Enumeration of Billions of Maximal Bicliques in Bipartite Graphs without Using GPUs

Zhe Pan, Shuibing He, Xu Li, Xuechen Zhang*, Yanlong Yin,
Rui Wang, Lidan Shou, Mingli Song, Xian-He Sun#, Gang Chen

Zhejiang University, *Washington State University Vancouver, #Illinois Institute of Technology

SC24 Atlanta, GA | hpc creates.

浙江大學 ZHEJIANG UNIVERSITY WASHINGTON STATE UNIVERSITY ILLINOIS TECH

Outline

- Introduction
  - Problem definition
  - Baseline MBE approach
  - Graph representation in memory
- Motivation
  - Computational subgraph
  - Key insights
- AdaMBE: Adaptive MBE algorithm
  - Redesign of key operations using local neighborhood information
  - Hybrid in-memory representation of computational subgraphs
- Evaluation

Introduction

- Problem definition
- Baseline MBE approach
- Graph representation in memory

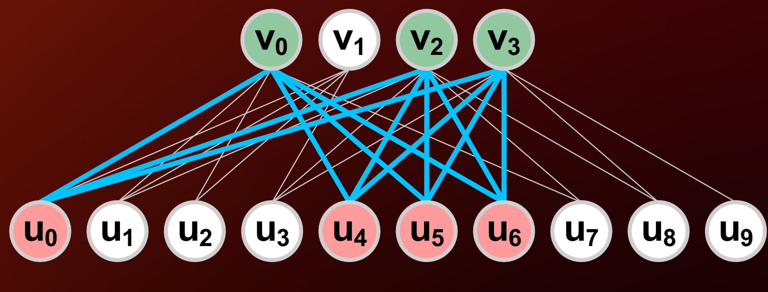
Introduction: Problem Definition

➢ **Preliminaries**

- Bipartite graph $G(U,V,E)$: A graph structure contains two disjoint vertex sets U, V and an edge set E. $E \subseteq U \times V$.
- Biclique : A complete bipartite graph in which every vertex is connected to every vertex in the opposite subset.
- Maximal biclique : a biclique that can not be further enlarged to form a large biclique.

➢ **Problem definition**

- Maximal biclique enumeration (MBE) aims to find all maximal bicliques in G.



An example of a bipartite graph G_0 and a maximal biclique ($\{u_0, u_4, u_5, u_6\}$, $\{v_0, v_2, v_3\}$) in G_0.

Introduction: Baseline MBE Approach

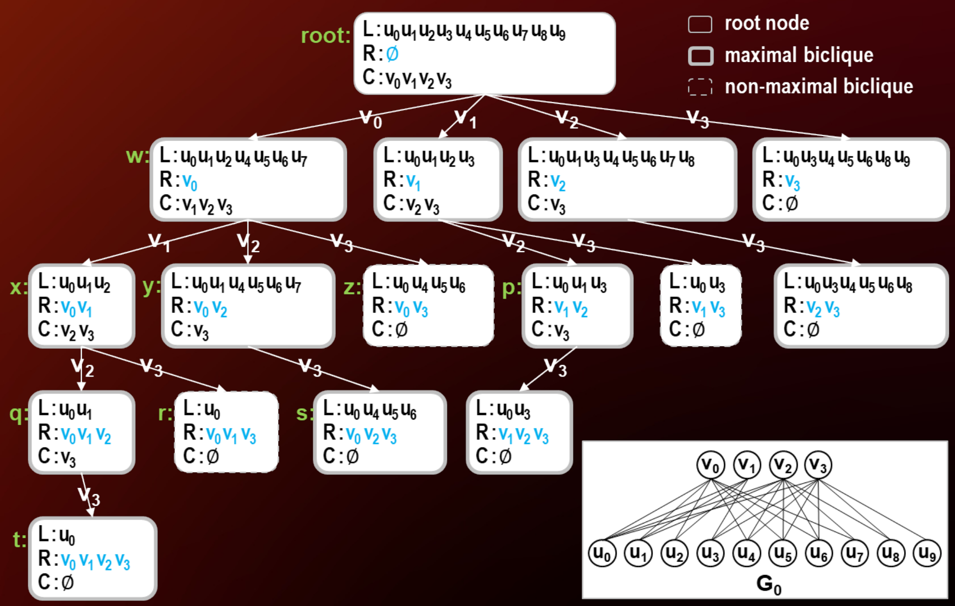
Are all vertices necessary for enumeration?

➢ Set enumeration tree for MBE

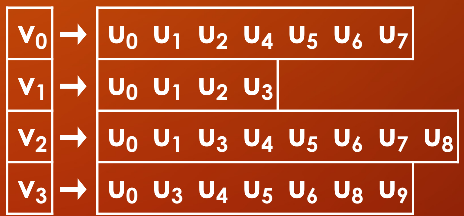
- Each tree node is a 3-tuple (L, R, C). (L, R) is the corresponding biclique and C stores candidate vertices for expanding R.

➢ Baseline approach

- Step 1 : Utilize a set enumeration tree to generate the powerset of V.
- Step 2 : Expand each subset of the powerset of V to a biclique (L, R) and enumerate maximal ones.



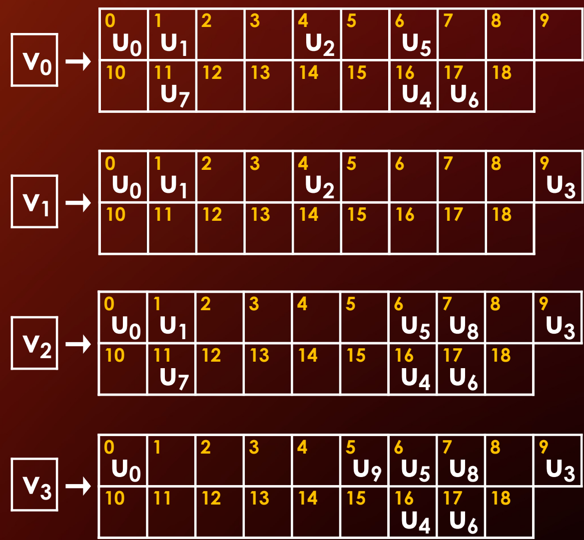
Introduction: Graph Representation in Memory



(a) Adjacency list. Time: $O(\Delta(V))$ Space: $O(|E|)$

	u_0	u_1	u_2	u_3	u_4	u_5	u_6	u_7	u_8	u_9
v_0	1	1	1	0	1	1	1	1	0	0
v_1	1	1	1	1	0	0	0	0	0	0
v_2	1	1	0	1	1	1	1	1	1	0
v_3	1	0	0	1	1	1	1	0	1	1

(b) Bitmap. Time: $O(|U|)$ ($O(1)$ for small graphs) Space: $O(|U| \times |V|)$



Can we combine the advantages of different graph representations?

(c) Hash table. Time: $O(\Delta(V))$ Space: $O(|E|)$

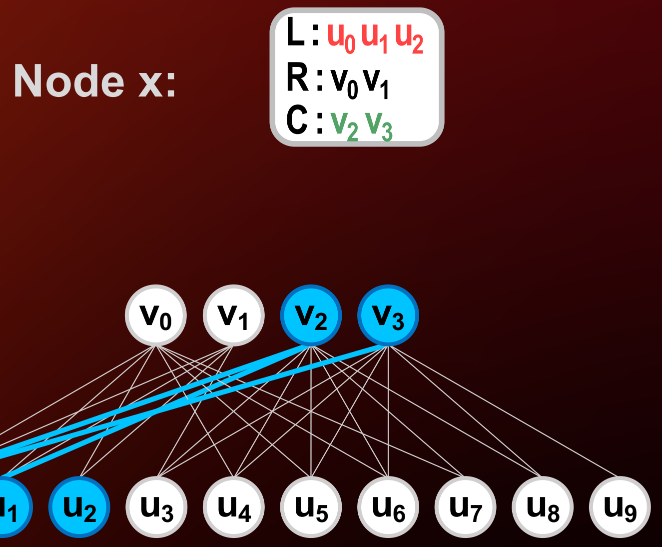
Motivation

- Computational subgraph
- Key insights

Motivation: Computational Subgraph

Computational subgraph (CG): At the current node (L, R, C), the CG in MBE is the subgraph formed by the vertices in LUC, along with all edges between L and C in the original bipartite graph.

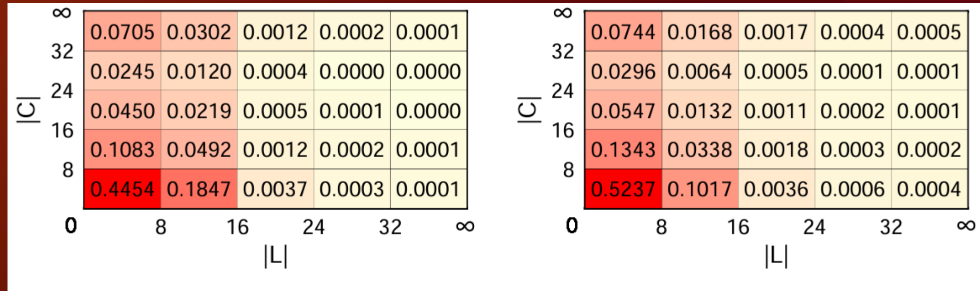
Example: Given a node x, we highlight the corresponding CG of node x in the original bipartite graph G_0 in blue.



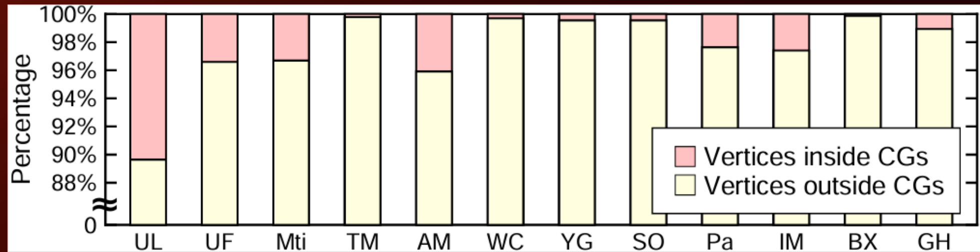
Bipartite graph G_0

Motivation: Key Insights

- Characteristics of CGs
  1. The size of CGs dynamically changes. Most of these CGs are relatively small.
  2. The computational subgraph of the current enumeration node can be directly used for node generation.
  3. Existing algorithms require access to vertices outside their corresponding CGs.
- Limitations of existing works
  1. They typically operate on the original graph, resulting in extensive access to vertices outside CGs.
  2. They commonly utilize the adjacency list as the default choice for representing graphs.



Distribution of CG sizes based on |L| and |C|.



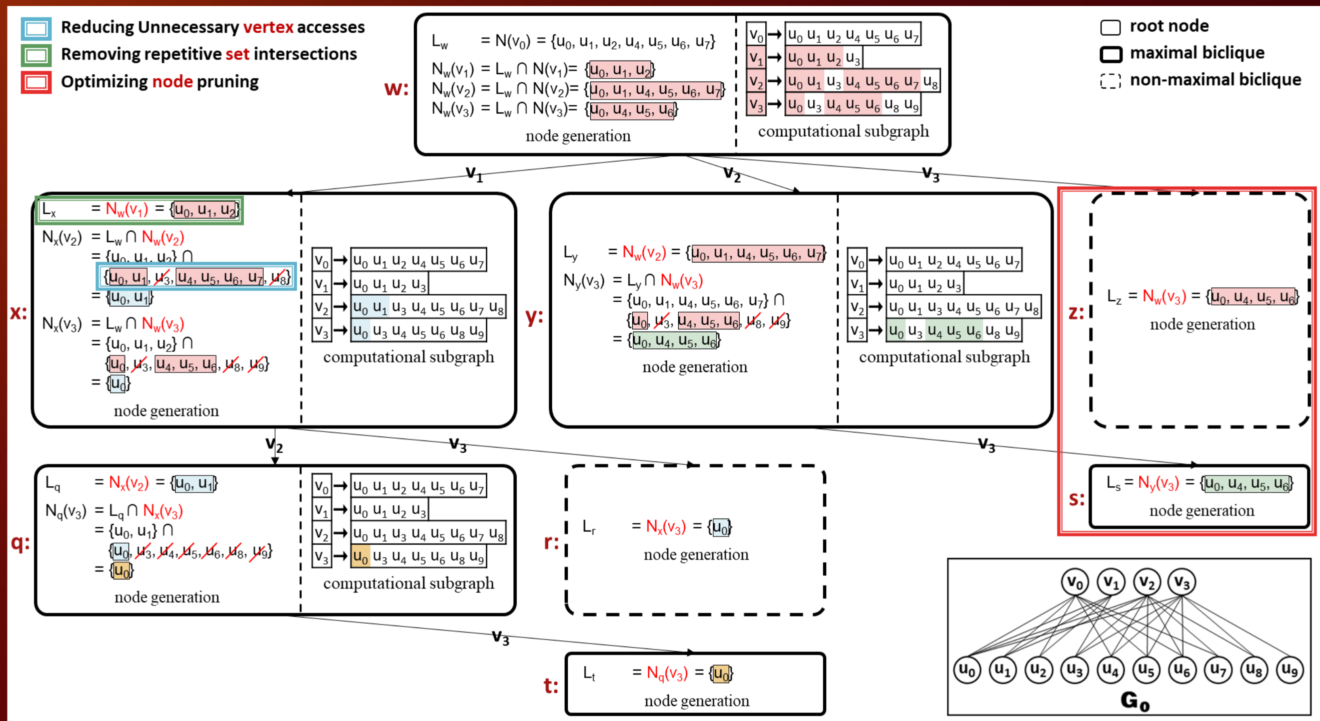
Percentage of vertices inside and outside CGs on real-world datasets.

AdaMBE: Adaptive MBE Algorithm

- Redesign of key operations using local neighborhood information
- Hybrid in-memory representation of computational subgraphs

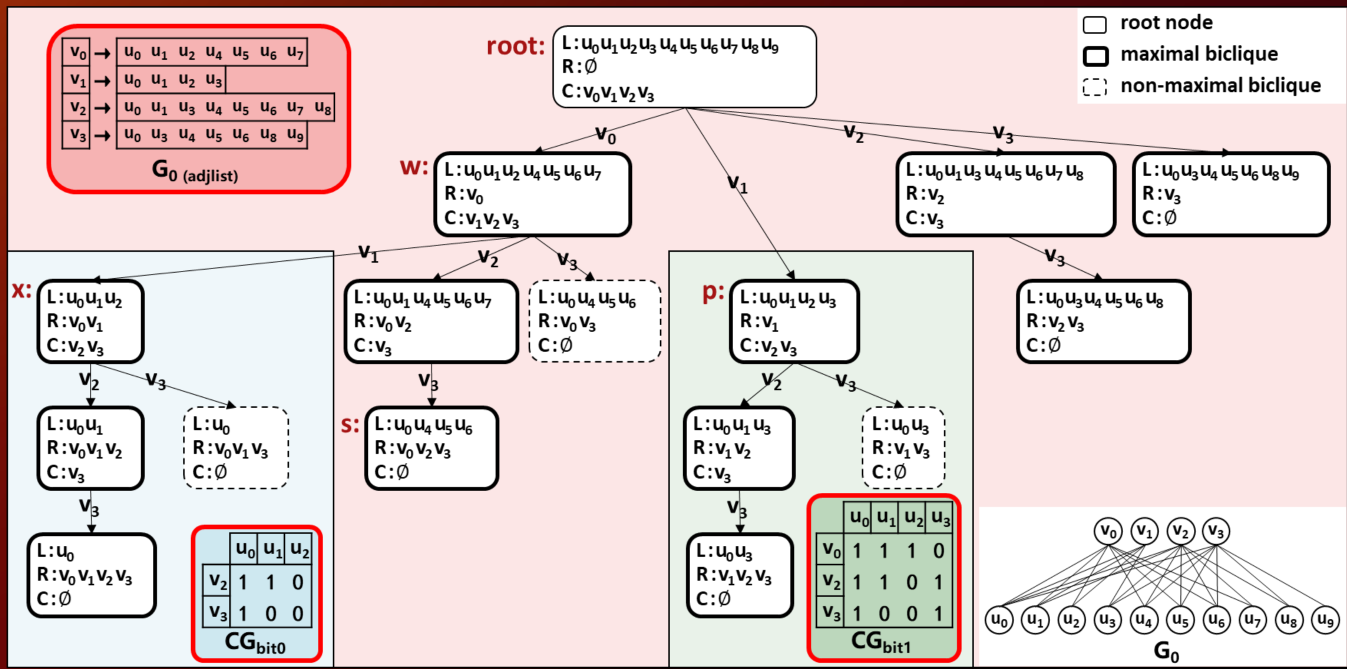
AdaMBE: Redesign of Key Operations Using Local Neighborhood Information

- Local neighbors
  - Neighbors of vertex v in the current CG.
- Main idea
  - We use local neighbors to redesign key operations to reduce unnecessary vertex accesses, repetitive set intersections, and unproductive tree nodes at the same time.



AdaMBE: Hybrid in-memory representation of computational subgraphs

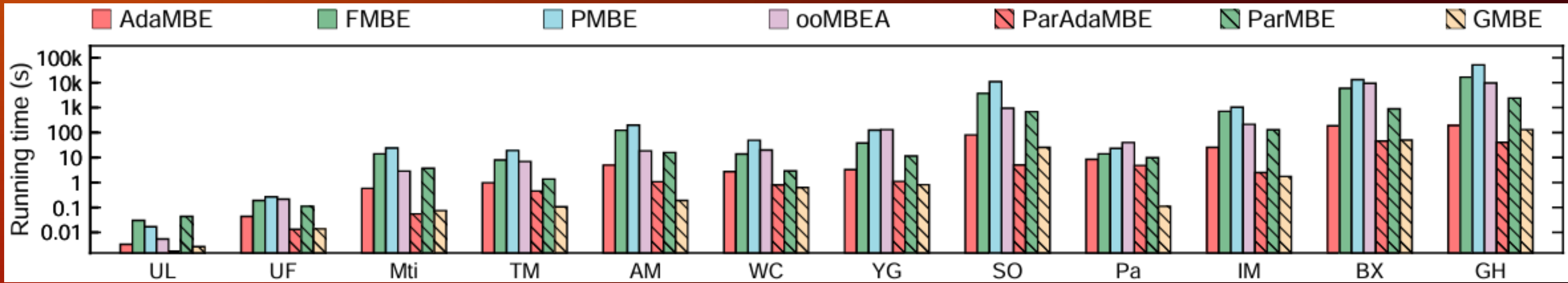
- Main idea
  - For large CGs, we use the adjacency list for its memory efficiency
  - For small CGs, we use the bitmap to boost computational efficiency.



Evaluation

- Overall evaluation
- Breakdown analysis
- Sensitivity analysis

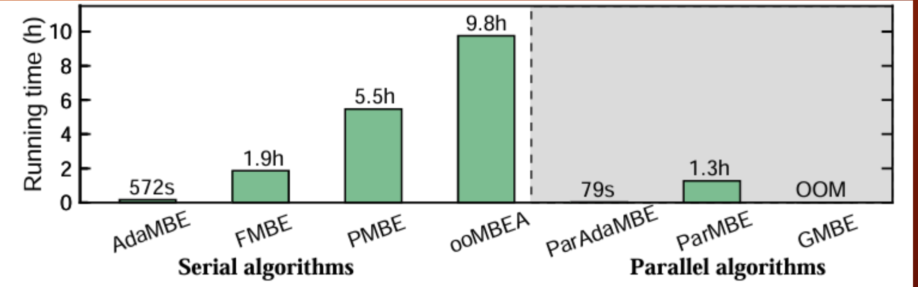
Evaluation: Overall Evaluation



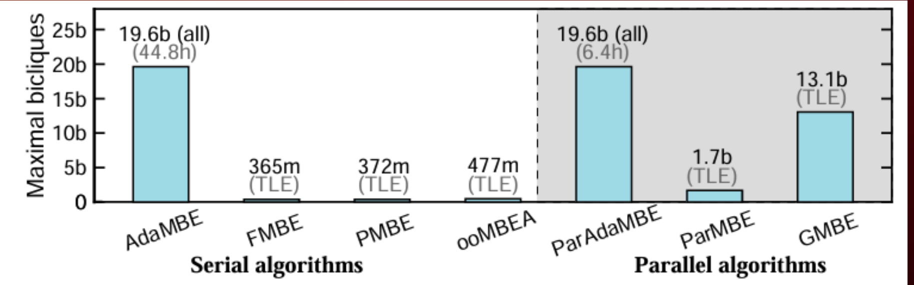
Running time evaluation on general datasets (log scale). Parallel MBE algorithms are indicated by diagonal lines.

- Our AdaMBE outperforms all other serial competitors by 1.6x-49.7x across all datasets.
- Our parallel ParAdaMBE is 1.3x-33.7x faster than the CPU-based ParMBE on all datasets.
- ParAdaMBE on a 96-core CPU is up to 5.07x faster than GMBE on an A100 GPU on time-consuming datasets like StackOverflow, BookCrossing, and GitHub, .

Evaluation: Overall Evaluation



(a) Evaluation on CebWiki. We report the running time of all competitors excluding GMBE, which runs out of memory (OOM).

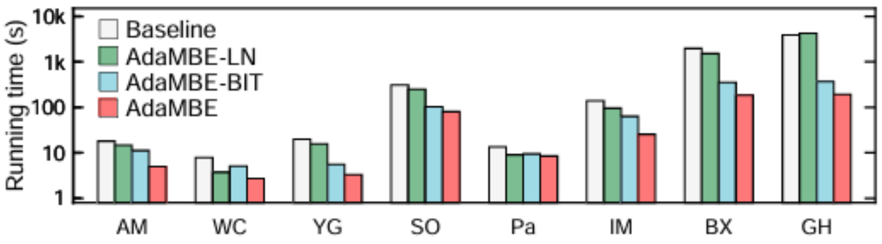


(b) Evaluation on TVTropes. We report the maximal biclique count after 48 hours. The acronym "TLE" stands for "Time Limit Exceeded."

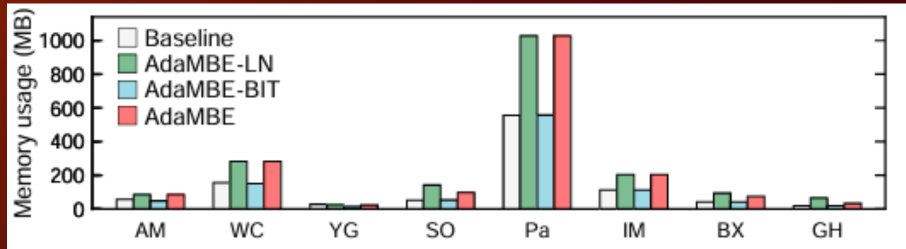
Overall evaluation on two large datasets.

- On the CebWiki dataset, AdaMBE and ParMBE complete in 572 and 79 seconds, respectively, while all other competitors take several hours.
- On the TVTropes dataset, only AdaMBE and ParMBE can enumerate all 19.6 billion maximal bicliques within 48 hours.

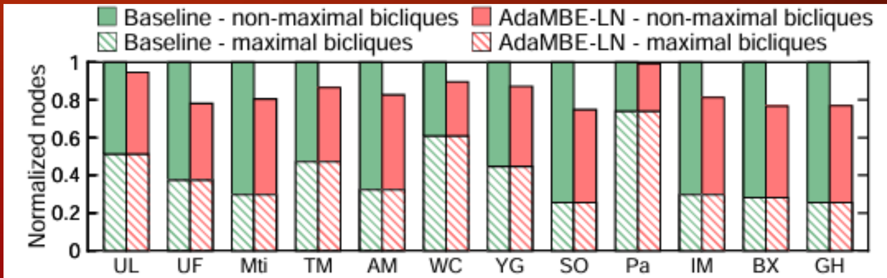
Evaluation: Breakdown Analysis



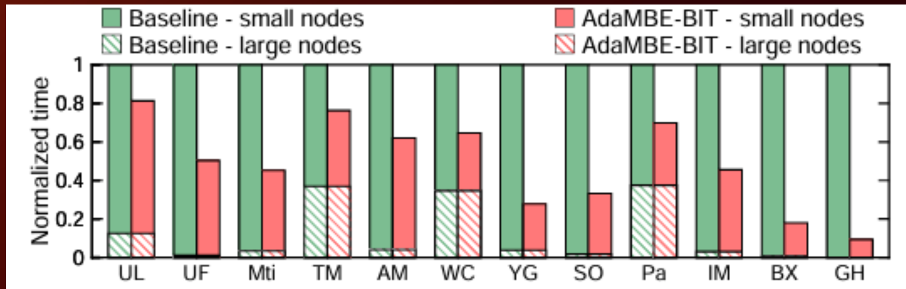
(a) Running time (log scaled).



(b) Memory usage.



(c) Node pruning efficiency of LN. AdaMBE-LN efficiently reduces the nodes with non-maximal bicliques.



(d) Time breakdown of BIT. Small nodes represent the nodes with $|L|$ smaller than τ. AdaMBE-BIT reduces the running time of these nodes.

Both local-neighbor-based optimizations (LN) and the hybrid in-memory bitmap representation (BIT) enhance AdaMBE's performance.

Evaluation: Sensitivity Analysis

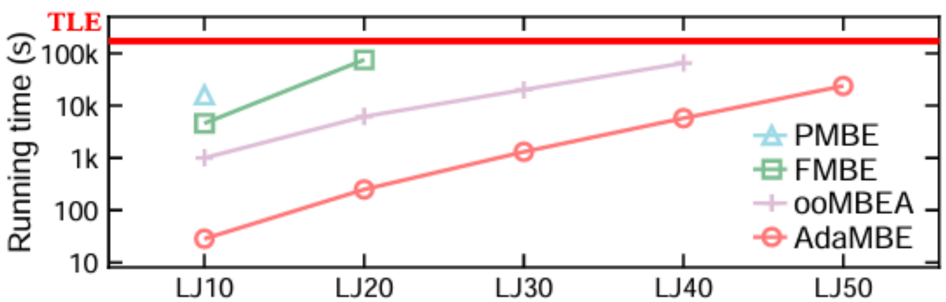


Fig. 13. Impact of dataset size (log scaled).

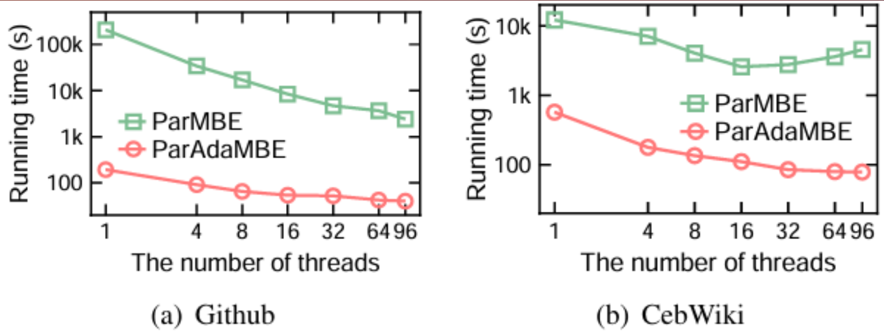


Fig. 14. Impact of number of threads (log scaled).

- AdaMBE excels over all serial competitors on large synthetic datasets with billions of maximal bicliques.
- ParAdaMBE consistently outperforms ParMBE across all thread configurations.

Q & A

Open source: https://github.com/ISCS-ZJU/AdaMBE

Contact information: panzhe@zju.edu.cn

SC24 Atlanta, GA | hpc creates.