Understanding Graph Computation Behavior to Enable Robust Benchmarking

Fan Yang* and Andrew A. Chien*†
*University of Chicago, †Argonne National Laboratory
{fanyang, achien}@cs.uchicago.edu

HPDC, June 18, 2015
Portland

Background

- Graph processing is challenging because of
  - Extreme scale
  - Complex computation
- Many graph-processing systems are designed to meet these challenges
  - Pregel, Giraph, GraphLab, GPS, GraphX, GraphChi ……

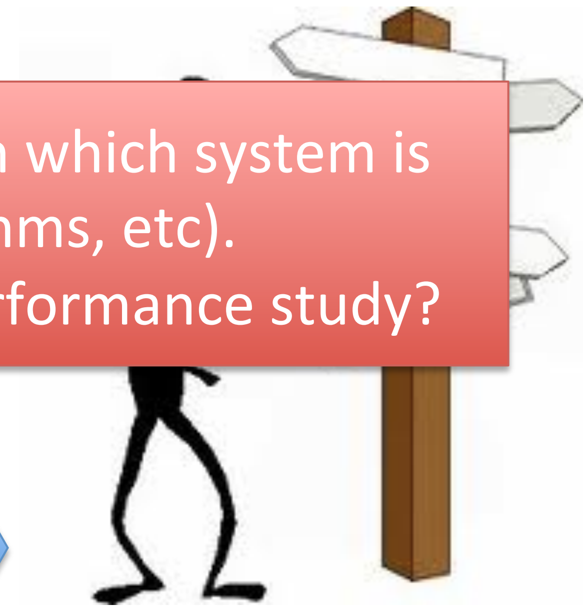
Motivation

- Graph computation spans a large diversity of algorithms and graphs.
- Graph system performance studies reflect this diversity.

Description	Benchmarks	Graphs
M. Han [1]: Giraph,	PageRank,	LiveJournal,Orkut,Arabi
F St Giraph, GraphLab		Orkut
Y. Guo [3]: Hadoop, YARN, Stratosphere, Giraph, Graphlab, Neo4j	Statistic algorithm, BFS, CC, CD, GE	Amazon, WikiTalk, KGS, Citation, DotaLeague, Synth, Friendster

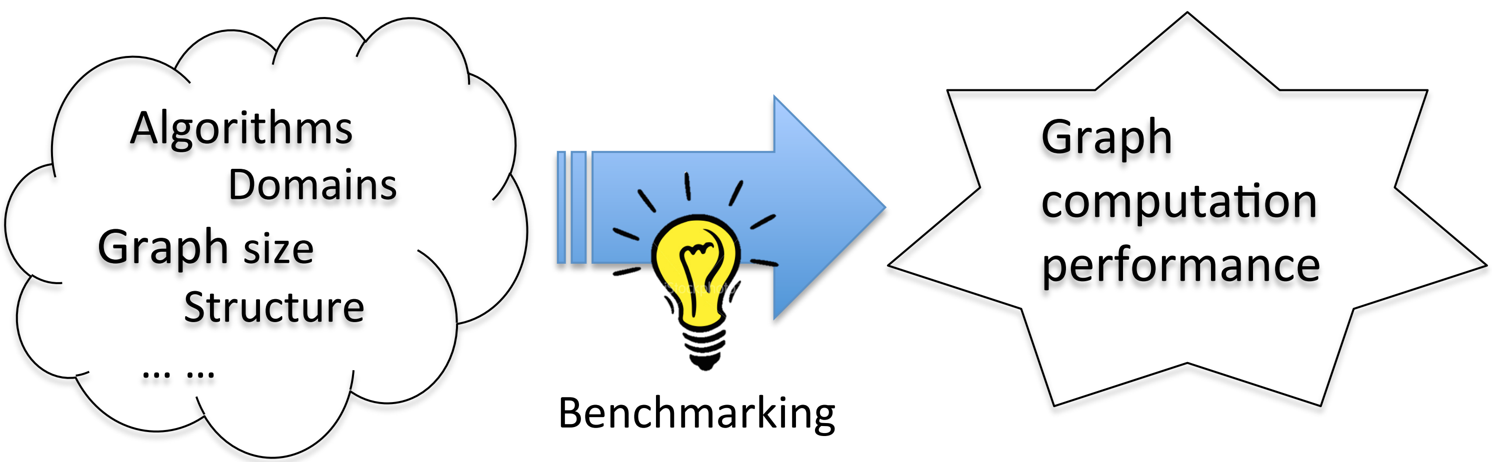
Their results provide no clear perspective on which system is preferable in a given context (graph, algorithms, etc).

→ How to do a more complete, efficient performance study?



Our Contribution

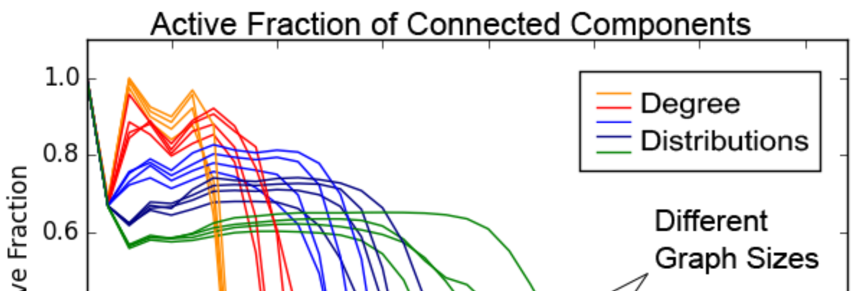
We provide a systematic understanding of the performance impact of various algorithms and graphs, and thereby enable robust, systematic, efficient benchmarking.

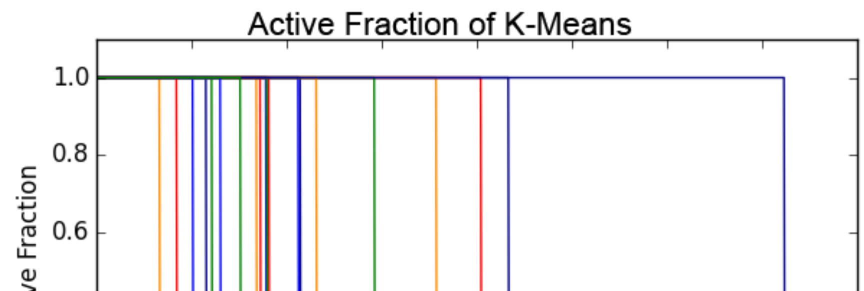


Characterize the Variation

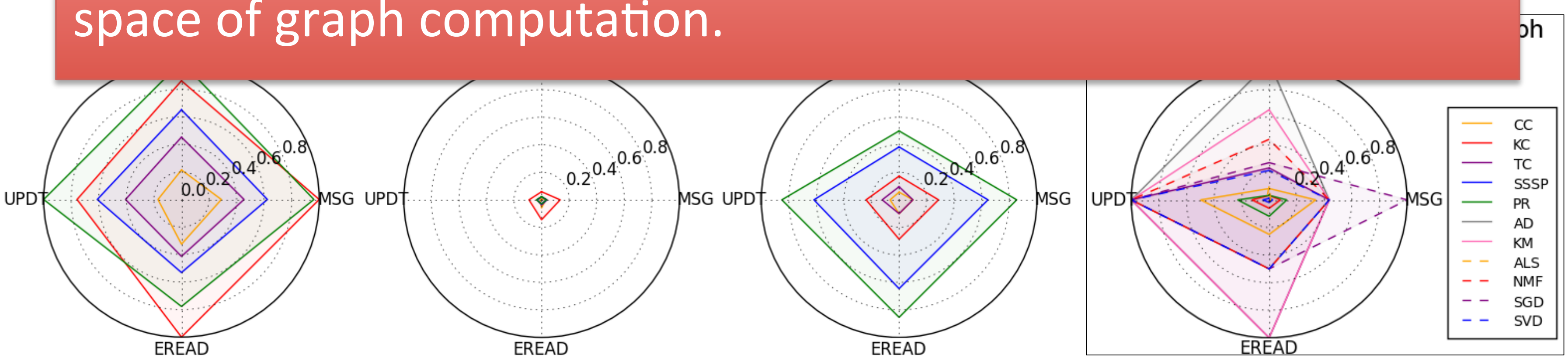
- Reflect the complexity of graph computation:
  - We select 11 algorithms from multiple domains.
  - We generate 20 graphs with different sizes and degree distributions.
- We run each <graph, algorithm> on GraphLab, and capture its fundamental behavior:
  - Active fraction: fraction of active vertices
  - {UPDT, WORK, EREAD, MSG}: #vertex updates, CPU time, #edge reads, #messages

- Behavior variation across graph algorithms
- Behavior variation across graphs



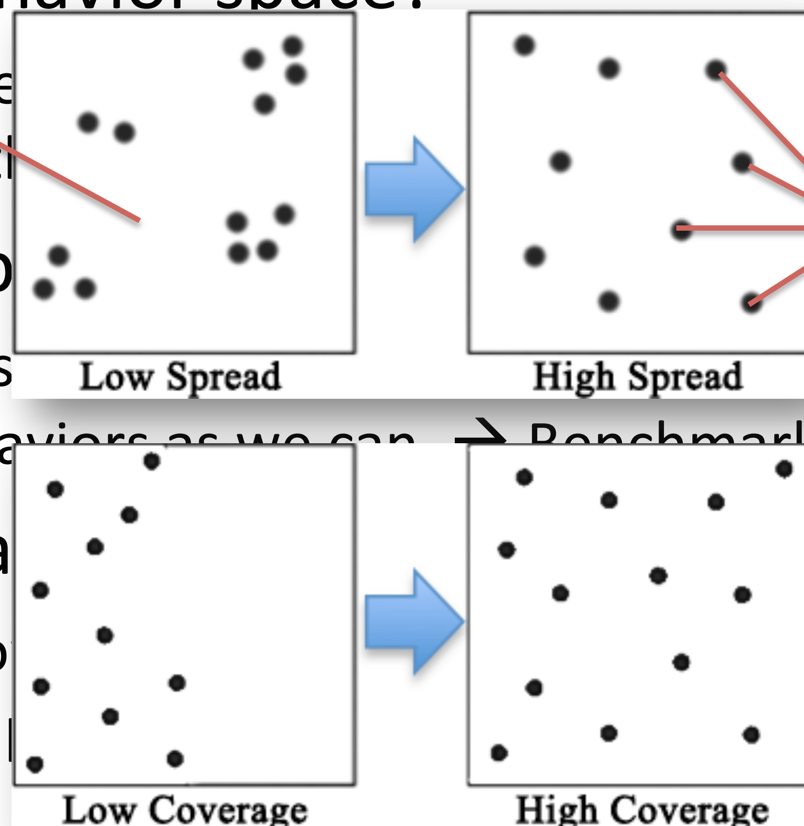


Graph computation behavior exhibits a wide variation across algorithms and graphs, forming a broad space.
→ We need a more efficient way to explore the behavior space of graph computation.



Understand the Behavior Space

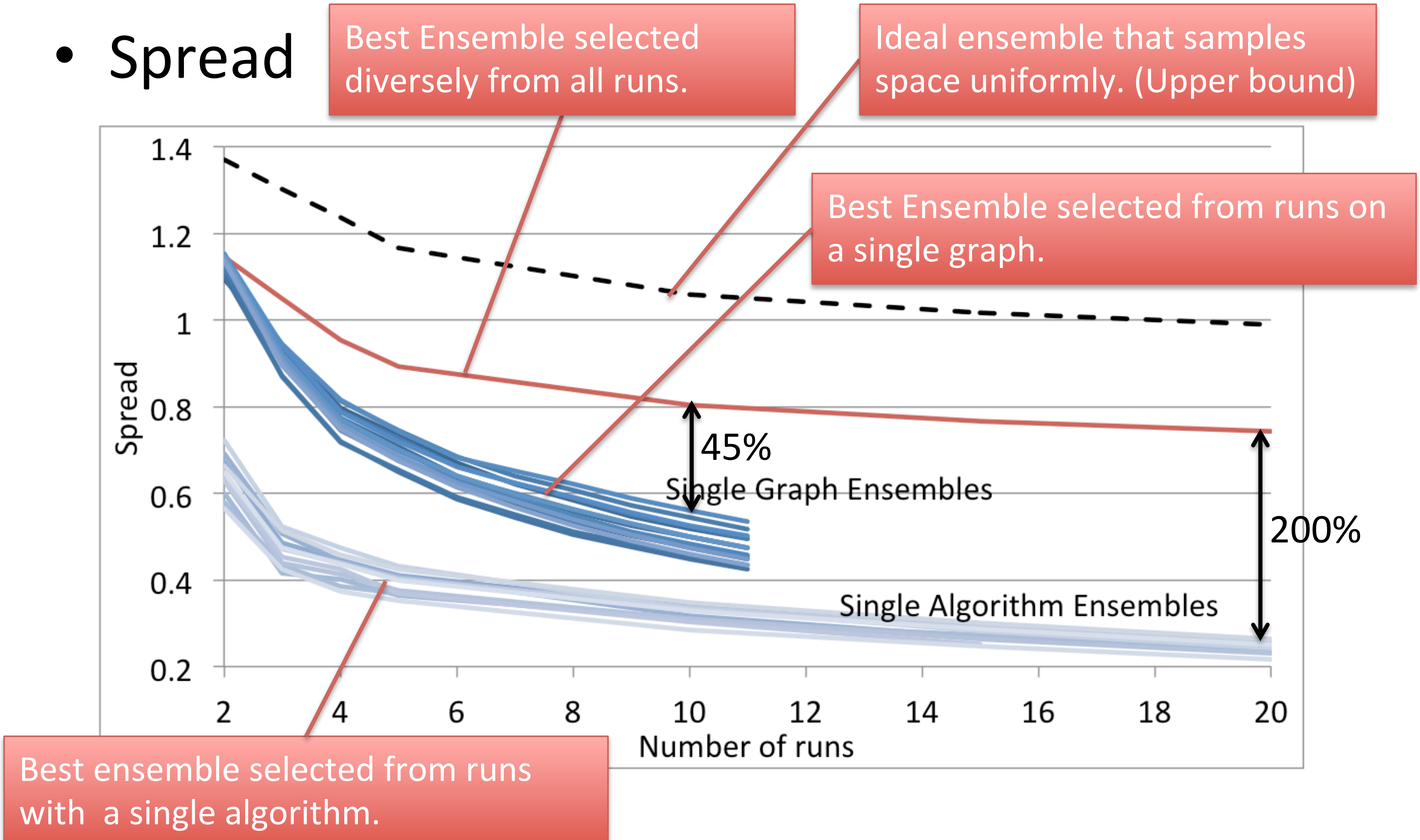
- What is behavior space?
  - ble … ed by executing any algorit…
- How to exp … space?
  - Run an ens … airs to extract as broad behaviors as we can → Benchmarking!
- How to eva … lity?
  - Spread: ho … ores the space.
  - Coverage: … xplores the space



[For precise definition of spread and coverage see the paper.]

Which ensemble is better?

- Spread



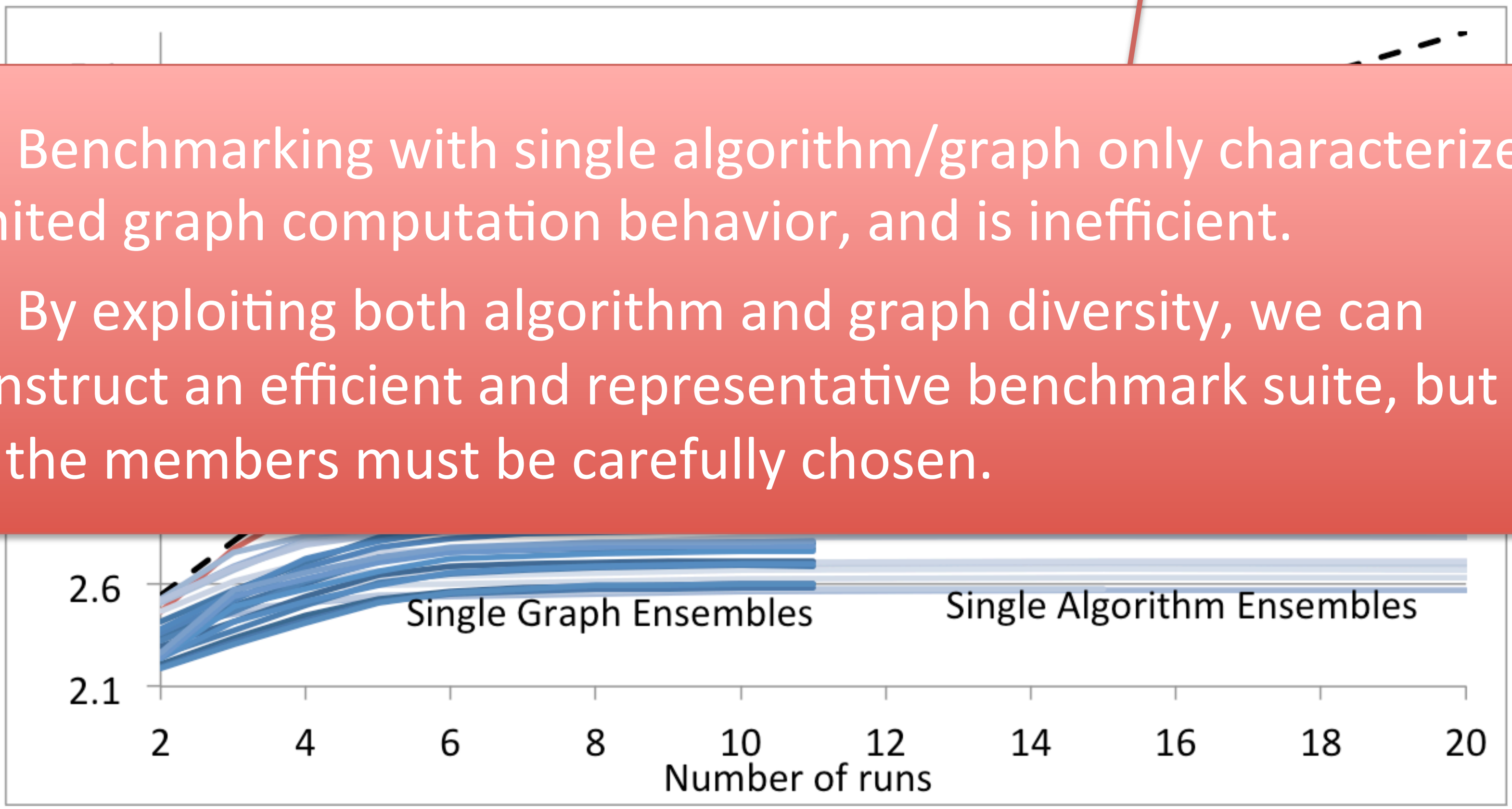
Which ensemble is better?

- Coverage

Ideal ensemble that samples space uniformly. (Upper bound)

(1) Benchmarking with single algorithm/graph only characterizes limited graph computation behavior, and is inefficient.

(2) By exploiting both algorithm and graph diversity, we can construct an efficient and representative benchmark suite, but all the members must be carefully chosen.



Members of ensembles achieving best spread and coverage

Type	Ensemble Size	Ensemble Members (Runs)
Best Spread	5	<ALS, 10^5, 3.0>, <SGD, 10^8, 2.0>, <TC, 10^9, 2.0>, <SSSP, 10^9, 3.0>, <ALS, 10^5, 2.75>
	10	ALS, SGD, TC, SSSP, ALS, TC, SGD, ALS, KM, SVD
	15	SSSP, ALS, KM, SGD, ALS, TC, SGD, ALS, TC, SSSP, ALS, SGD, TC, SVD, ALS
	20	SSSP, ALS, TC, SGD, ALS, TC, SGD, ALS, KM, SSSP, ALS, SGD, KM, SVD, ALS, TC, SGD, ALS, SGD, TC
Best Coverage	5	<TC, 10^6, 2.5>, <KM, 10^6, 2.25>, <AD, 10^7, 3.0>, <ALS, 10^8, 2.0>, <KC, 10^6, 2.5>
	10	AD, SVD, KM, ALS, TC, KC, KM, ALS, KM, NMF
	15	KM, NMF, ALS, AD, SVD, KC, KM, ALS, KM, KM, SVD, PR, ALS, TC, NMF
	20	AD, SVD, KM, ALS, TC, KC, KM, ALS, KM, SGD, NMF, KM, ALS, NMF, PR, TC, NMF, SSSP, ALS, AD

More Implications for Benchmarking

- Some algorithms are more useful in behavior space exploration than others.
  - Alternating Least Square, K-means, Triangle counting
- We can further reduce benchmarking complexity without much loss of quality.
  - Employ less algorithms, run less iterations, etc.

[More details can be found in the paper. Full description can be found in my master's thesis]

Summary & Future Work

- We find graph computation exhibits large variation of behavior across both algorithms and graphs.
- Our study shows that diverse and careful selection of algorithms and graphs is important for robust and efficient benchmarking.
- We present a systematic approach to constructing robust, efficient benchmark set.
- Future work:
  - Study temporal and spatial dynamic variation of graph computation behavior.
  - Use our framework to analyze performance studies.
  - Understand if we can model a graph computation and predict its performance.
  - Use our framework to find optimal configurations for graph computations

Fan Yang, HPDC 2015, Portland

Thanks!

Workload: Graph Algorithms

- Select from multiple domains to capture the variety and breadth of graph algorithms.
  - Graph Analytics: Connected Components (CC), K-Cores (KC), Triangle Counting (TC), SSSP, PageRank (PR), Approximate Diameter (AD).
  - Clustering: K-Means (KM).
  - Collaborative Filtering: Alternating Least Squares (ALS), Non-negative Matrix Factorization (NMF), Stochastic Gradient Descent (SGD), Singular Value Decomposition (SVD).

Workload: Graphs

- Capture the major properties that significantly impact graph computation.
  - Graph size is defined as the number of edges (*nedges*). ($10^5 \sim 10^9$).
  - Degree distribution of a graph follows a *power law*, defined as the following formula:

$$P(k) \sim k^{-\alpha}$$

Where $P(k)$ is the fraction of vertices in the graph with degree k, and α is a constant. (2.0 ~ 3.0)

- A synthetic graph is represented as <*nedges*, α>.

Performance Metrics

- Capture the fundamental behavior of graph computation
  - Active Fraction: the ratio of active vertices to all vertices in a single iteration.
  - UPDT: a
    iteratio
  - WORK:
    vertex v
  - EREAD:
  - MSG: a

Graph size vary by orders of magnitude

↓

Metric values vary by orders of magnitude

Relative difference is more important than absolute values

Normalization

< Note: {UPDT, WORK, EREAD, MSG} are normalized to [0, 1]. >

Experimental Setup

- We execute 11 algorithms over 20 graphs (215 runs in total)
- Platform: Midway (up to 48 nodes, 16 cores each)
- Graph-processing system: GraphLab v2.2

How to understand the Behavior Space?

- Definitions:
  - The behavior of a graph computation (a graph-algorithm pair):

$$Behavior(GC_i) = < UPDT, WORK, EREAD, MSG >$$

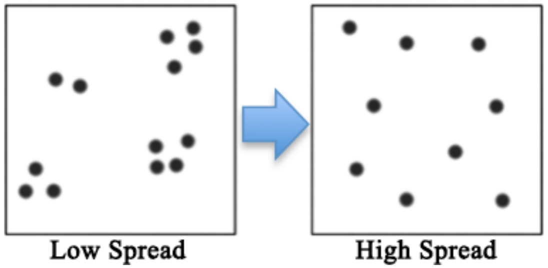
  - An ensemble of graph computations to model any sets of experiments:

$$Ensemble_k = \{GC_1, GC_2, \ldots, GC_N\}$$

How well an ensemble sample the Behavior Space?

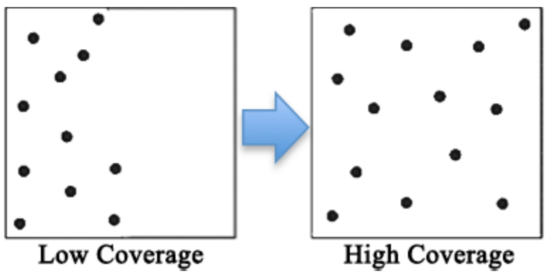
- Ensemble metrics:
  - Spread is how efficiently an ensemble explores the behavior space.

$$Spread(Ensemble_k) = \frac{\sum_{i=1}^{N}\sum_{j=1}^{N} d(Behavior(GC_i), Behavior(GC_j))}{N(N-1)}$$



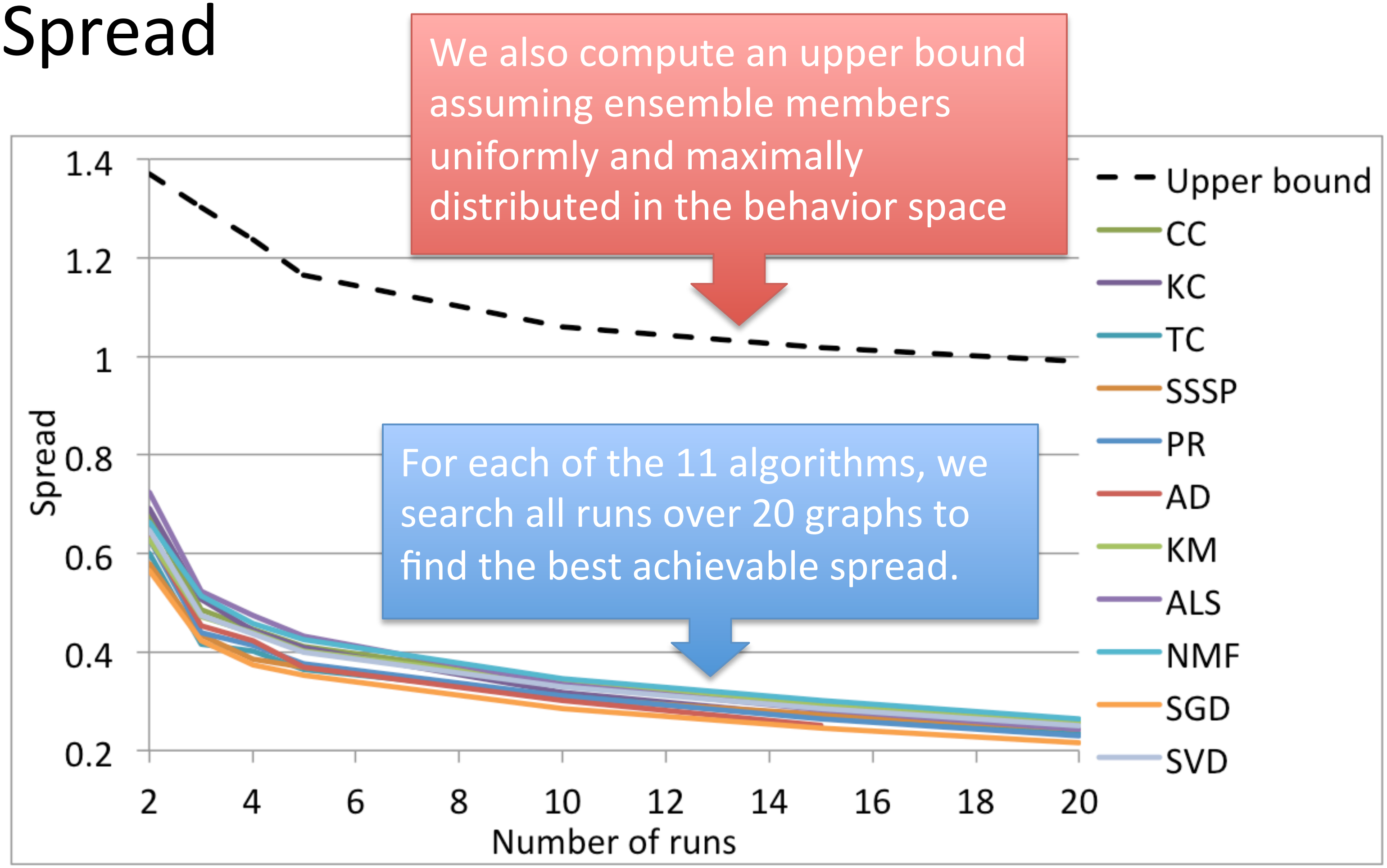
  - Coverage is how completely an ensemble explores the behavior space.

$$Coverage(Ensemble_k) = \frac{N_S}{\sum_{i=1}^{N_S} \min_{k=1\ldots N} \{d(Sample_i, Behavior(GC_k))\}}$$



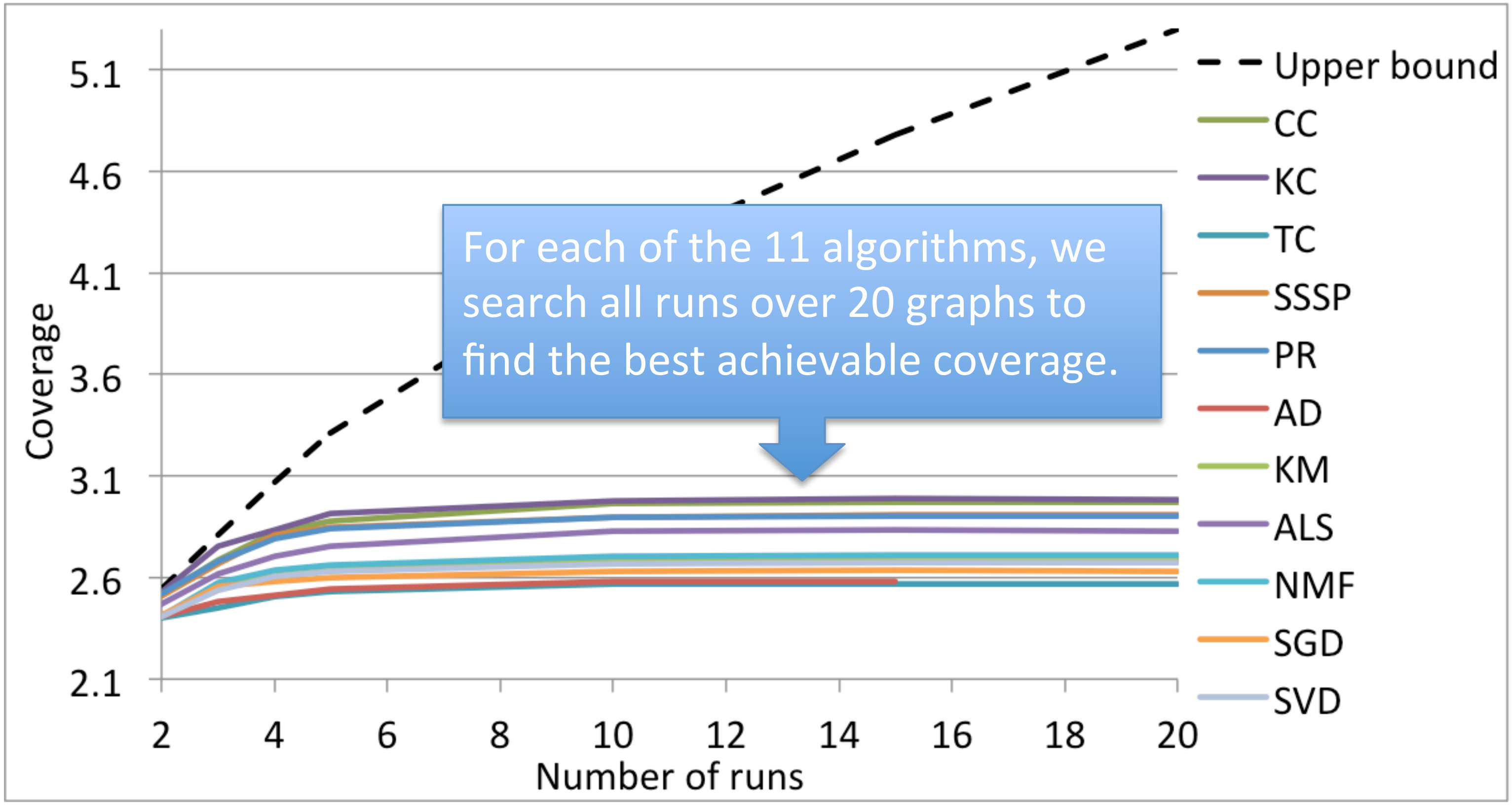
Q1: How efficiently can an ensemble with a single algorithm explore the behavior space?

- Spread



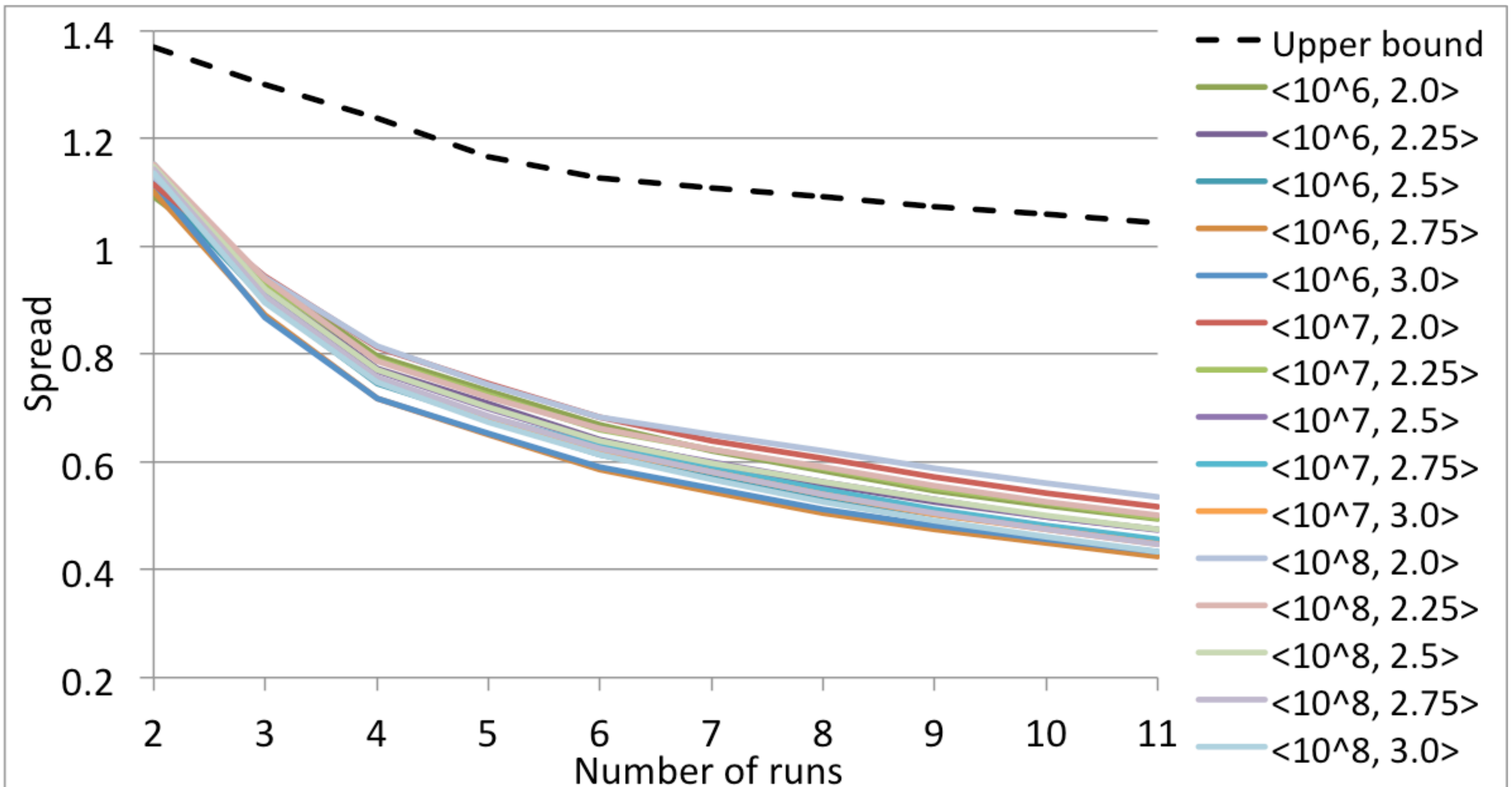
Q1: How completely can an ensemble with a single algorithm explore the behavior space?

- Coverage



Q2: How efficiently can an ensemble with a single graph explore the behavior space?

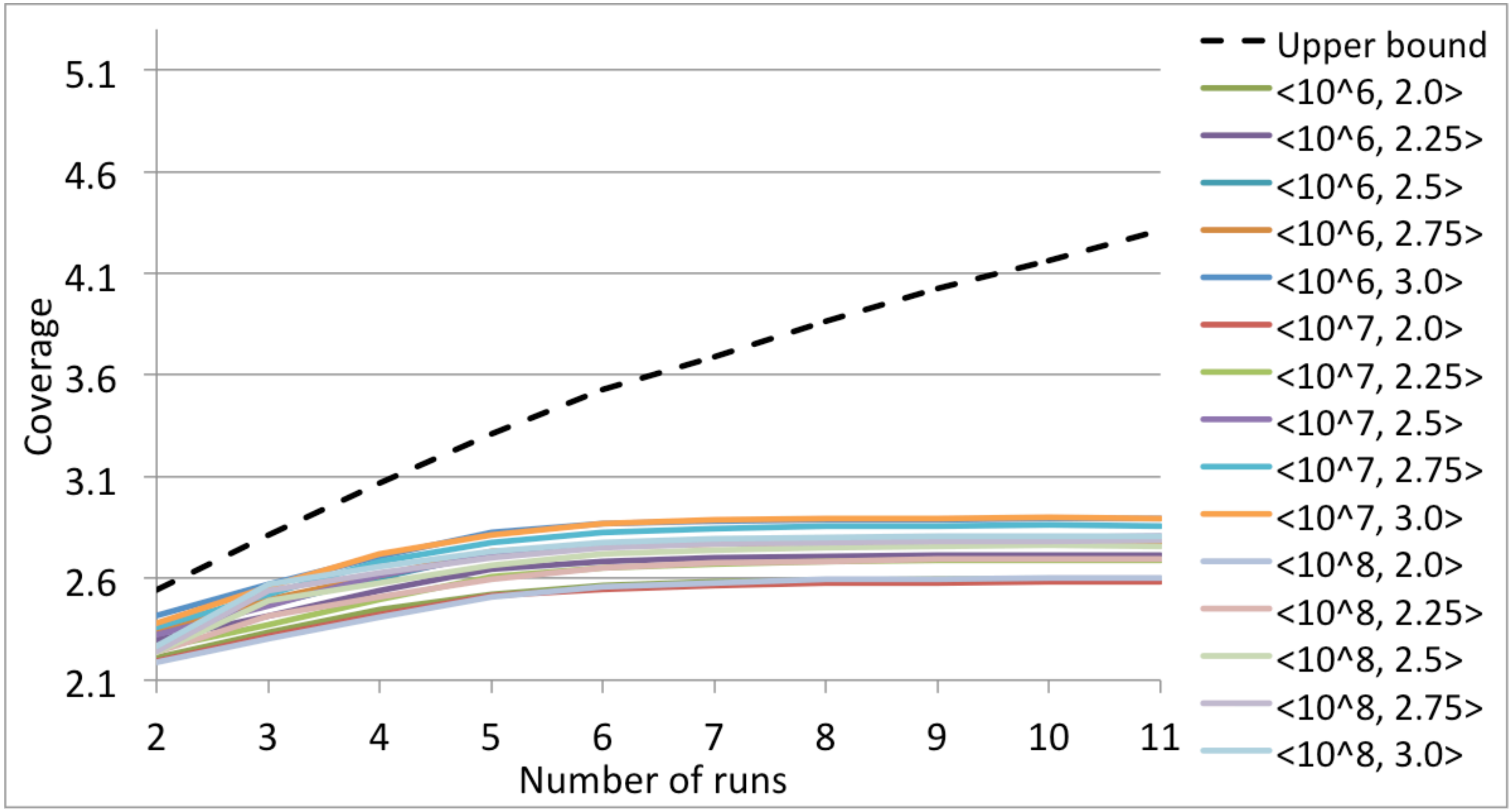
- Spread



Single-graph ensembles achieve higher spread than single-algorithm ensembles.

Q2: How completely can an ensemble with a single graph explore the behavior space?

- Coverage

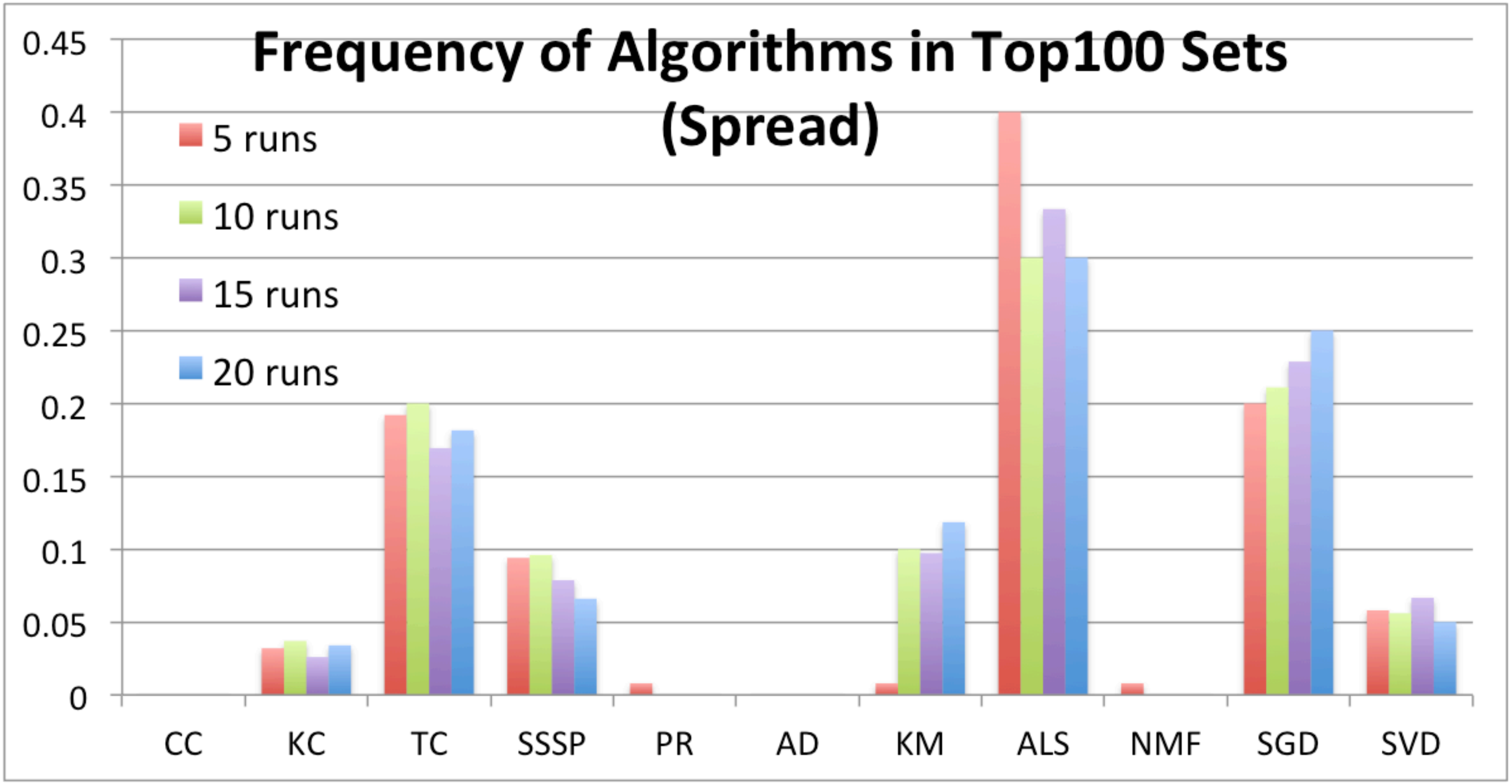


What aspects of diversity in algorithms and graphs contribute to this improvement? Let’s look into the members of ensembles achieving best spread and coverage…

Type	Ensemble Size	Ensemble Members (Runs)
Best Spread	5	<ALS, 10^5, 3.0>, <SGD, 10^8, 2.0>, <TC, 10^9, 2.0>, <SSSP, 10^9, 3.0>, <ALS, 10^5, 2.75>
	10	ALS, SGD, TC, SSSP, ALS, TC, SGD, ALS, KM, SVD
	15	SSSP, ALS, KM, SGD, ALS, TC, SGD, ALS, TC, SSSP, ALS, SGD, TC, SVD, ALS
	20	SSSP, ALS, TC, SGD, ALS, TC, SGD, ALS, KM, SSSP, ALS, SGD, KM, SVD, ALS, TC, SGD, ALS, SGD, TC
Best Coverage	5	<TC, 10^6, 2.5>, <KM, 10^6, 2.25>, <AD, 10^7, 3.0>, <ALS, 10^8, 2.0>, <KC, 10^6, 2.5>
	10	AD, SVD, KM, ALS, TC, KC, KM, ALS, KM, NMF
	15	KM, NMF, ALS, AD, SVD, KC, KM, ALS, KM, KM, SVD, PR, ALS, TC, NMF
	20	AD, SVD, KM, ALS, TC, KC, KM, ALS, KM, SGD, NMF, KM, ALS, NMF, PR, TC, NMF, SSSP, ALS, AD

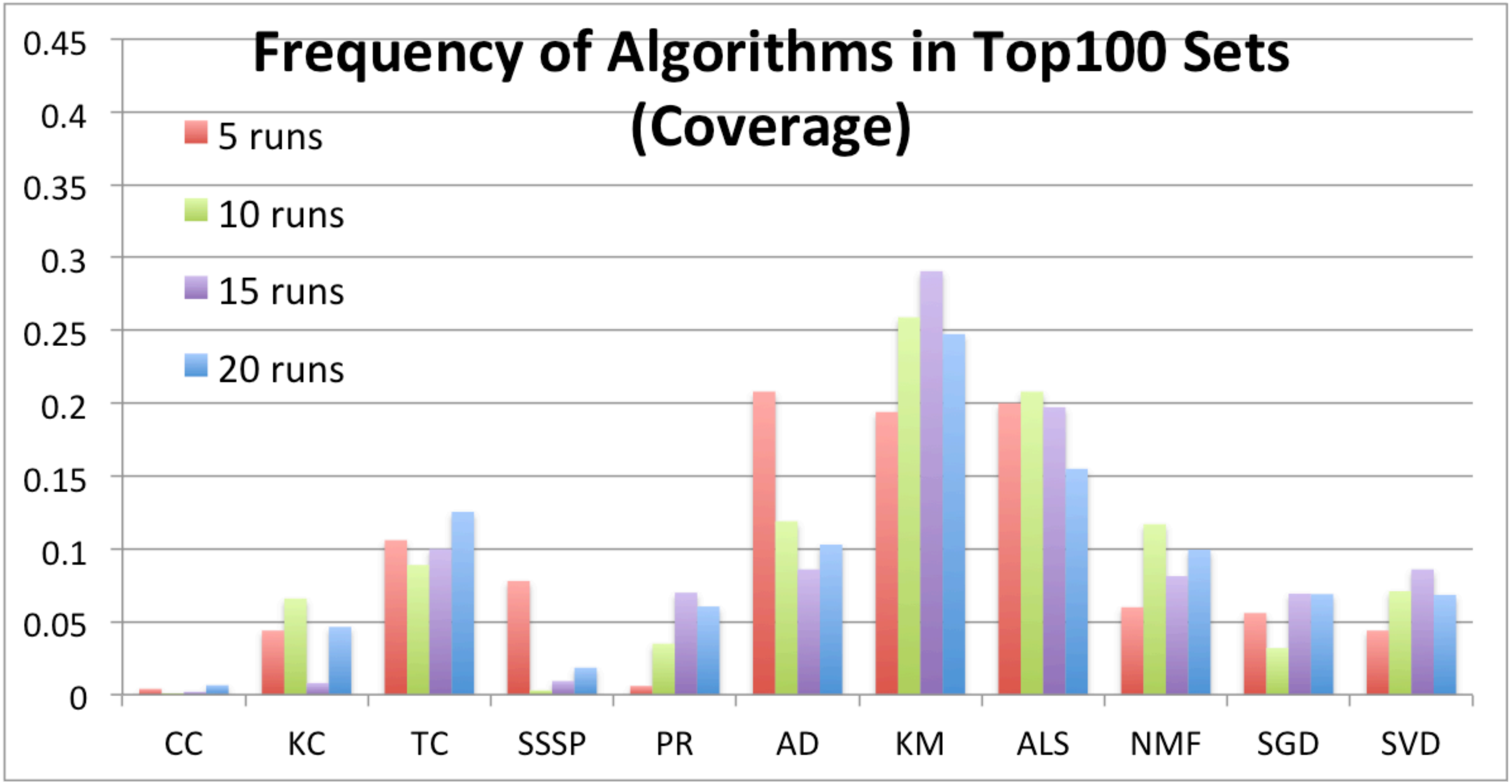
Q4: Which algorithms contribute most often to the best ensembles for spread and coverage?

- For spread: ALS, SGD, TC, …



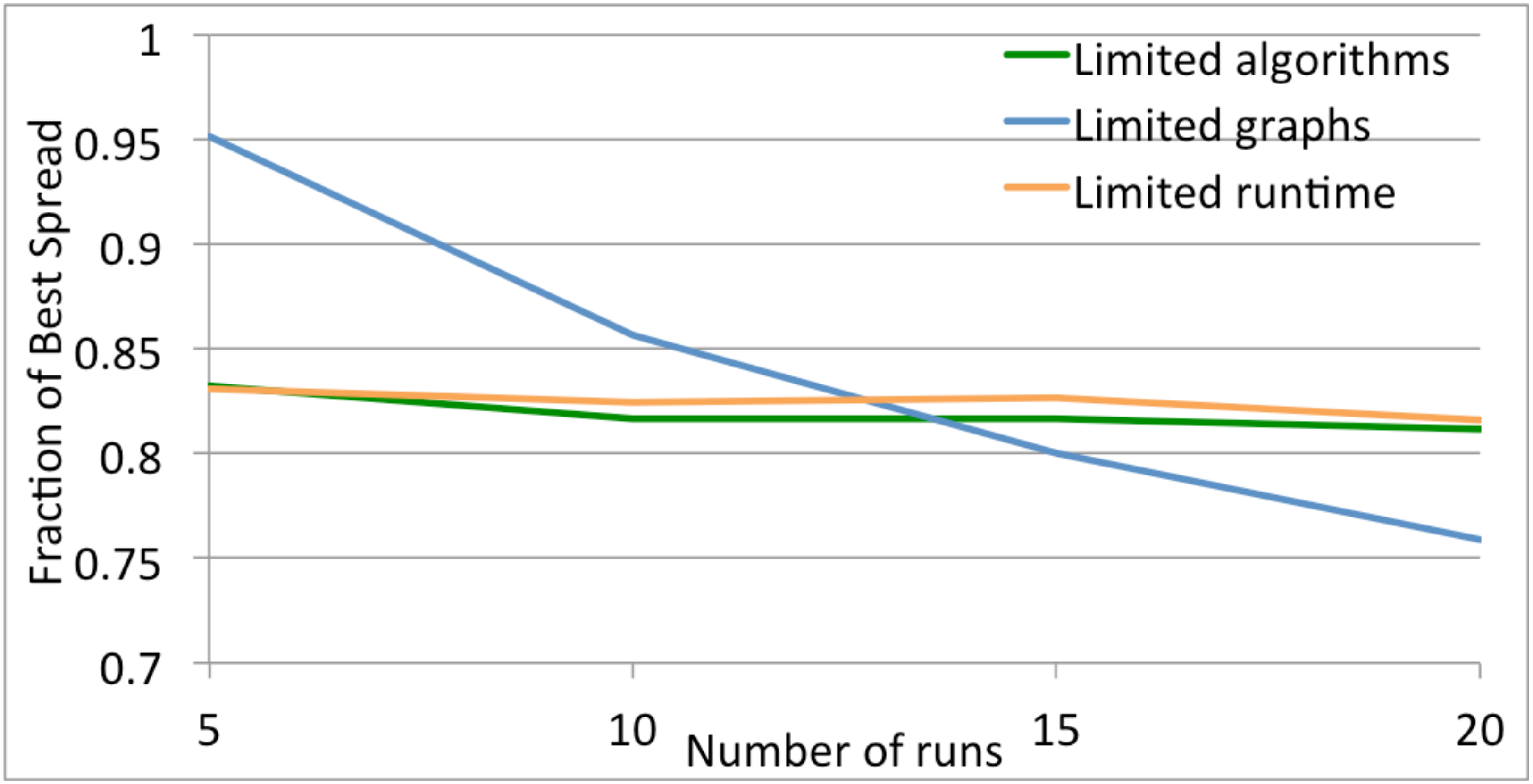
Q4: Which algorithms contribute most often to the best ensembles for spread and coverage?

- For coverage: KM, ALS, AD, TC, …



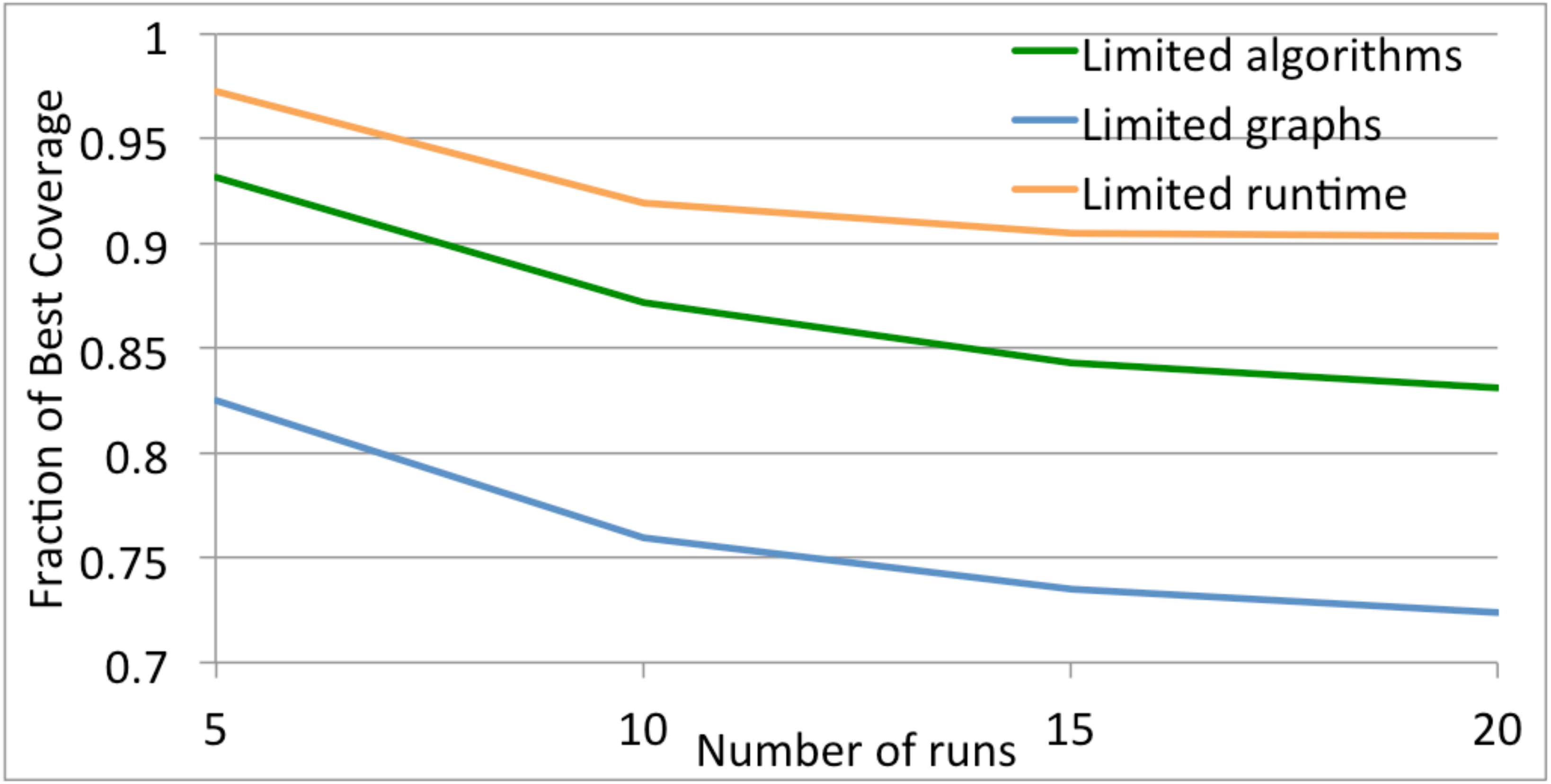
Reduce Ensemble Complexity (1 of 2)

- Spread



Reduce Ensemble Complexity (2 of 2)

- Coverage



Previous Benchmarking Efforts (1 of 2)

- Graph500 (BFS on single graph)
- B. Elser et al. (K-Core over 7 graphs)

→ Benchmarks drawn from single graph or algorithm.

- W. Han et al. (PageRank, Connected Components over 3 graphs)

→ Benchmarks combining only a small set of simple algorithms.

- M. Han et al. (4 simple algorithms over 5 graphs)
- S. Salihoglu et al. (5 algorithms over 5 graphs)

→ Ad-hoc benchmarks exploring only a small part of the whole behavior space.

Previous Benchmarking Efforts (2 of 2)

- Y. Guo's work (5 algorithms and 7 graphs) is the closest to ours. They recognize the need to explore algorithm and graph diversity.

→ "Thorough" benchmarking without any proof of real thoroughness.

← In contrast, we have formulated a space and clear metrics for assessing thoroughness.