Speeding Up PPR Computations Via Graph Compression

Graph Compression

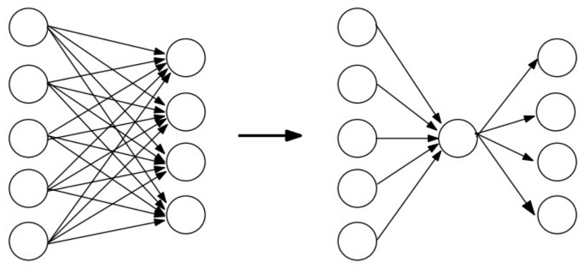


Figure 1. The bipartite clique–star transformation.

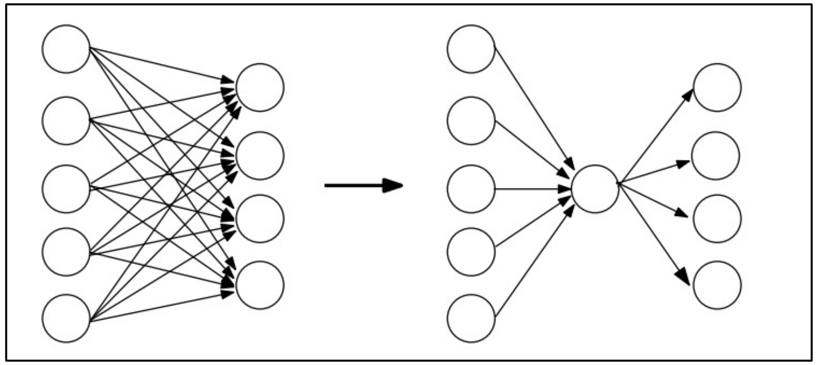
Modified Compressed-Graph Algorithms

PageRank $\vec{r} = cP\vec{r} + \frac{1}{N}(1-c)\vec{1}$

PPR $\vec{r} = cP\vec{r} + (1-c)\vec{s}$

Graph Compression

Transformation



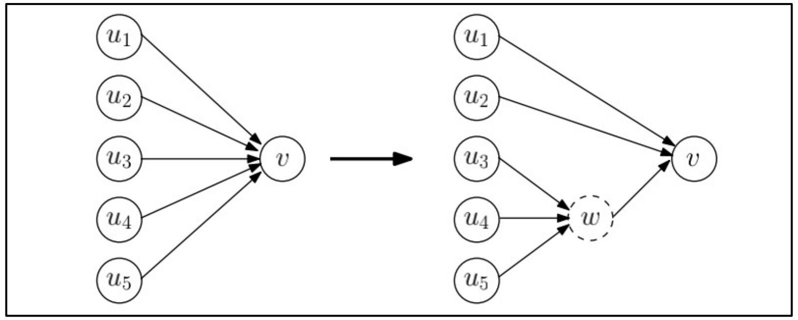
Add virtual vertex to **reduce edges**

Aiming at dense subgraphs, i.e. the **bipartite subgraph**.

Accelerate Adjacency Matrix Multiplication

$$\mathbf{y} = E^T\mathbf{x} \qquad \mathbf{y}[v] = \sum_{uv \in E} \mathbf{x}[u].$$

Considering a virtual node w:



$$\begin{aligned}\mathbf{y}[v] &= \mathbf{x}[u_1] + \mathbf{x}[u_2] + \mathbf{x}[u_3] + \mathbf{x}[u_4] + \mathbf{x}[u_5] \\ &= \mathbf{x}[u_1] + \mathbf{x}[u_2] + \mathbf{x}[w].\end{aligned}$$

Graph Compression

Accelerate Adjacency Matrix Multiplication

Algorithm 2. (Compressed-multiply(E, $\mathbf{x}$))

```
forall Real nodes v do
    y[v] = 0;
end
forall Virtual nodes v do
    x[v] = 0;
end
forall Nodes u ∈ V' do
    forall Edges uv ∈ E' do
        if v is real then
            y[v] = y[v] + x[u];
        else
            x[v] = x[v] + x[u];
        end
    end
end
```

For virtual node v:

$$\mathbf{x}[v] = \sum_{uv \in E'} \mathbf{x}[u].$$

Then, sum up for y results.

$$\mathbf{y}[v] = \sum_{uv \in E'} \mathbf{x}[u].$$

$$O(|E'| + |V'|)$$

PageRank and PPR

PageRank $\vec{r} = cP\vec{r} + \frac{1}{N}(1-c)\vec{1}$

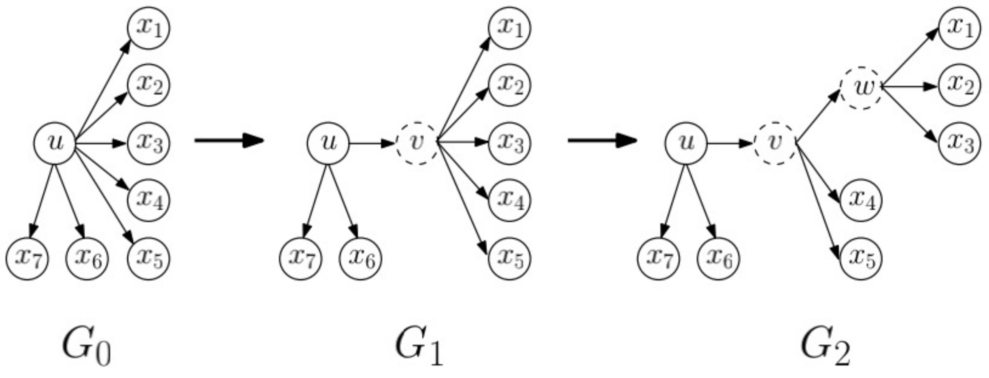
	Uncompressed	Black-Box	Markov Chain
Time/Iteration (sec)	263.52	60.80	60.06
No. of Iterations	21	21	53
Speedup	1	4.33	1.74

$\downarrow$

PPR $\vec{r} = cP\vec{r} + (1-c)\vec{s}$

Directly run on a compressed graph?

(Personalized) PageRank On Compressed-Graph



- Random walk on G’ is not uniform.
- Jump vector has zeros on virtual nodes
- Transitions made from virtual nodes have zero jump p.

This ensures that the Markov chain on ***G’*** models exactly **the uniform random walk on *G***

$$\Delta_G(u) = \begin{cases} 1 & \text{if } u \text{ is real,} \\ \sum_{w \in \delta_{\text{out}}^G(u)} \Delta_G(w) & \text{if } u \text{ is virtual.} \end{cases}$$

$$\Gamma_G(u) = \sum_{w \in \delta_{\text{out}}^G(u)} \Delta_G(w). \quad X[u,v] = \frac{\Delta_{G'}(v)}{\Gamma_{G'}(u)}.$$

$$Y[u,v] = \begin{cases} (1-\alpha)X[u,v] & \text{if } u \text{ is real,} \\ X[u,v] & \text{if } u \text{ is virtual.} \end{cases}$$

$Z = (Y + \alpha J'^T)$

Compute P, the steady state represented by Z

P': Discard probability for virtual nodes in P.

P'': the L_1 norm scale from P'

P'' is the steady state of original uniform random walk on G.

FORA On Compressed-Graph？

Algorithm 1: Forward Push

Input: Graph G, source node s, probability α, residue threshold r_{max}

Output: $\pi^{\circ}(s, v)$, $r(s, v)$ for all $v \in V$

$r(s, s) \leftarrow 1$; $r(s, v) \leftarrow 0$ for all $v \neq s$;

$\pi^{\circ}(s, v) \leftarrow 0$ for all v;

while $\exists v \in V$ *such that* $r(s, v)/|N^{out}(v)| > r_{max}$ **do**

  for *each* $u \in N^{out}(v)$ **do**

    $r(s, u) \leftarrow r(s, u) + (1 - \alpha) \cdot \frac{r(s,v)}{|N^{out}(v)|}$

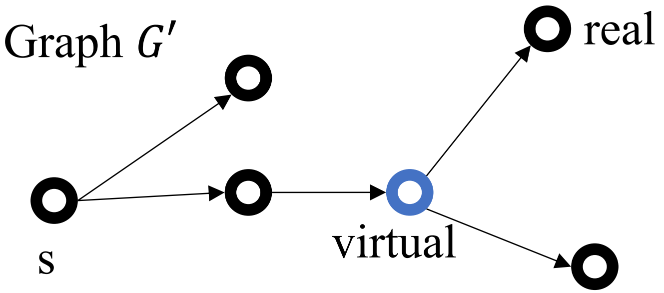
  $\pi^{\circ}(s, v) \leftarrow \pi^{\circ}(s, v) + \alpha \cdot r(s, v)$;

  $r(s, v) \leftarrow 0$;

$$\boxed{\pi(s,t) = \pi^{\circ}(s,t) + \sum_{v \in V} r(s,v) \cdot \pi(v,t)}$$

Using MC to estimate $\pi(v,t)$

Compressed Graph G'



Modified Forward Push

While $\exists v \in V$ such that $\frac{r(s,v)}{\tau(v)} > r_{max}$ **do**

  for each $u \in N^{out}(v)$ **do**

    If v is real **do**

      $r(s,u) \mathrel{+}= \frac{(1-\alpha)\Delta(u)}{\tau(v)} r(s,v)$

      $\pi^{o}(s,v) \mathrel{+}= \alpha \cdot r(s,v)$

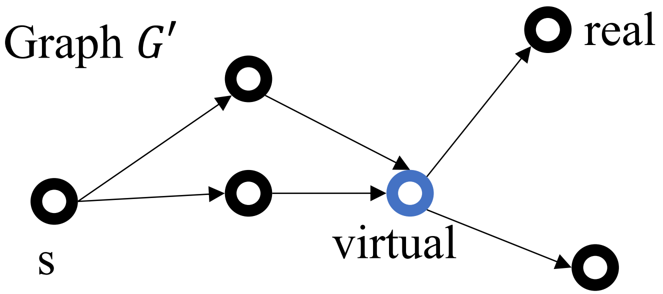
    If v is virtual **do**

      $r(s,u) \mathrel{+}= \frac{\Delta(u)}{\tau(v)} r(s,v)$

  $r(s,v) \leftarrow 0$

FORA On Compressed-Graph

Compressed Graph G'



Modified Forward Push

While $\exists v \in V$ such that $\frac{\mathrm{r}(s,v)}{\tau(v)} > r_{max}$ **do**

for each $u \in N^{out}(v)$ **do**

If v is real **do**

$$\mathrm{r}(s,u) \mathrel{+}= \frac{(1-\alpha)\Delta(u)}{\tau(v)} \mathrm{r}(s,v)$$

$$\pi^{o}(s,v) \mathrel{+}= \alpha \cdot \mathrm{r}(s,v)$$

If v is virtual **do**

$$\mathrm{r}(s,u) \mathrel{+}= \frac{\Delta(u)}{\tau(v)} \mathrm{r}(s,v)$$

$$\mathrm{r}(s,v) \leftarrow 0$$

?

$$\boxed{\pi_{G'}(s,t) = \pi^{o}_{G'}(s,t) + \textstyle\sum_{v \in G'} r(s,v) \cdot \pi_{G'}(v,t)}$$

Using MC to estimate $\pi_{G'}(v,t)$.

Monte-Carlo Random Walk On Compressed Graph

Consider one step, $a(u) = \begin{cases} 1-\alpha, & if\ u\ is\ real \\ 1, & if\ u\ is\ virtual \end{cases}$

Any $v \in N_{out}(u)$,

$$p(u \to v) = a(u) \cdot \frac{\Delta(v)}{\tau(u)}$$

$$p(u \to u) = 1 - a(u)$$

Proof it's an unbiased estimation.

Vital PPR Components

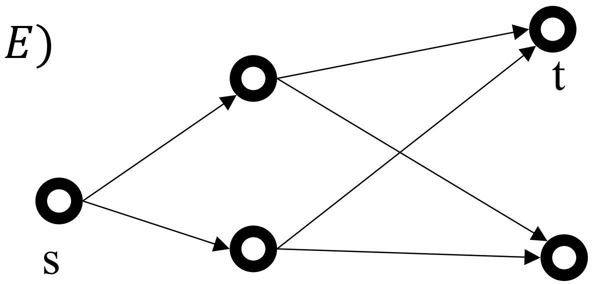
TopPPR

- Forward Push
- Backward Push
- MC simulations

SpeedPPR

- Forward Push
- Power Iteration
- MC simulations

Graph $G\ (V, E)$



α-random walk:

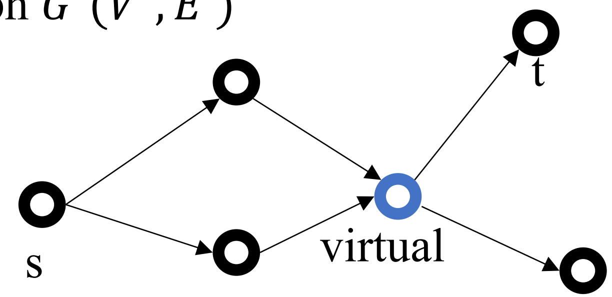
Any $v \in N_{out}(u)$,

$$p(u \rightarrow v) = \frac{1-\alpha}{|N_{out}(u)|}$$

$$p(u \rightarrow u) = \alpha$$

$\pi\ (s, t)$: Probability of **α-random walk** starts from s, ends at t.

Compressed Graph $G'(V', E')$



Virtual α-random walk:

Consider one step, $a(u) = \begin{cases} 1-\alpha, & if\ u\ is\ real \\ 1, & if\ u\ is\ virtual \end{cases}$

Any $v \in N_{out}(u)$,

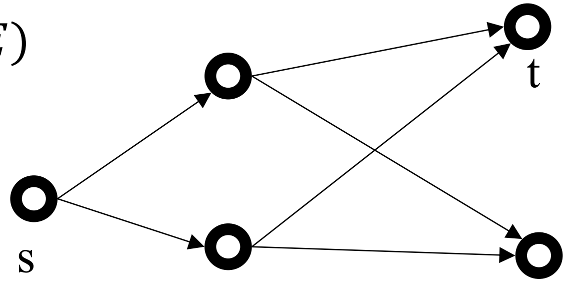
$$p(u \rightarrow v) = a(u) \cdot \frac{\Delta(v)}{\tau(u)}$$

$$p(u \rightarrow u) = 1 - a(u)$$

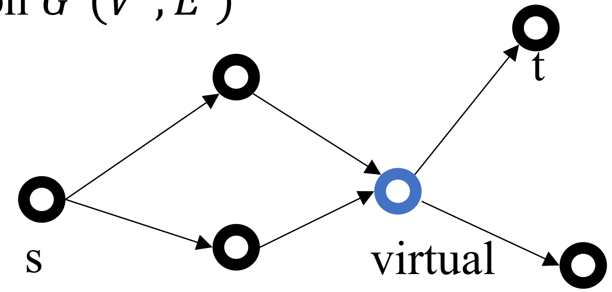
$\pi'\ (s, t)$: Probability of **Virtual α-random walk** starts from s, ends at t.

- $\boldsymbol{\pi'\ (s, t) = \pi\ (s, t)}$ **holds** and MC simulations is **unbiased.**

Graph $G\ (V, E)$



Compressed Graph $G'(V', E')$



Algorithm 1: Forward Push

Input: Graph G, source node s, probability α, residue threshold r_{max}

Output: $\pi^{\circ}(s, v)$, $r(s, v)$ for all $v \in V$

$r(s, s) \leftarrow 1$; $r(s, v) \leftarrow 0$ for all $v \neq s$;

$\pi^{\circ}(s, v) \leftarrow 0$ for all v;

while $\exists v \in V$ *such that* $r(s, v)/|N^{out}(v)| > r_{max}$ **do**

  for *each* $u \in N^{out}(v)$ **do**

    $r(s, u) \leftarrow r(s, u) + (1 - \alpha) \cdot \frac{r(s,v)}{|N^{out}(v)|}$

  $\pi^{\circ}(s, v) \leftarrow \pi^{\circ}(s, v) + \alpha \cdot r(s, v)$;

  $r(s, v) \leftarrow 0$;

$$\boxed{\pi(s, t) = \pi^{\circ}(s, t) + \sum_{v \in V} r(s, v) \cdot \pi(v, t)}$$

Virtual Forward Push

While $\exists v \in V$ such that $\frac{r'(s,v)}{\tau(v)} > r_{max}$ **do**

  for each $u \in N^{out}(v)$ **do**

    If v is real **do**

      $r'(s, u) \mathrel{+}= \frac{(1-\alpha)\Delta(u)}{\tau(v)} r'(s, v)$

      $\pi^{o\prime}(s, v) \mathrel{+}= \alpha \cdot r'(s, v)$ once

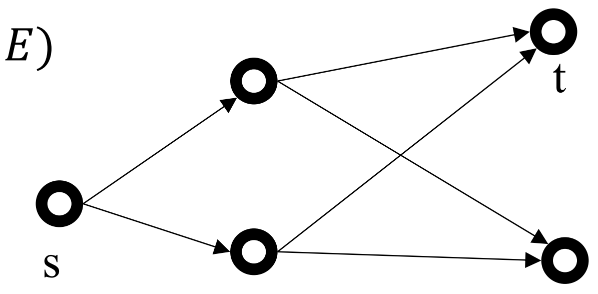
    If v is virtual **do**

      $r'(s, u) \mathrel{+}= \frac{\Delta(u)}{\tau(v)} r'(s, v)$

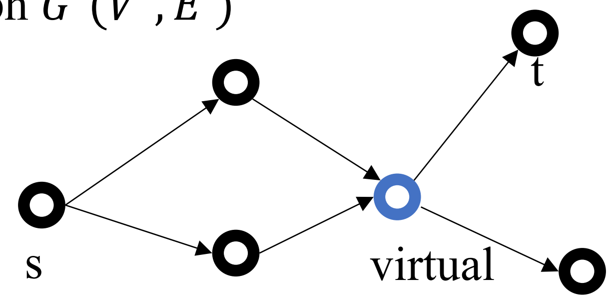
  $r'(s, v) \leftarrow 0$

$$\boxed{\pi'(s, t) = \pi^{o\prime}(s, t) + \sum_{v \in V'} r'(s, v) \cdot \pi'(v, t)}$$

Graph $G\ (V, E)$



Compressed Graph $G'(V', E')$



Algorithm 2: Backward Search

Input: Graph G, target node t, decay factor α, threshold r^b_{max}
Output: Backward residue $r^f(s, v)$ and reserve $\pi^f(s, v)$ for all $v \in V$

$r^b(t, t) \leftarrow 1$ and $r^b(v, t) \leftarrow 0$ for all $v \neq t$ and $i = 1, \ldots, j$;
$\pi^b(v, t) \leftarrow 0$ for all v;
while $\exists v$ *such that* $r^b(v, t) > r^b_{max}$ **do**
  for *each* $u \in \mathcal{N}^{in}(v)$ **do**
    $r^b(u, t) \leftarrow r^b(u, t) + (1 - \alpha) \cdot \frac{r^b(v,t)}{d_{out}(u)}$
  $\pi^b(v, t) \leftarrow \pi^b(v, t) + \alpha \cdot r^b(v, t)$;
  $r^b(s, v) \leftarrow 0$;

Virtual Backward Push

While $\exists v \in V$ such that $\mathrm{r}'(\mathrm{v}, \mathrm{t}) > r_{max}$ **do**
  for each $u \in N^{in}(v)$ **do**
    If v is real **do**
      $\mathrm{r}'(u, t) \mathrel{+}= \frac{(1-\alpha)\Delta(v)}{\tau(u)} \mathrm{r}(v, t)$
      $\pi^{o\prime}(v, t) \mathrel{+}= \alpha \cdot \mathrm{r}'(v, t)$ once
    If v is virtual **do**
      $\mathrm{r}'(u, t) \mathrel{+}= \frac{\Delta(v)}{\tau(u)} \mathrm{r}'(v, t)$
  $\mathrm{r}'(v, t) \leftarrow 0$

$$\boxed{\pi(s,t) = \pi^o(s,t) + \sum_{v \in V} \pi(s,v) \cdot r(v,t)}$$

$$\boxed{\pi'(s,t) = \pi^{o\prime}(s,t) + \sum_{v \in V'} \pi'(s,v) \cdot r'(v,t)}$$

Node Order in Forward Push in SpeedPPR

Simultaneous way vs asynchronous way.

$O(m \cdot \log \frac{1}{\lambda})$ $\qquad$ $O(\frac{m}{\lambda})$

Algorithm 2: First-In-First-Out Forward Push

Input: $G, \alpha, s, r_{\max}$

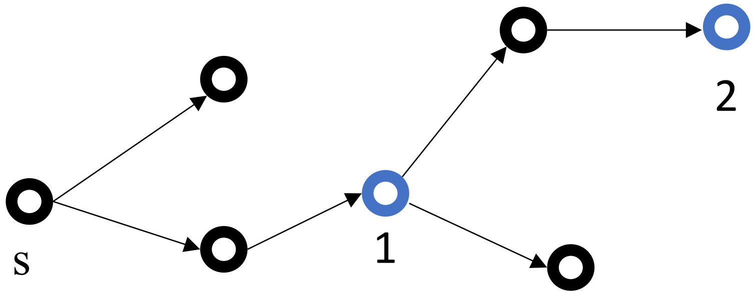
Output: an estimation $\hat{\vec{\pi}}_s$ of $\vec{\pi}_s$ and the resulted residues $\vec{r}_s$

$\hat{\pi}(s,v) \leftarrow 0$ and $r(s,v) \leftarrow 0$ for all $v \in V$; $r(s,s) \leftarrow 1$;
initialize a *first-in-first-out* queue $Q \leftarrow \emptyset$;
$Q.append(s)$; // append s at the end of Q;
while $Q \neq \emptyset$ **do**
  $v \leftarrow Q.pop()$; // pop and remove the front node from Q;
  $\hat{\pi}(s,v) \leftarrow \hat{\pi}(s,v) + \alpha \cdot r(s,v)$;
  for *each* $u \in N_{out}(v)$ **do**
    $r(s,u) \leftarrow r(s,u) + \frac{(1-\alpha)\cdot r(s,v)}{d_v}$;
    if $r(s,u) > d_u \cdot r_{\max}$ *and* $u \notin Q$ **then**
      // u is active and not in Q;
      $Q.append(u)$; // append u at the end of Q;
  $r(s,v) \leftarrow 0$;
return $\hat{\pi}(s,v)$ for all $v \in V$ as a vector $\hat{\vec{\pi}}_s$, and $r(s,v)$ for all $v \in V$ as the resulted residue vector $\vec{r}_s$;

Push Order in Compressed Graph

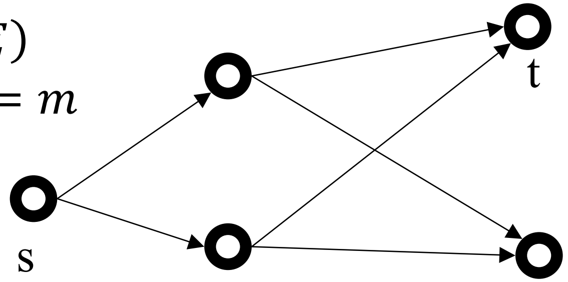
In each iteration:

Real Nodes $\longrightarrow$ Virtual Nodes In Rank Order

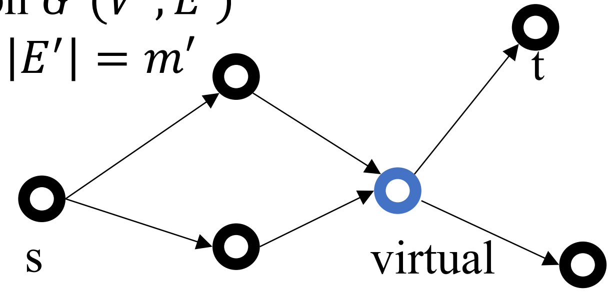


Theoretical Analysis

Graph $G\ (V, E)$
$|V| = n, |E| = m$



Compressed Graph $G'(V', E')$
$|V'| = n,' |E'| = m'$



For $\delta, 0 < \delta < 1$ *and* $m \geq n^{2-\delta}$, *with compressied graph* $G'(V', E')$ *holds*:

- $m' = O\left(\frac{m}{k(n,m,\delta)}\right)$
- $O\left(\frac{m}{n^{1-\delta}}\right)$ *additional(virtual) nodes*
- *runs in time* $O(m \cdot n^{\delta} \cdot \log^2 n)$