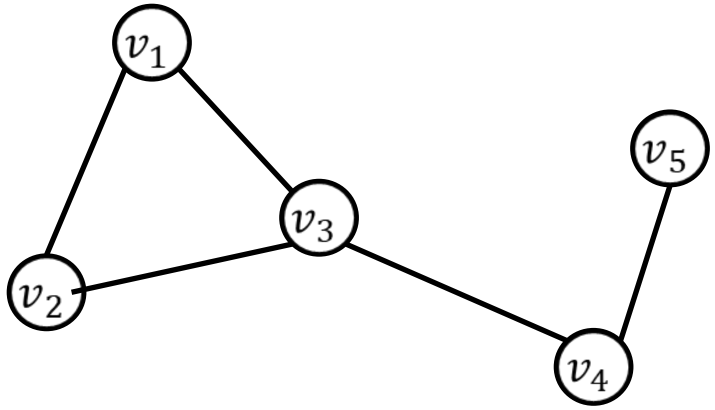
# Global PageRank

- PageRank is an algorithm of measuring the importance of website pages.

- $PR(v_i) = \frac{1-\alpha}{N} + \alpha \times \sum_{v_j \in N(v_i)} \frac{PR(v_j)}{L(v_j)}$

- Vectorize representation:
  - $\vec{r} = \alpha P \vec{r} + \frac{1}{N}(1-\alpha)\vec{1}$




$$P^T = \begin{pmatrix} 0 & 1/2 & 1/2 & 0 & 0 \\ 1/2 & 0 & 1/2 & 0 & 0 \\ 1/3 & 1/3 & 0 & 1/3 & 0 \\ 0 & 0 & 1/2 & 0 & 1/2 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

# Calculate global PageRank

- Iterative process: Each vertex is initialized with a random PageRank value. Iteratively apply the transition function until convergence

- $PR^{(t+1)}(v_i) = \frac{1-\alpha}{N} + \alpha \times \sum_{v_j \in N(v_i)} \frac{PR^{(t)}(v_j)}{L(v_j)}$

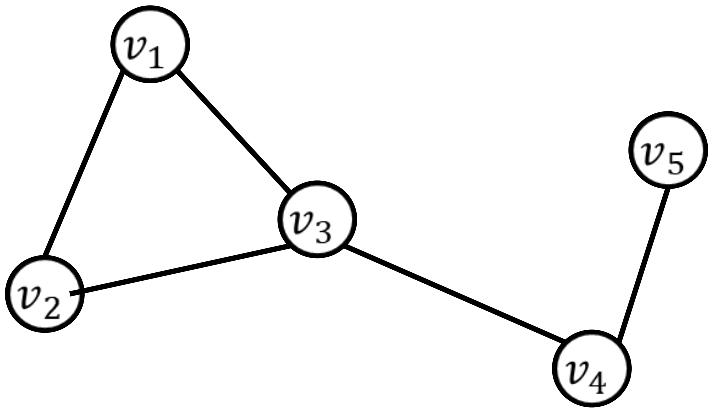
- $\overrightarrow{r^{(t+1)}} = \alpha P \overrightarrow{r^{(t)}} + \frac{1}{N}(1-\alpha)\vec{1}$

- Output: global importance of each vertex

# Personalized PageRank

- global PageRank:
  - $\vec{r} = \alpha P\vec{r} + \frac{1}{N}(1-\alpha)\vec{1}$

$$\begin{bmatrix}1\\1\\1\\1\\1\end{bmatrix}$$

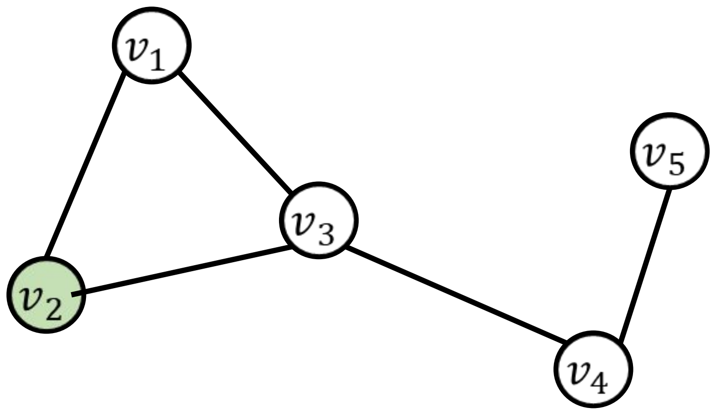



Randomly start from some vertex, each vertex has equal probability

- Personalized PageRank:
  - $\vec{r} = \alpha P\vec{r} + (1-\alpha)\vec{s}$

Start from target vertex

$$\begin{bmatrix}0\\1\\0\\0\\0\end{bmatrix}$$



# Accuracy and latency issue of PPR

- Treat PPR as a linear equation:
  - $\vec{r} = (1 - \alpha P)^{-1}(1 - \alpha)\vec{s}$
  - Exact answer of PPR (very accurate)
  - Need to calculate the inverse of the transition matrix $O(n^3)$

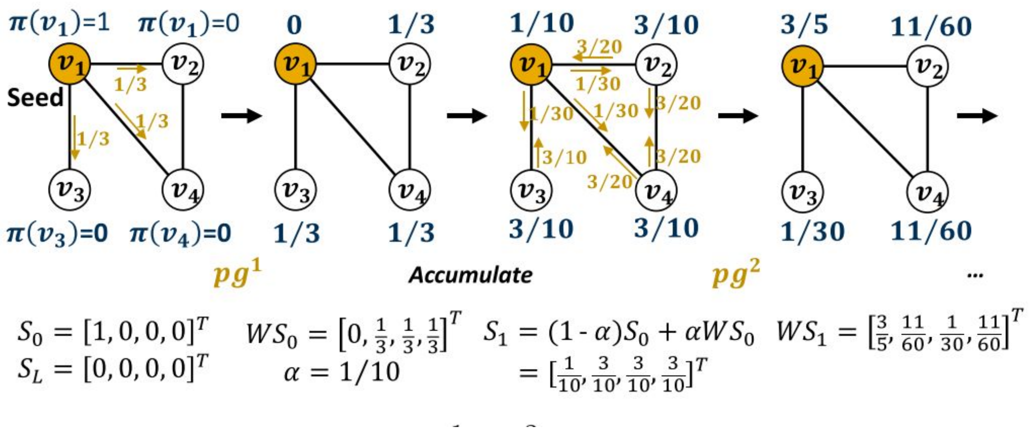
# Fast and approximate algorithm

- Information propagate in a local range, no need to apply the full transition matrix.

# Graph Diffusion algorithm

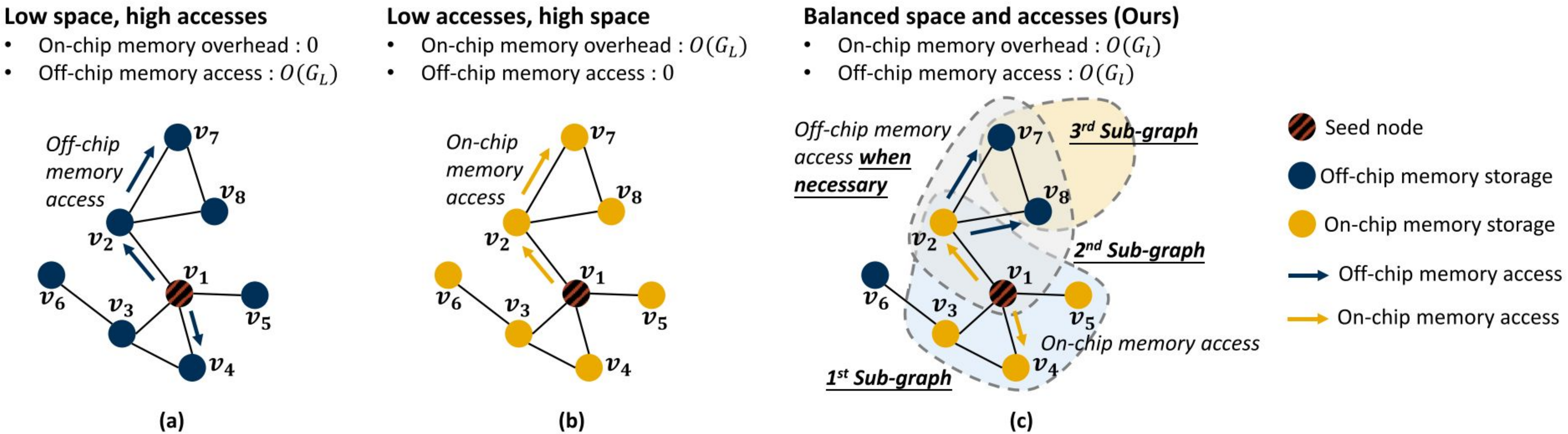
- MELOPPR: Software/Hardware Co-design for Memory-efficient Low-latency Personalized PageRank
- Starting from target vertices, iteratively distribute the PageRank value to the neighbors

- Propagate k iterations -> can reach $k$ hop neighbors
- $k-hop$ neighbors grows exponentially, large memory requirement
- More iteration higher accuracy

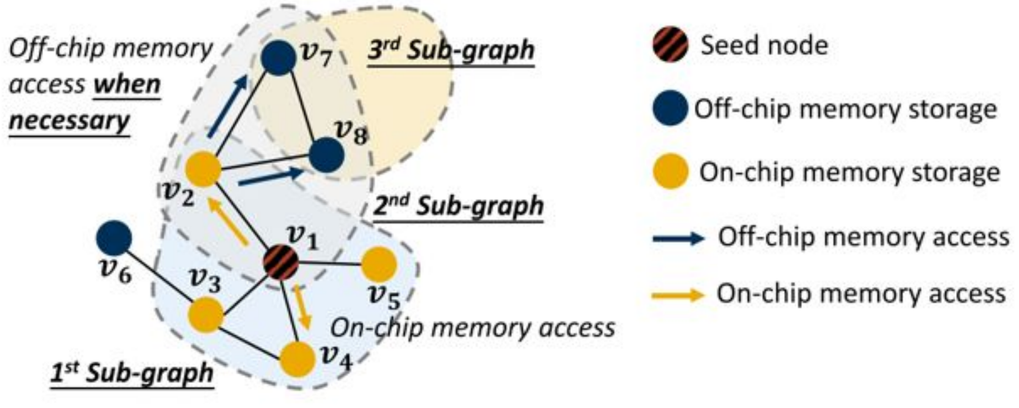


# Multi-stage PPR

- PPR has linearity
  - $PPR(w_1\vec{u} + w_2\vec{v}) = w_1 * PPR(\vec{u}) + w_2 * PPR(\vec{v})$



# Multi-stage PPR



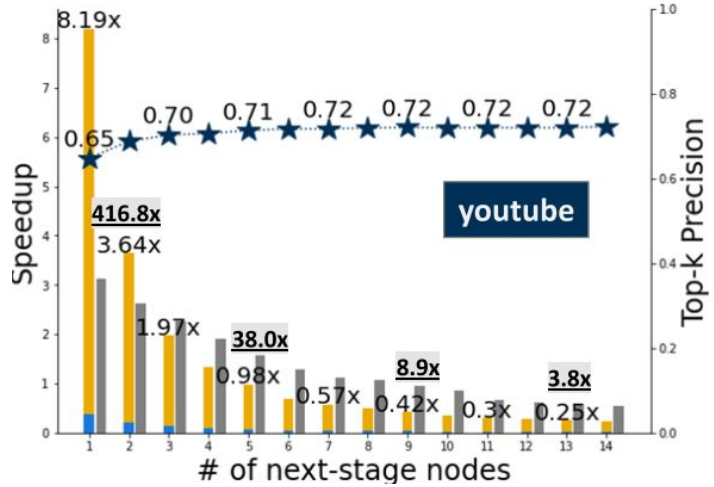

(c)

- $k$ iterations $\rightarrow$ $l_1 + l_2 = k$ iteration
  - Example; 2 iterations $\rightarrow$ $1 + 1 = 2$ iteration
- $In$ the first $l_1$ iteration, reach the $l_1$ hop neighbors of the target vertex. Then start from each $l_1$ hop neighbors, perform PPR $l_2$ iteration.
- $GD^l(S)$: pagerank vector start from initialization for $l$ iterations.

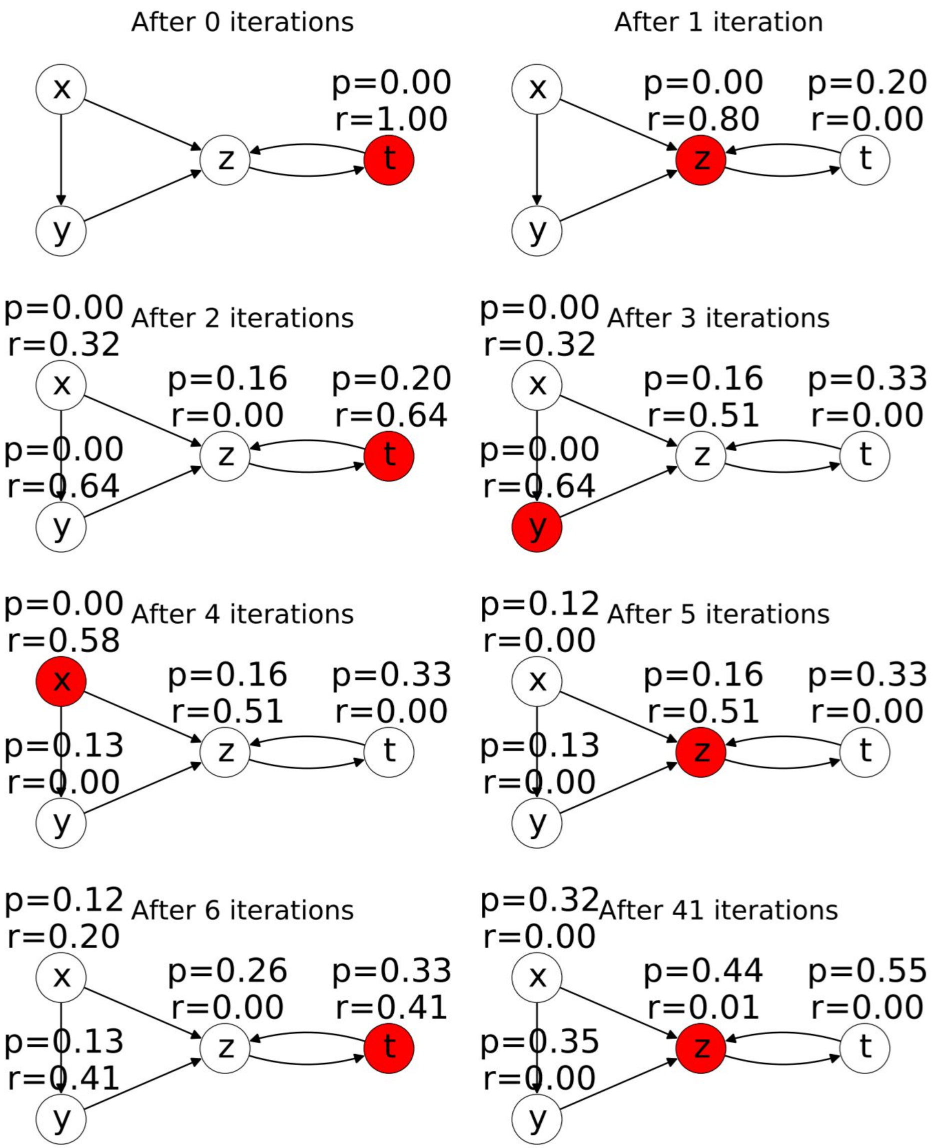
$$\mathcal{GD}^{(l_1+l_2)}(S_0) = \mathcal{GD}^{(l_1)}(S_0) - \alpha^{l_1} S^r_{l_1} + \alpha^{l_1} \sum_{v \in G_{l_1}(s)} \mathcal{GD}^{(l_2)}(S^r_{l_1,v})$$

# MELOPPR

- Fast, memory efficient
- No guaranteed error bound
- Error measure:
  - $\hat{T}(s,k)$: $top - k$ important vertices selected by the algorithm
  - $T(s,k)$: $top - k$ important vertices in the ground truth
  - Precision: $prec(s,k) = \left|\{v | v \in \hat{T}(s,k) \wedge c \in T(s,k)\}\right| / \mathrm{k}$



# Approximate Personalized PageRank using local push




`ApproximatePageRank` $(v, \alpha, \epsilon)$:

1. Let $p = \vec{0}$, and $r = \chi_v$.
2. While $\max_{u \in V} \frac{r(u)}{d(u)} \geq \epsilon$:
   - (a) Choose any vertex $u$ where $\frac{r(u)}{d(u)} \geq \epsilon$.
   - (b) Apply $\texttt{push}_u$ at vertex $u$, updating $p$ and $r$.
3. Return $p$, which satisfies $p = \text{apr}(\alpha, \chi_v, r)$ with $\max_{u \in V} \frac{r(u)}{d(u)} < \epsilon$.

$\texttt{push}_u(p, r)$:

1. Let $p' = p$ and $r' = r$, except for the following changes:
   - (a) $p'(u) = p(u) + \alpha r(u)$.
   - (b) $r'(u) = (1 - \alpha) r(u)/2$.
   - (c) For each v such that $(u, v) \in E$: $\quad r'(v) = r(v) + (1 - \alpha) r(u)/(2d(u))$.
2. Return $(p', r')$.

# local push

- Convergent until an error bound $\epsilon$ is reached
- No guaranteed execution time
- The computation complexity has a upper bound $O(\frac{1}{\epsilon})$
- Can be applied to dynamic graph:
  - Approximate Personalized PageRank on Dynamic Graphs
  - When a new edge is added, the complexity of updating the PPR has the complexity of O(1).