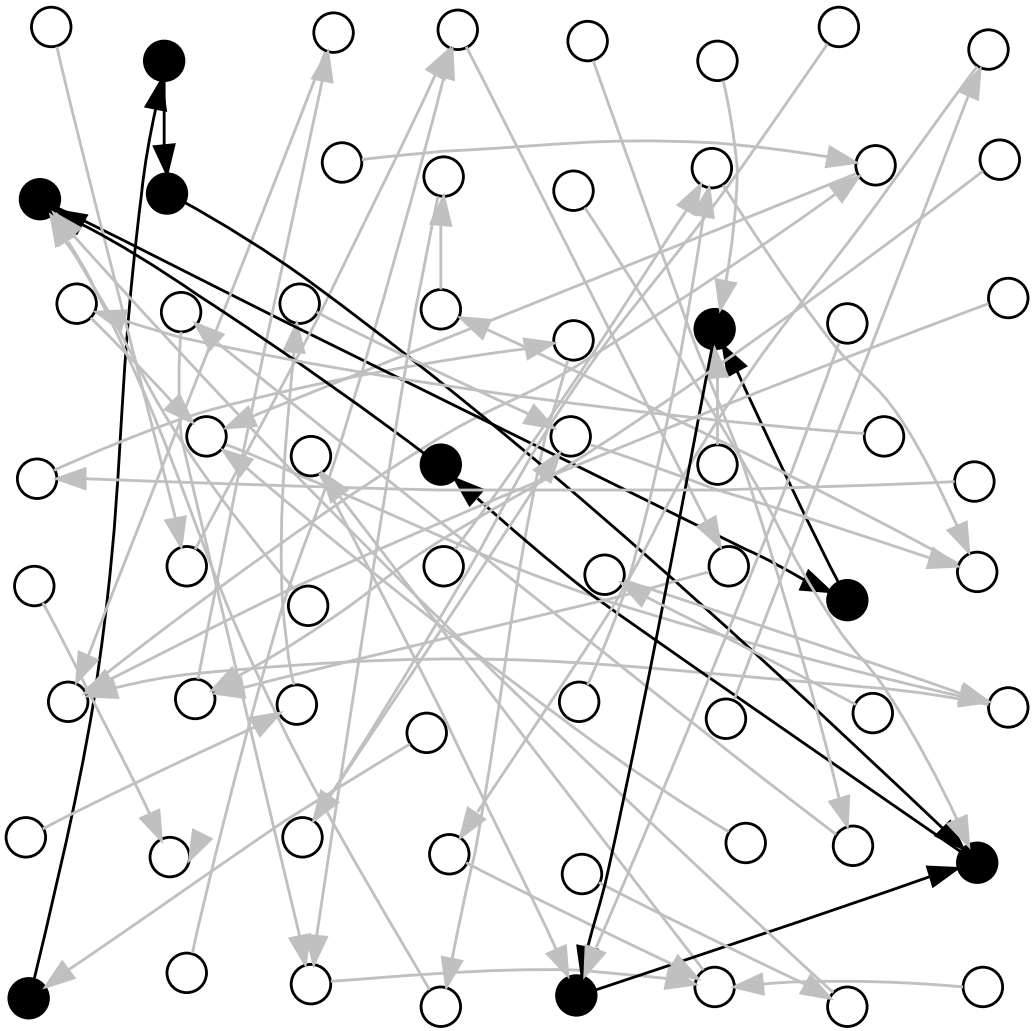
Pollard's rho method



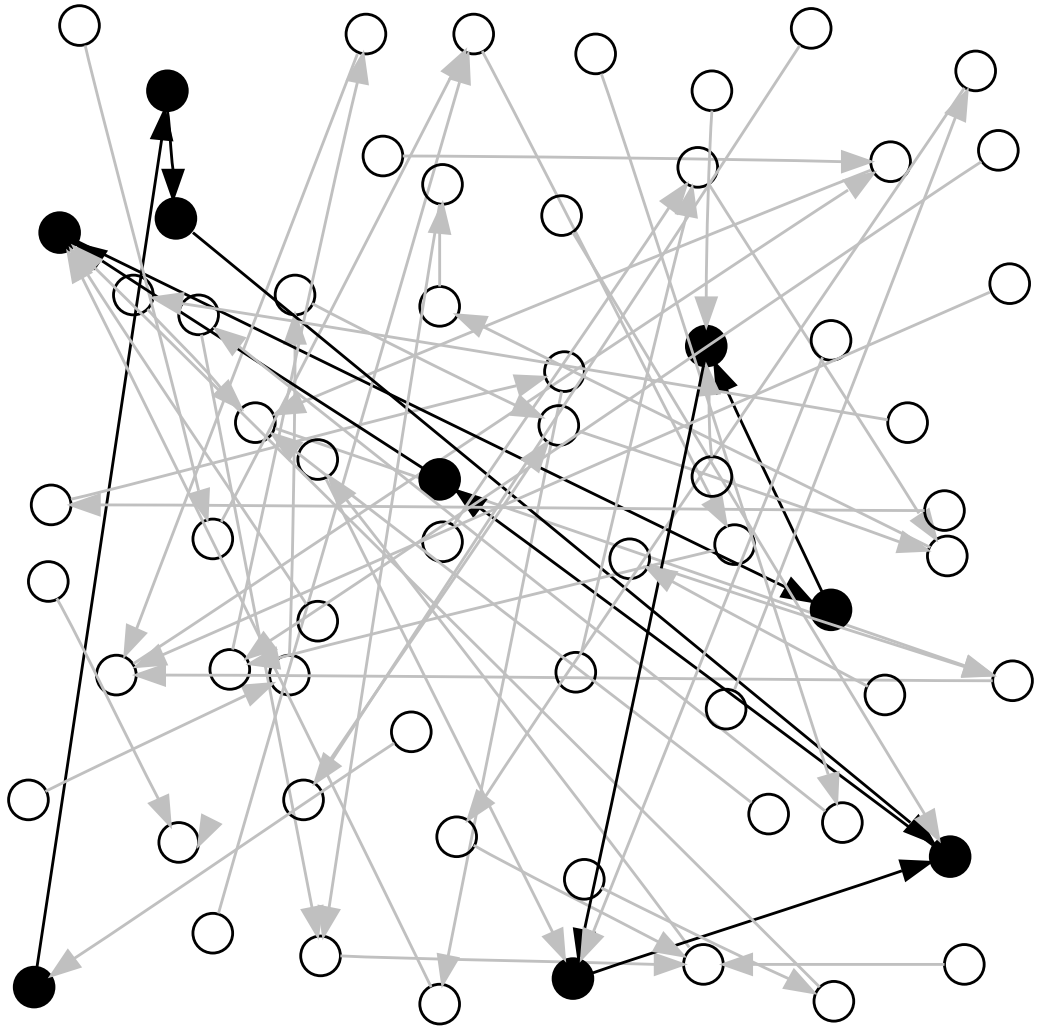
Pollard's rho method



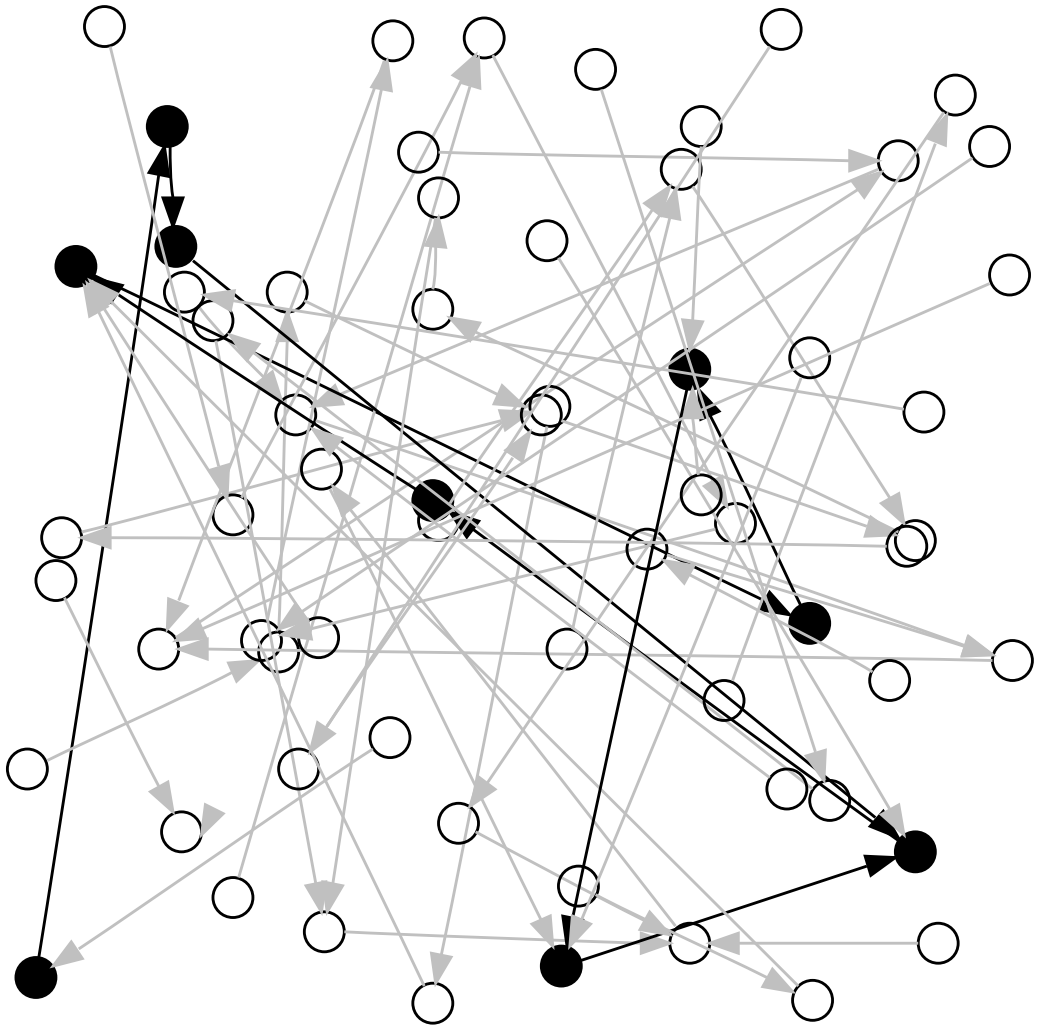
Pollard's rho method



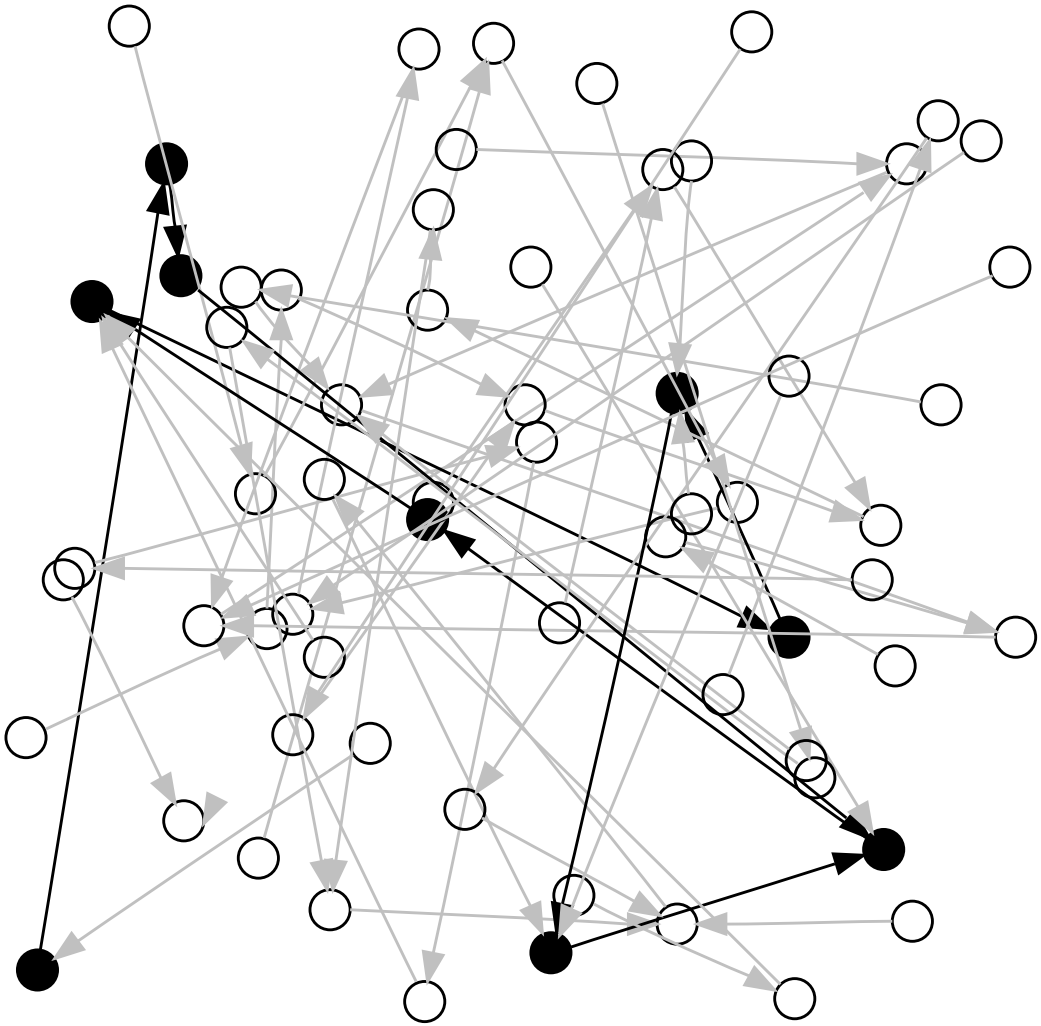
Pollard's rho method



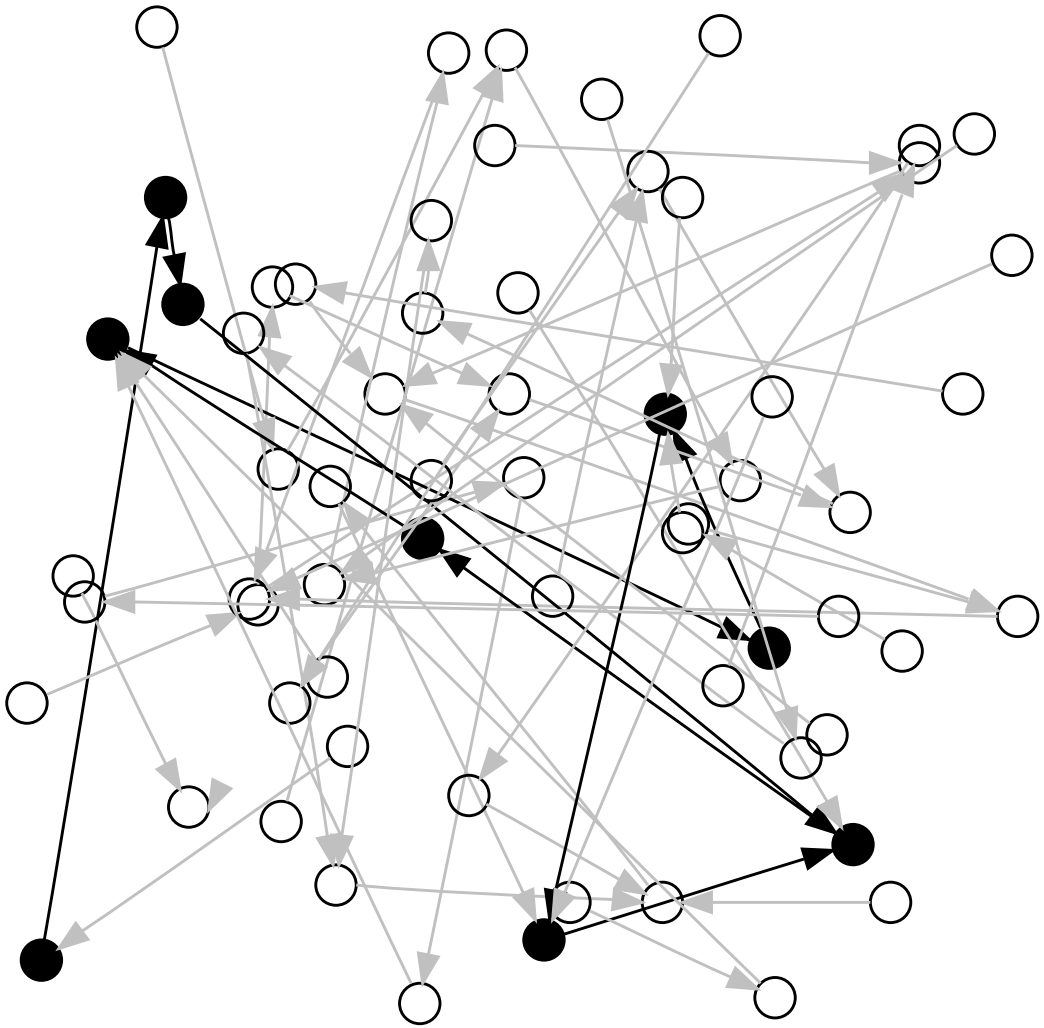
Pollard's rho method



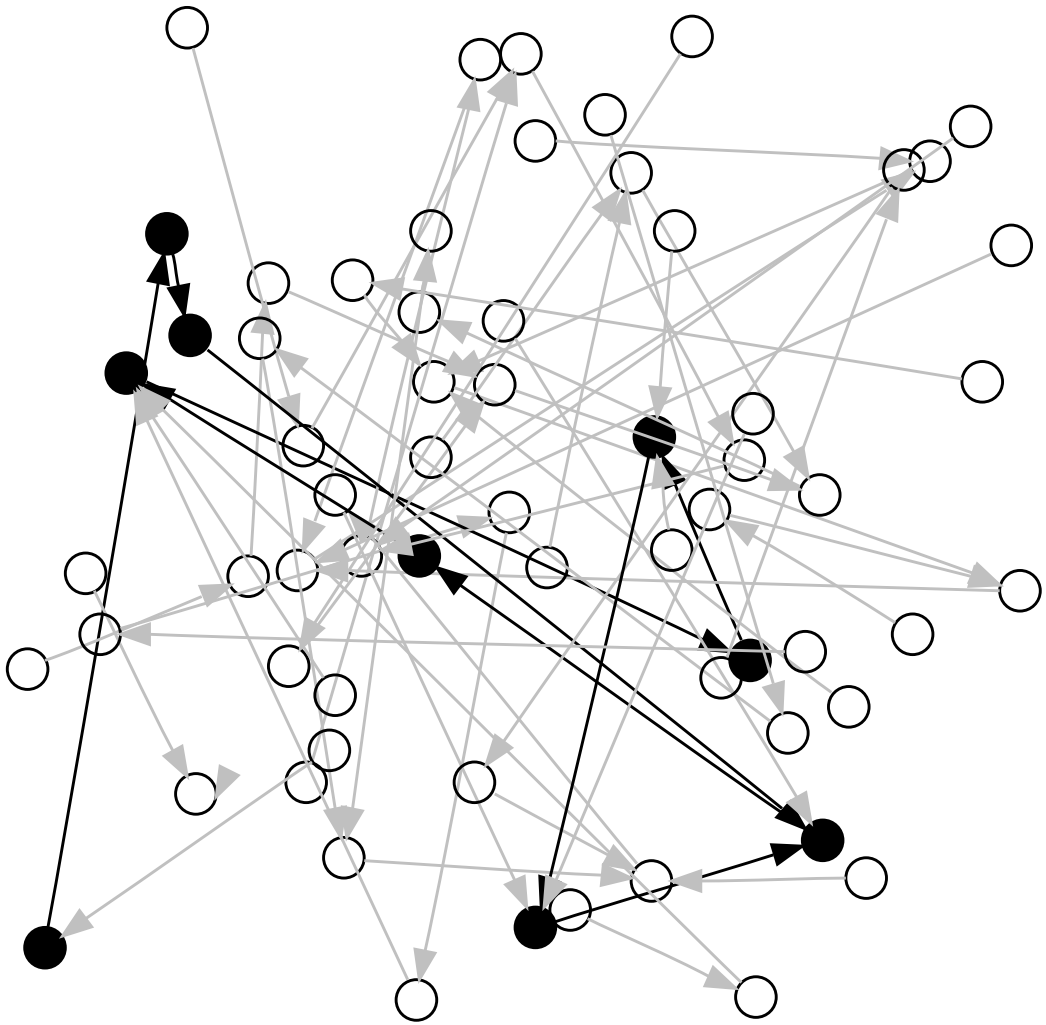
Pollard's rho method



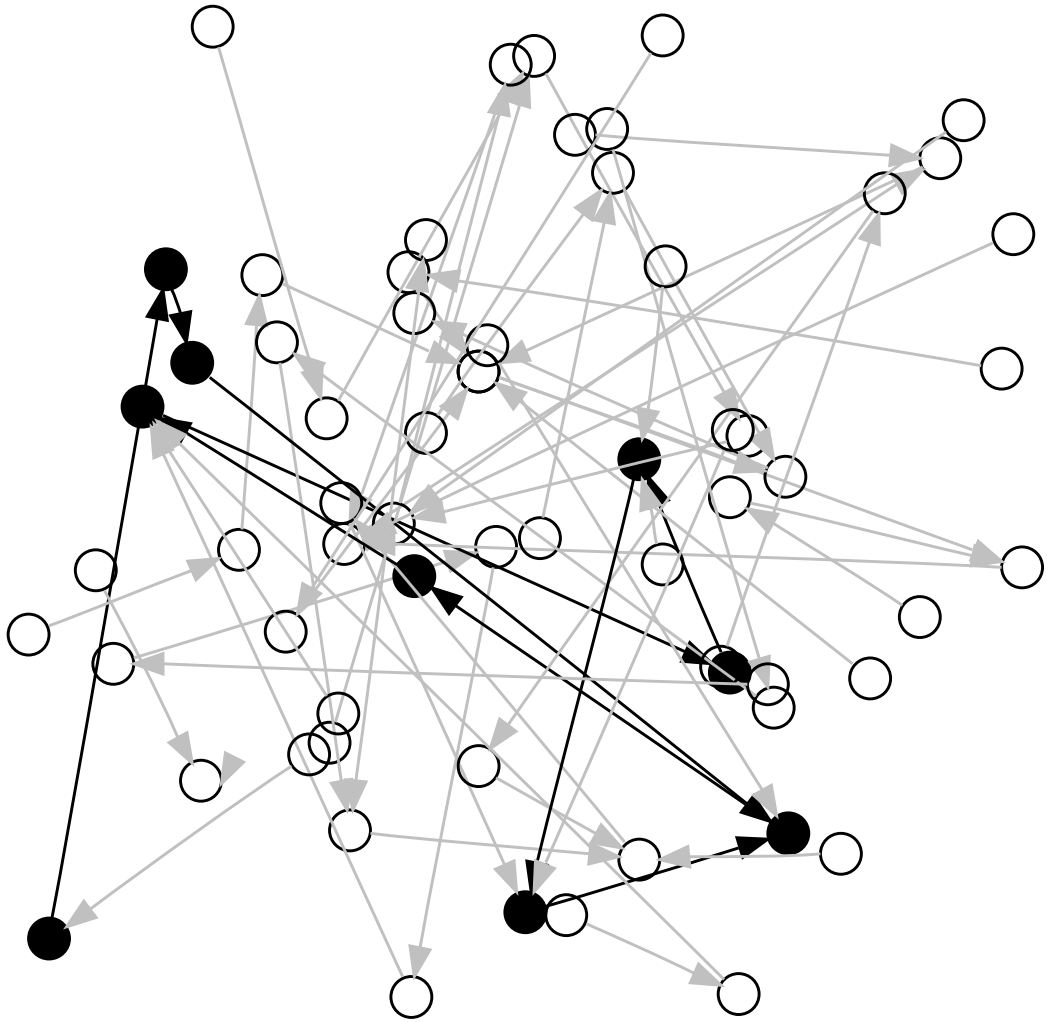
Pollard's rho method



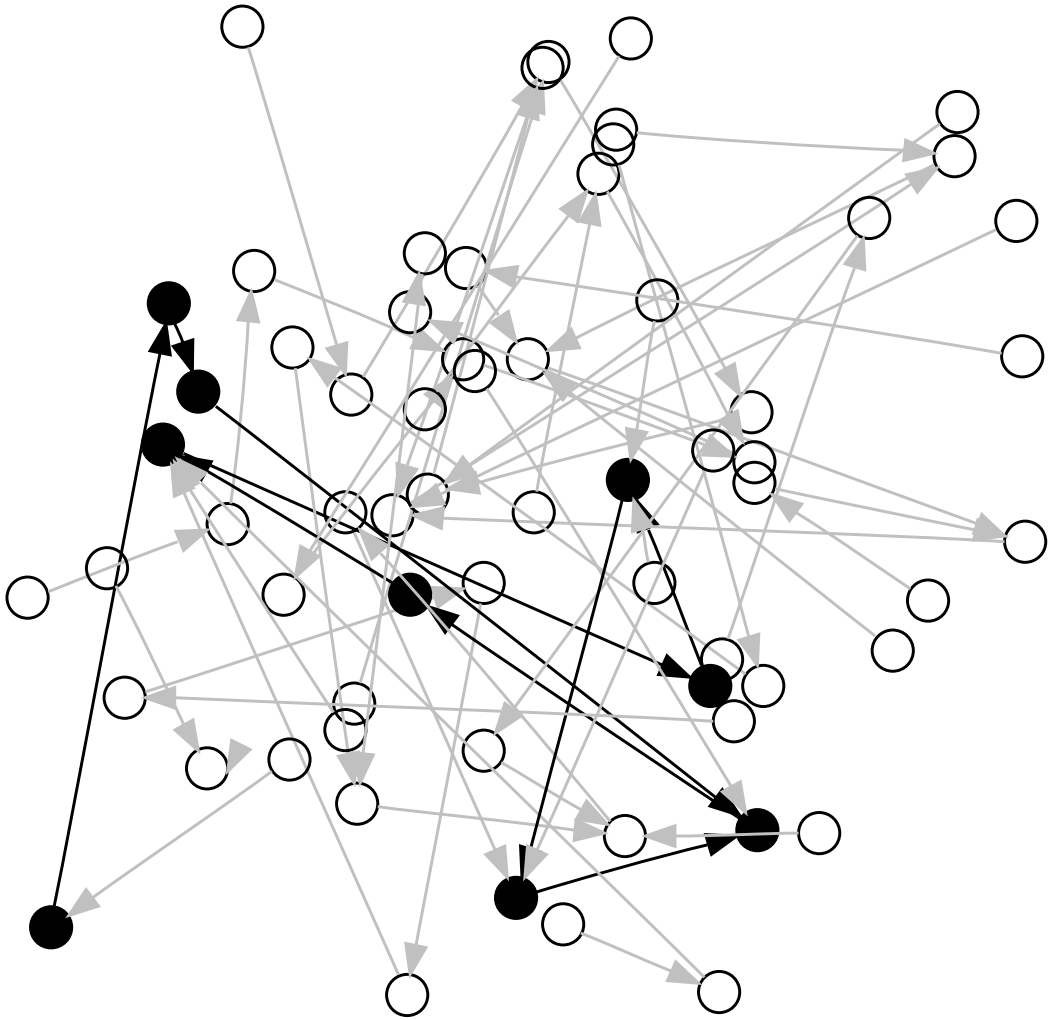
Pollard's rho method



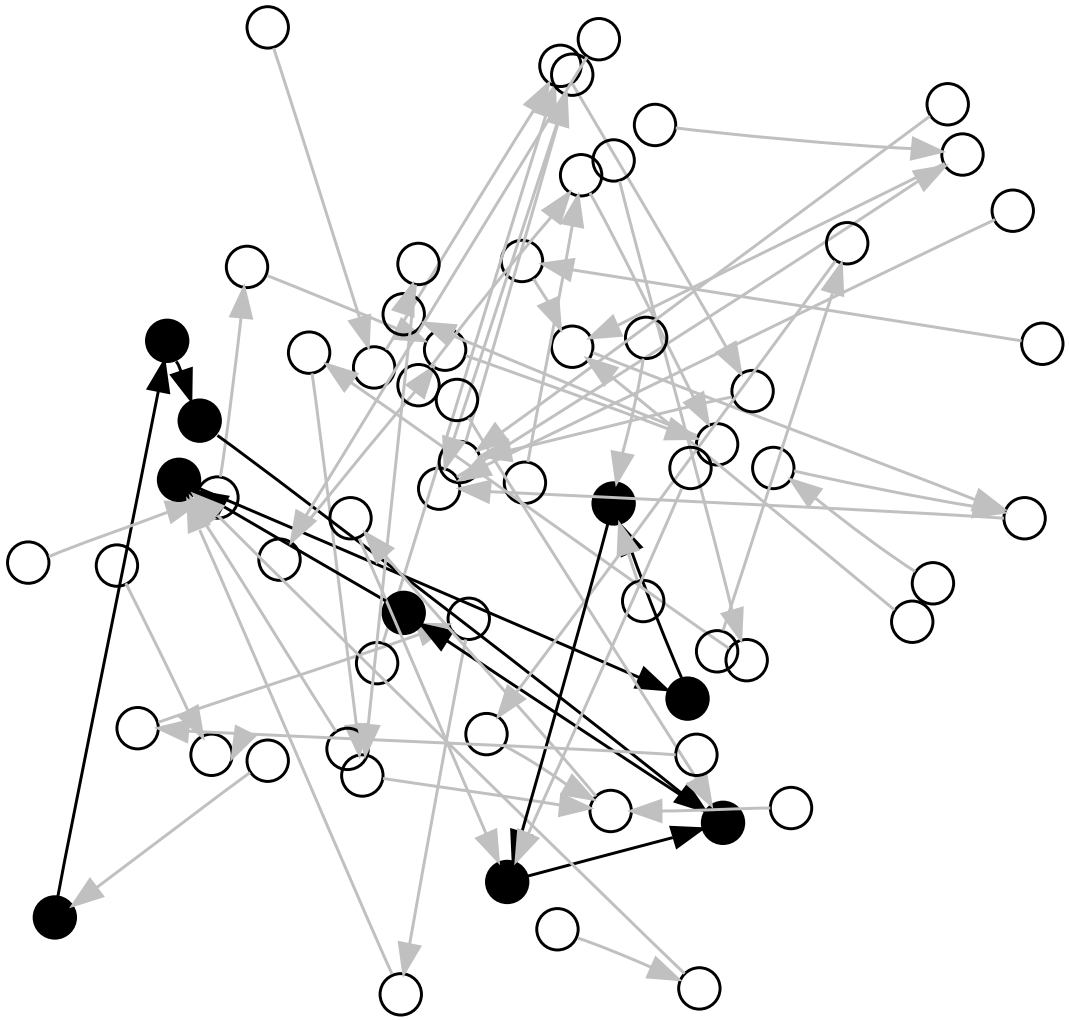
Pollard's rho method



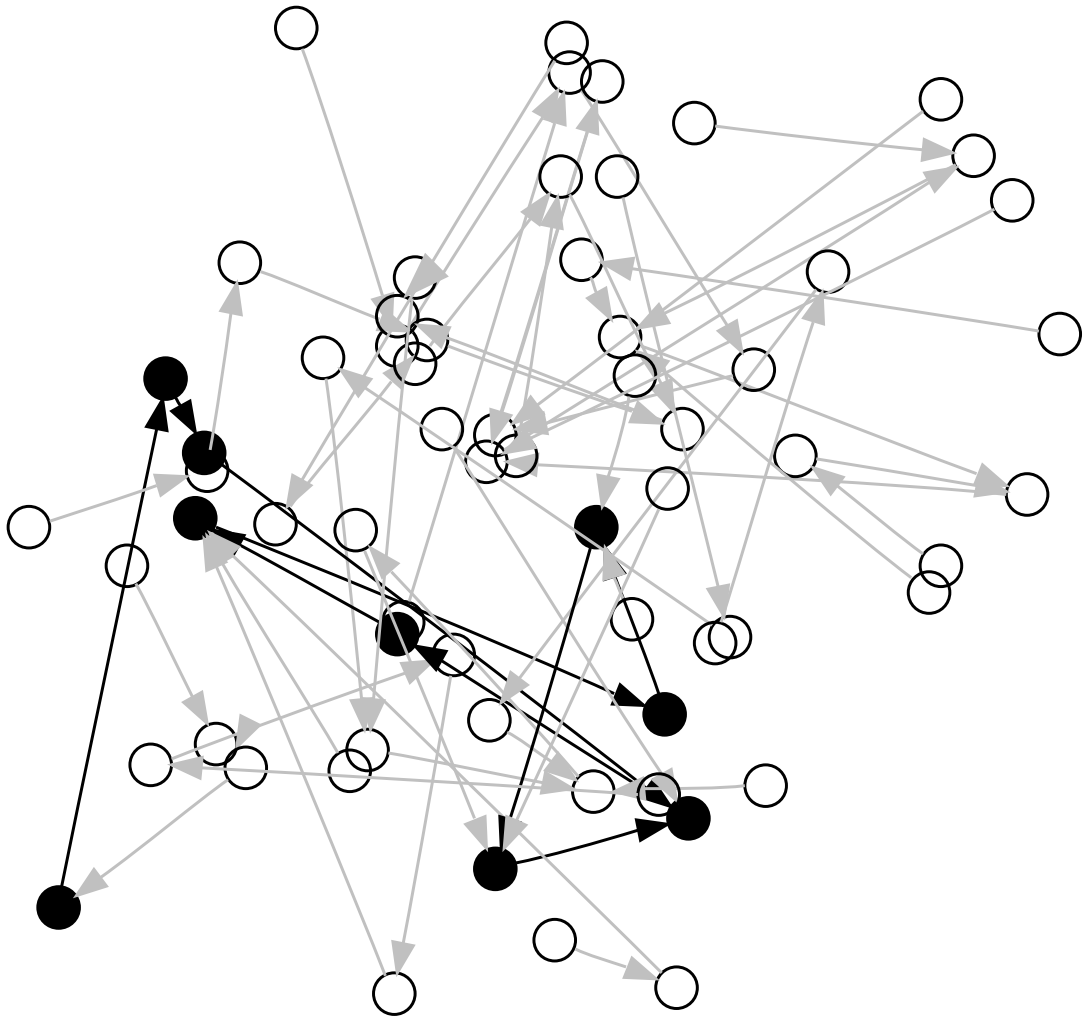
Pollard's rho method



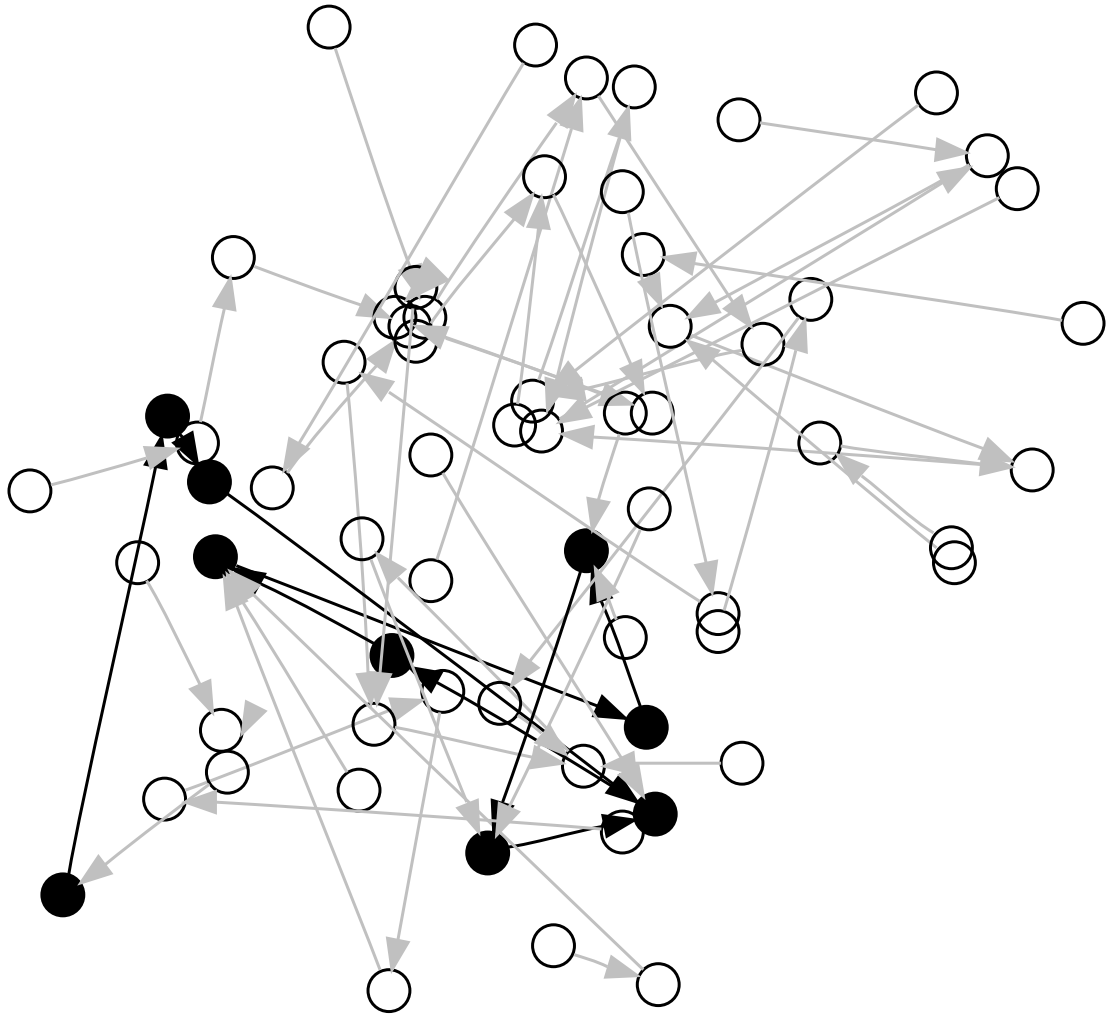
Pollard's rho method



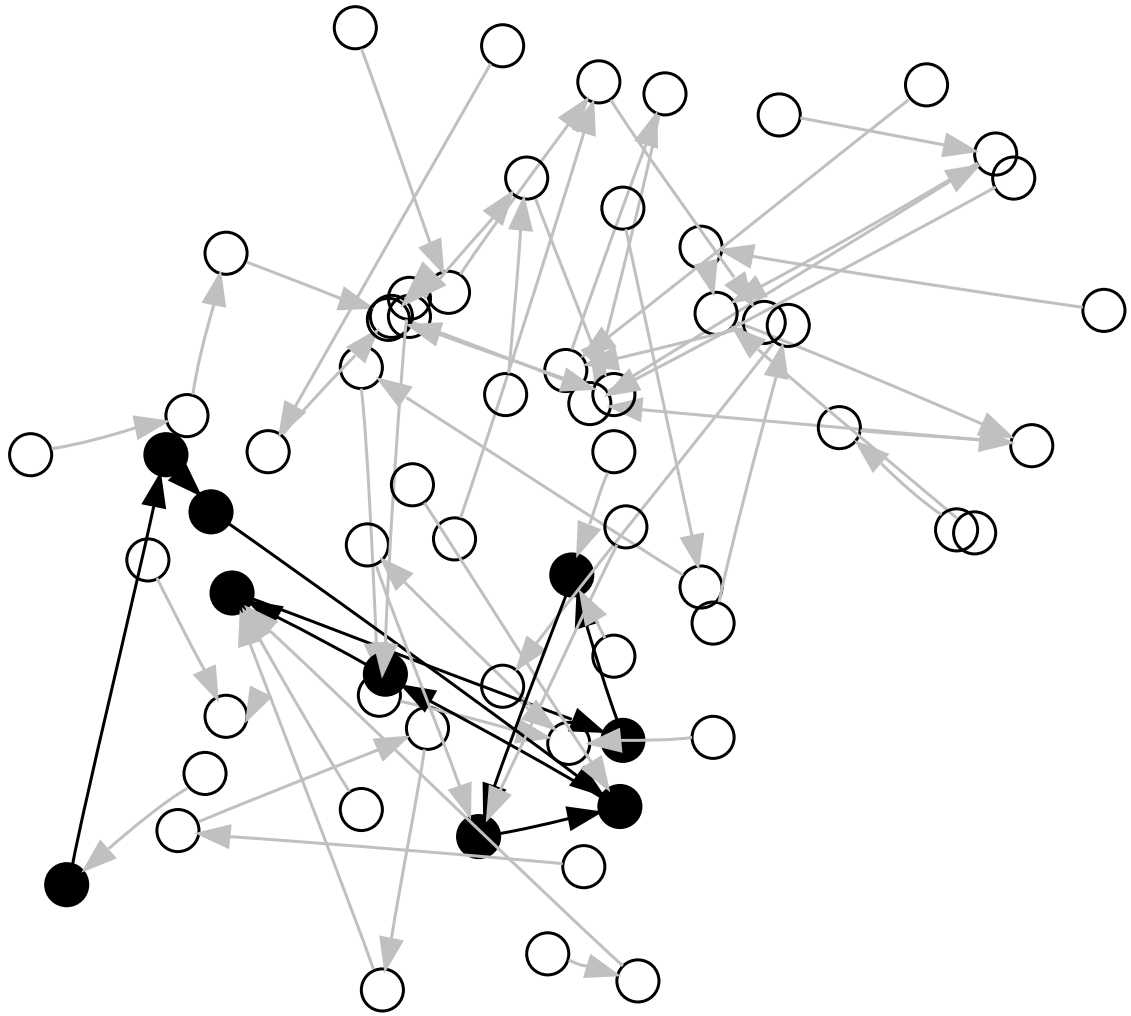
Pollard's rho method



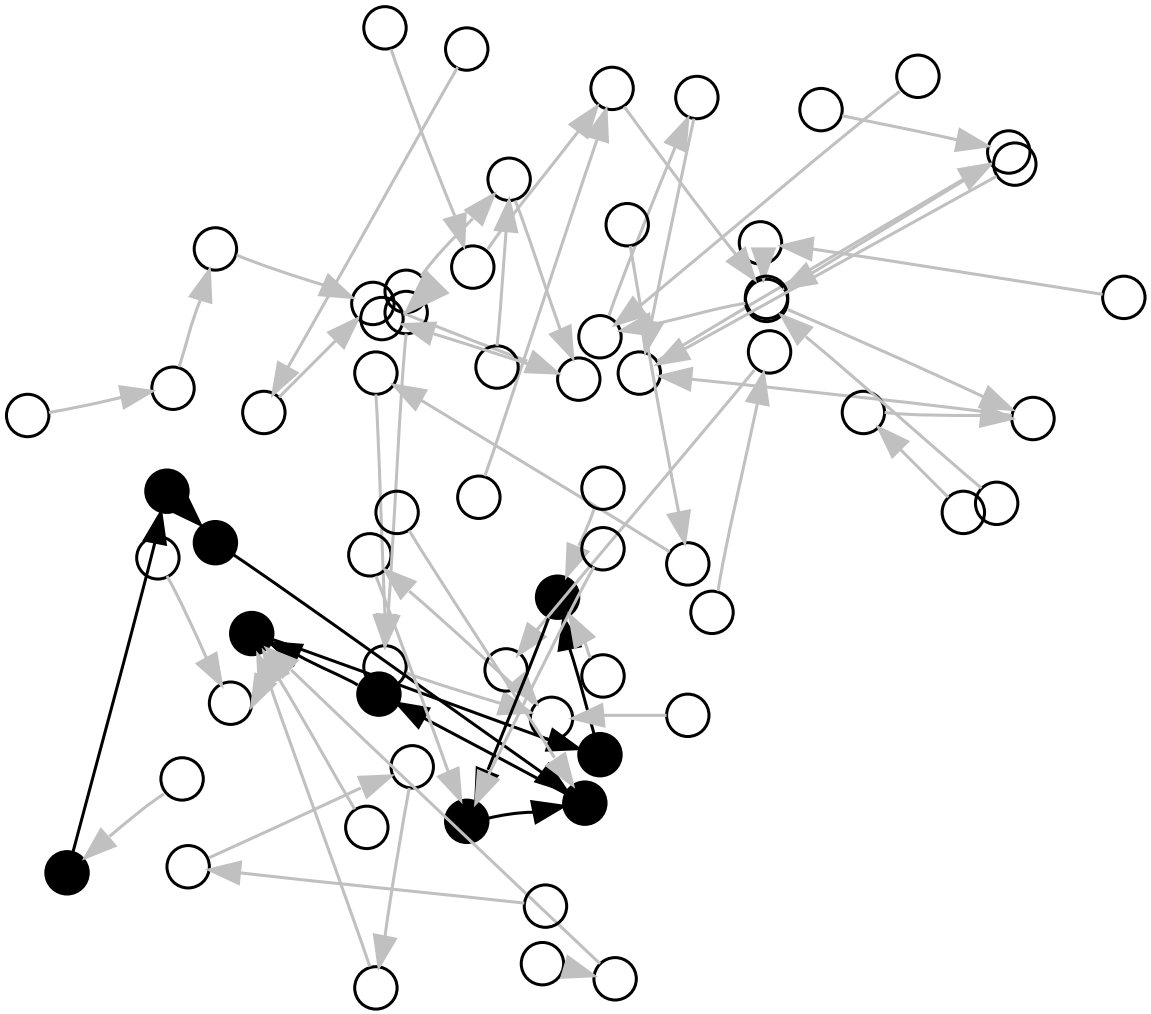
Pollard's rho method



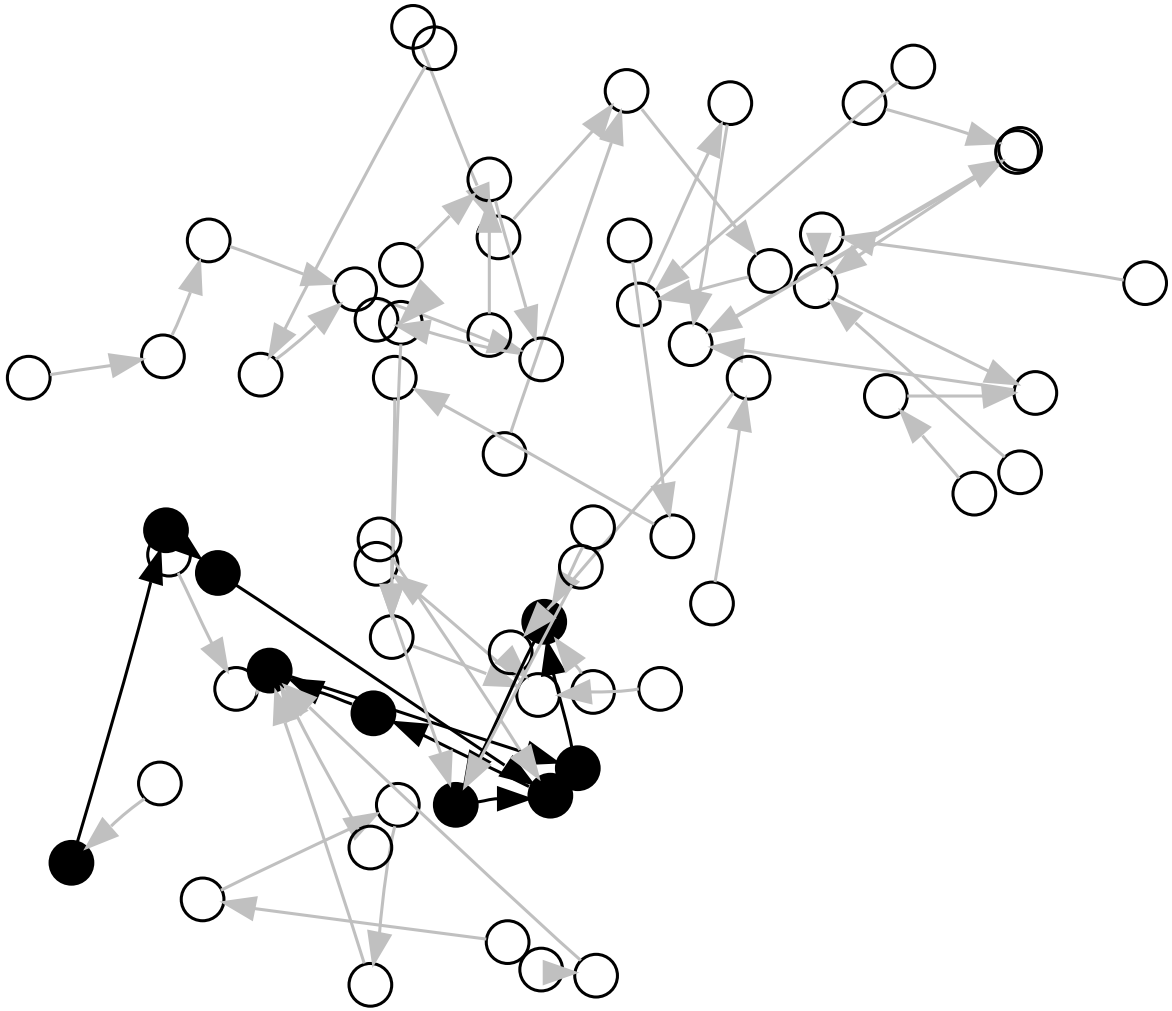
Pollard's rho method



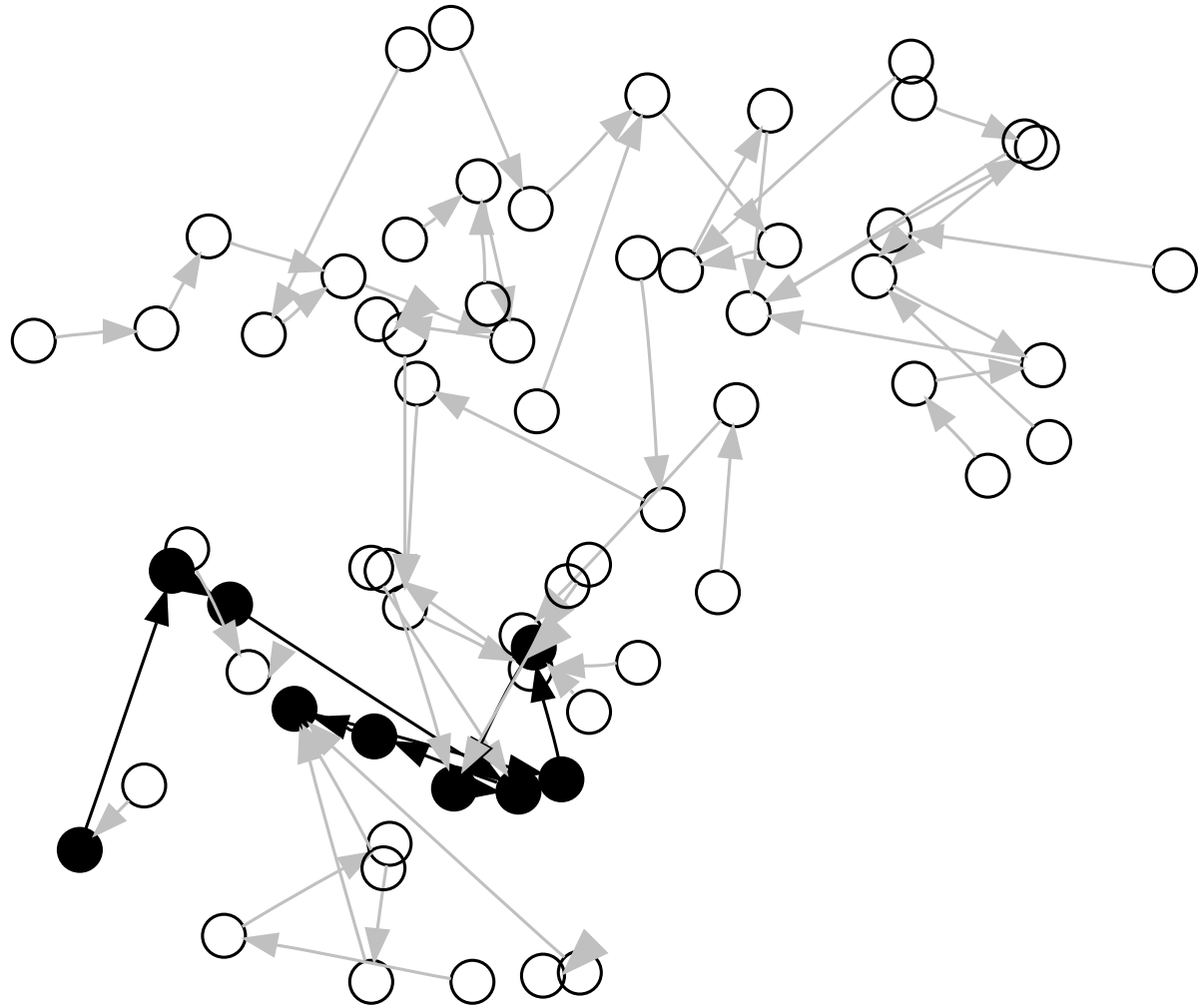
Pollard's rho method



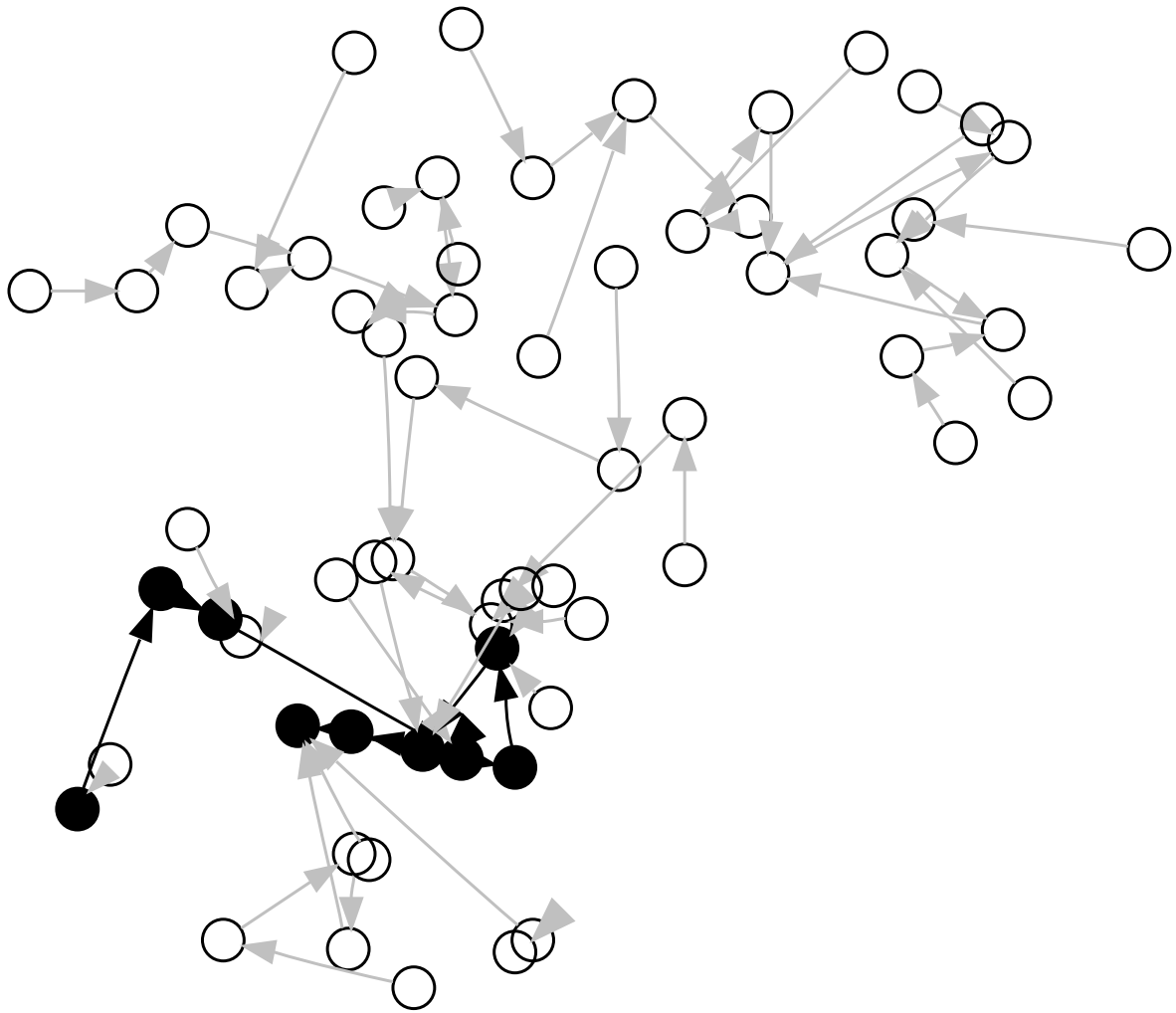
Pollard's rho method



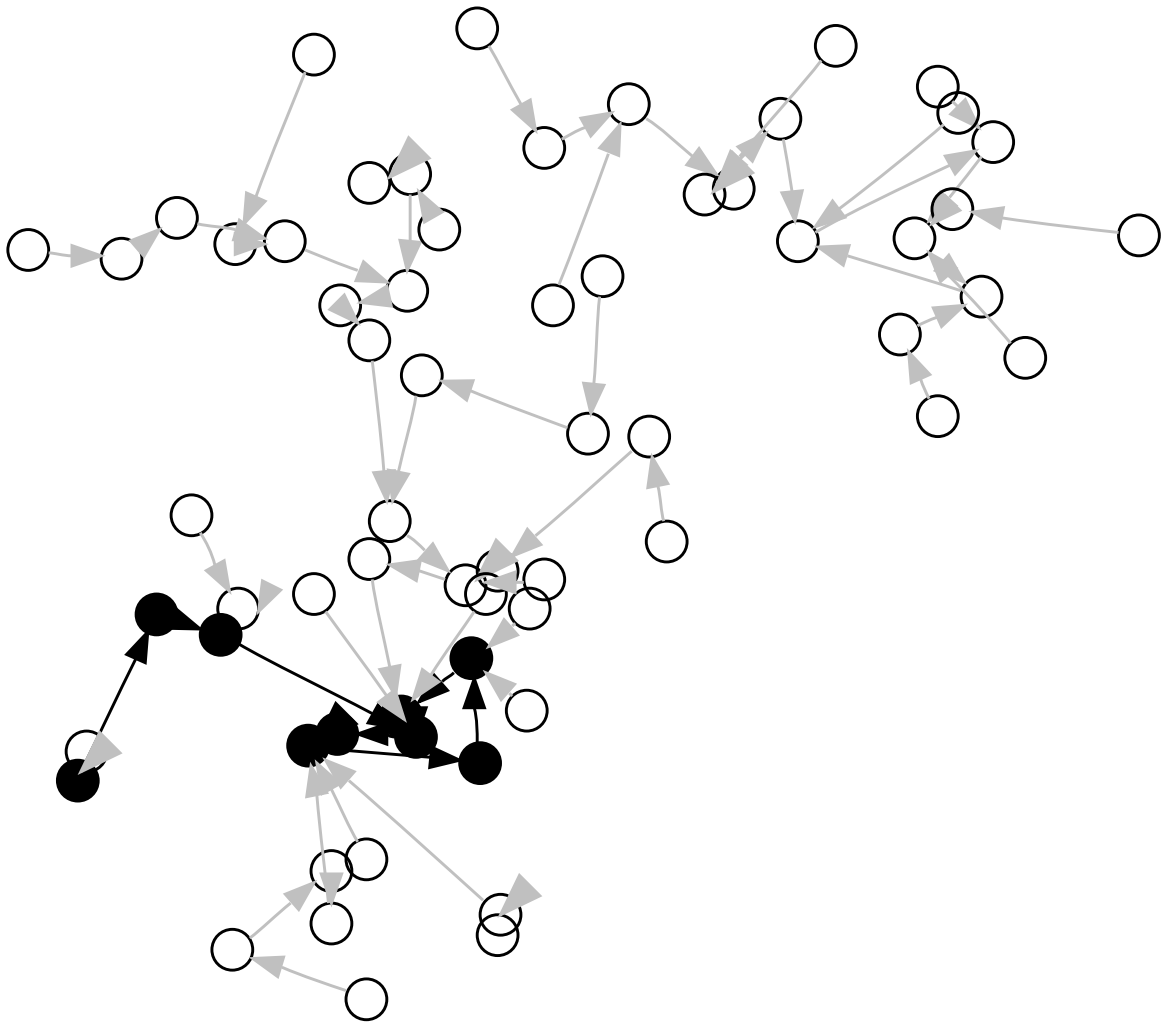
Pollard's rho method



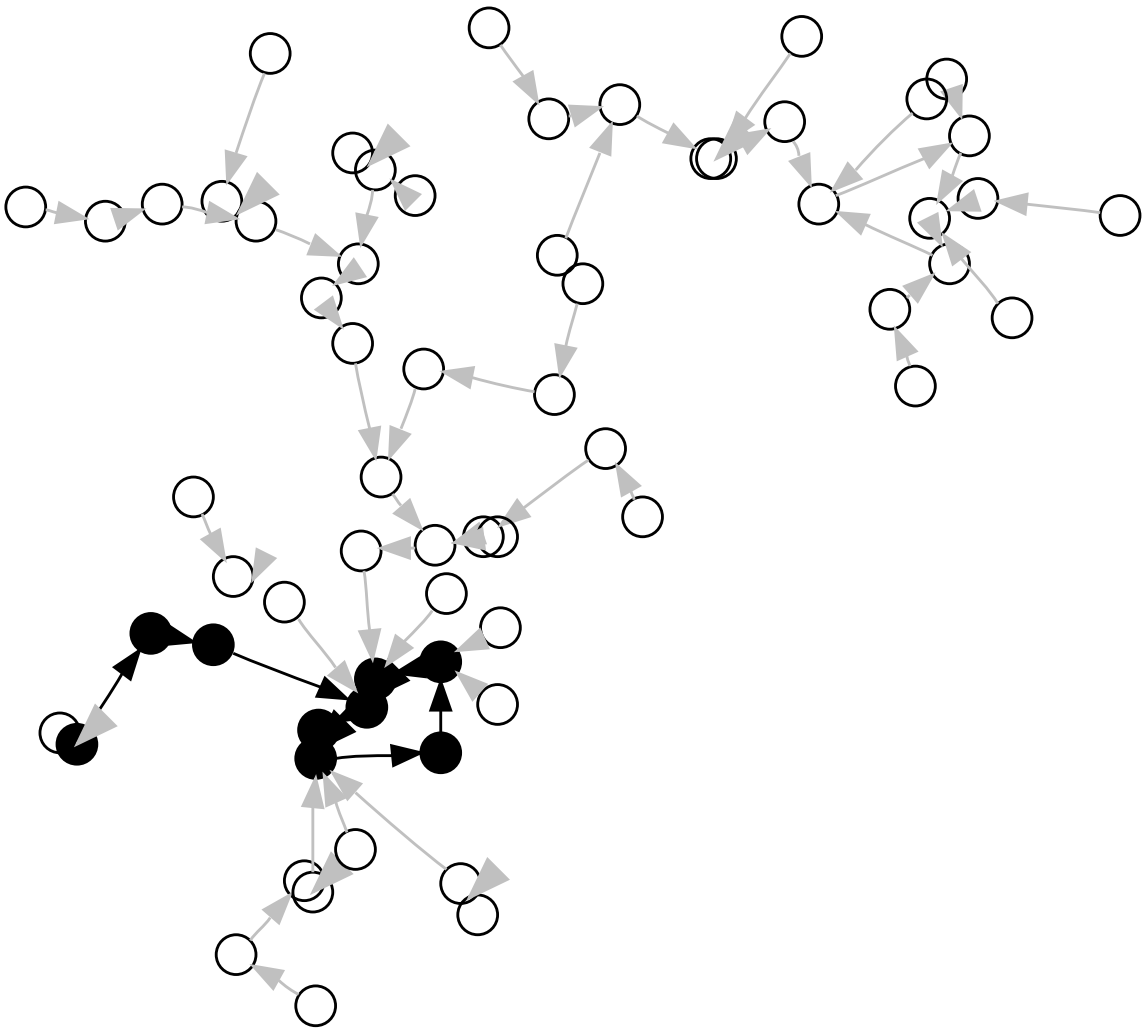
Pollard's rho method



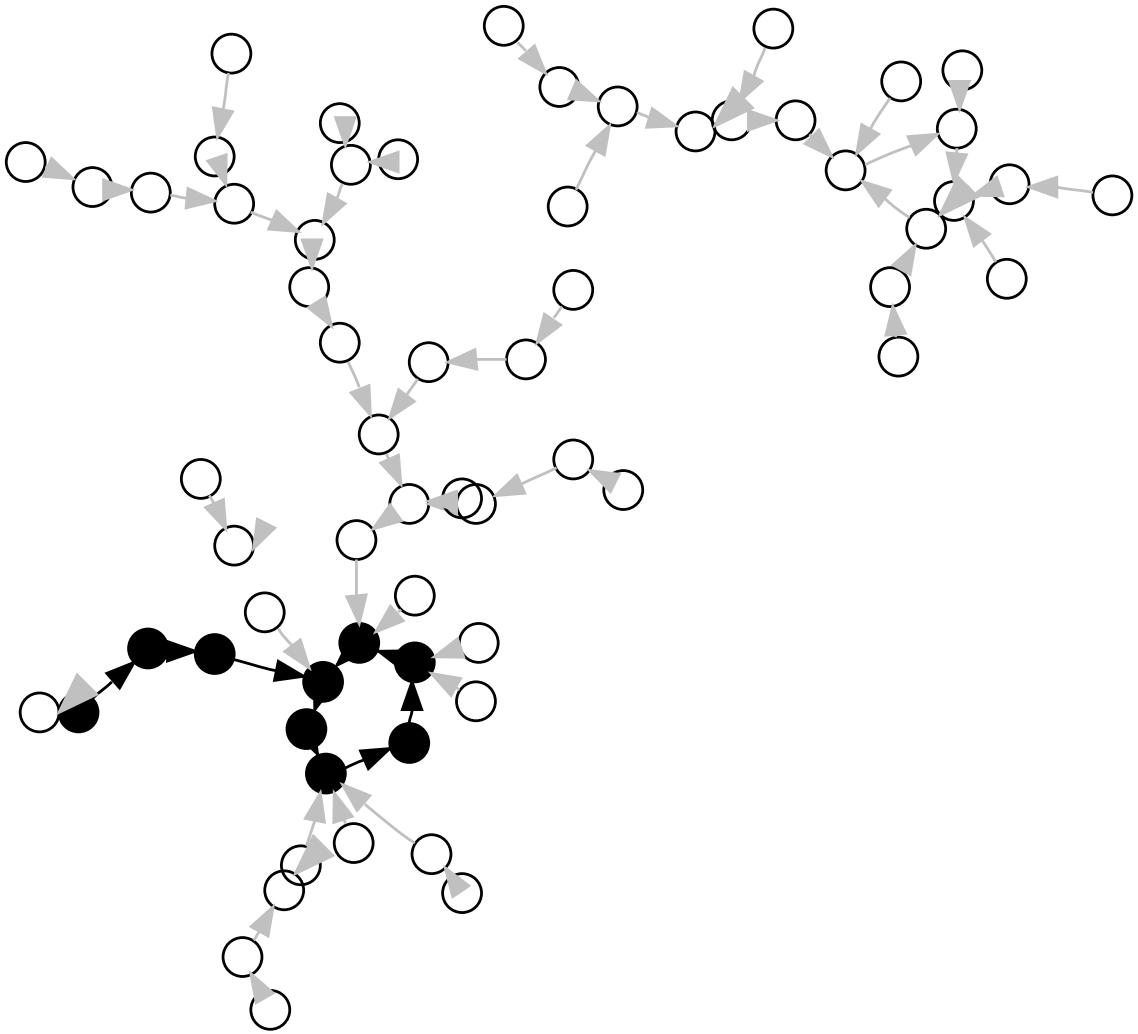
Pollard's rho method



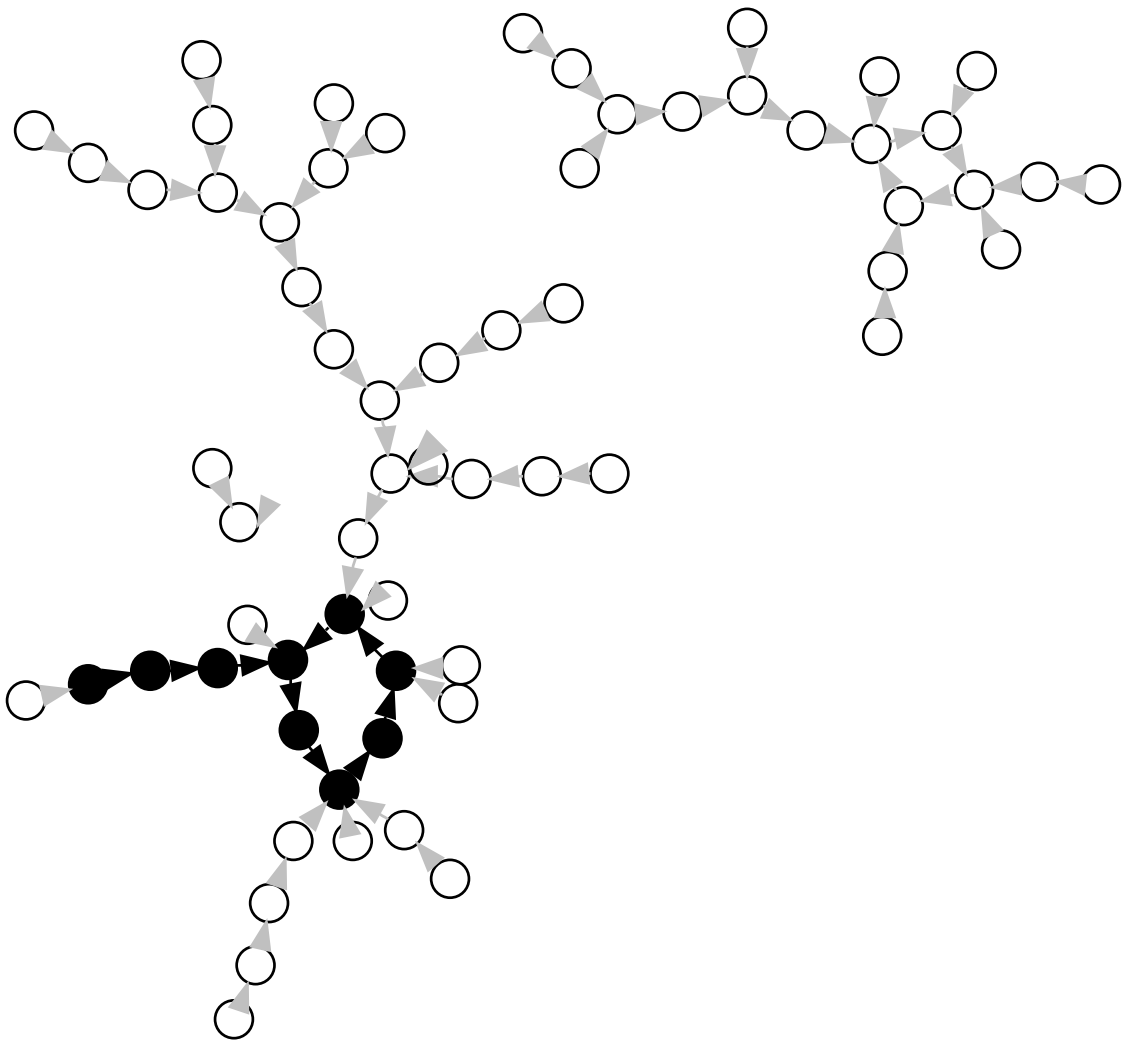
Pollard's rho method



Pollard's rho method



Pollard's rho method



Tail length:
$\sqrt{\pi\ell/8}$
Cycle length:
$\sqrt{\pi\ell/8}$

Pollard's rho method

The motivation is to remove the storage costs by turning solving the DLP into a random walk on elements of G.

This is a common strategy – see factorization, collision attacks for hash functions and now DLPs.

By the birthday paradox, after $\sqrt{\pi\ell/2}$ random draws from a set of ℓ elements, one element will be drawn twice with 50% probability.

Two problems need to be solved to use the rho method successfully:

1. Design the step function so that it randomly samples elements (so that the birthday paradox applies) while being deterministic (so we can use Floyd's cycle finding method to remove storage).
2. Design the step function so that a collision produces a meaningful result.

For hash functions the last part is obvious, but one needs to backtrack to the first collision.

Pollard rho for DLP: Attempt 1

Assume that we know for each step a way to write the result as $g^b h^c$.

If we find a match, then $g^{b_1} h^{c_1} = g^{b_2} h^{c_2}$ and we can sort these into

$$g^{b_2 - b_1} = h^{c_1 - c_2} = g^{a(c_1 - c_2)}.$$

Thus $a \equiv (b_2 - b_1)/(c_1 - c_2) \bmod \ell$ if $c_1 \not\equiv c_2 \bmod \ell$.

This means that collisions are meaningful (most of the time).

To make the step function deterministic for each group element, design a function that takes the representation of a group element and produces the exponents b and c.

If these exponents are chosen from a large interval then the walk resembles a random walk.

Pick a starting point in terms of g and h, use one double exponentiation for each step.

Pollard rho for DLP: Attempt 2

[Note: We have a solution, we're just haggling about the price.]

Multiplicative walks make each step cheaper, while keeping the rest of the features.
The idea is that having some fixed set of step directions suffices to make the walk look random.

Pick some small set of steps, e.g. 32 random $(r_i, t_i) \in [0, \ell - 1]^2$. and precompute $s_i = g^{r_i} g^{t_i}$ for these. This is a fixed (small) number of double exponentiations and a fixed (small) amount of storage.

Define the step function as $x \mapsto x \cdot s_{f(x)}$,
where

$$f : G \to 0, 1, \ldots, 31,$$

assigns one step to each group element

E.g. take $f(x)$ the last 5 bits of x
(using some fixed representation of G).

Rho for DLP with multiplicative walks

If the set of steps is large enough, this is close to random.

If there are k steps then the runtime increases by a factor of

$$1/\sqrt{1-1/k} \approx 1 + 1/(2k).$$

Problem: we don't know anymore how to write $x = g^b h^c$.

Rho for DLP with multiplicative walks

If the set of steps is large enough, this is close to random.

If there are k steps then the runtime increases by a factor of

$$1/\sqrt{1-1/k} \approx 1 + 1/(2k).$$

Problem: we don't know anymore how to write $x = g^b h^c$.

Solution: start from *known* starting point, e.g. $x = g$ or $x = h$ and keep track of the exponents.

Each step updates

$$x \leftarrow x \cdots_{f(x)}, \quad b \leftarrow b + r_{f(x)}, \quad c \leftarrow c + t_{f(x)}.$$

starting from $x = g, b = 1, c = 0$ or $x = h, b = 0, c = 1$.

Schoolbook method for Pollard rho

The schoolbook method uses a step function with only 3 types of steps.

This would be very bad for randomness with the $1 + 1/(2k)$ formula!

The method escapes that by using squaring as one of the steps.

$$x \leftarrow \begin{cases} x \cdot g \\ x \cdot h, \\ x^2 \end{cases} \quad b \leftarrow \begin{cases} b+1 \\ b, \\ 2b \end{cases} \quad c \leftarrow \begin{cases} c \\ c+1, \\ 2c \end{cases} \quad \text{for } f(x) = \begin{cases} 0 \\ 1. \\ 2 \end{cases}$$

Typically $f(x)$ takes an integer representation of x and outputs the remainder of division by 3.