SWeG: Lossless and Lossy Summarization of Web-Scale Graphs

Kijung Shin
KAIST
kijungs@kaist.ac.kr

Amol Ghoting
LinkedIn
aghoting@linkedin.com

Myunghwan Kim
Mesh Korea
mykim@cs.stanford.edu

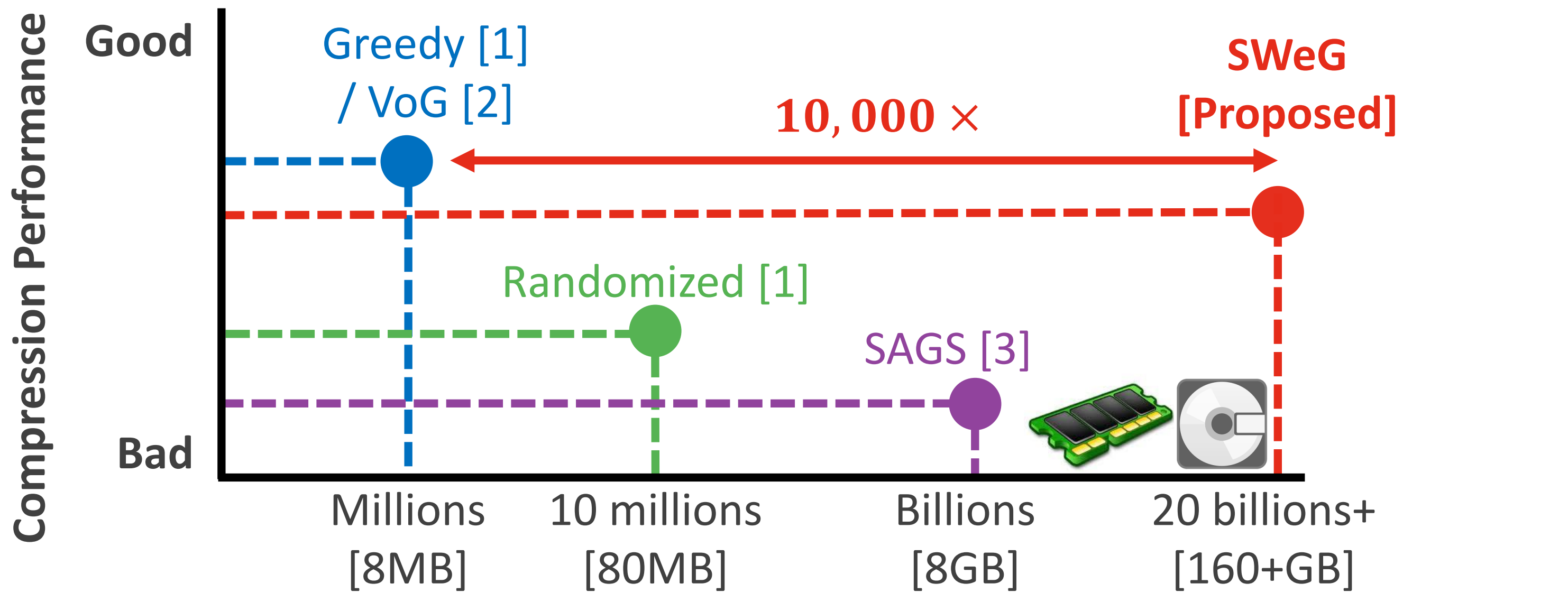
Hema Raghavan
LinkedIn
hraghavan@linkedin.com

Summary

- **Goal:** to compactly represent graphs with tens or hundreds of billions of edges
- **Previous Work:**
  - **Graph summarization:** a promising graph-compression technique
  - **Algorithms** for summarizing graphs that are small enough to fit in main memory
- **Proposed Algorithm (SWeG):**
  - a parallel and distributed algorithm for compactly summarizing large-scale graphs
  - scales near linearly with the size of the input graph and requires sub-linear memory
- **Results:**
  - **Speed:** up to 5,400× faster than competitors, with similarly compact representations
  - **Scalability**: scales to graphs with over 20 billions of edges
  - **Compression**: achieves up to 3.4× additional compression when combined with other advanced graph-compression techniques

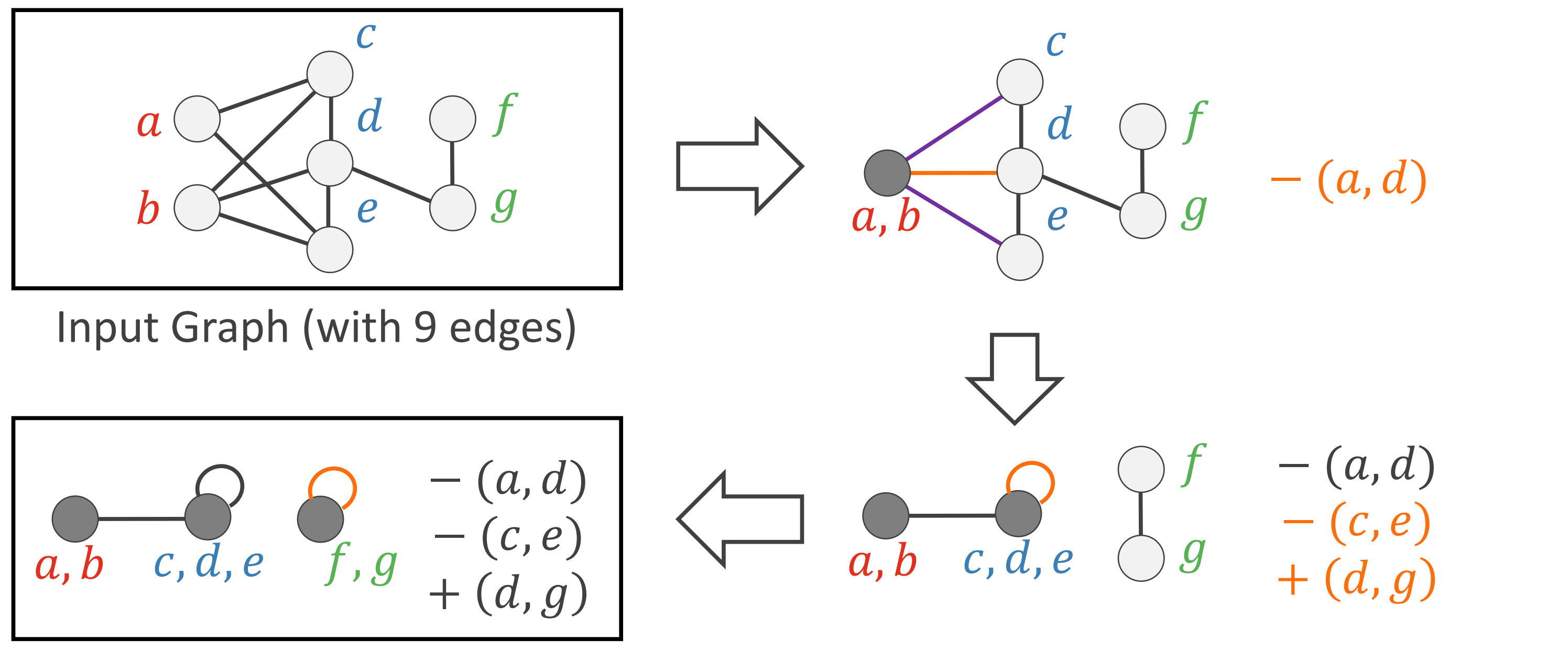
Motivation

- **Graph summarization** is a promising graph-compression technique
- Existing algorithms are not satisfactory in terms of speed and compression rates
- Existing algorithms assume that the input graph is small enough to fit in main memory
- **Question.** *How can we concisely summarize graphs with **tens or hundreds of billions of edges** that are too large to fit in main memory or even on a disk?*



Problem Definition: Graph Summarization

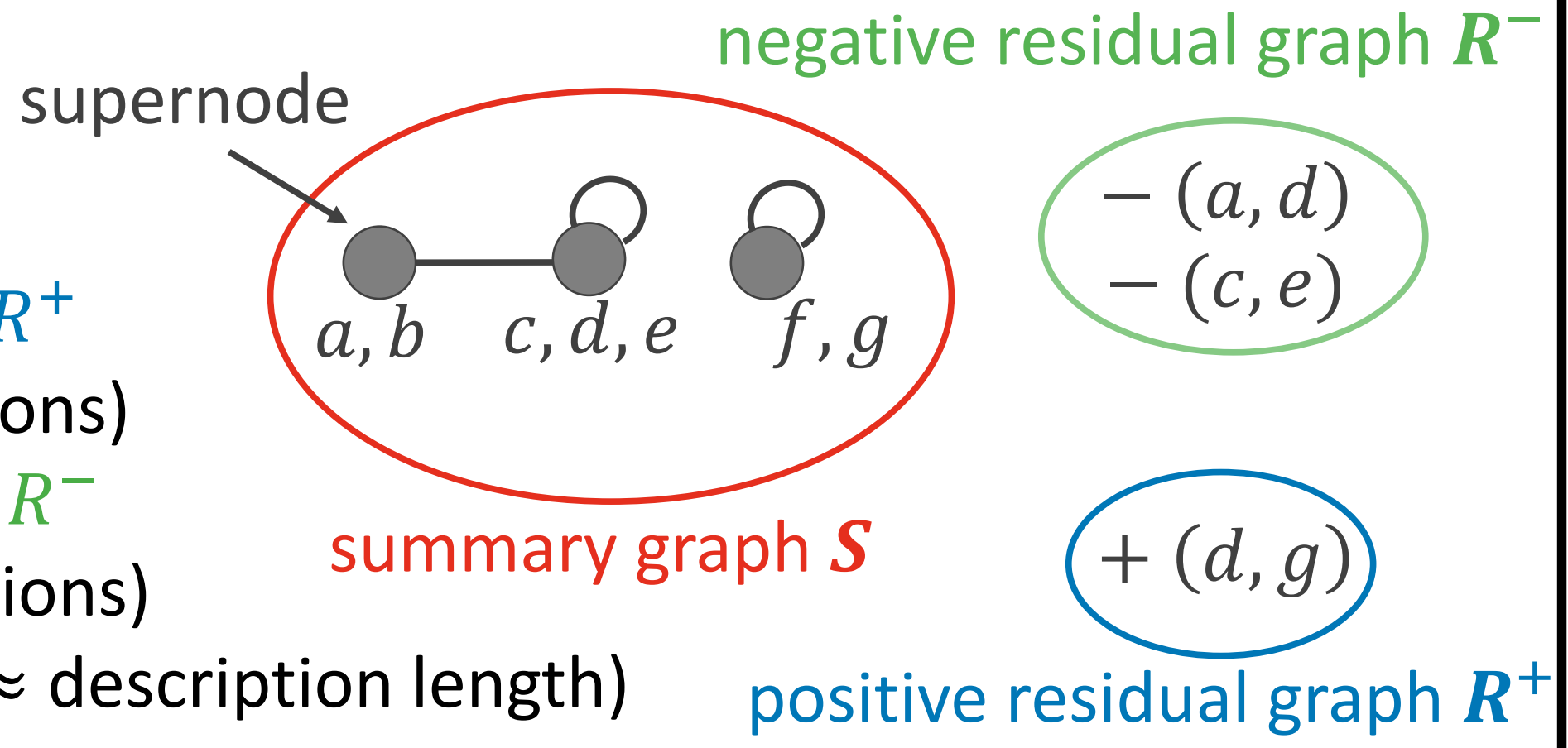
- **Example of graph summarization:** Notice that it is a *lossless* process!



Input Graph (with 9 edges)

Output (with 6 edges)

- **Formal Problem Definition:**
  - **Given:** an input graph
  - **Find:** (a) a summary graph S
    - (b) a positive residual graph R^+ (i.e., positive edge corrections)
    - (c) a negative residual graph R^- (i.e., negative edge corrections)
  - **To Minimize**: sum of edge counts (≈ description length)



- **Why Graph Summarization** (as a graph-compression technique)?
  - supporting efficient **neighbor queries**
  - applicable to **lossy compression**
  - **combinable** with other graph-compression techniques (since the outputs are graphs)

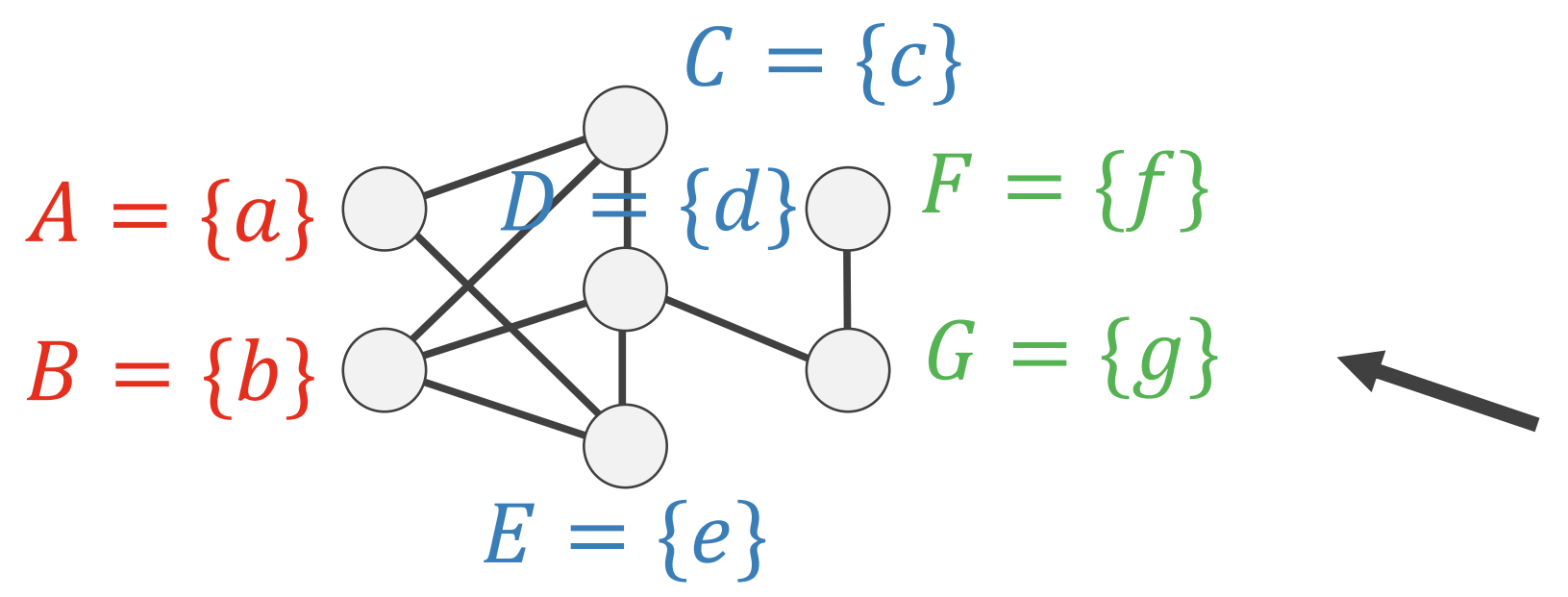
discussed in the paper

Proposed Algorithm: SWeG (Summarizing Web-Scale Graphs)

(1) Initializing Step:

- Initialize the summary and residual graphs

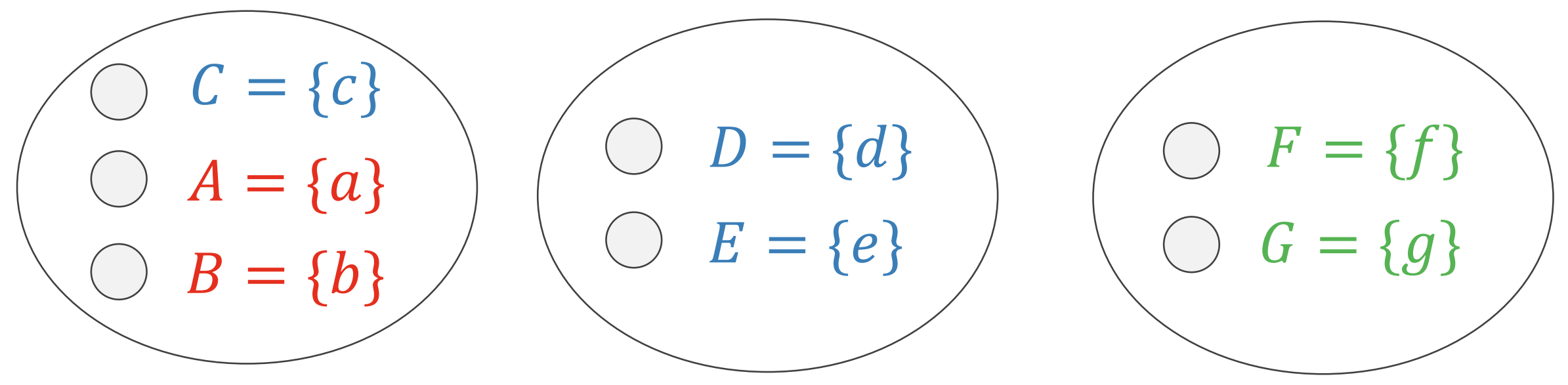
Summary Graph $S = G$ *Residual Graphs* $R^+ = R^- = \emptyset$



each supernode consists of a single subnode

(2) Grouping Step: *"Min Hashing"*

- Divide supernodes into groups of supernodes with similar connectivity

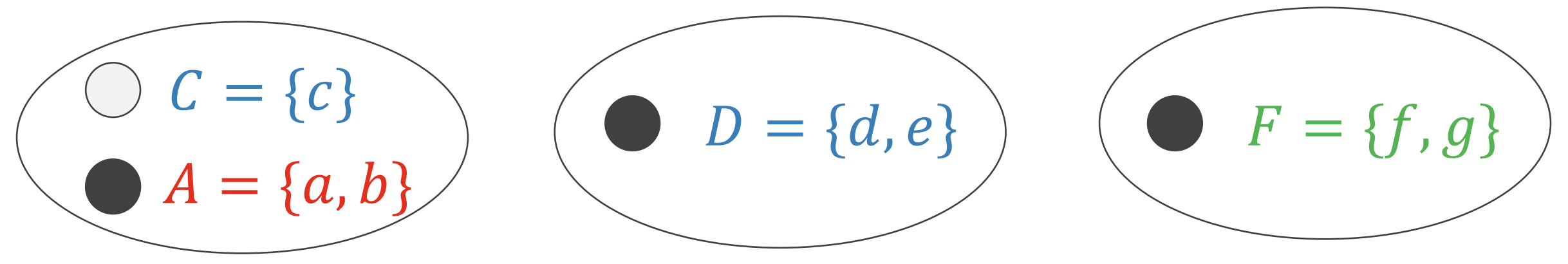


(3) Merging Step: *"Greedy Search"*

- Greedily merge supernodes within each group if $\mathit{Saving} > \theta^{(t)}$

saving in the encoding cost

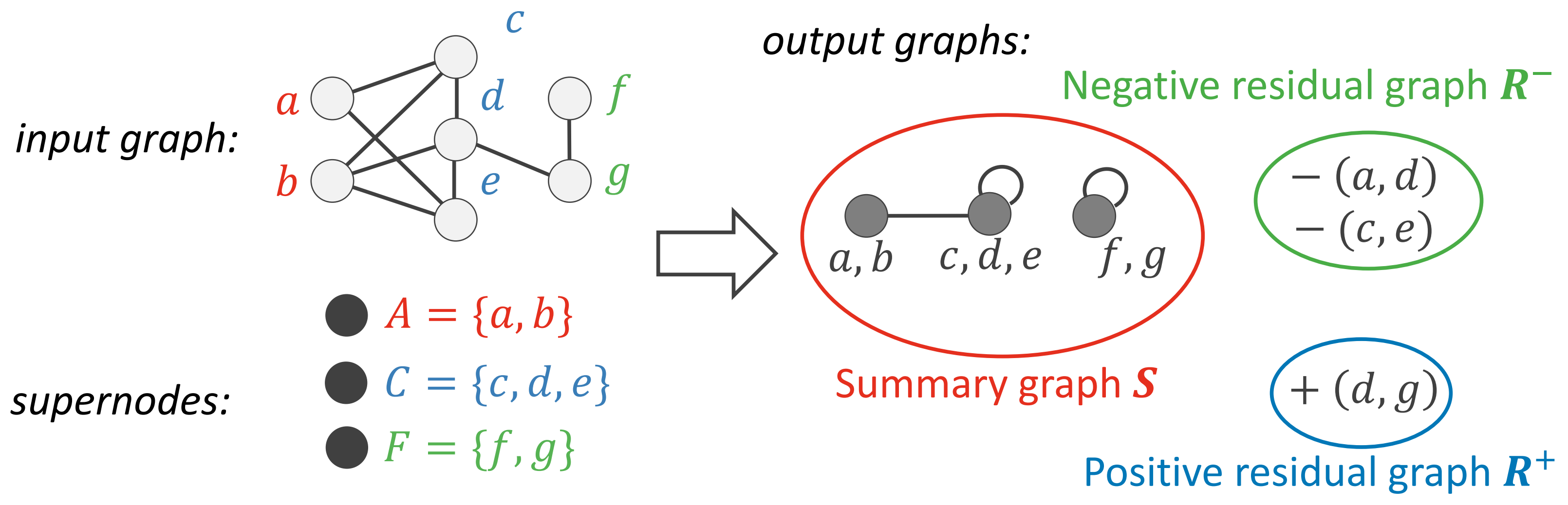
threshold dropping over iterations



Repeat T times

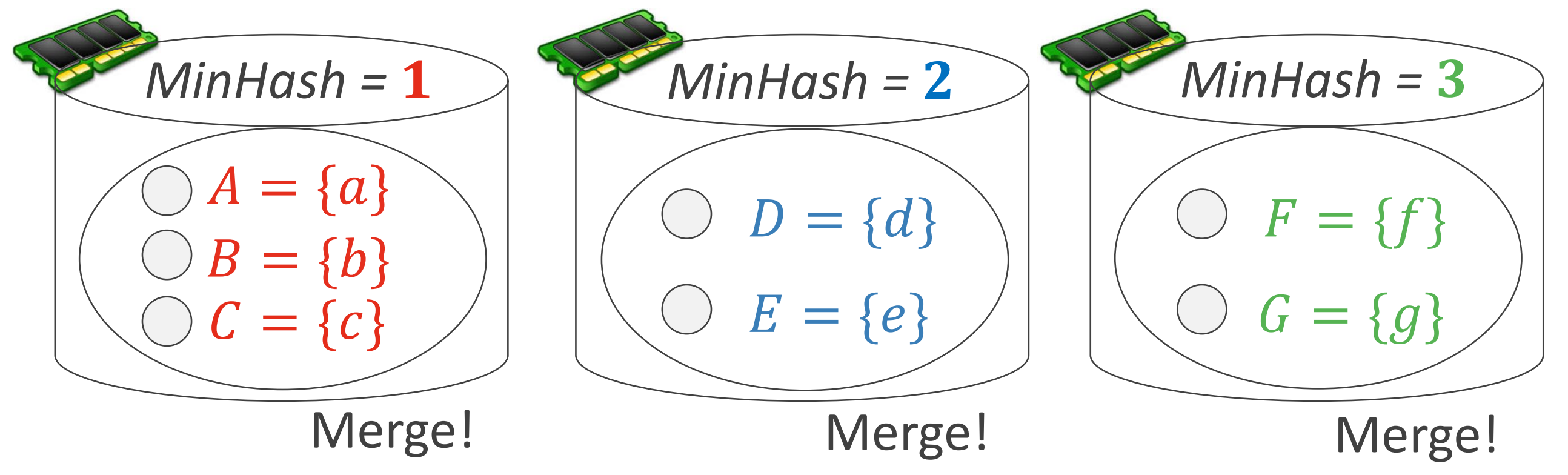
(4) Encoding Step:

- Given supernodes, encode the input graph with minimum (super) edges



Parallel & Distributed Implementation

- **Map Stage:** compute *min hashes* in parallel
- **Shuffle Stage:** group super nodes using min hashes
- **Reduce Stage:** process groups independently in parallel



No need to load the entire graph in memory!

Complexity Analysis

Let the input graph be $G = (V, E)$. For each node group V_i in the grouping step, consider subgraph $G_i = (V_i', E_i)$, which induced from V_i and their neighbors.

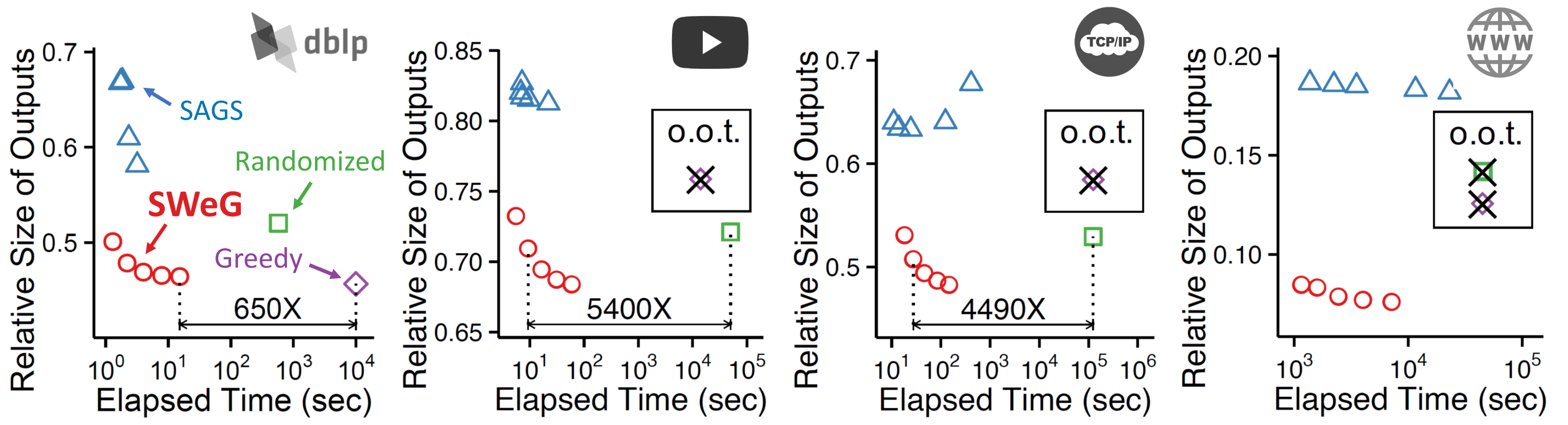
- **Time Complexity:** $O(T \times \sum_i(|V_i| \times |E_i|))$ $(= O(T \times |E|)$ if we divide groups finely)
- **Memory Requirements:** $O(|V| + max_i|E_i|)$

Further Compression: SWeG+

- **Main Idea:** further compress the outputs of SWeG, which are three graphs, using any off-the-shelf graph-compression techniques, such as BFS [4], BP [5], and VNMiner [6].

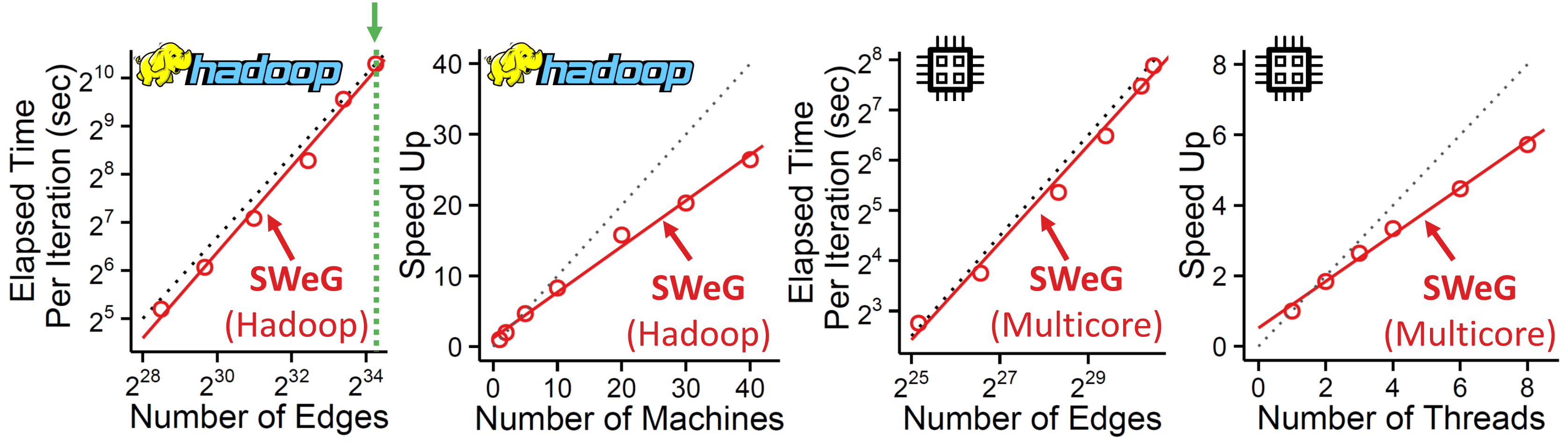
Experimental Results

- **Dataset:** 13 real graphs (w/ 10K – **20B** edges) ...
- **Competitors** (summarization algorithms) : Greedy [1], Randomized [1], and SAGS [3]
- **Q1. Speed and Compression:** *SWeG significantly outperforms its competitors*

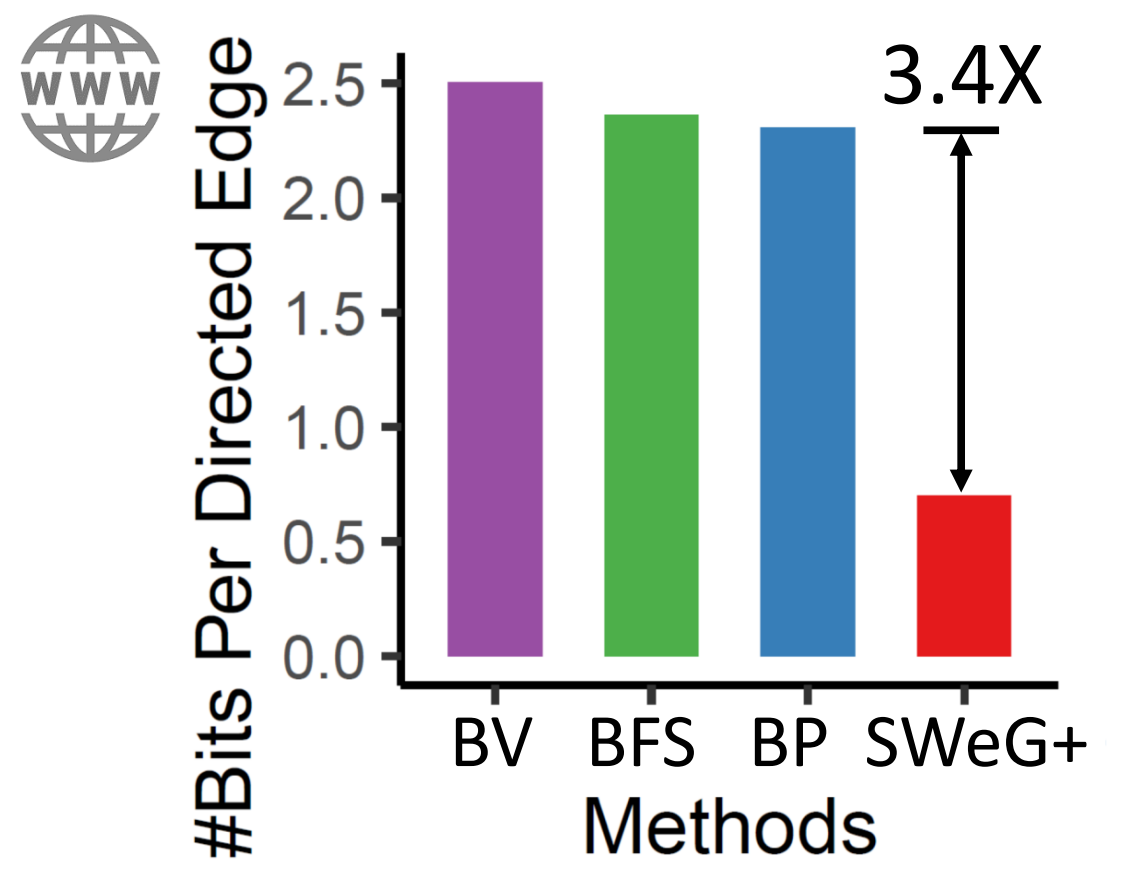


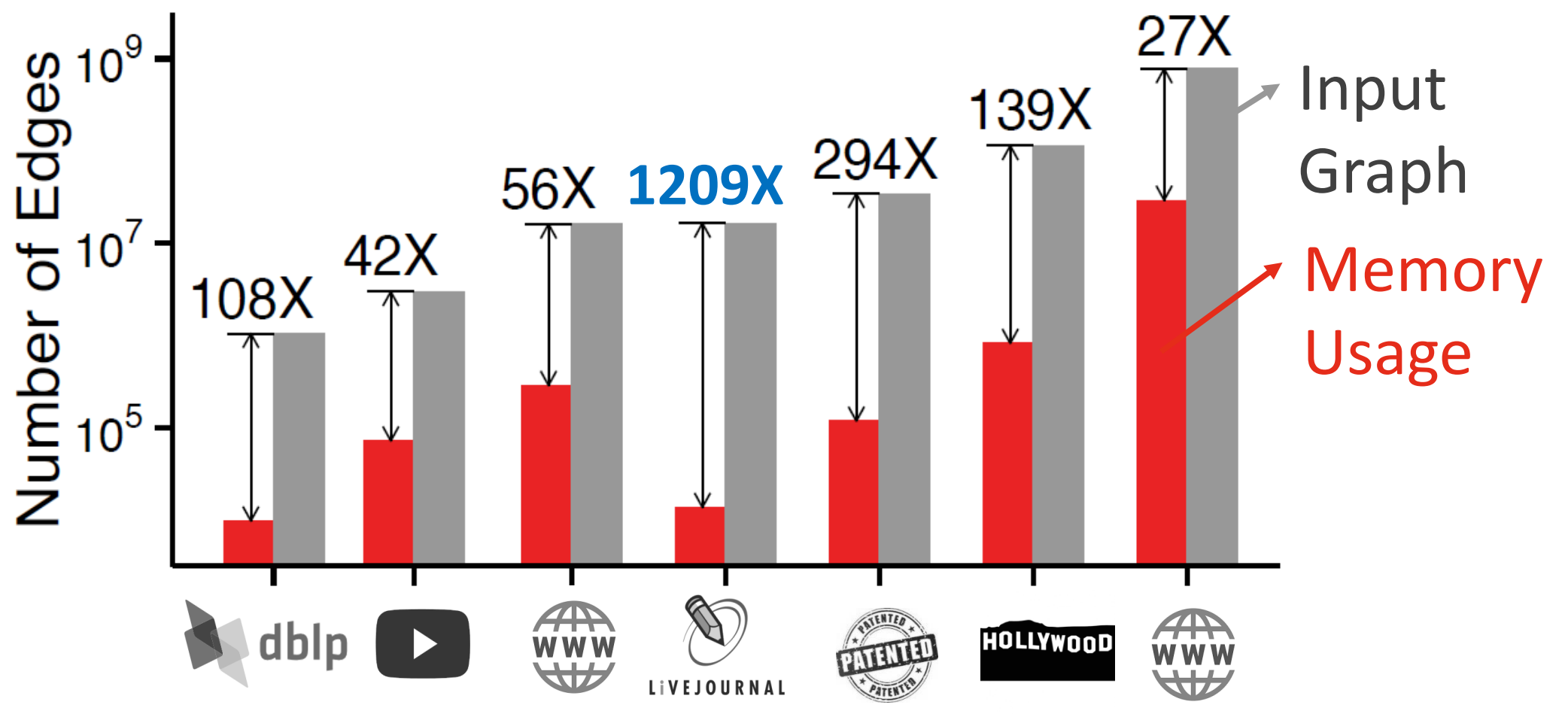
- **Q2. Scalability:** *SWeG is linear in the number of edges & SWeG scales up*

≥ 20 billion edges



- **Q3. Compression:** *SWeG+ achieves unprecedented compression rates*
- **Q4. Memory Requirements:** *SWeG loads* $\leq 0.1 - 4\%$ *of edges in main memory at once*





- **Q5. Effects of Iterations (Figure 7 of the paper):** *~20 iterations (i.e.,* $T = 20$*) are enough for obtaining concise outputs*

References

[1] S. Navlakha, R. Rastogi, N. Shrivastava. *"Graph summarization with bounded error."* In SIGMOD (2008)

[2] D. Koutra, U.Kang, J. Vreeken, C. Faloutsos. *"Vog: Summarizing and understanding large graphs."* In SDM (2014)

[3] K. U. Khan, W. Nawaz, Y Lee. *"Set-based approximate approach for lossless graph summarization."* Computing 97(12):1185-1207 (2015)

[4] A. Apostolico, G. Drovandi. *"Graph compression by BFS."* Algorithms, 2(3):1031-1044 (2009)

[5] L. Dhulipala, I. Kabiljo, B. Karrer, G. Ottaviano, S. Pupyrev, A. Shalita. *"Compressing graphs and indexes with recursive graph bisection."* In KDD (2016)

[6] G. Buehrer, K. Chellapilla, *"A scalable pattern mining approach to web graph compression with communities."* In WSDM (2008)