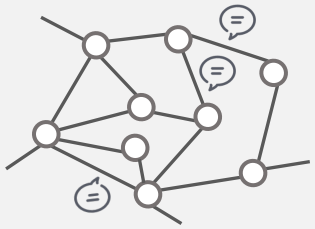
Revisiting Graph Analytics Benchmarks: Unveiling the Practicality of Graph Analytics Platforms

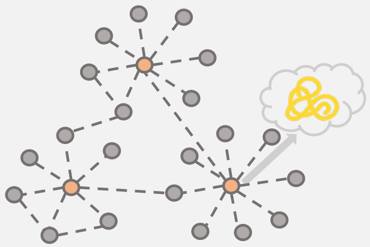
Long Yuan

Graph are everywhere

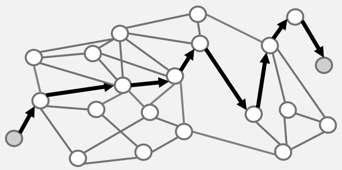
- Social Network, Biological Network, Road Network, E-Commerce, Web, Scientific domains ……
- Interest in graph analytics continues to increase
- Many different graph processing platforms are proposed, e.g., GraphX, PowerGraph, Flash, Grape, Pregel+, Ligra, etc.



Social Network Analysis



Biological Network Analysis



Road Network Routing

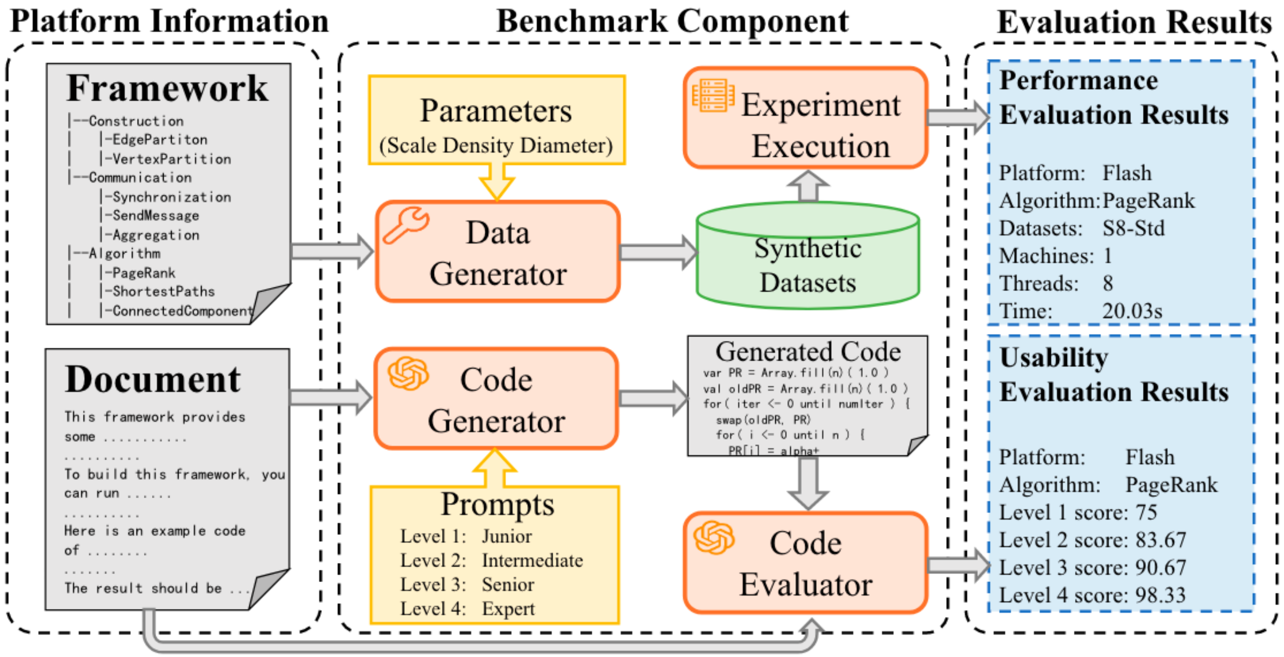
Graph Benchmark

- **Graph Analytics Benchmarks**, like Graph500 and LDBC have played significant roles in evaluating graph analytics platforms and giving suggestions on selecting proper ones.

- But, there are still some **limitations**:
  - The core algorithm set lacks diversity, typically selecting only simple algorithms or those frequently used in research papers.
  - Existing graph data generators focus on vertex and edge counts but overlook other crucial properties like diameter and density.
  - Evaluation metrics emphasize objective performance, while neglecting usability from the users' perspective.

A new graph analytics benchmark

- Select eight algorithms as the core algorithm set: PageRank, Label Propagation Algorithm, SSSP, etc.
- A new data generator: Hop Distance Generator
- LLM-based usability evaluation framework



An overview of our benchmark

The Core Algorithm Set

In our previous work, we have surveyed and analyzed the main challenges of various graph algorithms in distributed environments, including **parallelism, load balance, communication overhead, and bandwidth**, and categorized them into seven topics based on the challenges they address.

Centrality	PageRank, Personalized PageRank, Betweenness Centrality, Closeness Centrality
Community Detection	Louvain, Label Propagation, Connected Components
Similarity	Jaccard Similarity, Cosine Similarity, SimRank
Cohesive Subgraph	*k*-Core, *k*-Truss, Maximal Clique
Traversal	BFS, Single Source Shortest Path, Minimum Spanning Tree, Cycle Detection, Maximum Flow
Pattern Matching	Triangle Counting, *k*-Clique, Subgraph Matching, Subgraph Mining
Covering	Minimum Vertex Covering, Maximum Matching, Graph Coloring

Lingkai Meng, Yu Shao, Long Yuan, Longbin Lai, Peng Cheng, Xue Li, Wenyuan Yu, Wenjie Zhang, Xuemin Lin, and Jingren Zhou. 2024. A Survey of Distributed Graph Algorithms on Massive Graphs. arXiv:2404.06037 [cs.DC] https://arxiv. org/abs/2404.06037

The Core Algorithm Set

We considered the following factors: **(1) Topic Diversity and Coverage; (2) Breadth and Popularity; (3) Computation Workload and Paradigm.**

Algorithms	Workload	Paradigm
PR	$O(k \cdot m)$	Iterative
LPA	$O(k \cdot m)$	Iterative
SSSP	$O(m + n \cdot \log n)$	Sequential
CC	$O(m + n)$	Sequential
BC	$O(n^3)$	Sequential
CD	$O(m + n)$	Sequential
TC	$O(m^{1.5})$	Computative
KC	$O(k^2 \cdot n^k)$	Computative

- PageRank (PR)
- Label Propagation Algorithm (LPA)
- Single Source Shortest Path (SSSP)
- Connected Component (CC)
- Betweenness Centrality (BC)
- Core Decomposition (CD)
- Triangle Counting (TC)
- k-Clique (KC)

Lingkai Meng, Yu Shao, Long Yuan, Longbin Lai, Peng Cheng, Xue Li, Wenyuan Yu, Wenjie Zhang, Xuemin Lin, and Jingren Zhou. 2024. A Survey of Distributed Graph Algorithms on Massive Graphs. arXiv:2404.06037 [cs.DC] https://arxiv. org/abs/2404.06037

Data Generator-Distance Hop Generator

- Extract resulting edges directly for avoiding the incidence of sampling failure.
- Adopt a tuning factor, α, during the sampling step to foster the creation of closed edges to improve the efficiency of sparse graph generation.

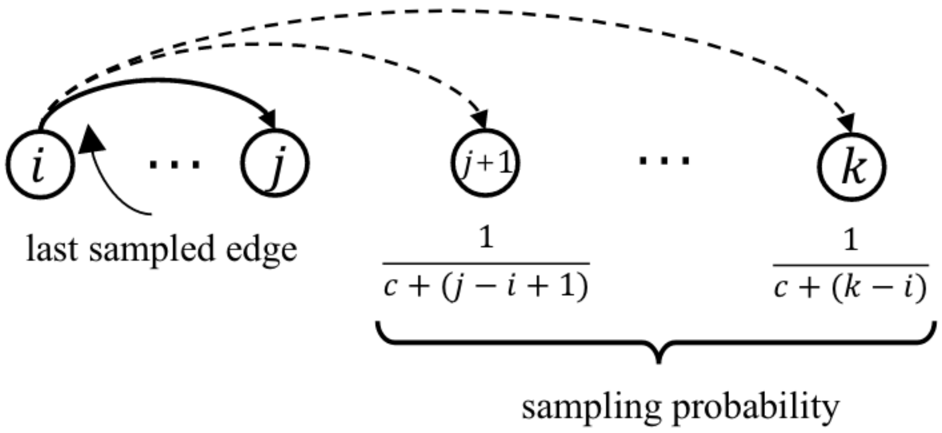


Figure: The sampling process resumes from the point at which the last edge was successfully sampled.

Data Generator-Distance Hop Generator

- Extract resulting edges directly for avoiding the incidence of sampling failure.
- Adopt a tuning factor, α, during the sampling step to foster the creation of closed edges to improve the efficiency of sparse graph generation.
- Restrict the span of each edge, organize the vertices into groups, and generate edges within each group to maintain a consistent average diameter.

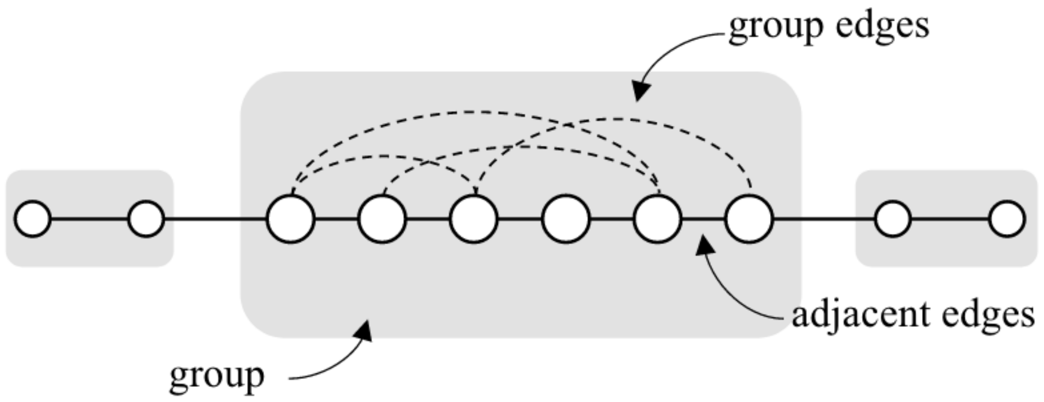


Figure: Generating graphs with adjustable diameters

Data Generator-Distance Hop Generator

- Extract resulting edges directly for avoiding the incidence of sampling failure.
- Adopt a tuning factor, α, during the sampling step to foster the creation of closed edges to improve the efficiency of sparse graph generation.
- Restrict the span of each edge, organize the vertices into groups, and generate edges within each group to maintain a consistent average diameter.

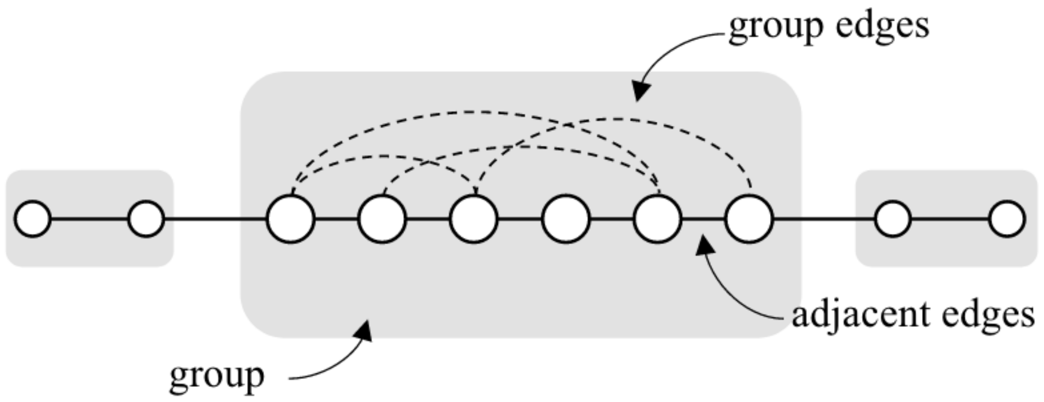
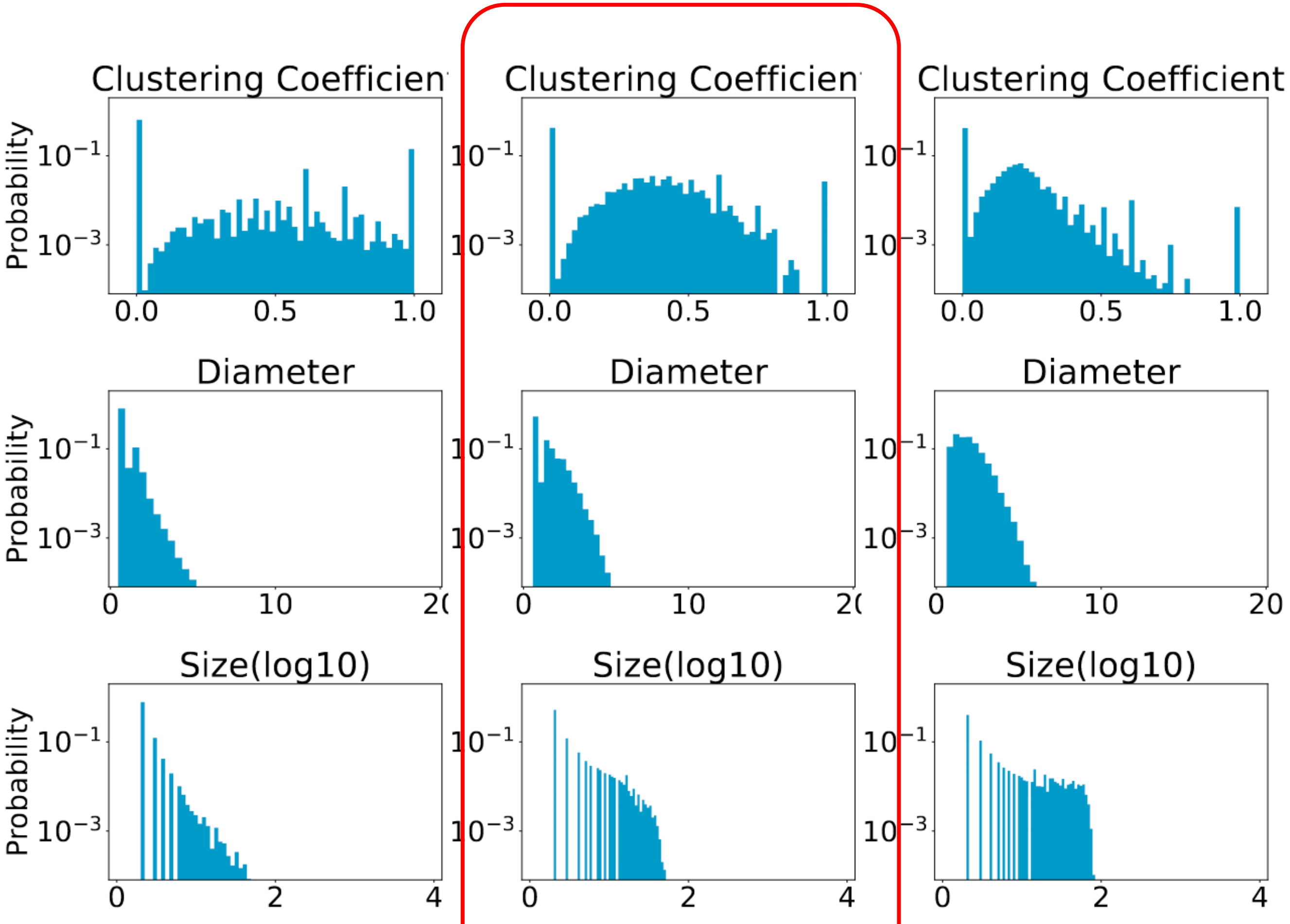


Figure: Generating graphs with adjustable diameters

Scale   Density   Diameter

Data Generator-Distance Hop Generator

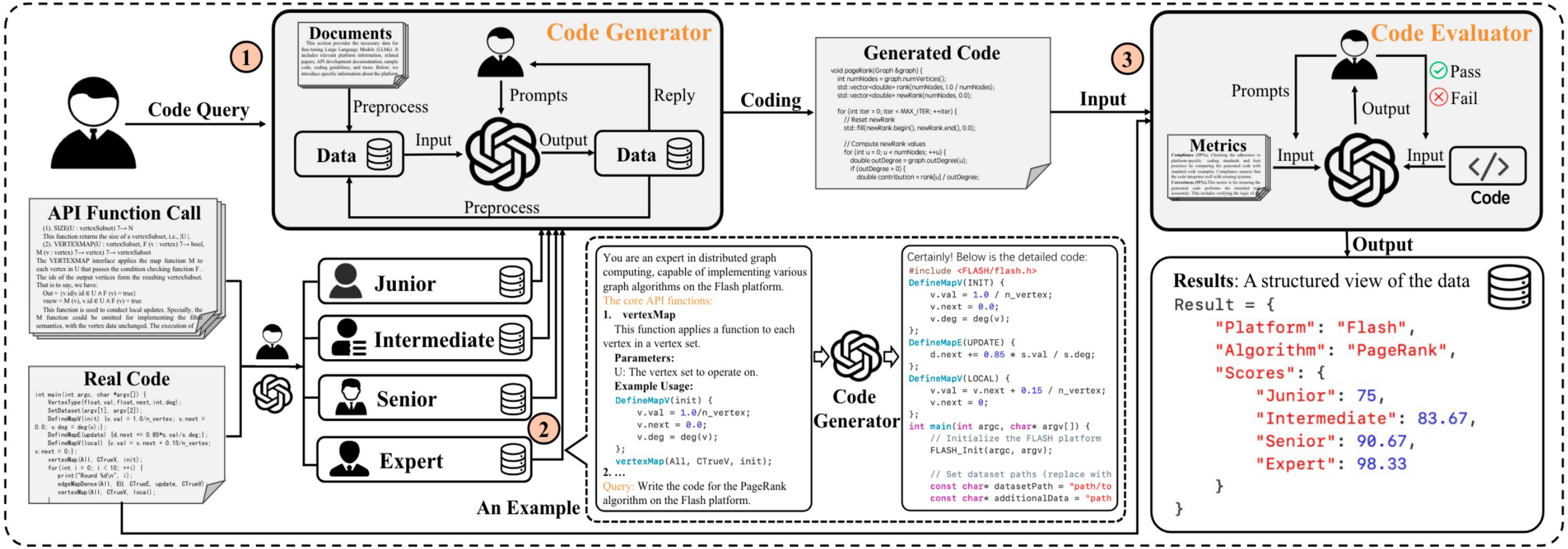


(a) LiveJournal Dataset. (b) Our Generator. (c) LDBC Generator.

A multi-Dimensional Evaluation Framework

LLM-Based Usability Evaluation Framework

- Step 1: Instruction-Tuning of LLMs.
- Step 2: Multi-Level Prompts.
- Step 3: Code Evaluation.



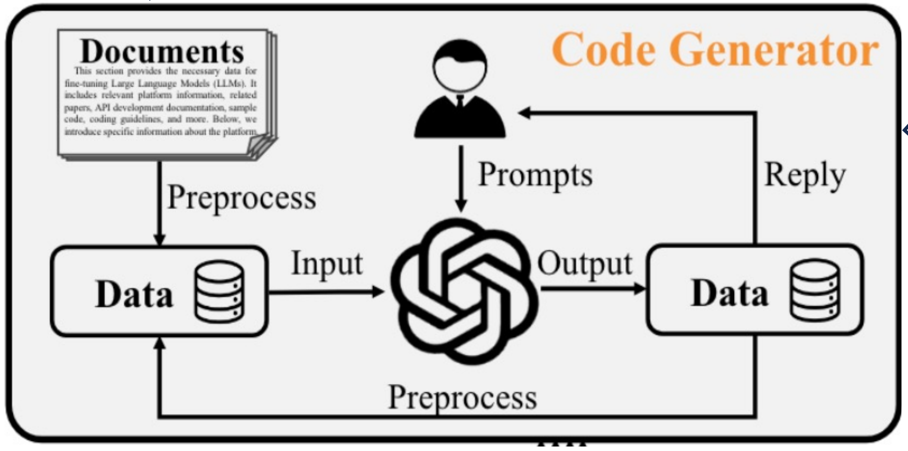
A multi-Dimensional Evaluation Framework

LLM-Based Usability Evaluation Framework

- **Step 1: Instruction-Tuning of LLMs.**
- Step 2: Multi-Level Prompts.
- Step 3: Code Evaluation.

1 Gather data from each platform, including research papers, API documentation, sample codes, and coding guidelines.

2 Preprocess and structure the data for LLM recognition and processing.



4 Human evaluators provide feedback and prompts on generated results.

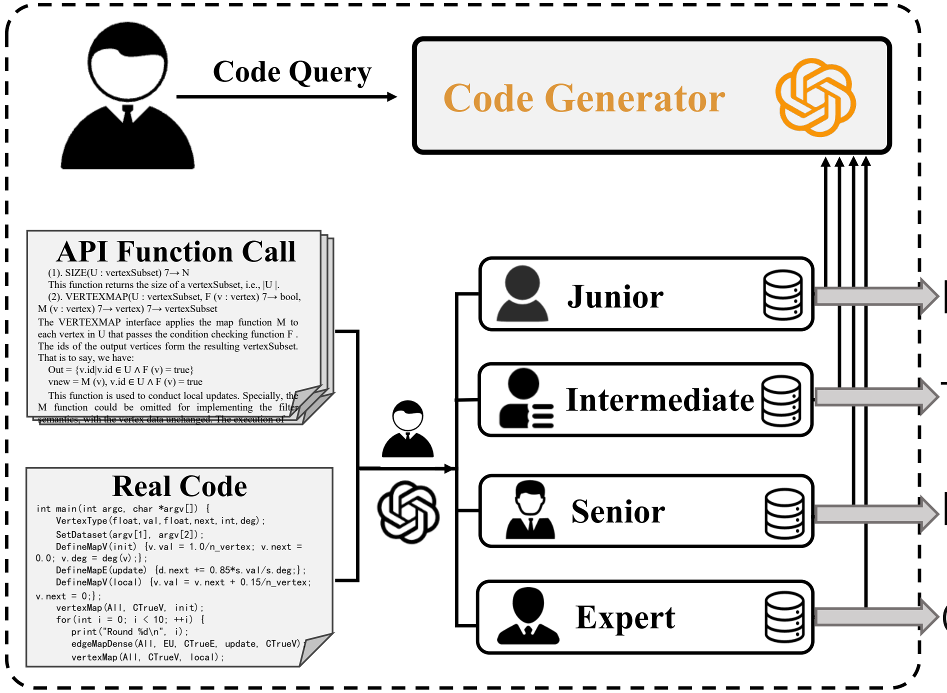
3 Feed the structured data into LLMs to learn specific coding styles and conventions.

5 Use feedback to refine the data and improve the instruction-tuning process.

A multi-Dimensional Evaluation Framework

LLM-Based Usability Evaluation Framework

- Step 1: Instruction-Tuning of LLMs.
- **Step 2: Multi-Level Prompts.**
- Step 3: Code Evaluation.



Description of the task

The names of core APIs and parameters

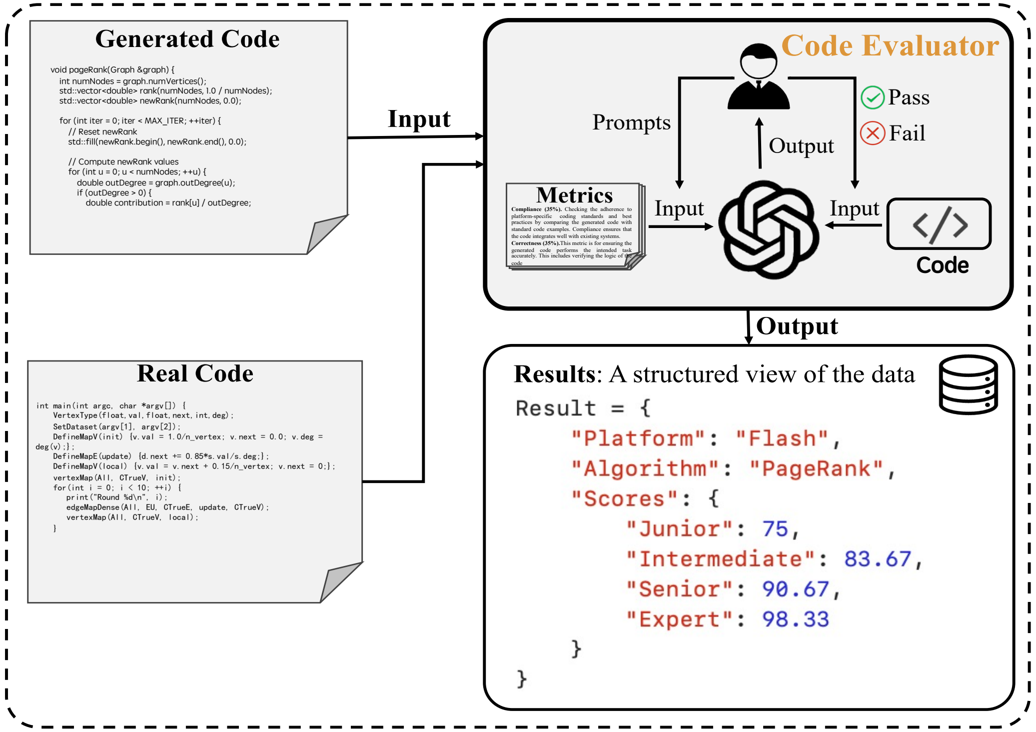
Detailed usage instructions for the relevant APIs and example code

On the basis of the previous prompt, add the pseudocode

A multi-Dimensional Evaluation Framework

LLM-Based Usability Evaluation Framework

- Step 1: Instruction-Tuning of LLMs.
- Step 2: Multi-Level Prompts.
- **Step 3: Code Evaluation.**



Compliance (35%)

Correctness (35%)

Readability (30%)

A multi-Dimensional Evaluation Framework

Performance evaluation metrics

Category	Metric	Description
Timing	**Upload Time**	Time required to read, convert, partition, and load graph data into memory.
	Running Time	Total time required to complete an algorithm execution task.
	Makespan	Overall time for graph operations, including reading, processing, and writing data.
Throughput	**Edges/sec**	Number of edges processed per second.
	Edges+Vertices/sec	Combined number of edges and vertices processed per second.
Scalability	**Speedup**	Rate of performance improvement with additional computational resources.
Robustness	**Stress Test**	Platform's stability and reliability under high-stress conditions.

Experiments

Experimental Setup

Platforms: GraphX, PowerGraph, Flash, Grape, Pregel+, Ligra

Hardware Information:

Hardware	Information
Cluster	$16 \times$ Machines
CPUs	$4 \times$ Intel® Xeon® Platinum 8163 @ 2.50GHz
Cores	4×24
Memory	512 GB
Disk	3 TB
Network	15 Gbps

Experiments

Experimental Setup

Platforms: GraphX, PowerGraph, Flash, Grape, Pregel+, Ligra

Hardware Information:

Selected Synthetic Datasets:

Datasets	n	m	Density	Diameter
S8-Std	3.6M	153M	2.4×10^{-5}	6
S8-Dense	1.2M	159M	2.2×10^{-4}	5
S8-Diam	3.6M	155M	2.4×10^{-5}	101
S9-Std	27.2M	1.42B	3.8×10^{-6}	6
S9-Dense	9.1M	1.47B	3.6×10^{-5}	5
S9-Diam	27.2M	1.48B	4.0×10^{-6}	102
S9.5-Std	77M	4.36B	1.5×10^{-6}	6
S10-Std	210M	12.62B	5.7×10^{-7}	6

Experiments

Experimental Setup

Platforms: GraphX, PowerGraph, Flash, Grape, Pregel+, Ligra

Hardware Information:

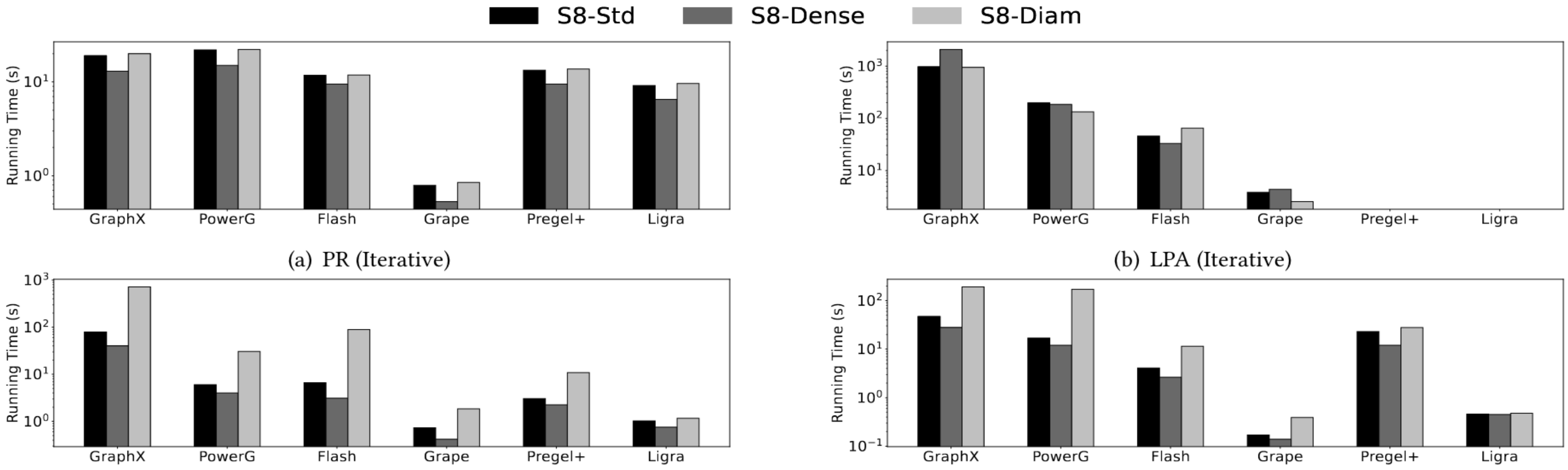
Selected Synthetic Datasets:

Experimental Methodology:

Aspects	Section	Algorithms	Datasets	#threads	#machines
Algorithm Impact Statistics Impact	Section 7.1	All	S8-Std, S8-Dense, S8-Diam	32	1
Scalability Sensitivity	Section 7.2	PR, SSSP, TC	S8-Std, S8-Dense, S8-Diam	1, 2, 4, 8, 16, 32	1
			S9-Std, S9-Dense, S9-Diam	32	1, 2, 4, 8, 16
Throughput	Section 7.3	PR, SSSP, TC	S8-Std, S8-Dense, S8-Diam, S9-Std, S9-Dense, S9-Diam	32	16
Stress Test	Section 7.4	PR	S8-Std, S9-Std, S9.5-Std, S10-Std	32	16
Usability Evaluation	Section 7.5	All	—	—	—

Experiments

Algorithm & Statistics Impact



(a) PR (Iterative)

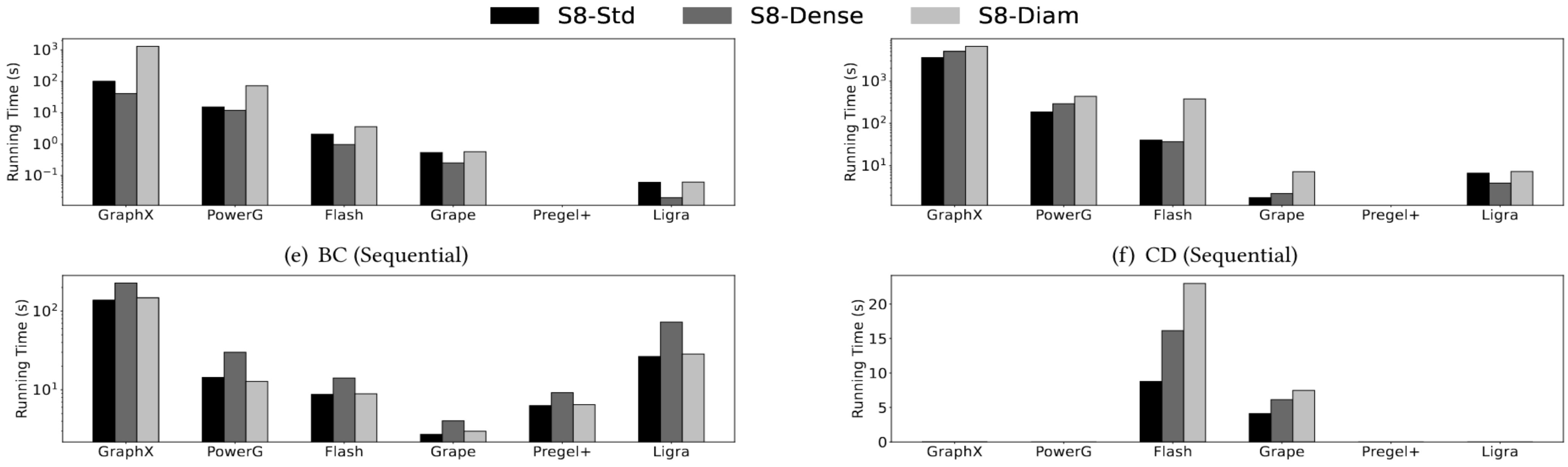
(b) LPA (Iterative)

(c) SSSP (Sequential)

(d) CC (Sequential)

Experiments

Algorithm & Statistics Impact



(e) BC (Sequential)

(f) CD (Sequential)

(g) TC (Computative)

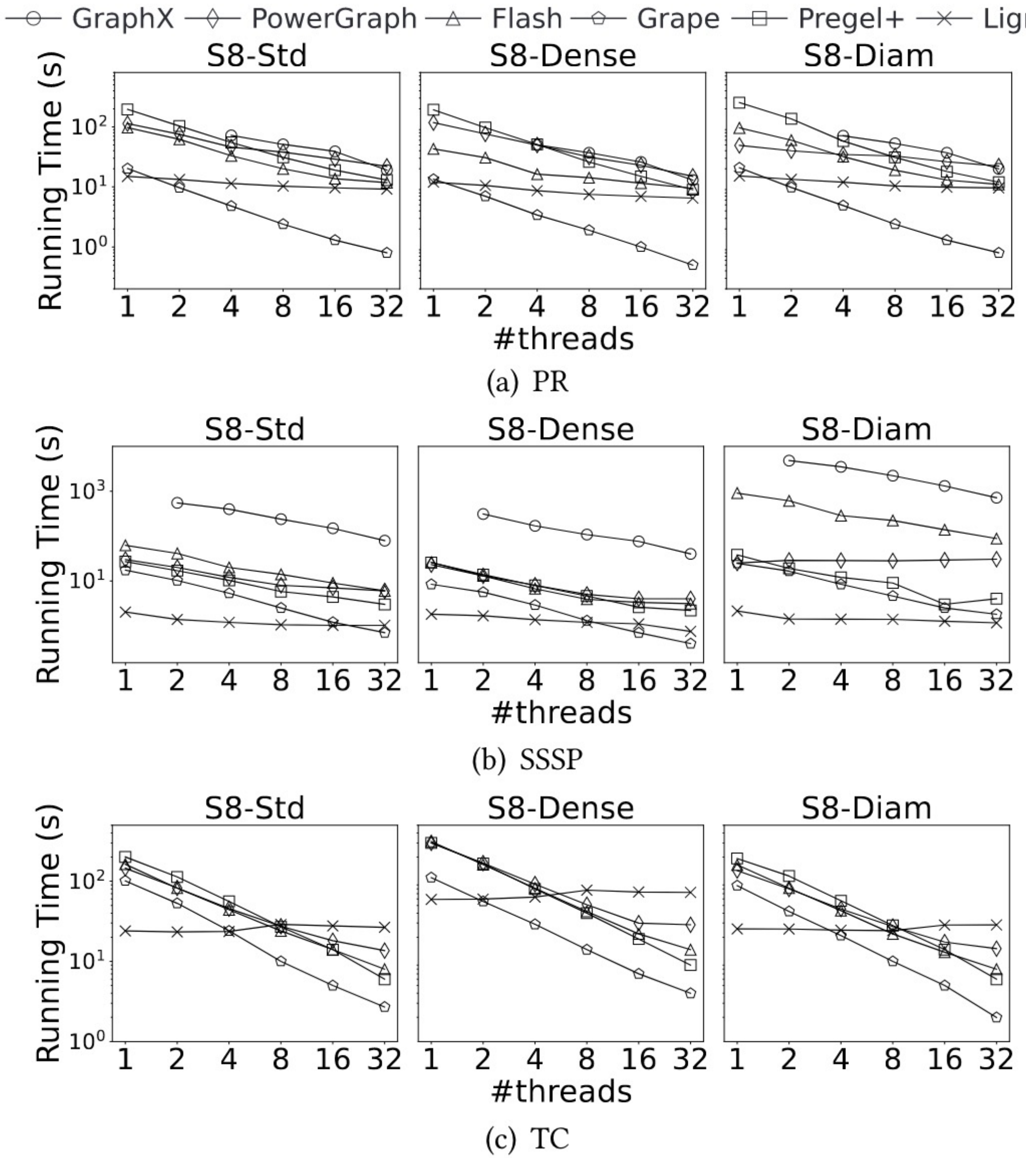
(h) KC (Computative)

Experiments

Scalability Sensitivity-Varying Number of Threads

Algo.	Dataset	GraphX	PowerG	Flash	Grape	Pregel+	Ligra
PR	S8-Std	3.8	5.1	8.2	25.3	14.6	1.6
	S8-Dense	3.8	7.8	4.5	25.2	20.1	1.8
	S8-Diam	3.6	2.2	8.2	24.2	19.5	1.6
SSSP	S8-Std	6.9	5.0	9.3	23.5	8.8	2.0
	S8-Dense	7.8	5.8	8.5	19.7	11.3	2.4
	S8-Diam	6.7	0.9	10.2	13.2	10.1	1.8
TC	S8-Std	—	10.7	18.7	37.2	32.0	1.1
	S8-Dense	—	10.5	22.2	27.5	32.4	0.9
	S8-Diam	—	9.4	18.1	29.6	29.6	1.1

Scaling Factor: the best performance over single **thread** performance



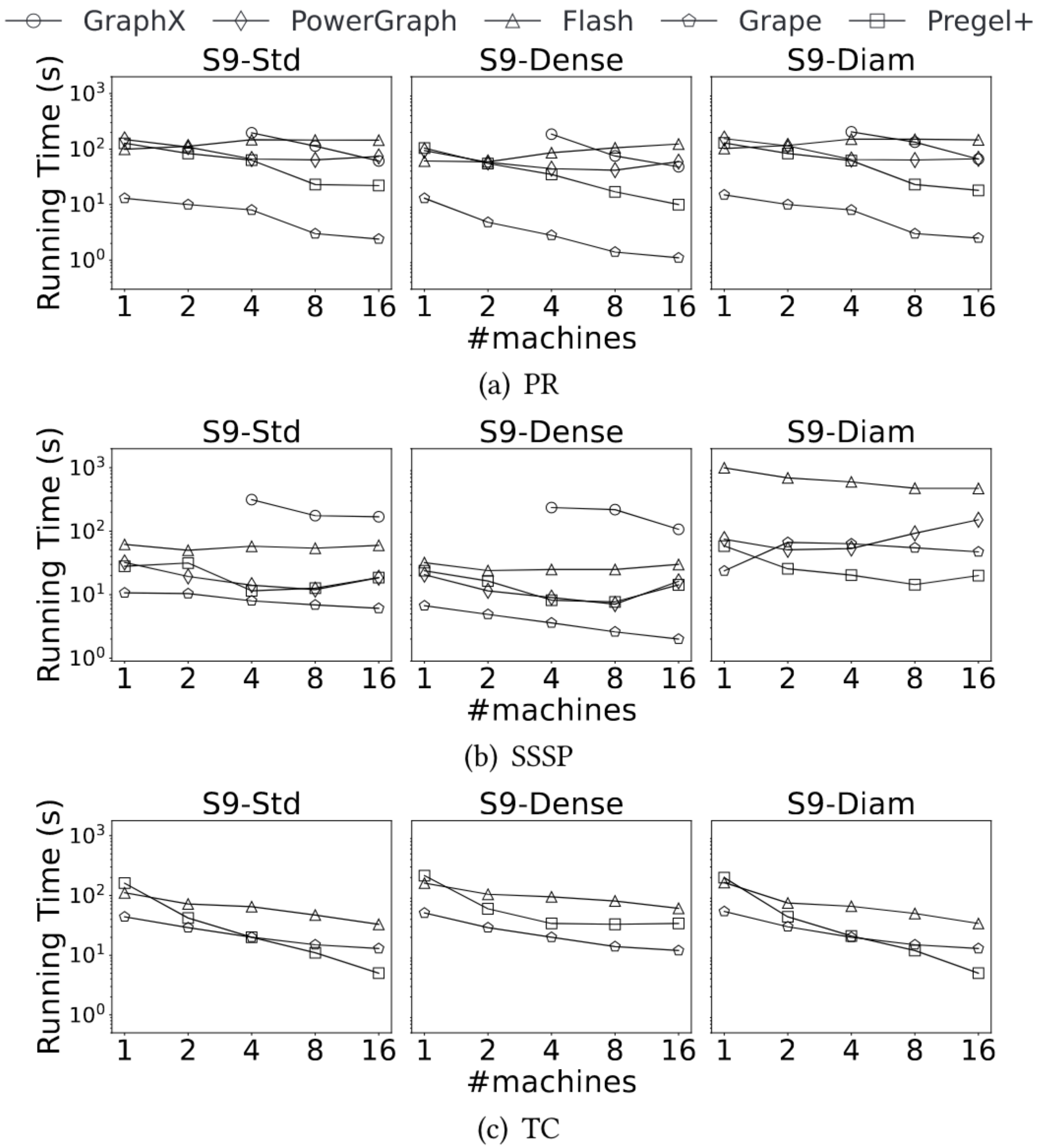
Running time of PR, SSSP and TC, with three datasets (Scale = 8), **varying #threads**

Experiments

Scalability Sensitivity-Varying Number of Machines

Algo.	Dataset	GraphX	PowerG	Flash	Grape	Pregel+
PR	S9-Std	3.2	2.3	0.8	5.8	5.7
	S9-Dense	3.8	2.2	1.0	11.5	9.9
	S9-Diam	3.0	2.4	0.8	6.1	7.5
SSSP	S9-Std	1.8	2.6	1.2	1.7	2.4
	S9-Dense	2.2	2.9	1.3	3.3	3.1
	S9-Diam	—	1.4	2.0	0.5	4.0
TC	S9-Std	—	—	3.3	3.2	27.6
	S9-Dense	—	—	2.6	4.1	6.5
	S9-Diam	—	—	4.7	3.9	35.4

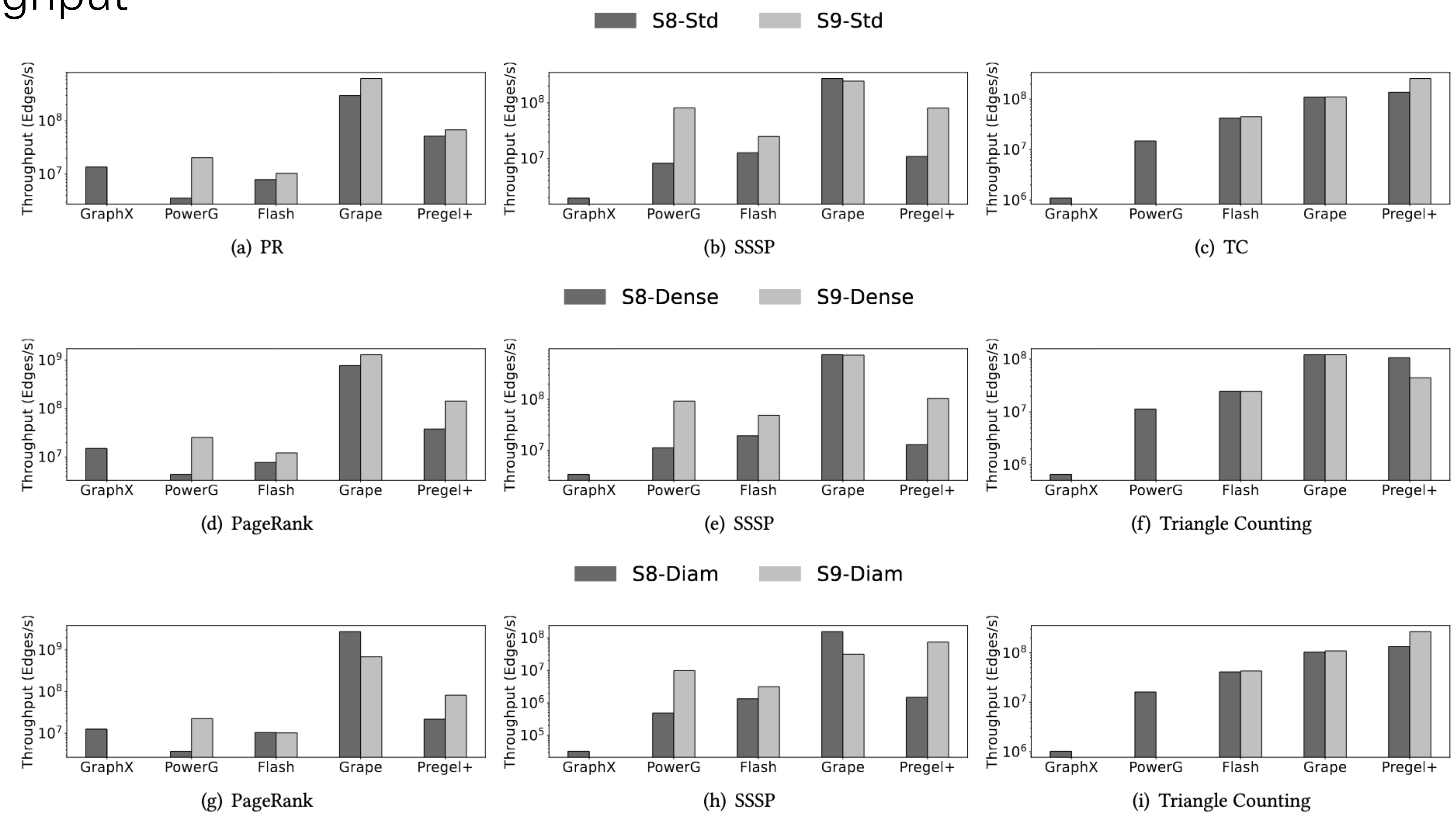
Scaling Factor: the best performance over single **machine** performance



Running time of PR, SSSP and TC, with three datasets (Scale = 9), varying #**machines**

Experiments

Throughput



Experiments

Stress Test

Platforms	S8-Std	S9-Std	S9.5-Std	S10-Std
GraphX	✓	✓	✓	
PowerGraph	✓	✓		
Flash	✓	✓	✓	
Grape	✓	✓	✓	✓
Pregel+	✓	✓	✓	✓

Experiments

Usability Evaluation

- **GraphX** stands out with the highest usability scores across all expertise levels.
- **PowerGraph** and **Pregel+** exhibit balanced usability, particularly favoring junior and intermediate users.
- **Grape**'s API has a steep learning curve, receiving low scores from beginners but significantly improving in usability for senior and expert users.
- **Flash** and **Ligra** show a pattern of lower usability for beginners, with scores improving as users gain more expertise.

Platforms	Junior	Intermediate	Senior	Expert
GraphX	**71.25**	74.00	**93.50**	**98.25**
PowerG	69.11	74.33	77.33	85.67
Flash	61.54	66.38	76.38	91.17
Grape	40.87	62.07	74.87	88.27
Pregel+	70.00	**78.33**	83.50	91.67
Ligra	61.00	69.17	82.28	91.89

Summarization & Platform Selection Guide

- **GraphX:**
  - An ideal choice for users of all experience levels, provided that performance and scalability are not their primary concerns.
- **PowerGraph & Pregel+:**
  - Recommended for beginners and intermediate users due to balanced performance and usability, especially with large data.
- **Flash & Ligra:**
  - Best for users with strong performance needs.
  - Flash is preferred for its multi-machine support.
  - Some experience required to fully leverage their capabilities.
- **Grape:**
  - Best for users demanding top performance and scalability, despite a steeper learning curve.

Conclusion

- We select eight representative algorithms and introduce the Distance Hop Generator that enhances dataset generation efficiency and flexibility by adjusting scale, density, and diameter.
- we adopt a multi-level usability evaluation framework based on LLMs to assess API usability. This is the first time usability evaluation metrics have been introduced in the field of graph analytics benchmarks.
- Extensive experiments evaluate both the performance and API usability of various platforms, providing valuable insights for developers, researchers, and practitioners in selecting the appropriate platform.

Thanks