Your LOGO

Taming Computational Complexity :

Efficient and Parallel SimRank Optimizations on Undirected Graphs

Contents

What is SimRank?

- The similarity in a domain can be modeled as graphs.
  [vertices→ objects , edges → relationships]
- SimRank is an important similarity measure which exploits the relationships between vertices on web graphs.
  (Glen Jeh & Jennifer Widom , ’02)
- Basic intuition:
  - Two objects are similar if their neighbors are similar.
    (the recursive definition)
  - Objects are maximally similar to themselves.
    (the base case)

Existing Similarity measures

- Textual-Content Similarity (text-based)
  - Vector-cosine similarity, Pearson correlation in IR
- Structural-Context Similarity (link-based)
  - PageRank
    - One page's authority is decided by its neighbors' authorities.
  - SimRank
    - Two objects are similar if they are referenced by similar objects.

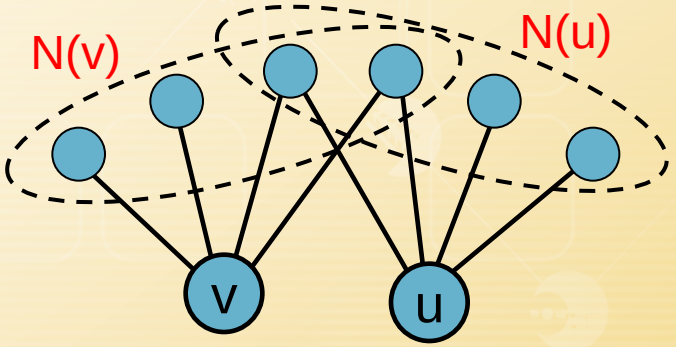
Contents

G vs. G^2 Model

- Basic Graph Model: G = (V, E)

  For each vertex $v \in V$, we define:

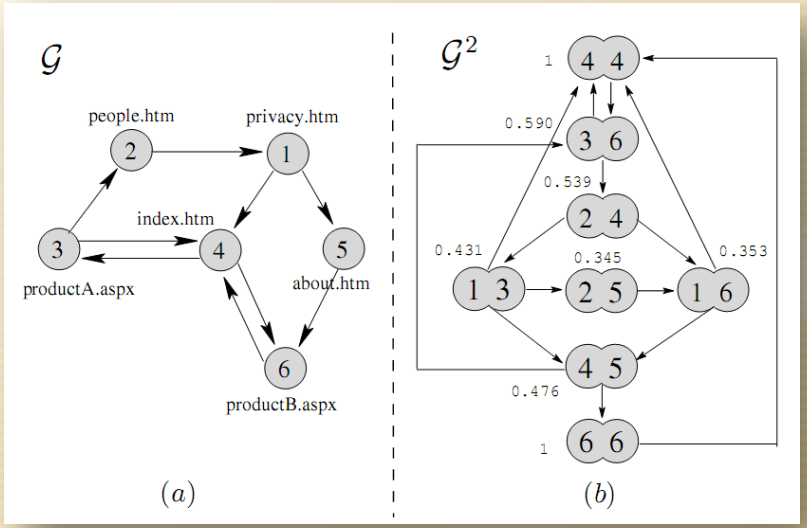
  - N(v): all the neighbors of vertex v
  - $N_i(v)$: individual member of N(v)



- Node-pair Graph: $G^2 = (V^2, E^2)$

  $\forall$ (a, b) $\in V^2$ represents a pair (a, b) of nodes in G.

  $\forall \langle (a_1, b_1) , (a_2, b_2) \rangle \in E^2$ denotes the edges $\langle a_1, a_2 \rangle$ and $\langle b_1, b_2 \rangle$ exist in G.



SimRank propagating similarity from node to node in G is associated with the propagation from pair to pair in G^2.

SimRank Equation

- Definition 1 (SimRank similarity)

Let s: $V^2 \rightarrow [0, 1]$ be a similarity function on G^2

- if a = b, → s (a, b) = 1,
- if N(a) or N(b) = ∅, → s (a, b) = 0,
- otherwise:

$$s(a,b) = \frac{c}{|N(a)| \cdot |N(b)|} \cdot \sum_{i=1}^{|N(a)|} \sum_{j=1}^{|N(b)|} s\big(N(a), N(b)\big)$$

where c is a decay factor btw. 0 & 1

Similarity btw. a & b is the average similarity btw. neighbors of a and neighbors of b.

Existing Techniques for SimRank Optimization

- **Deterministic** Method [VLDB J. '10, EDBT '10, APWEB '10, etc]

(computing s(·, ·) iteratively for finding a fixed point)

$$s^{(k+1)}(a,b) = \frac{c}{|N(a)| \cdot |N(b)|} \cdot \sum_{i=1}^{|N(a)|} \sum_{j=1}^{|N(b)|} s^{(k)}(N(a), N(b))$$

Advantage: accurate

Disadvantage: high time complexity ($O(Kn^3)$ in the worse case)

- **Probabilistic** Method [WWW '05, SIGIR '06]

(estimating s(·, ·) stochastically by using Monte-Carlo)

$s(a,b) = E(c^{T(a,b)})$, where $T(a,b)$: the first meeting time btw. a & b

Advantage: scalable (linear time)

Disadvantage: low similarity quality

Existing Techniques for SimRank Deterministic Computation

- Jeh and Widom first proposed a SimRank model, taking $O(Kn^4)$ worst-case time. [WWW '02]
- Li et al. proposed a non-iterative approximate algorithm, yielding $O(r^4n^2)$ time for dynamic information networks. [EDBT '10]
- Lizorkin et al. used a partial sum function, reducing the time to $O(Kn^3)$ in the worst case. [VLDB '10]
- Yu et al. showed a fast matrix multiplication for digraph, requiring $O(K \cdot \min(m \cdot n, n^r))$, where $2<r<\log_2 7$. [APWEB '10]

Motivations

- The time required for SimRank deterministic algorithms is still about cubic in the number of vertices for each iteration, which is costly over large graphs.

- As for SimRank deterministic computation, parallel implementation has not been addressed in scientific literature yet.

Our Contributions

- We present an efficient spectral decomposition based algorithm for SimRank computation over undirected graphs, which reduces the time complexity from $O(Kn^3)$ to $O(n^3 + Kn^2)$.
- We develop a block partition technique in combination with the Parallel Linear Algebra Package (PLAPACK) to parallelize SimRank algorithm on distributed memory multi-processors.
- We perform extensive evaluations of our proposed methods demonstrating the efficiency and effectiveness of our algorithms.

Contents

Your
LOGO

Efficient and Parallel SimRank Optimizations on Undirected Graphs

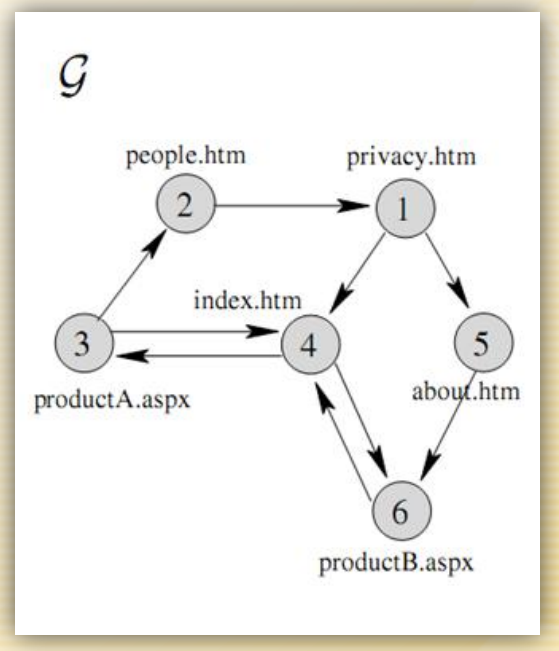
3.1 AUG-SimRank




Graph Spectrum

- Definition 1. (Graph Spectrum) Given a web graph G, let Q_G denote its transition probability matrix. The spectrum of G is defined to be the set of the eigenvalues of Q_G. In symbols,

$$\sigma(\mathcal{G}) := \{\lambda \in \mathbb{C} | \mathbf{Q}_{\mathcal{G}} \cdot \mathbf{u} = \lambda \cdot \mathbf{u}, \quad \forall \mathbf{u} \neq \mathbf{0}\}$$



directed graph

$$\mathbf{Q}_{\mathcal{G}} = \begin{pmatrix} 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 & 0 & \frac{1}{2} \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \quad \sigma(\mathcal{G}) = \begin{pmatrix} -1 \\ 1 \\ -0.338 + 0.4892i \\ -0.338 - 0.4892i \\ 0.338 + 0.4892i \\ 0.338 - 0.4892i \end{pmatrix}$$

undirected graph

$$\mathbf{Q}_{\mathcal{G}} = \begin{pmatrix} 0 & \frac{1}{3} & 0 & \frac{1}{3} & \frac{1}{3} & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 \\ \frac{1}{3} & 0 & \frac{1}{3} & 0 & 0 & \frac{1}{3} \\ \frac{1}{2} & 0 & 0 & 0 & 0 & \frac{1}{2} \\ 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 \end{pmatrix} \quad \sigma(\mathcal{G}) = \begin{pmatrix} -1 \\ 1 \\ 0.1667 \\ -0.1667 \\ 0.5 \\ -0.5 \end{pmatrix}$$

Graph Spectrum

- For a digraph G,
  - some elements in σ(G) might be complex numbers.
- For an undirected graph G,
  - all elements in σ(G) must be real numbers.

- Theorem 1. Given an undirected graph G, all the eigenvalues of its transition probability matrix Q_G are real numbers associated with a complete set of orthonormal eigenvectors.

Graph Spectrum

- Theorem 2. For an undirected graph G, let $Q = U \cdot \Lambda \cdot U^{-1}$ be a complete spectral decomposition of Q, where
  - U is an orthogonal matrix with real entities whose columns are eigenvectors of Q,
  - Λ is a real diagonal matrix whose diagonal entities give the corresponding eigenvalues.

Then we can construct the following iteration:

$$\tilde{\mathbf{S}}_k = \begin{cases} \mathbf{I}_n & k = 0 \\ \left(\left[c \cdot diag(\mathbf{\Lambda}) \cdot diag(\mathbf{\Lambda})^T \right] \odot \tilde{\mathbf{S}}_{k-1} \right) \vee \mathbf{I}_n, & k = 1, 2, \cdots \end{cases}$$

And SimRank similarity can be thereby obtained as follows:

$$\mathbf{S}_k = \mathbf{U} \cdot \tilde{\mathbf{S}}_k \cdot \mathbf{U}^{-1}$$

Key Observations

- 0. SimRank Matrix Representation

$$s^{(0)}(a,b) = \begin{cases} 1, & a = b; \\ 0, & a \neq b. \end{cases}$$

$$s^{(k)}(a,b) = \begin{cases} 1, & a = b; \\ \frac{c}{|N(a)||N(b)|} \sum_{j=1}^{|N(b)|} \sum_{i=1}^{|N(a)|} s^{(k-1)}\left(N_i(a), N_j(b)\right), & N(a), N(b) \neq \varnothing; \\ 0, & otherwise. \end{cases}$$

$$\begin{cases} \mathbf{S}^{(0)} = \mathbf{I}_n \\ \mathbf{S}^{(k)} = \left(c \cdot \mathbf{Q} \cdot \mathbf{S}^{(k-1)} \cdot \mathbf{Q}^T\right) \bigvee \mathbf{I}_n \end{cases}$$

- 1. Spectral Predecomposition [Theorem 1] $O(n^3)$

$$\mathbf{Q} = \mathbf{U} \cdot \mathbf{\Lambda} \cdot \mathbf{U}^{-1}$$

- 2. Iterative Element-wise Matrix Multiplication

$$\tilde{\mathbf{S}}_k = \left(\left[c \cdot diag(\mathbf{\Lambda}) \cdot diag(\mathbf{\Lambda})^T\right] \odot \tilde{\mathbf{S}}_{k-1}\right) \vee \mathbf{I}_n \quad \Leftrightarrow \quad \tilde{\mathbf{S}}_k = \left(c \cdot \mathbf{\Lambda} \cdot \tilde{\mathbf{S}}_{k-1} \cdot \mathbf{\Lambda}\right) \vee \mathbf{I}_n$$

$O(Kn^2)$

- 3. SimRank Matrix Computation $O(n^3)$

$$\mathbf{S}_k = \mathbf{U} \cdot \tilde{\mathbf{S}}_k \cdot \mathbf{U}^{-1}$$

Key Observations (cont.)

- Notice that $\Lambda \cdot S \cdot \Lambda = [diag(\Lambda) \cdot diag(\Lambda)^T] \odot S$ is our trick to reduce the time complexity from $O(n^3)$ to $O(n^2)$ per iteration.

$$\mathbf{\Lambda} \cdot \mathbf{S} \cdot \mathbf{\Lambda} = \begin{pmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_n \end{pmatrix} \cdot \begin{pmatrix} s_{11} & s_{12} & \dots & s_{1n} \\ s_{21} & s_{22} & \dots & s_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ s_{n1} & s_{n2} & \dots & s_{nn} \end{pmatrix} \cdot \begin{pmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_n \end{pmatrix}$$

$$=$$

$$\begin{pmatrix} {\lambda_1}^2 & \lambda_1\lambda_2 & \dots & \lambda_1\lambda_n \\ \lambda_2\lambda_1 & {\lambda_2}^2 & \dots & \lambda_2\lambda_n \\ \vdots & \vdots & \ddots & \vdots \\ \lambda_n\lambda_1 & \lambda_n\lambda_2 & \dots & {\lambda_n}^2 \end{pmatrix} \odot \begin{pmatrix} s_{11} & s_{12} & \dots & s_{1n} \\ s_{21} & s_{22} & \dots & s_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ s_{n1} & s_{n2} & \dots & s_{nn} \end{pmatrix} = \begin{pmatrix} {\lambda_1}^2 \cdot s_{11} & \lambda_1\lambda_2 \cdot s_{12} & \dots & \lambda_1\lambda_n \cdot s_{1n} \\ \lambda_2\lambda_1 \cdot s_{21} & {\lambda_2}^2 \cdot s_{22} & \dots & \lambda_2\lambda_n \cdot s_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \lambda_n\lambda_1 \cdot s_{n1} & \lambda_n\lambda_2 \cdot s_{n2} & \dots & {\lambda_n}^2 \cdot s_{nn} \end{pmatrix}$$

$$=$$

$$[diag(\mathbf{\Lambda}) \cdot diag(\mathbf{\Lambda})^T] \odot \mathbf{S} = \left[\begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_n \end{pmatrix} \cdot (\lambda_1 \ \lambda_2 \ \cdots \ \lambda_n) \right] \odot \begin{pmatrix} s_{11} & s_{12} & \dots & s_{1n} \\ s_{21} & s_{22} & \dots & s_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ s_{n1} & s_{n2} & \dots & s_{nn} \end{pmatrix}$$

AUG-SimRank Algorithm

- Theorem 3. For undirected graphs, SimRank can be performed for K iterations in $O(n^3 + K \cdot n^2)$ time in the worst case, where n is the number of vertices, and $n \gg K$.

Algorithm 1: AUG-SimRank: Accelerative SimRank for Undirected Graphs

Input : adjacency matrix $\mathbf{P} = (p_{i,j}) \in \mathbb{R}^{n \times n}$, decay factor c, accuracy ϵ
Output: SimRank matrix $\mathbf{S} = (s_{i,j}) \in [0,1]^{n \times n}$, iteration number k

1 **begin**
2 Calculate transition probability matrix $\mathbf{Q}$ according to $q_{i,j} \leftarrow p_{j,i} / \sum_{i=1}^{n} p_{j,i}$
3 Decompose $\mathbf{Q} \rightarrow \mathbf{U} \cdot \mathbf{\Lambda} \cdot \mathbf{U}^{-1}$, yielding $\mathbf{U}$, orthogonal, and $\mathbf{\Lambda}$, diagonal
4 Initialize $\tilde{\mathbf{S}}_0 \leftarrow \mathbf{I}_n$, $\mathbf{M} \leftarrow c \cdot diag(\mathbf{\Lambda}) \cdot diag(\mathbf{\Lambda})^T$, $k \leftarrow 0$
5 **repeat** /* each iteration takes $O(n^2)$ time */
6 Set $k \leftarrow k + 1$
7 Update $\tilde{\mathbf{S}}_k \leftarrow (\mathbf{M} \odot \tilde{\mathbf{S}}_{k-1}) \vee \mathbf{I}_n$
8 **until** *the auxiliary matrix* $\tilde{\mathbf{S}}_k$ *converges with an error of* ϵ
9 Calculate SimRank matrix $\mathbf{S}_k \leftarrow \mathbf{U} \cdot \tilde{\mathbf{S}}_k \cdot \mathbf{U}^{-1}$
10 Return $\mathbf{S}_k$, k

- Preconditioning techniques may be adopted when we calculate diag (Λ) · diag(Λ)^T . Once computed, this rank-1 matrix is memorized and is therefore not recomputed when subsequently required.

Your
LOGO

Efficient and Parallel SimRank Optimizations on Undirected Graphs

3.2 PAUG-SimRank



Parallel AUG-SimRank

- To parallelize the AUG-SimRank algorithm, we utilize PLAPACK in combination with matrix partition techniques on distributed memory architectures.
- PLAPACK is a parallel ARPACK version based on MPI (Message Passing Interface) for constructing parallel linear algebra libraries.
  - It provides a high-level object-oriented programming interface.
  - The coding of parallel linear algebra routines becomes a straightforward translation of algorithms.

Parallel AUG-SimRank

- In the spectral predecomposition phase,
  - Use a PLAPACK eigen-solver to decompose $Q \rightarrow U \cdot \Lambda \cdot U^{-1}$.
  - Partition the row vector $\mathrm{diag}(\Lambda)^T \rightarrow (\Lambda^{(1)} \ \Lambda^{(2)} \ \cdots \ \Lambda^{(N)})$.
- In the iterative element-wise matrix multiplication phase,
  - Initialize the upper triangular part of $M^{(i)} \leftarrow c \cdot \mathrm{diag}(\Lambda) \cdot \Lambda^{(i)}$.
  - Partition the similarity matrix as

$$\left(\tilde{\mathbf{S}}_k^{(1)} \middle| \cdots \middle| \tilde{\mathbf{S}}_k^{(N)}\right) = \left(\mathbf{M}^{(1)} \odot \tilde{\mathbf{S}}_{k-1}^{(1)} \middle| \cdots \middle| \mathbf{M}^{(N)} \odot \tilde{\mathbf{S}}_{k-1}^{(N)}\right) \vee \left(\mathbf{I}^{(1)} \middle| \cdots \middle| \mathbf{I}^{(N)}\right)$$

  - Calculate each partition in parallel as

$$\tilde{\mathbf{S}}_k^{(i)} = \left(\mathbf{M}^{(i)} \odot \tilde{\mathbf{S}}_{k-1}^{(i)}\right) \vee \mathbf{I}^{(i)} \quad (\forall i = 1, \cdots, N, \quad k = 1, 2, \cdots)$$

Parallel AUG-SimRank (cont.)

- In the SimRank matrix computation phase,

  parallel computation of S_k can be performed by the following substeps:

  - A symmetric matrix-matrix multiplication $\mathbf{V}_k \leftarrow \tilde{\mathbf{S}}_k \cdot \mathbf{U}^{-1}$ can be parallelized in PLAPACK.

  - The upper (or lower) triangular part of S_k can be updated as

$$\mathbf{S}_k \leftarrow \mathbf{U} \cdot \mathbf{V}_k = \left(\mathbf{U}^{(1)} \middle| \cdots \middle| \mathbf{U}^{(N')} \right) \cdot \begin{pmatrix} \mathbf{V}_k^{(1)} \\ \hline \vdots \\ \hline \mathbf{V}_k^{(N')} \end{pmatrix} = \sum_{i=1}^{N'} \mathbf{U}^{(i)} \cdot \mathbf{V}_k^{(i)}$$

  hence, S_k can be computed as

$$\mathbf{S}_k \leftarrow \mathbf{S}_k + \mathbf{U}^{(i)} \cdot \mathbf{V}_k^{(i)} \quad (\forall i = 1, \cdots, N')$$

Contents

Experimental Studies

- Hardware
  - 2.0GHz Pentium(R) Dual-Core / 2GB RAM
  - Windows Vista OS / Visual C++ 6.0
- Data Sets
  - Synthetic
    - graph with an average of 8 links per page.
    - 10 sample adjacency matrices from 1K to 10K with $\xi \sim$uniform[0; 16] out-links on each row.
  - Real-life
    - Wikipedia (3.2M articles with 110M intra-wiki links / Oct. ’07)
    - We choose the relationship : “a category contains an article to be a link btw. the category and the article”.

Experimental Studies

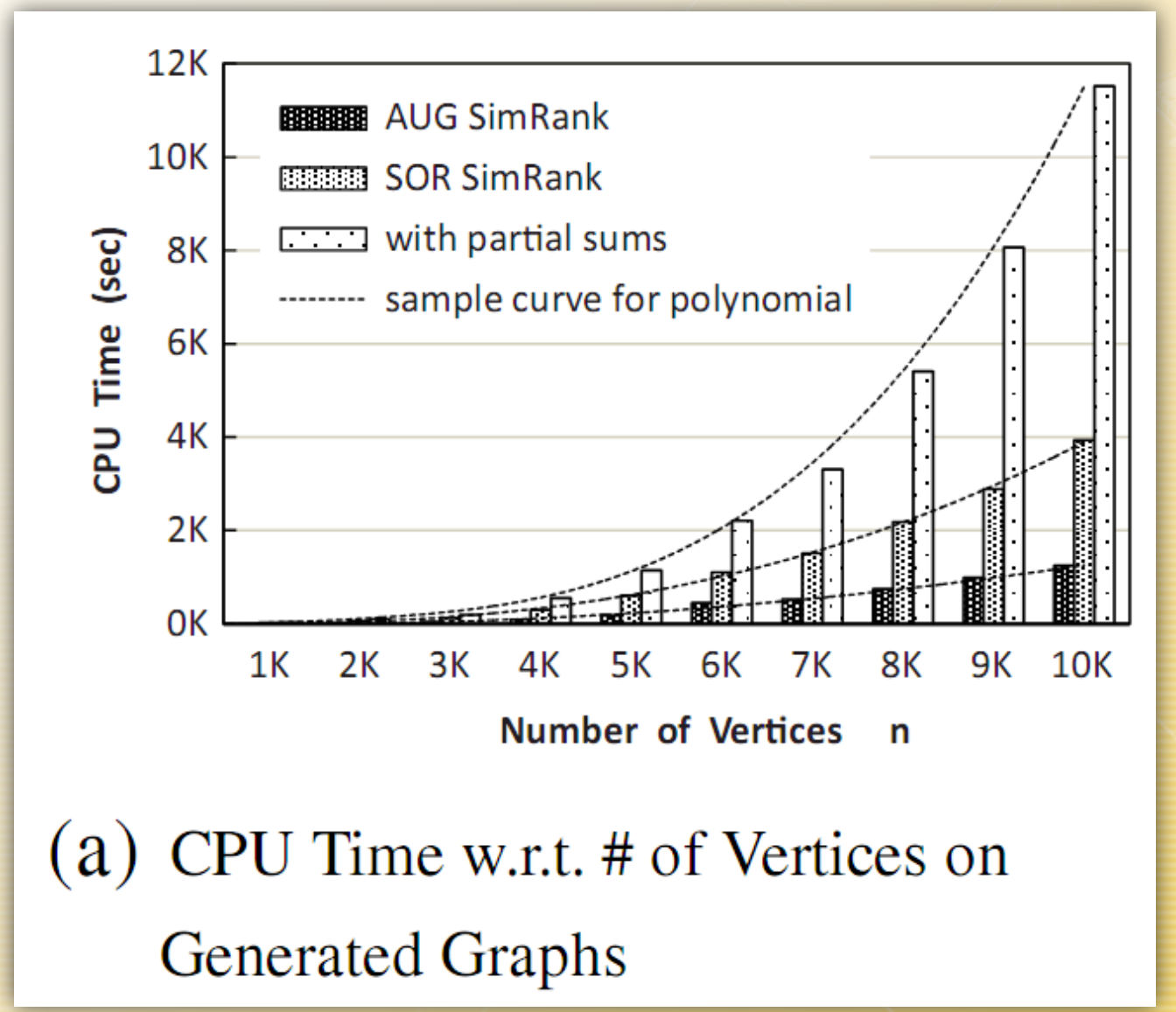
- Algorithms for Comparison
  - SimRank with partial sums. [VLDB '10]
  - SOR SimRank. [APWEB '10]
  - AUG SimRank. & Parallel AUG SimRank.
- Evaluation Measures
  - CPU time : computational complexity
  - absolute speedup : parallel efficiency

$$S_p := \frac{T_1}{T_p}$$

- p number of processors
- T_1 execution time of the sequential algorithm on one processor
- T_p time taken on p processors

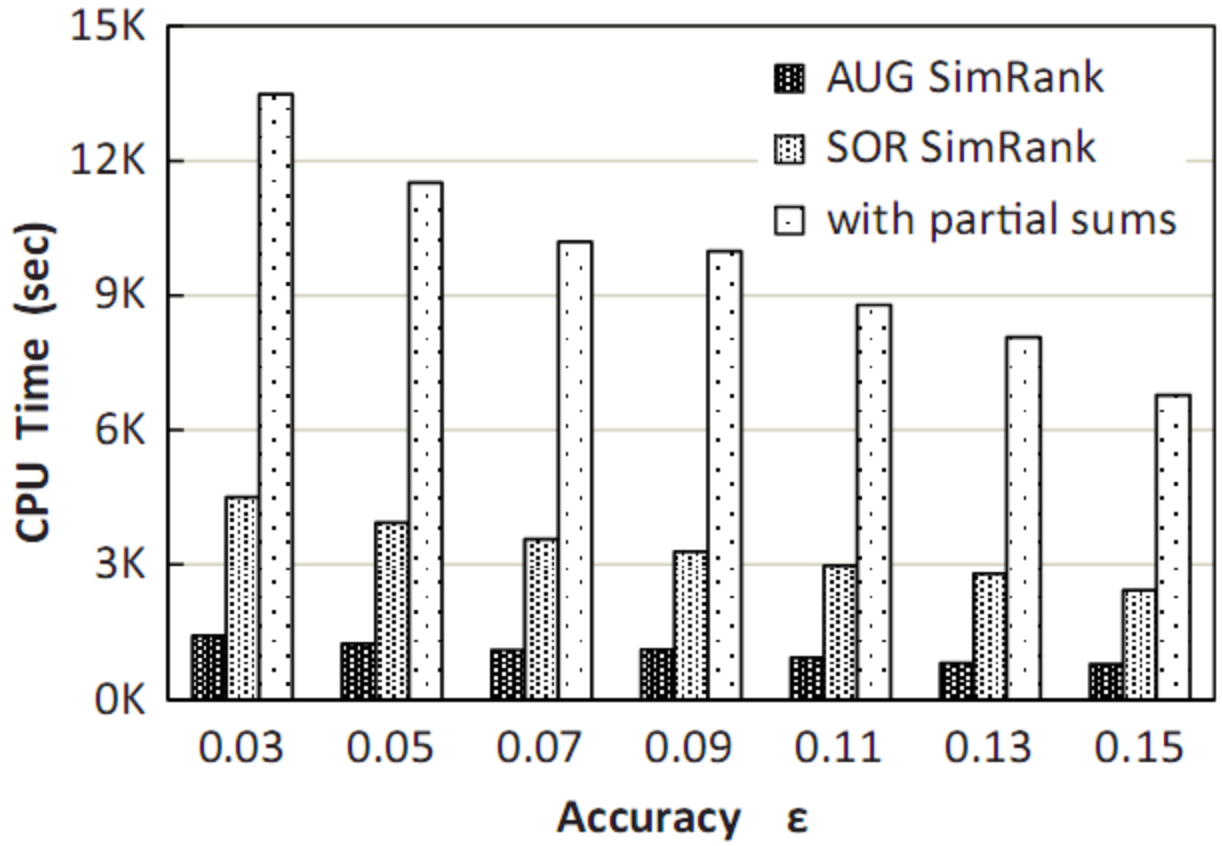
- Parameter Settings
  - c = 0.8, ω = 1.3, ϵ = 0.05

Time Efficiency Evaluation



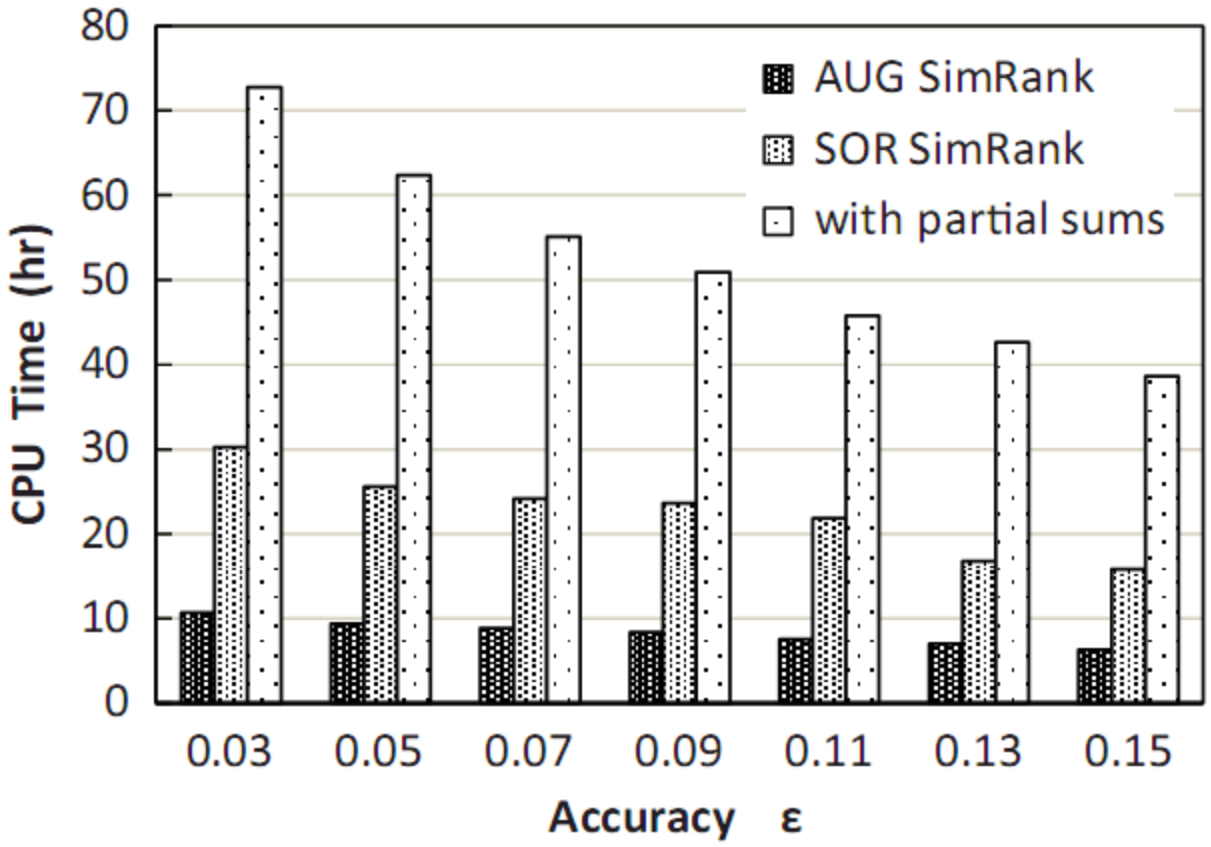
(a) CPU Time w.r.t. # of Vertices on Generated Graphs

Time Efficiency Evaluation



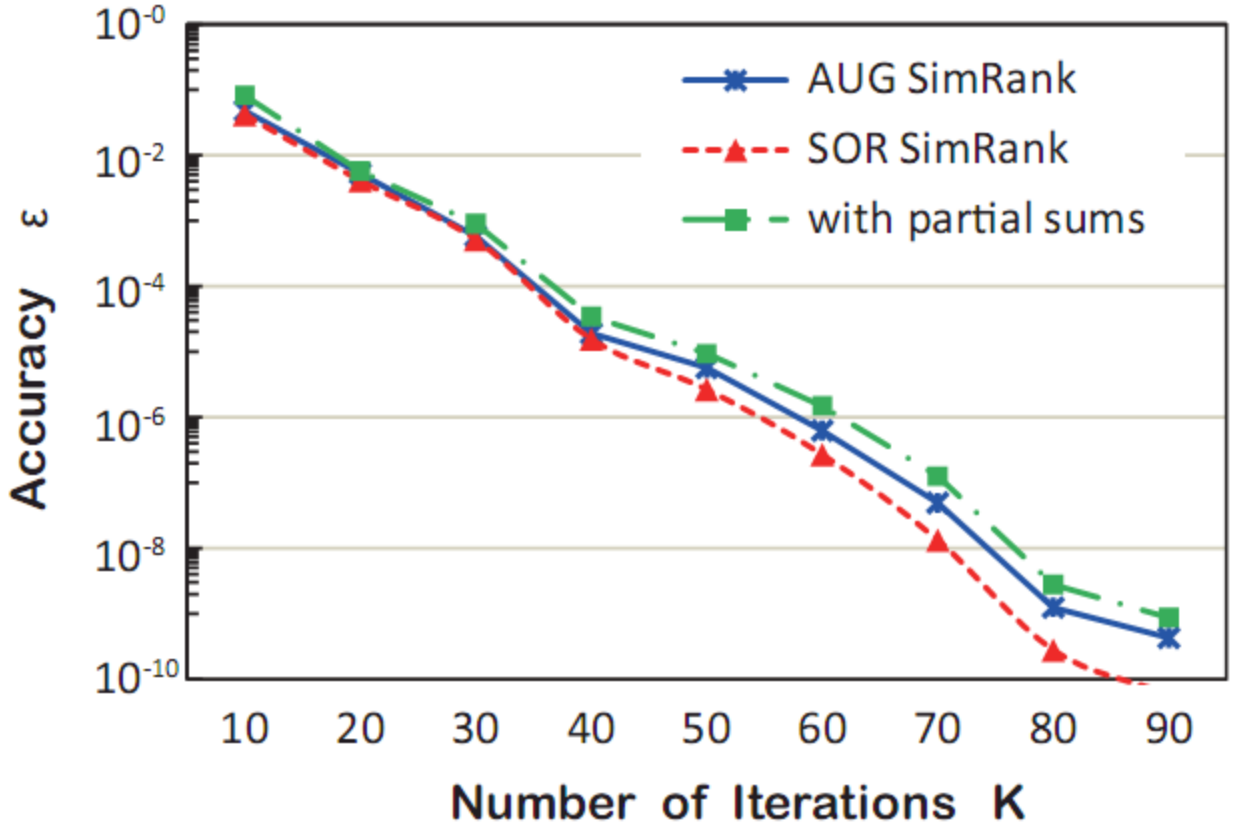
(b) CPU Time w.r.t. Accuracy on a Generated Graph (with n=10K)

Time Efficiency Evaluation



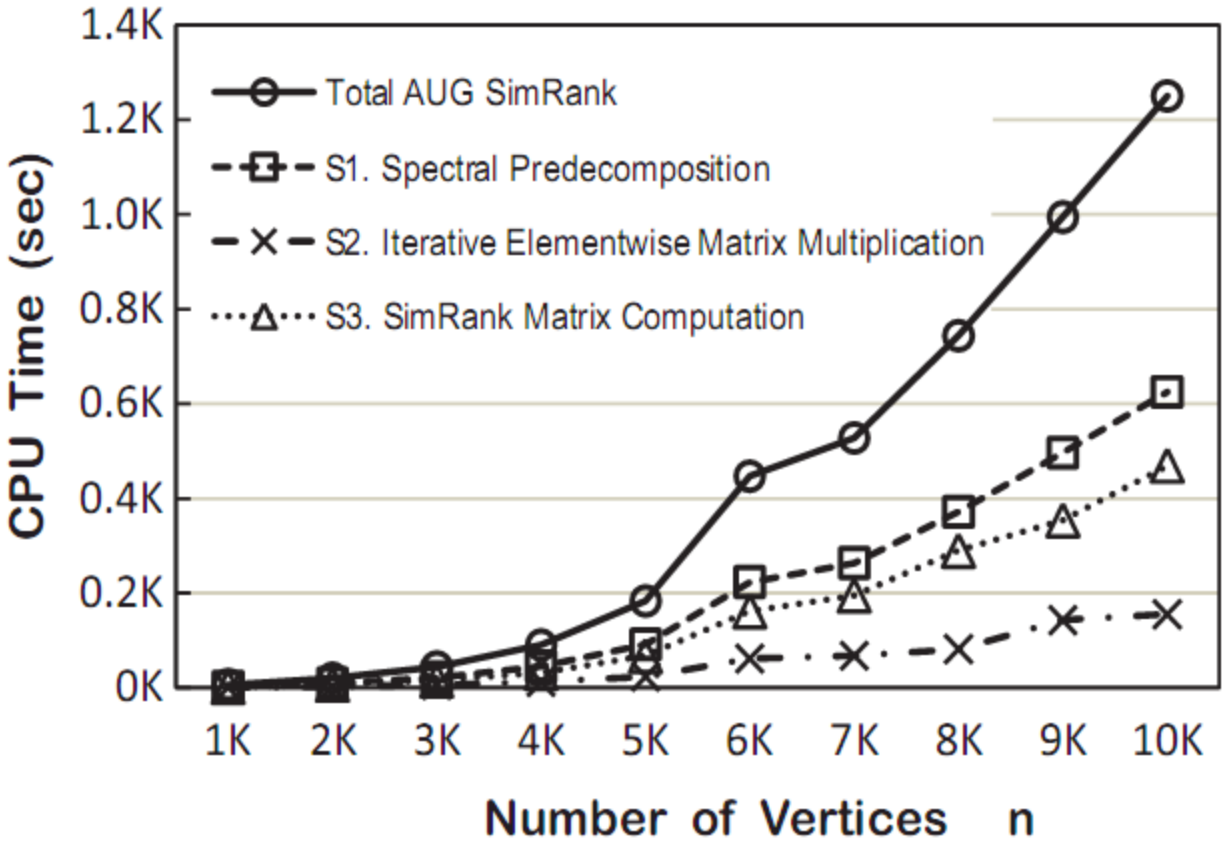
(c) CPU Time w.r.t. Accuracy on Wikipedia (with c=0.6)

Time Efficiency Evaluation



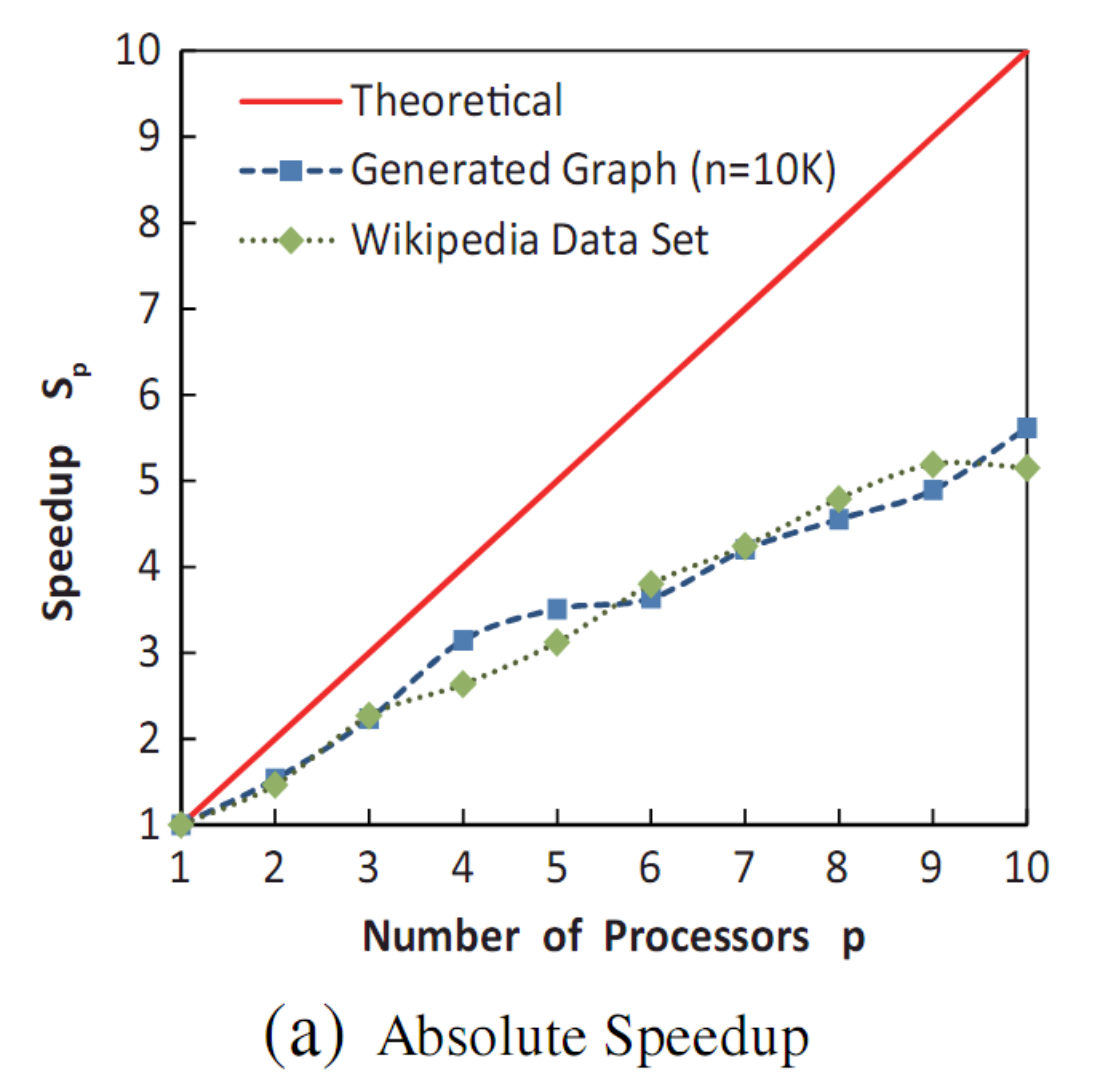
(d) Accuracy w.r.t. # of Iterations on Wikipedia (with c=0.6)

Time Efficiency Evaluation



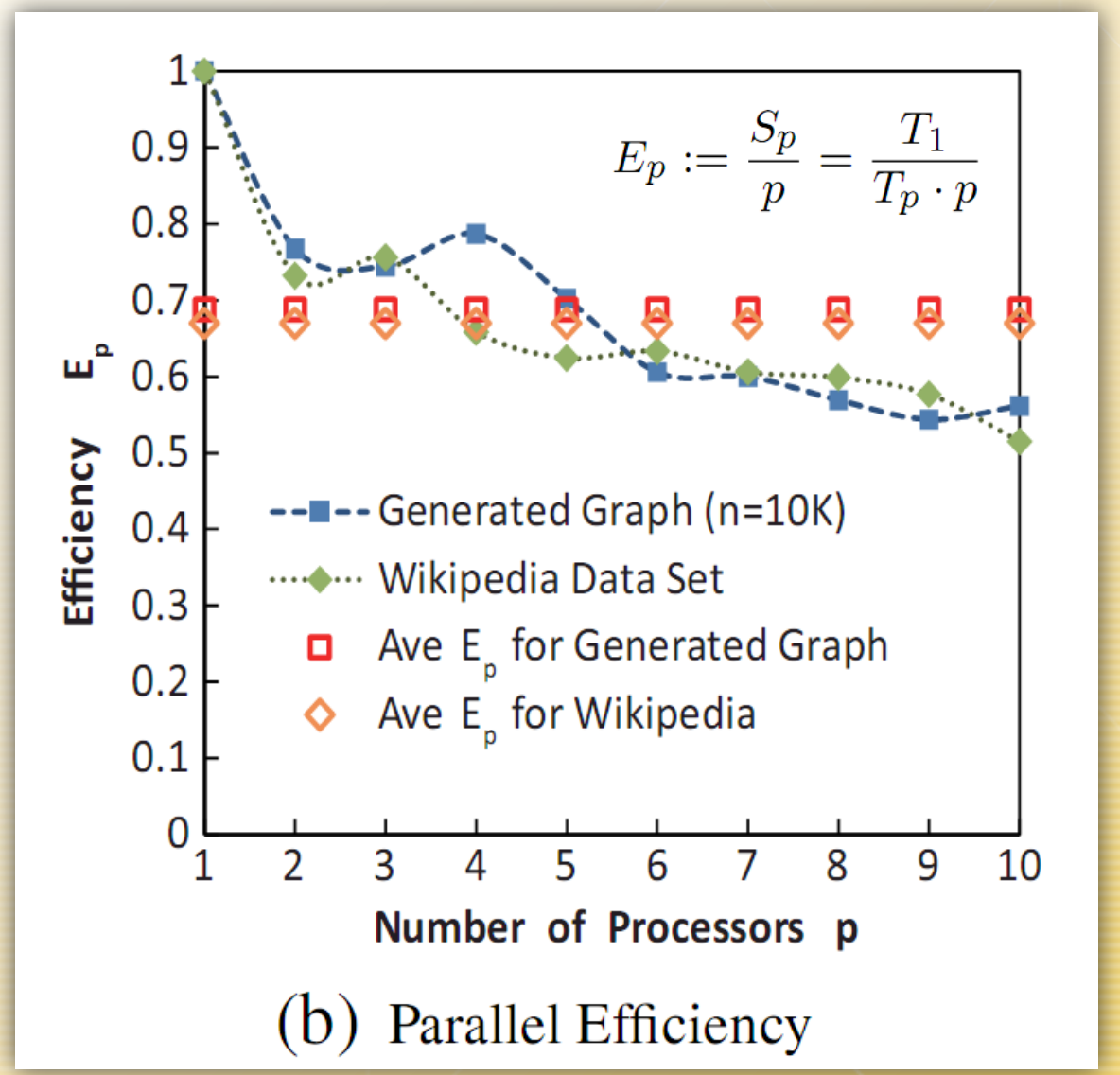
(e) AUG-SimRank Algorithm on Generated Graphs

Parallel Efficiency Evaluation



(a) Absolute Speedup

Parallel Efficiency Evaluation



(b) Parallel Efficiency

Thank You !