Discrete logarithm problem III

Random walks and cycle finding

Tanja Lange

Eindhoven University of Technology

2MMC10 – Cryptology

Random walks

Target $Q = aP$ in $\langle P \rangle$. Group has n elements.

Make a pseudo-random walk in the group $\langle P \rangle$,
where the next step depends on current point: $W_{i+1} = f(W_i)$.

Birthday paradox:
Randomly choosing from n elements picks one element twice after about

Random walks

Target $Q = aP$ in $\langle P \rangle$. Group has n elements.

Make a pseudo-random walk in the group $\langle P \rangle$,
where the next step depends on current point: $W_{i+1} = f(W_i)$.

Birthday paradox:
Randomly choosing from n elements picks one element twice after about $\sqrt{\pi n/2}$ draws.

Random walks

Target $Q = aP$ in $\langle P \rangle$. Group has n elements.

Make a pseudo-random walk in the group $\langle P \rangle$,
where the next step depends on current point: $W_{i+1} = f(W_i)$.

Birthday paradox:
Randomly choosing from n elements picks one element twice after about $\sqrt{\pi n/2}$ draws.

Assume that for each point we know $0 \leq a_i, b_i < n$ so that $W_i = a_i P + b_i Q$.

Then $W_i = W_j$ means that

$$a_i P + b_i Q = a_j P + b_j Q$$

Random walks

Target $Q = aP$ in $\langle P \rangle$. Group has n elements.

Make a pseudo-random walk in the group $\langle P \rangle$,
where the next step depends on current point: $W_{i+1} = f(W_i)$.

Birthday paradox:
Randomly choosing from n elements picks one element twice after about $\sqrt{\pi n/2}$ draws.

Assume that for each point we know $0 \leq a_i, b_i < n$ so that $W_i = a_i P + b_i Q$.

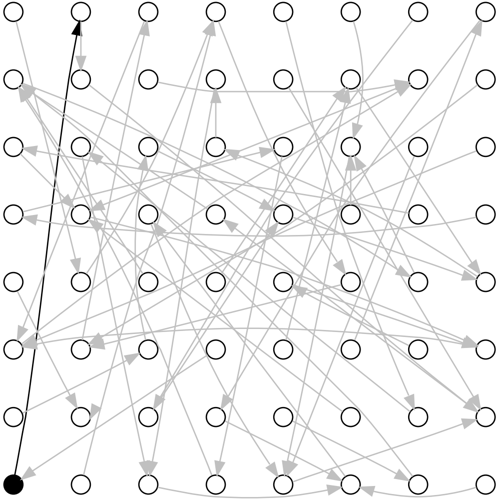
Then $W_i = W_j$ means that

$$a_i P + b_i Q = a_j P + b_j Q \Leftrightarrow (b_i - b_j)Q = (a_j - a_i)P.$$

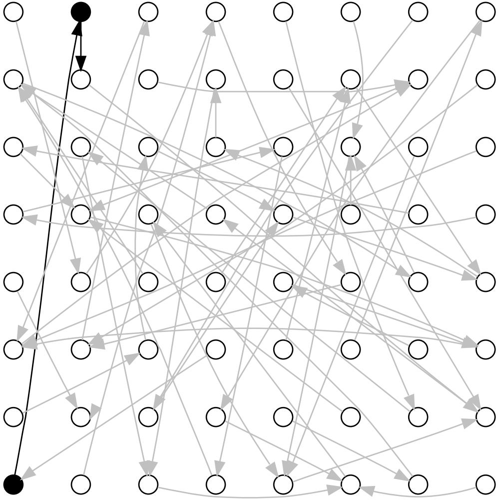
If $b_i - b_j$ invertible modulo n, the DLP is solved:

$$a \equiv (a_j - a_i)/(b_i - b_j) \bmod n$$

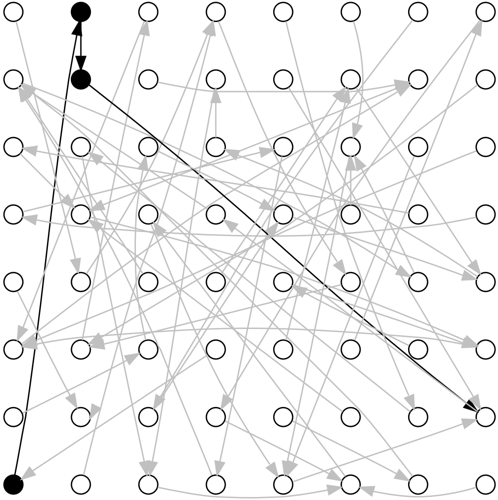
Random walks have collisions



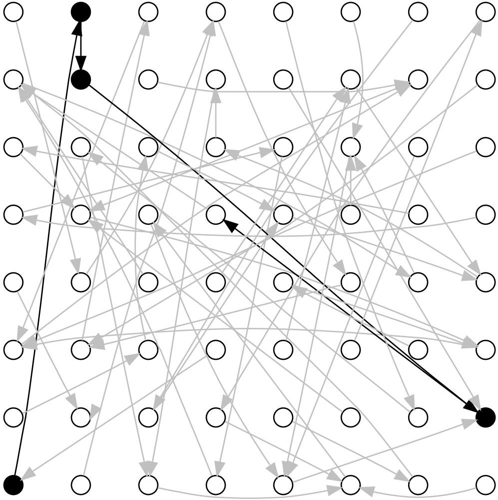
Random walks have collisions



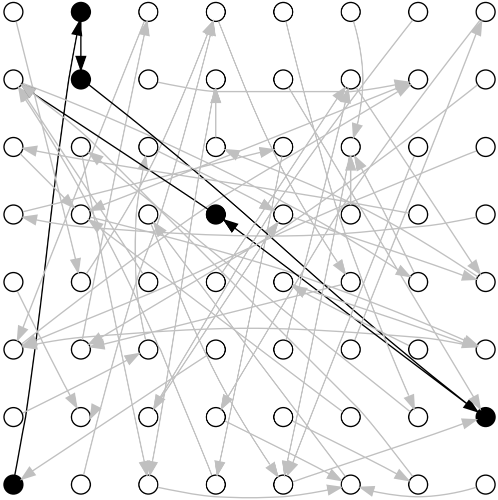
Random walks have collisions



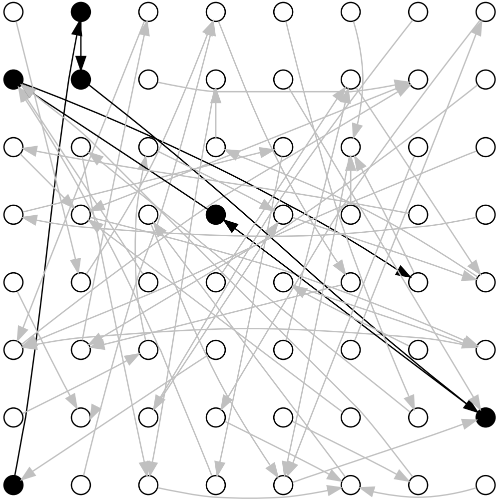
Random walks have collisions



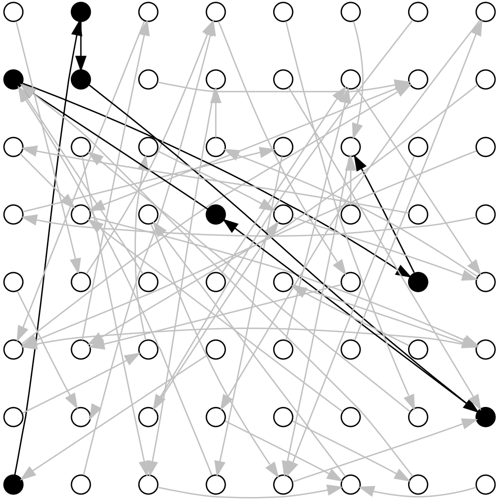
Random walks have collisions



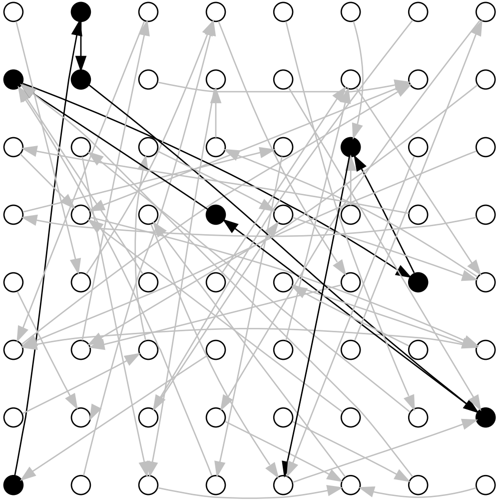
Random walks have collisions



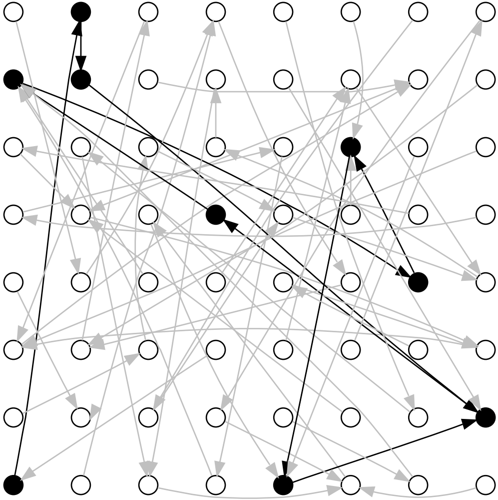
Random walks have collisions



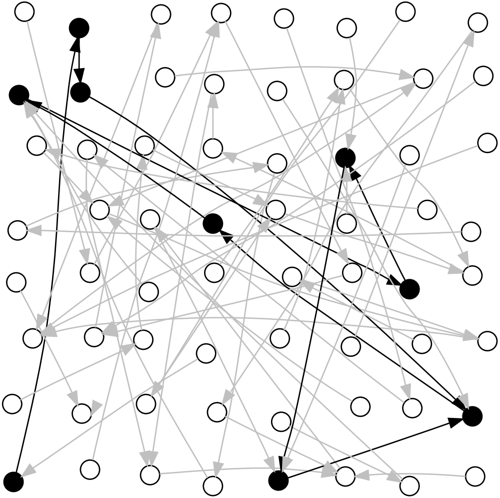
Random walks have collisions



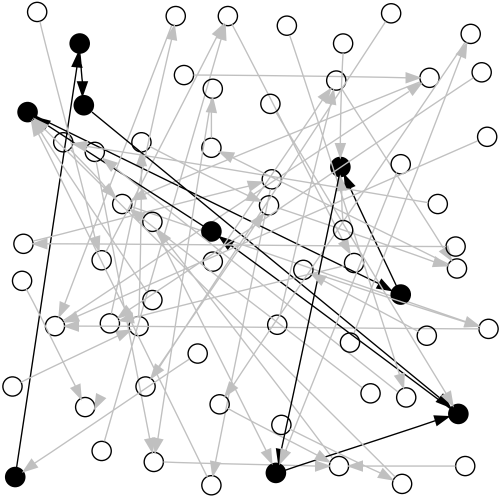
Random walks have collisions



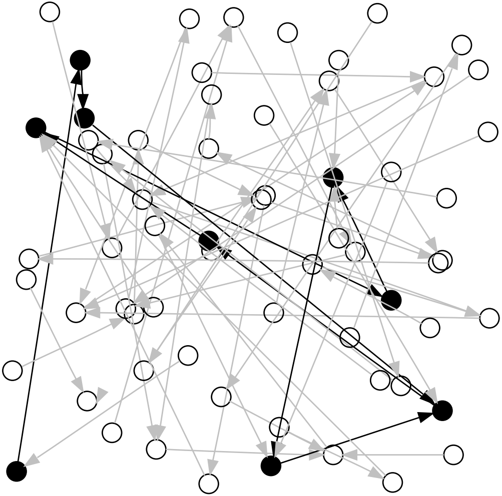
Random walks have collisions



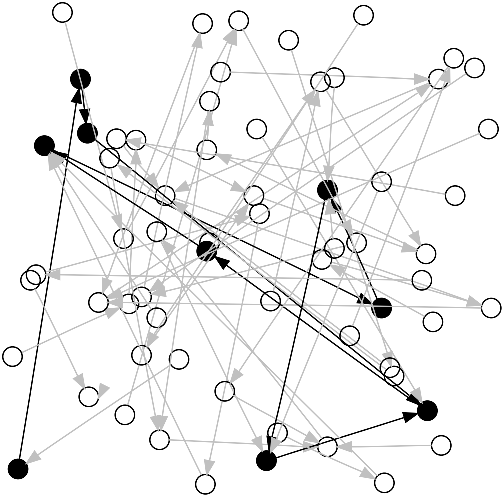
Random walks have collisions



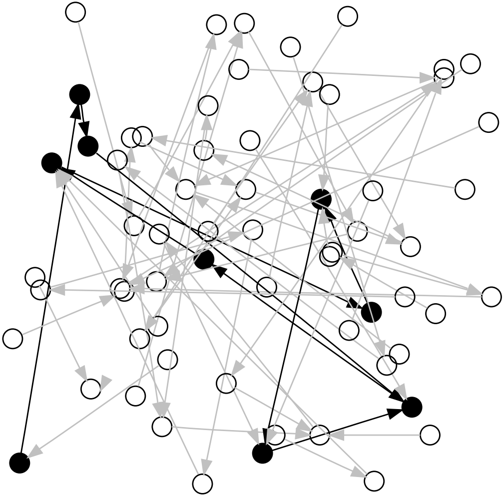
Random walks have collisions



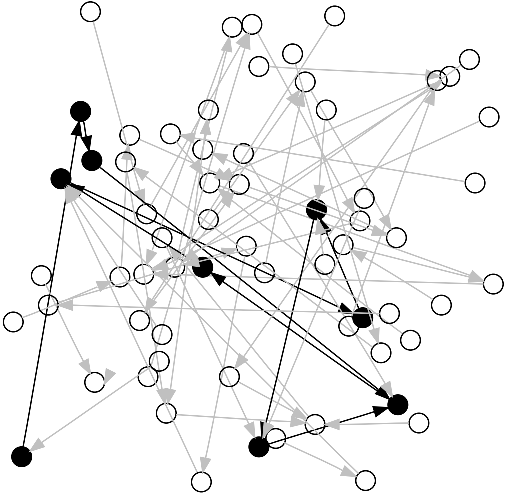
Random walks have collisions



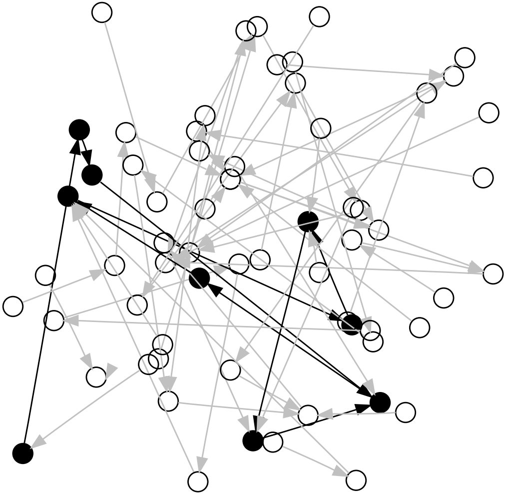
Random walks have collisions



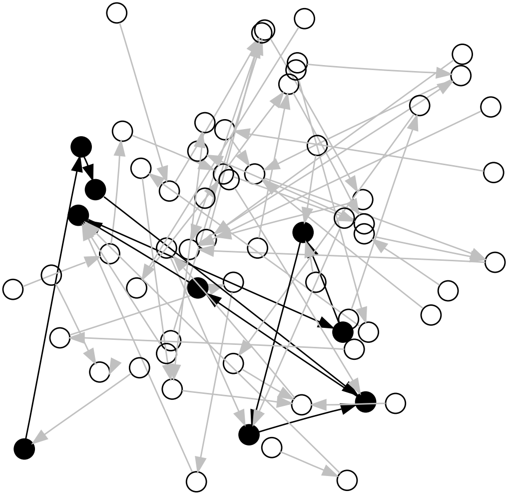
Random walks have collisions



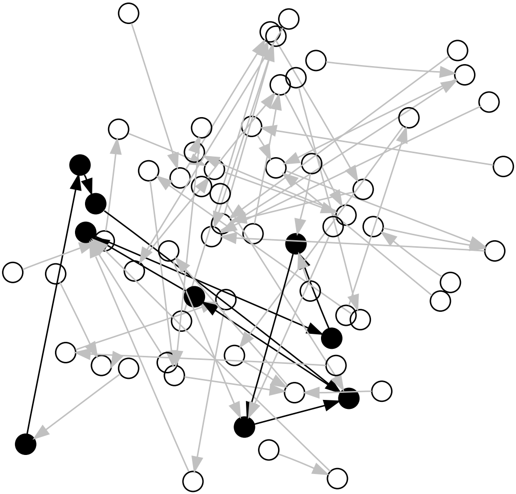
Random walks have collisions



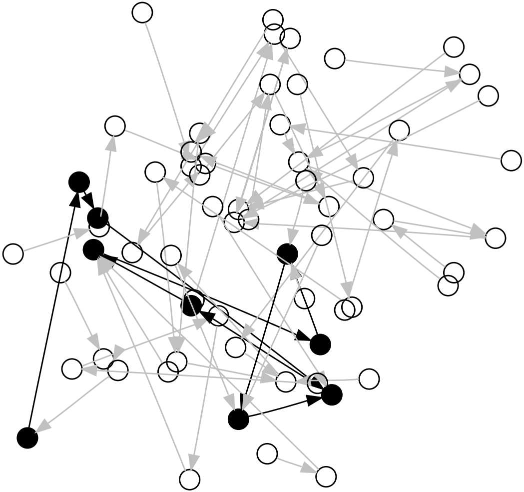
Random walks have collisions



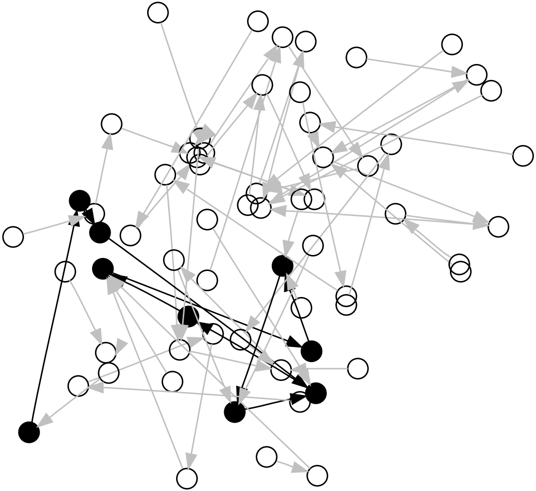
Random walks have collisions



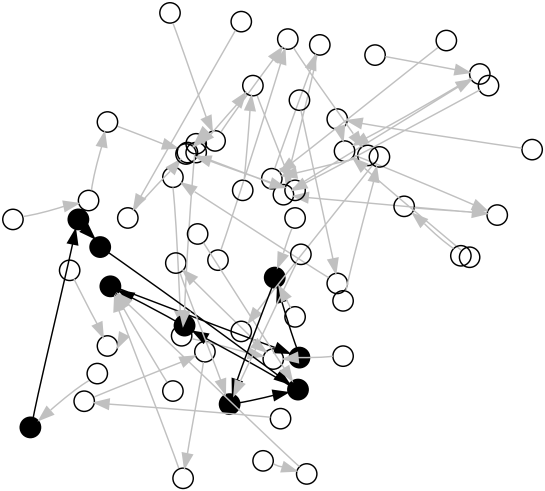
Random walks have collisions



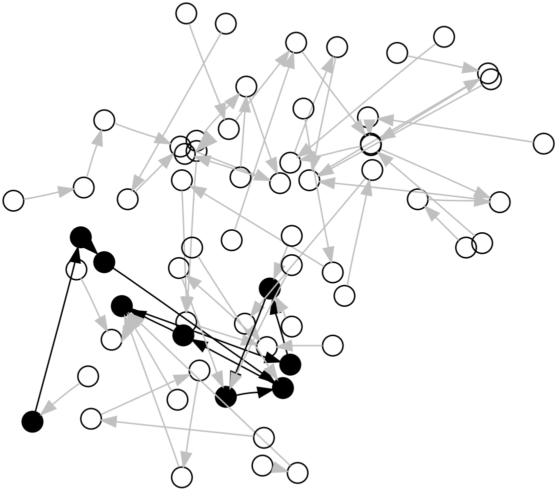
Random walks have collisions



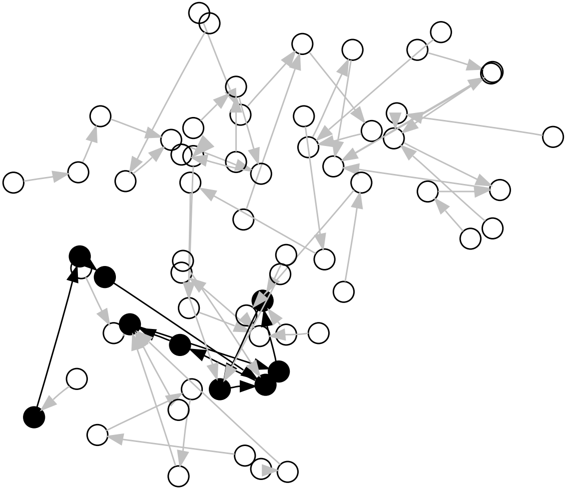
Random walks have collisions



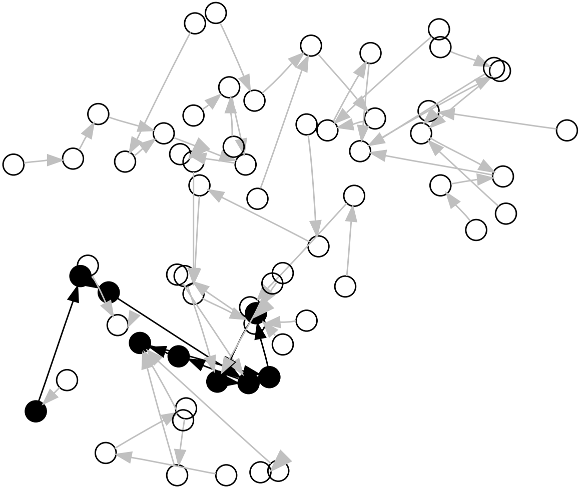
Random walks have collisions



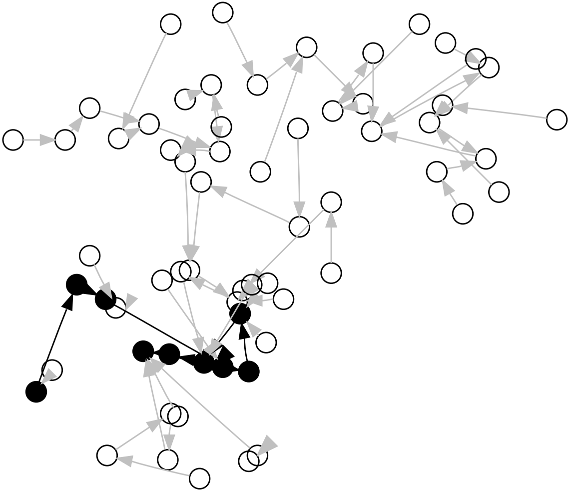
Random walks have collisions



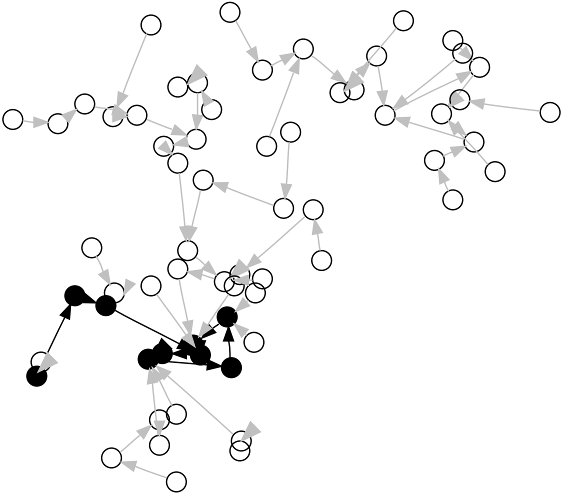
Random walks have collisions



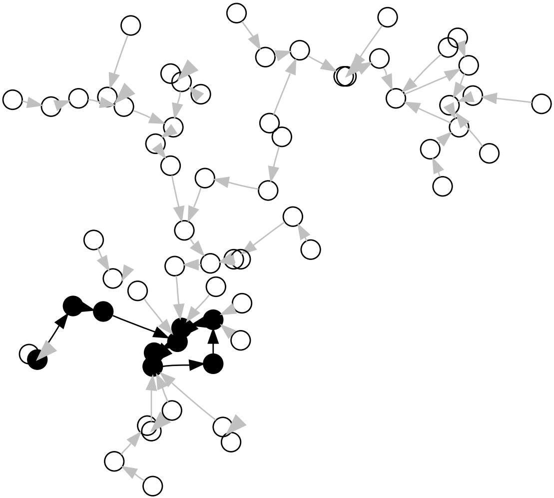
Random walks have collisions



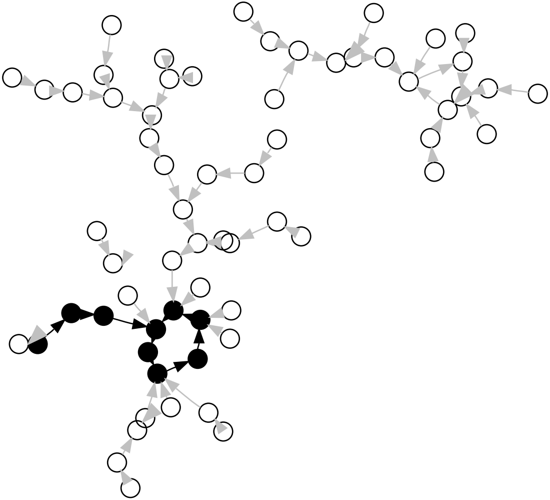
Random walks have collisions



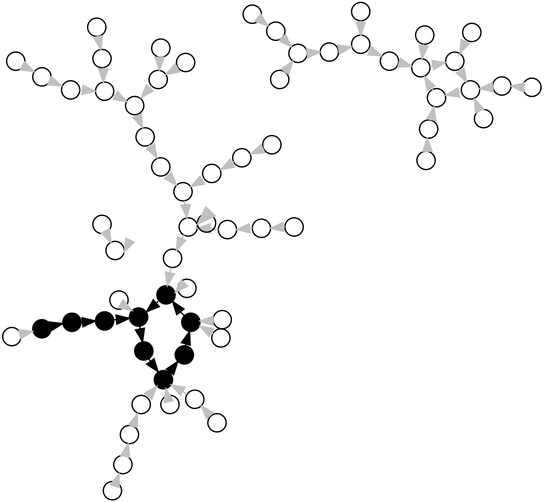
Random walks have collisions



Random walks have collisions



Random walks have collisions



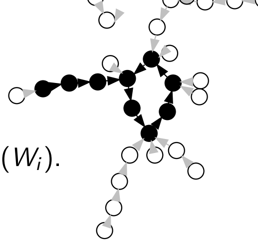
For random mappings:

Tail length: $\sqrt{\pi n/8}$

Cycle length: $\sqrt{\pi n/8}$

See Flajolet & Odlyzko URL.

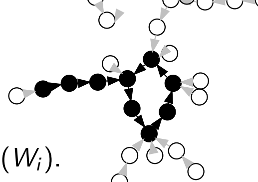
Eliminate the storage



Make a pseudo-random walk in the group $\langle P \rangle$,
where the next step depends on current point: $W_{i+1} = f(W_i)$.

After about $\sqrt{\pi n/2}$ steps we have a collision:
$W_i = W_j$.

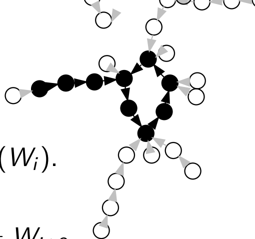
Eliminate the storage



Make a pseudo-random walk in the group $\langle P \rangle$,
where the next step depends on current point: $W_{i+1} = f(W_i)$.

After about $\sqrt{\pi n/2}$ steps we have a collision:
$W_i = W_j$. Then also $W_{i+1} = W_{j+1}, W_{i+2} = W_{j+2}, W_{i+3} = W_{j+3}, \ldots$.

Eliminate the storage

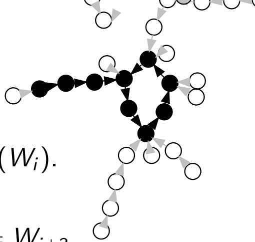


Make a pseudo-random walk in the group $\langle P \rangle$,
where the next step depends on current point: $W_{i+1} = f(W_i)$.

After about $\sqrt{\pi n/2}$ steps we have a collision:
$W_i = W_j$.Then also $W_{i+1} = W_{j+1}, W_{i+2} = W_{j+2}, W_{i+3} = W_{j+3}, \ldots$.

The walk now enters a cycle.
Cycle-finding algorithms (e.g., Floyd) quickly detects this without requiring storage.

Eliminate the storage



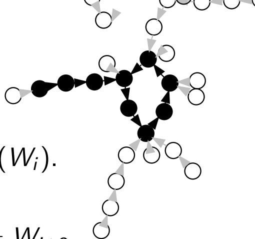
Make a pseudo-random walk in the group $\langle P \rangle$,
where the next step depends on current point: $W_{i+1} = f(W_i)$.

After about $\sqrt{\pi n/2}$ steps we have a collision:
$W_i = W_j$.Then also $W_{i+1} = W_{j+1}, W_{i+2} = W_{j+2}, W_{i+3} = W_{j+3}, \ldots$.

The walk now enters a cycle.
Cycle-finding algorithms (e.g., Floyd) quickly detects this without requiring storage.

Floyd runs two walks, a fast walk F_i and a slow walk S_i.
Only compares $F_i \stackrel{?}{=} S_i$, so does not need any older values.

Eliminate the storage



Make a pseudo-random walk in the group $\langle P \rangle$,
where the next step depends on current point: $W_{i+1} = f(W_i)$.

After about $\sqrt{\pi n/2}$ steps we have a collision:
$W_i = W_j$. Then also $W_{i+1} = W_{j+1}, W_{i+2} = W_{j+2}, W_{i+3} = W_{j+3}, \dots$.

The walk now enters a cycle.
Cycle-finding algorithms (e.g., Floyd) quickly detects this without requiring storage.

Floyd runs two walks, a fast walk F_i and a slow walk S_i.
Only compares $F_i \stackrel{?}{=} S_i$, so does not need any older values.

Start at $S_0 = F_0 = W_0$.
Each step does $S_{i+1} = f(S_i)$ and $F_{i+1} = f(f(F_i))$.
Once both walks enter the cycle, they will collide.

Cycle length: roughly $\sqrt{\pi n/8}$.
Fast walk might need to loop a few times.