M. BESTA, S. WEBER, L. GIANINAZZI, R. GERSTENBERGER, A. IVANOV, Y. OLTCHIK, T. HOEFLER

Slim Graph: Practical Lossy Graph Compression for Approximate Graph Processing, Storage, and Analytics

[Extre

Exascale Computing Project Selects Co-Design Center for Graph Analytics

Forbes
Billionaires Innovation Leadership Money Business Small Business Lif

Knowledge Graphs And Machine Learning -- The Future Of AI Analytics?

Graph Analytics keeps growing in popularity and possibilities

Graph continues to be the fastest growing segment of data managem

and collecting, thanks to the arrival of the internet and the a
society, powers all the incredible advances we see today in th
artificial intelligence (AI) and Big Data.

insideBIGDATA
Your Source for Machine Learning

AI Deep Learning Machine Learning Special S

CUSTOM DESIGN

Home » Uncategorized » Graph Analytics Expands to the Cloud

Graph Analytics Expands to the Cloud

aniel Gutierrez Leave a Comment

Graph Analytics and Mining

What are the concrete workloads we care about?

Graph Databases for Beginne

Graph Technology Is the Future

The Future of Data: A Decentralized Graph Database

October 14th 2019

Graph Databases and NoSQL Stores

Home › Architectures › DARPA ERI: HIVE and Intel PUMA Graph Processor

DARPA ERI: HIVE and Intel PUMA Graph Processor

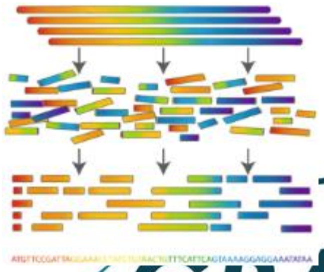
David Schor Architectures
Tagged DARPA, DARPA HIVE, G ure, PUMA

Modern microprocessors are d memory through a hierarchy of caches rkloads exhibit relatively predictable general memory patterns that can be exploited via spatial locality and temporal locality.

Graph Processing Architectures

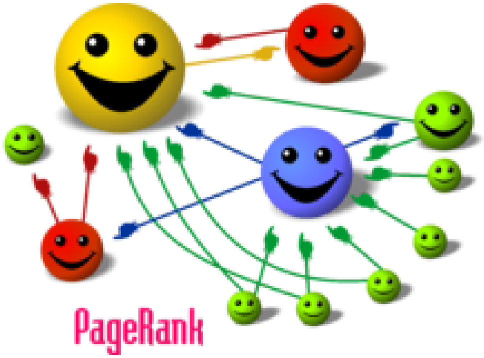
Graph Analytics Fundamental Problems and Algorithms [1, many others]

Genome traversals



Graph traversals (e.g., BFS, SSSP)

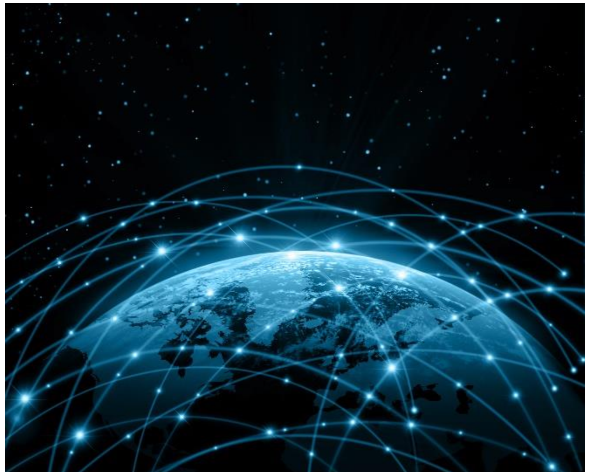
Iterative schemes (e.g., PageRank)



Connectivity related (e.g., #connected components)

Optimization problems (e.g., MST, colorings, …)

Graph mining & learning (e.g., clique listing)

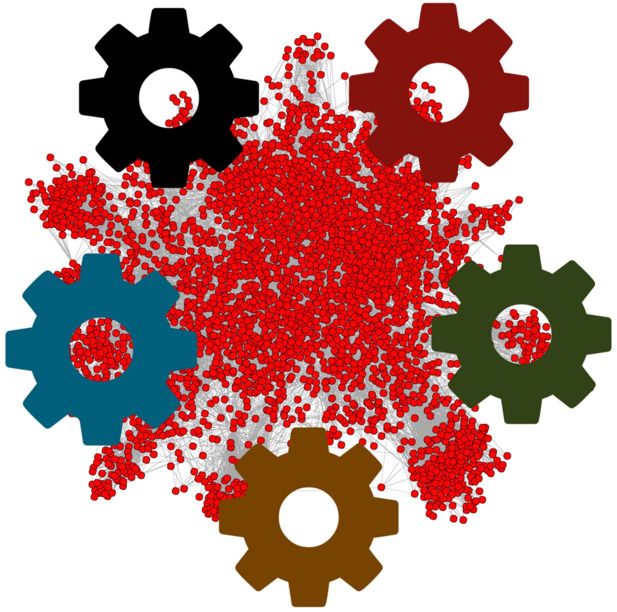


[1] A. Iosup et al.: LDBC Graphalytics: A benchmark for large-scale graph analysis on parallel and distributed platforms. VLDB'16.

Problems!

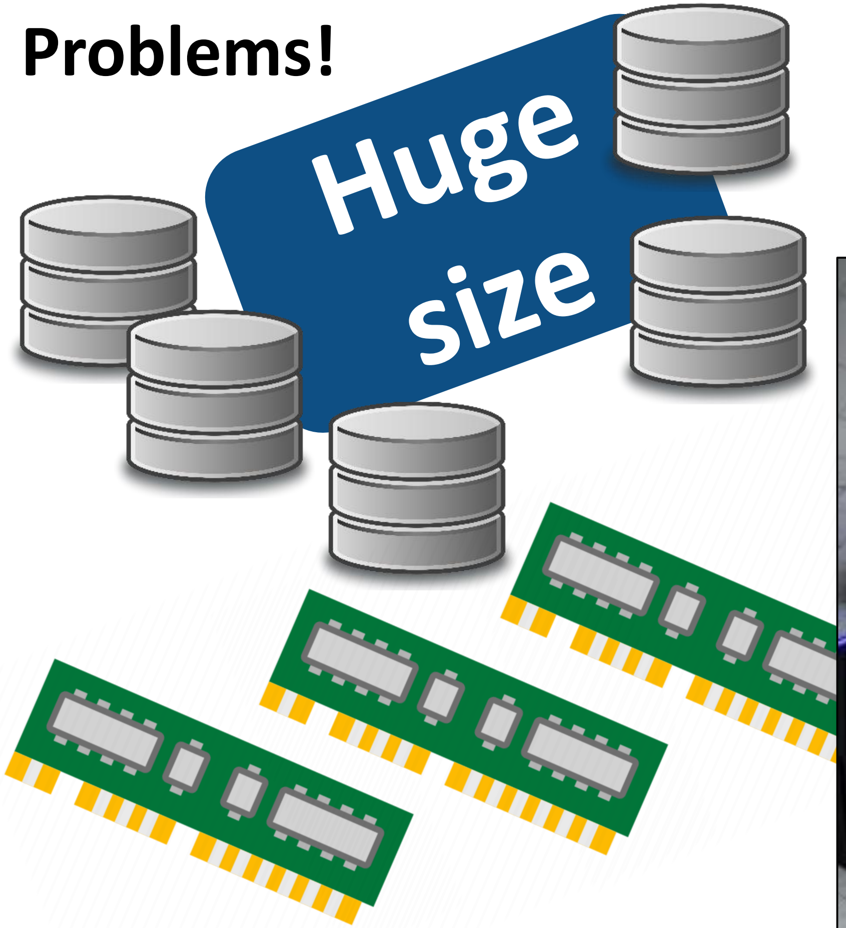
We need lots of hardware resources to store them

Running analytics on large graphs gets slow



What does “huge” mean?

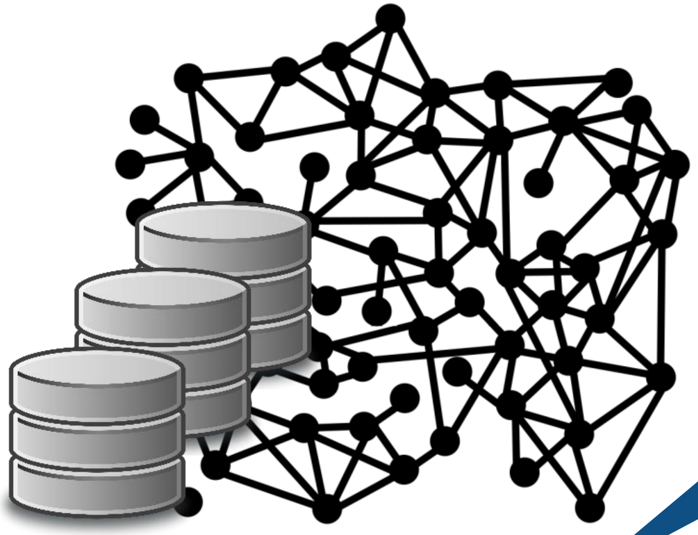
Problems!



Lossless compression incurs expensive decompression and it hits fundamental storage lower bounds [1,2]

What does "huge" mean?

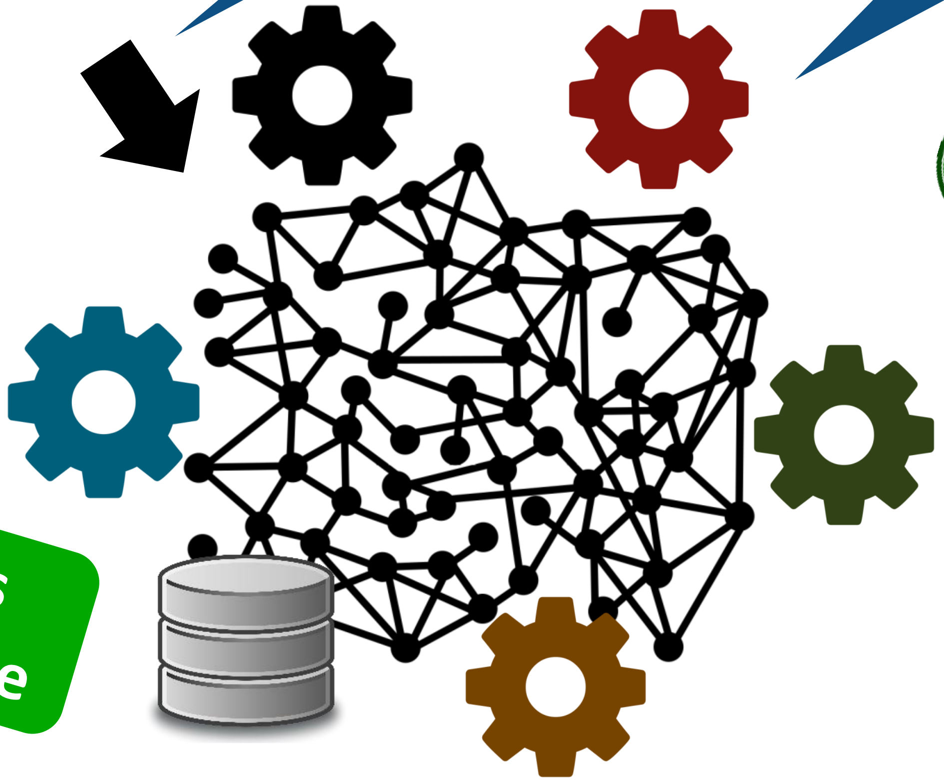
[1] M. Besta et al.: "Log(Graph): A Near-Optimal High-Performance Graph Representation", PACT'18
[2] M. Besta, T. Hoefler. "Survey and taxonomy of lossless graph compression and space-efficient graph representations", arXiv'19



Remove some edges and / or vertices (i.e., <u>**sparsification**</u>)

What if we don't want full precision?

Run graph analytics workloads on these sparsified graphs

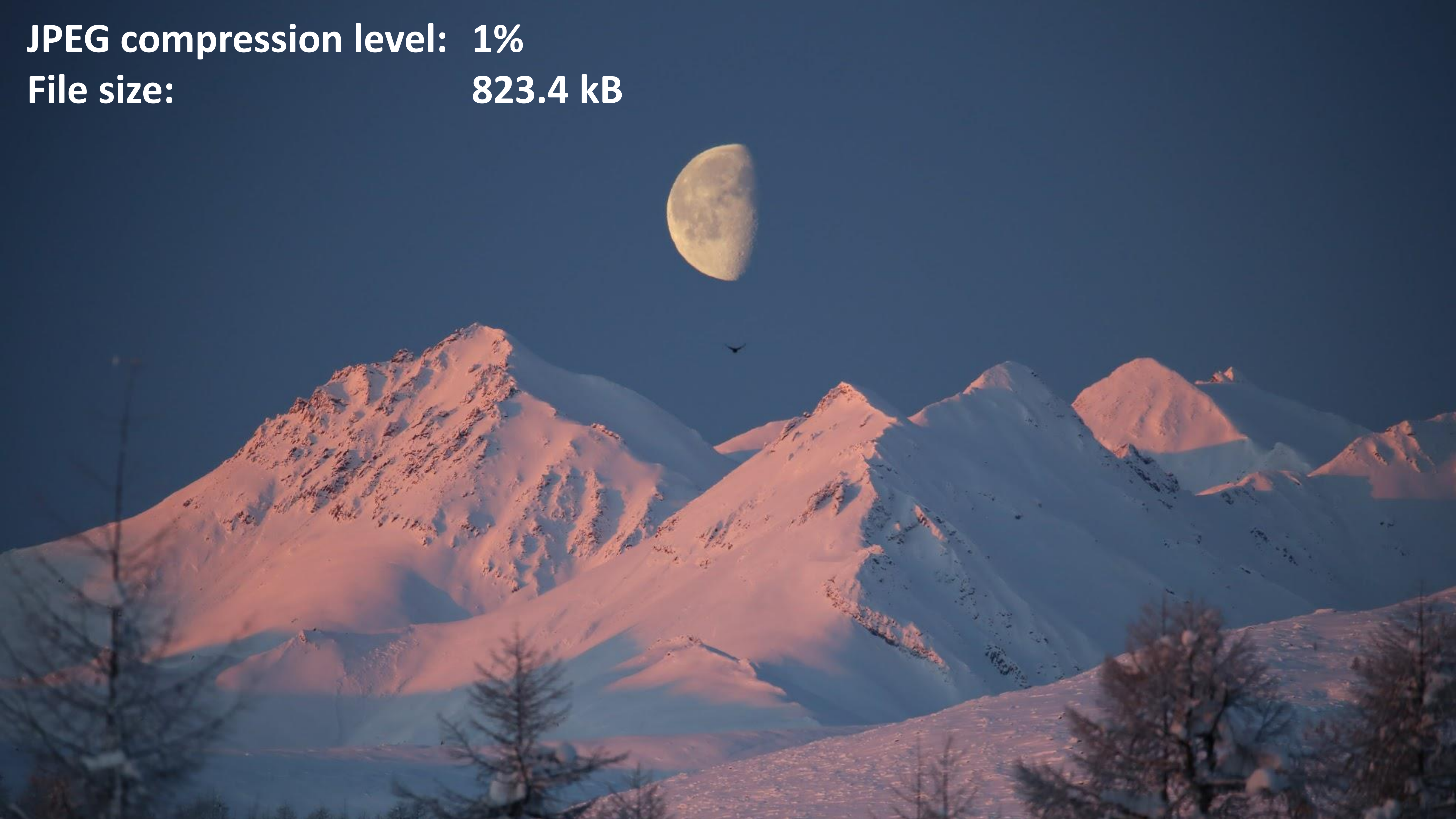


Less storage

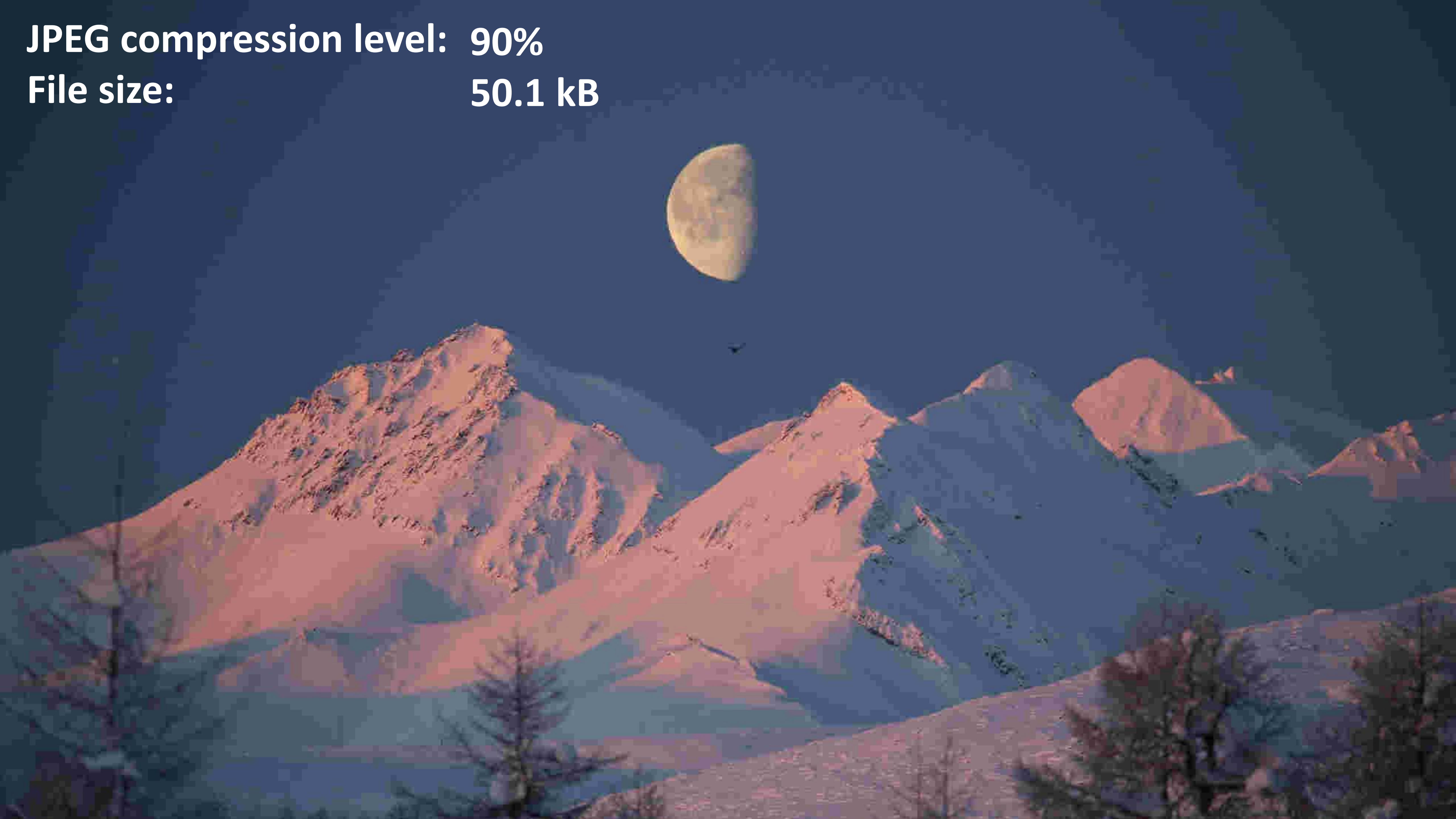
Faster workloads

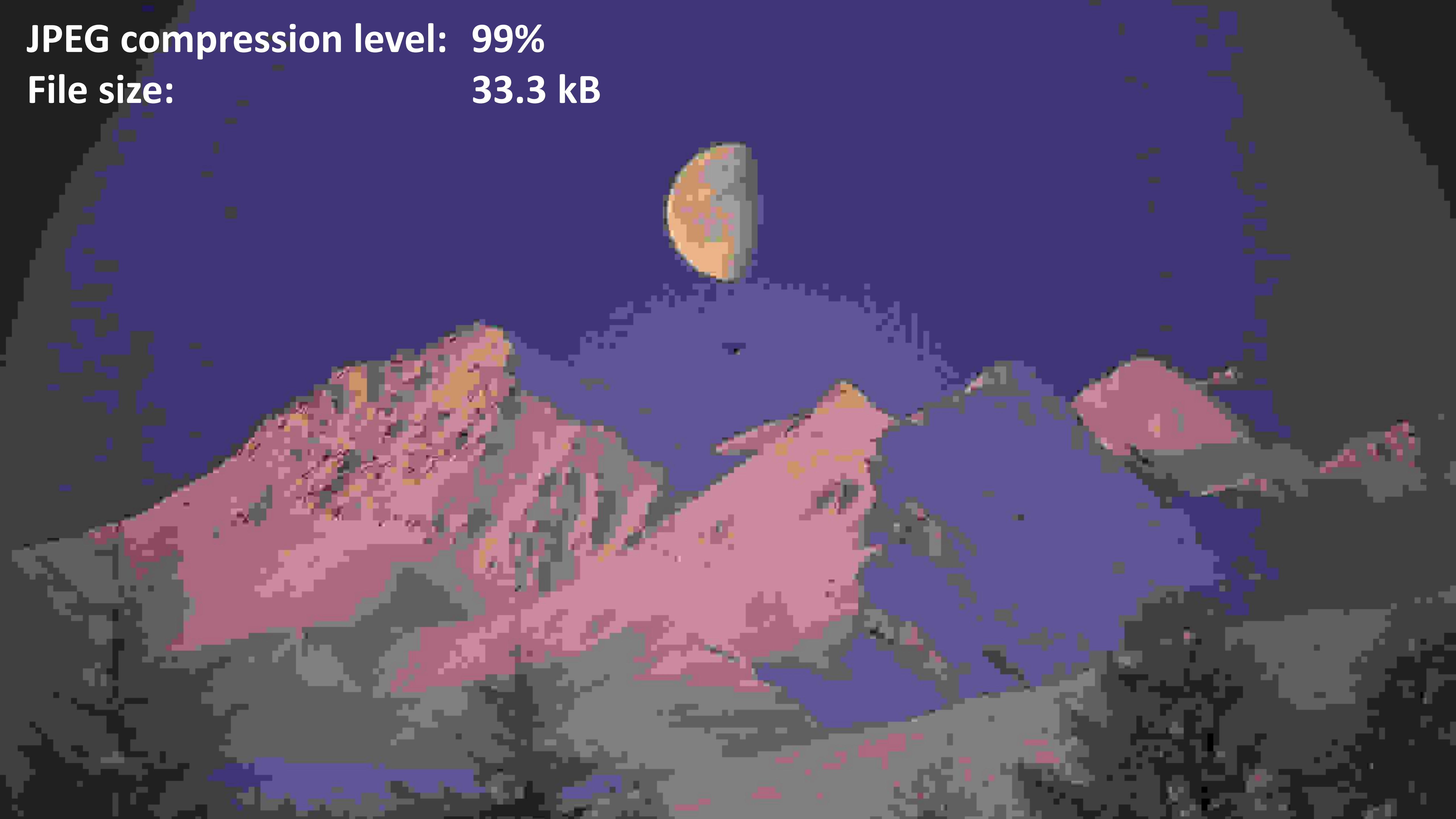
High Accuracy

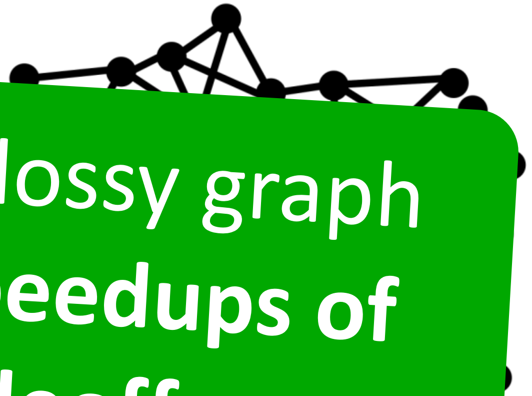
Let's see a curious motivation…


JPEG compression level: 1%
File size: 823.4 kB


JPEG compression level: 50%
File size: 130.2 kB


JPEG compression level: 90%
File size: 50.1 kB


JPEG compression level: 99%
File size: 33.3 kB



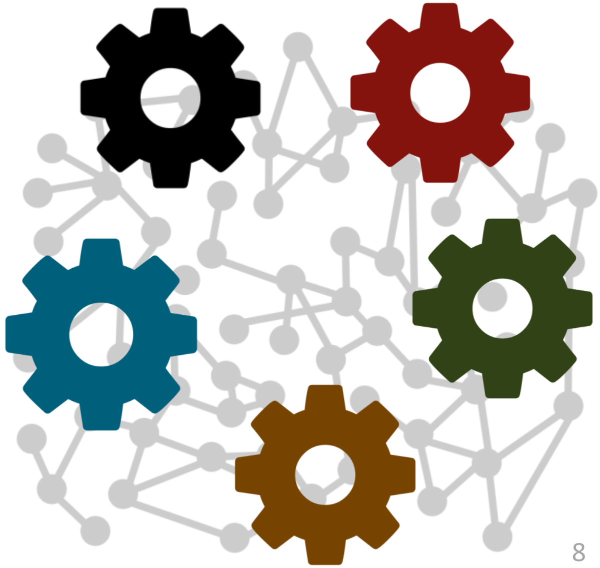
Slim Graph: A systematic approach for effective lossy graph compression, to enable **storage reductions** & **speedups of graph analytics**, with a **small accuracy tradeoff**

JPG & MP3 target specific things (**pictures** & **sound**)

Slim Graph targets specific **classes of graph workloads** / **properties**

.jpg

.mp3



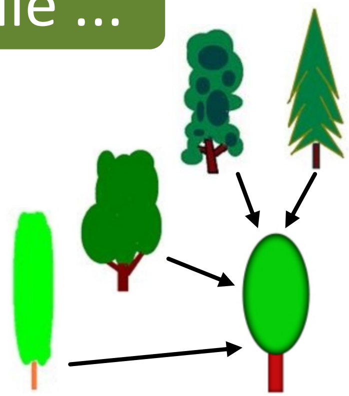
Slim Graph delivers a simple, intuitive, versatile ...

1 ... **Abstraction & programming model** for easy development and rapid prototyping of lossy graph compression methods

2 ... **Compression method** that preserves different graph properties that are important for the practice of graph processing

3 ... **Criterion (criteria?)** to assess the accuracy of lossy graph compression methods

4 ... *High-performance* and *extensible* **system** for implementing and executing lossy graph compression

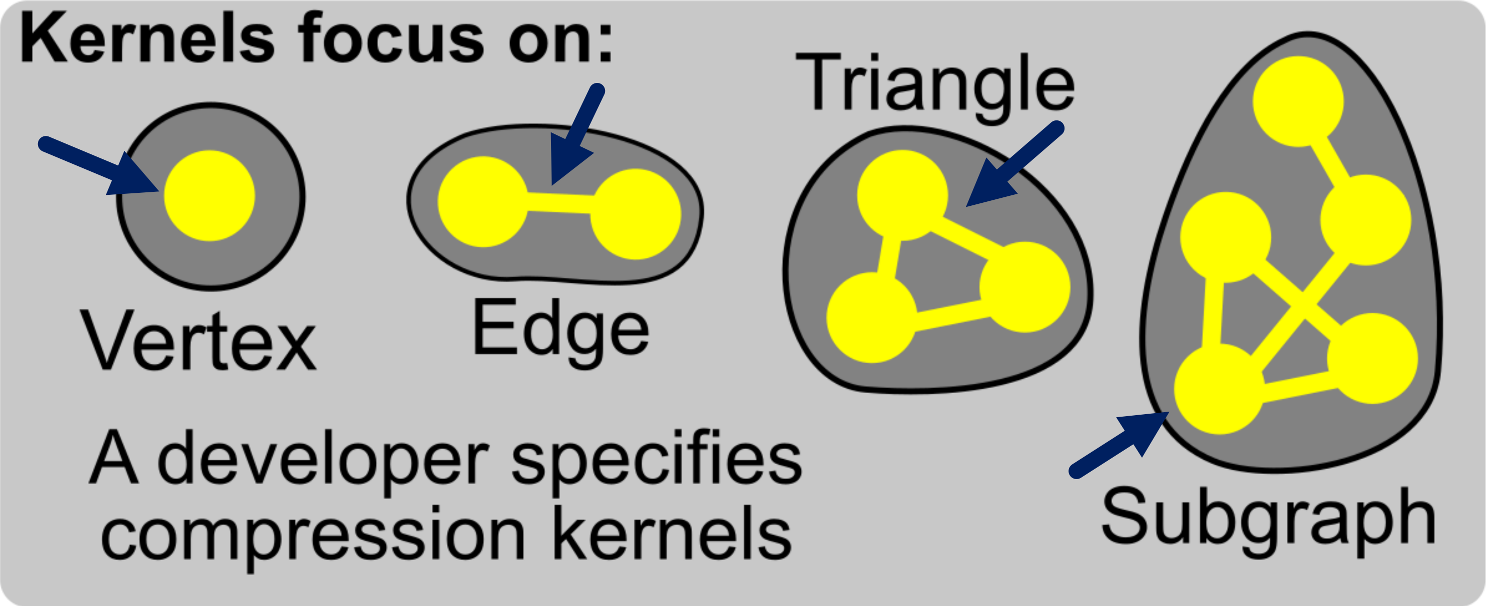


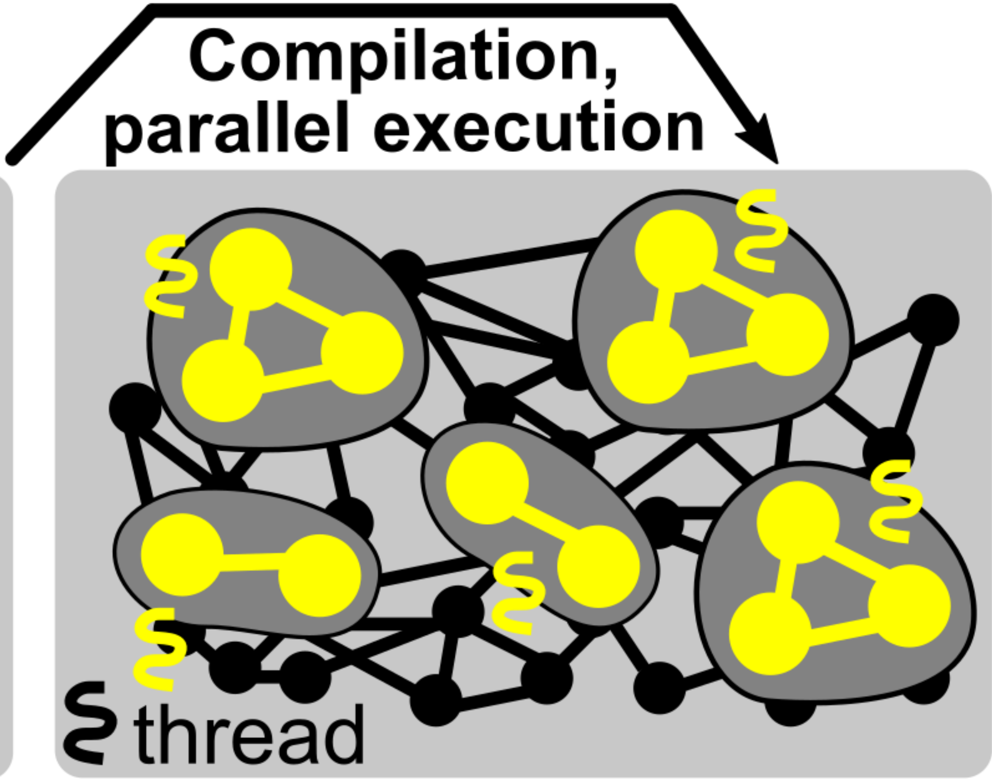
Number of ways [1] to sparsify (compress) a graph with *n* vertices

$$O\left(2^{\binom{n}{2}}\right)$$

[1] R. C. Entringer, P. Erdos. "On the Number of Unique Subgraphs of a Graph", Journal of Combinatorial Theory 1972

Slim Graph: Abstraction & Programming Model





Central concept is **compression kernels**: small code snippets that remove specified local parts of the graph

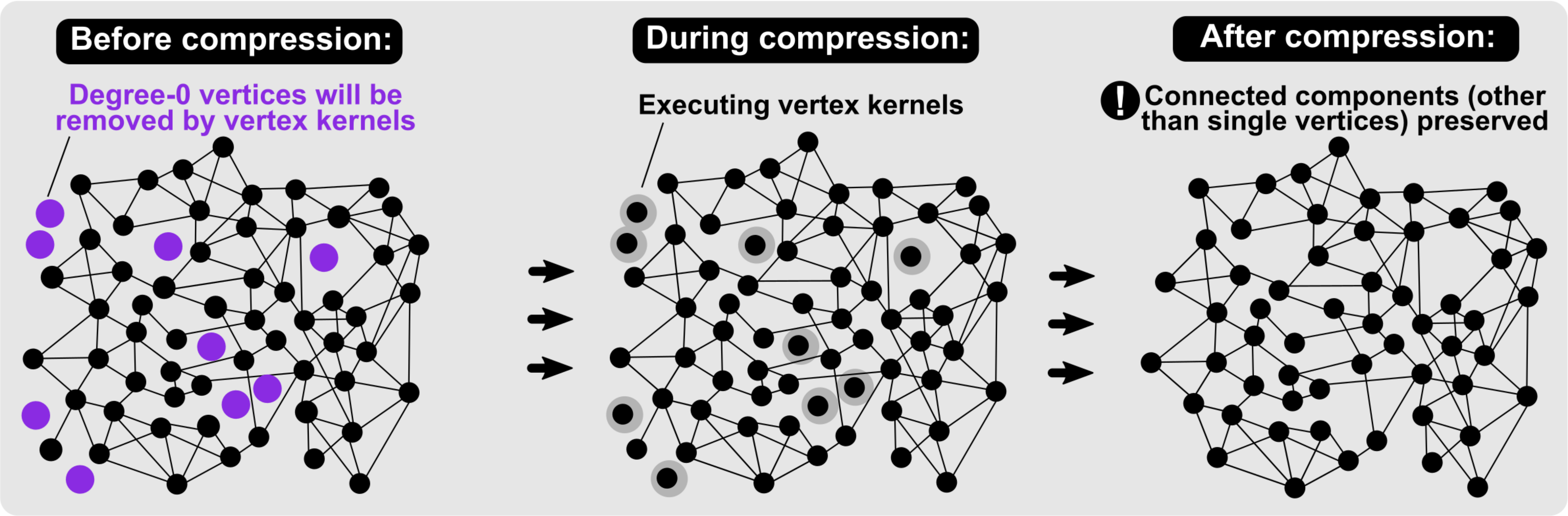
Different kernels enable different compression methods

Let's see some examples...

Slim Graph: Abstraction & Programming Model

Vertex kernels: removing degree-0 vertices

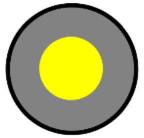
Connected components (other than single vertices) are preserved



Slim Graph: Abstraction & Programming Model

Vertex kernels: removing degree-0 vertices

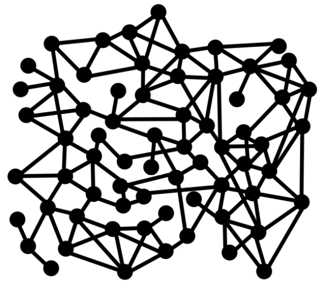
Connected components (other than single vertices) are preserved



```
kernel(V v) {
  if(v.deg==0)
    atomic SG.del(v);
}
```

Slim Graph: Abstraction & Programming Model

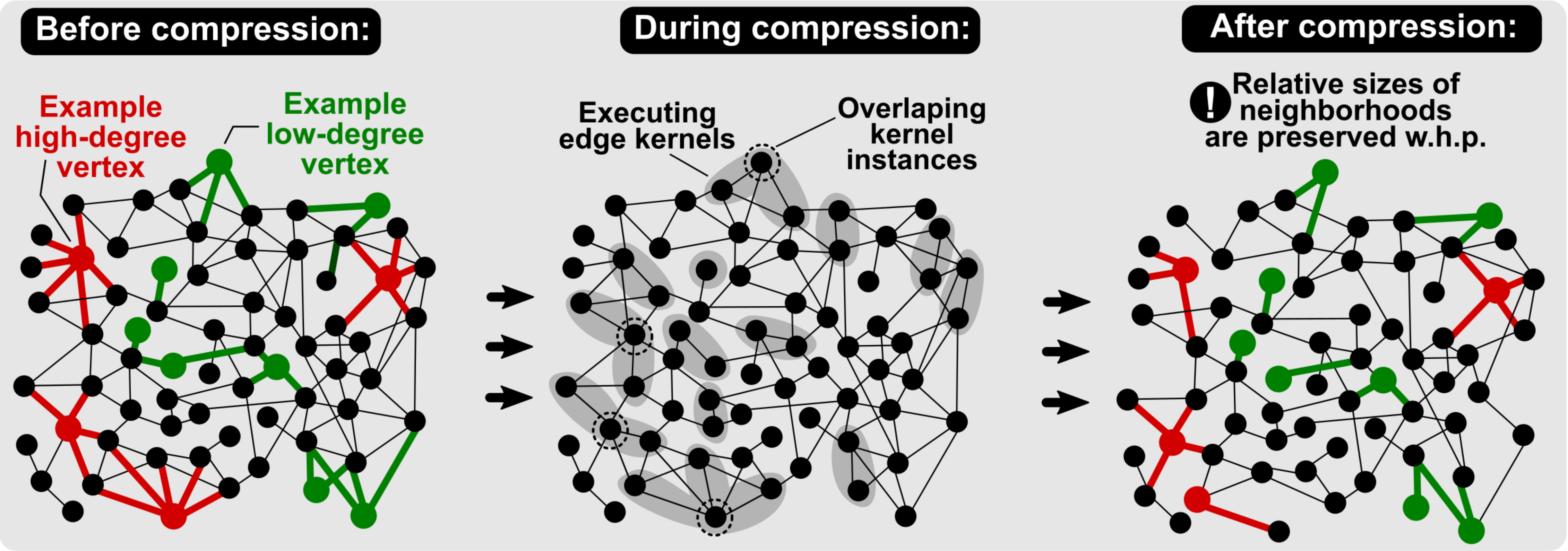
Input:



Edge kernels: random uniform sampling

Sparse/dense neighborhoods preserved w.h.p.

Relative neighborhood sizes provide information about., e.g., vertex importance



Slim Graph: Abstraction & Programming Model

Edge kernels: random uniform sampling

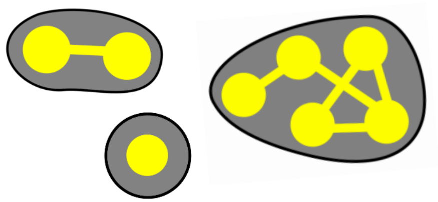
Sparse/dense neighborhoods preserved w.h.p.

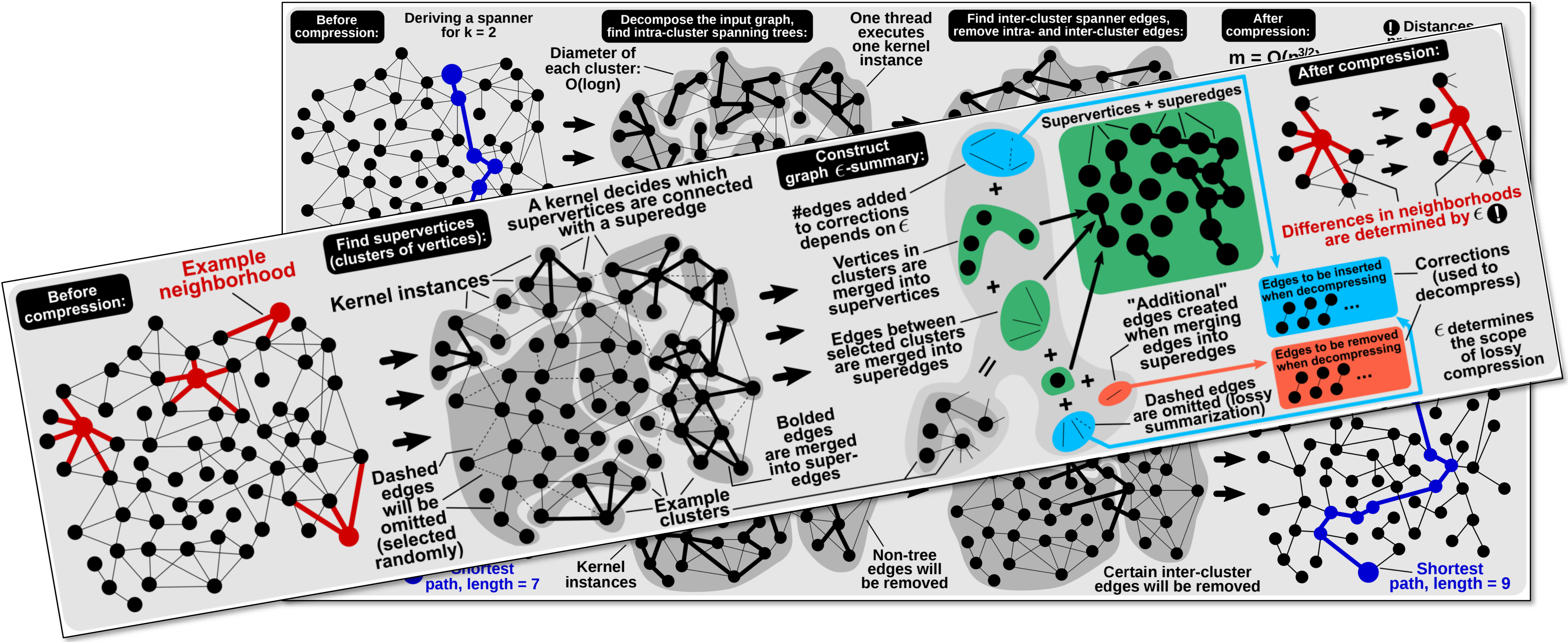
Relative neighborhood sizes provide information about., e.g., vertex importance

```
random_uniform_kernel(E e){
  double edge_stays = SG.p;
  if(edge_stays < SG.rand(0,1))
    atomic SG.del(e);
}
```

Slim Graph: Abstraction & Programming Model

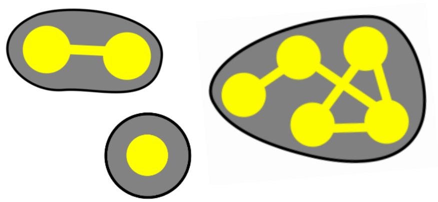
More kernels





Slim Graph: Abstraction & Programming Model

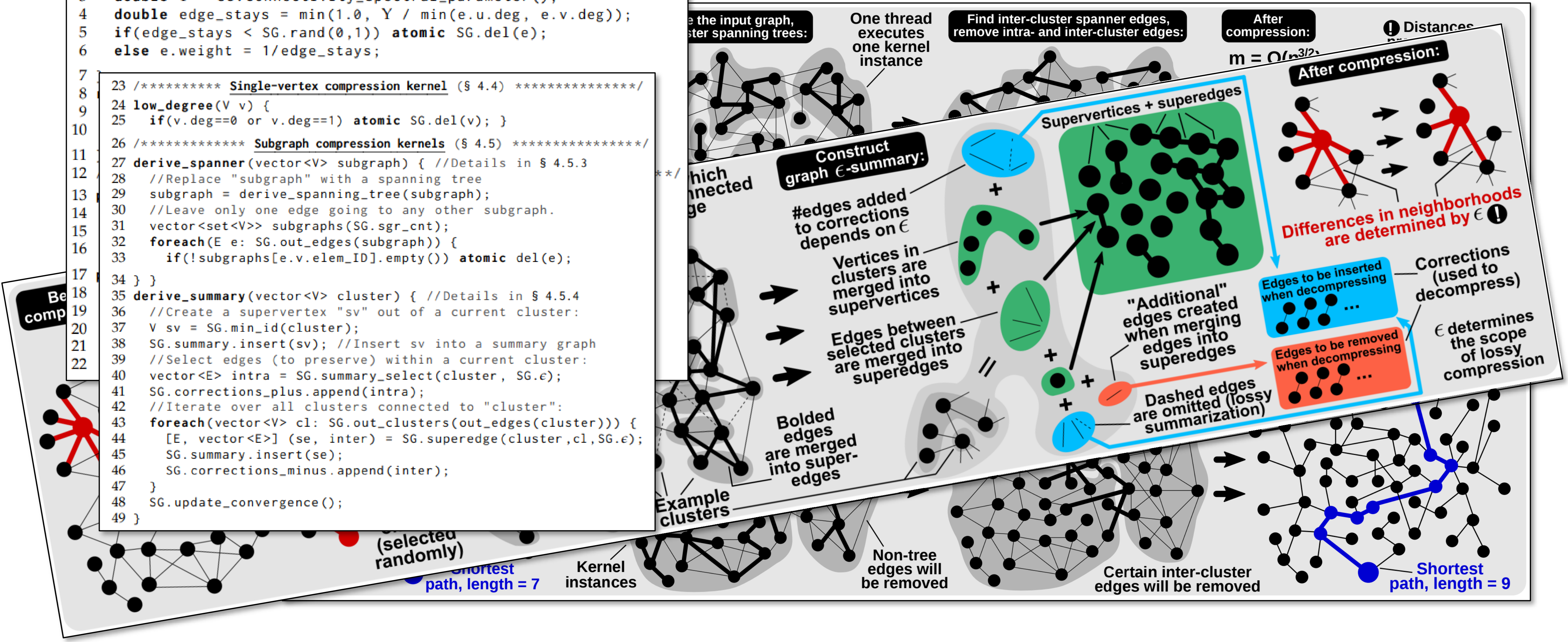
More kernels



```
/*********** Single-edge compression kernels (§ 4.2) *****************/
spectral_sparsify(E e) { //More details in § 4.2.1
  double Υ = SG.connectivity_spectral_parameter();
  double edge_stays = min(1.0, Υ / min(e.u.deg, e.v.deg));
  if(edge_stays < SG.rand(0,1)) atomic SG.del(e);
  else e.weight = 1/edge_stays;
```

```
/********** Single-vertex compression kernel (§ 4.4) ***************/
low_degree(V v) {
  if(v.deg==0 or v.deg==1) atomic SG.del(v); }

/************* Subgraph compression kernels (§ 4.5) ****************/
derive_spanner(vector<V> subgraph) { //Details in § 4.5.3
  //Replace "subgraph" with a spanning tree
  subgraph = derive_spanning_tree(subgraph);
  //Leave only one edge going to any other subgraph.
  vector<set<V>> subgraphs(SG.sgr_cnt);
  foreach(E e: SG.out_edges(subgraph)) {
    if(!subgraphs[e.v.elem_ID].empty()) atomic del(e);
} }
derive_summary(vector<V> cluster) { //Details in § 4.5.4
  //Create a supervertex "sv" out of a current cluster:
  V sv = SG.min_id(cluster);
  SG.summary.insert(sv); //Insert sv into a summary graph
  //Select edges (to preserve) within a current cluster:
  vector<E> intra = SG.summary_select(cluster, SG.ϵ);
  SG.corrections_plus.append(intra);
  //Iterate over all clusters connected to "cluster":
  foreach(vector<V> cl: SG.out_clusters(out_edges(cluster))) {
    [E, vector<E>] (se, inter) = SG.superedge(cluster,cl,SG.ϵ);
    SG.summary.insert(se);
    SG.corrections_minus.append(inter);
  }
  SG.update_convergence();
}
```



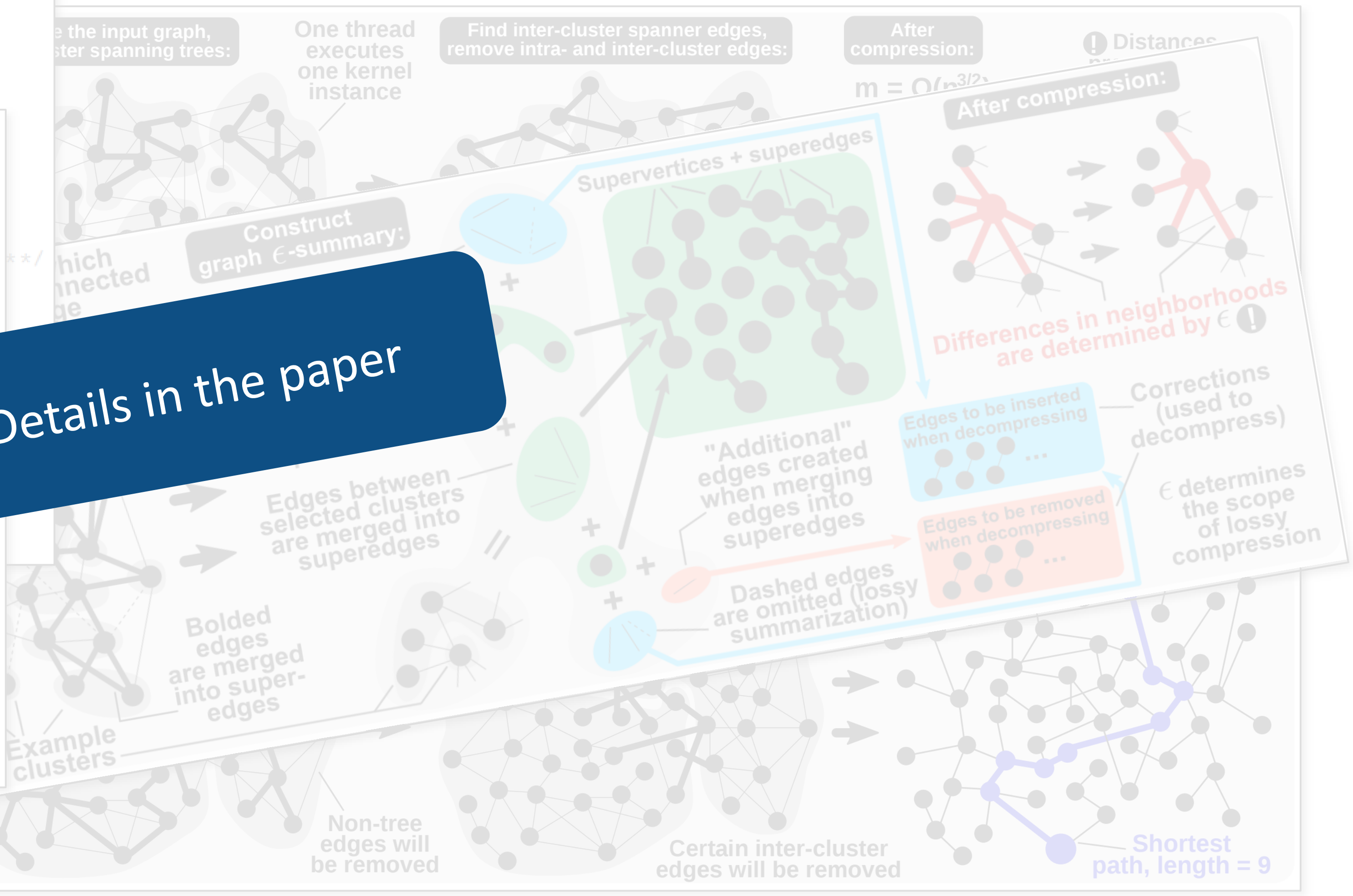
Slim Graph: Abstraction & Programming Model

More kernels

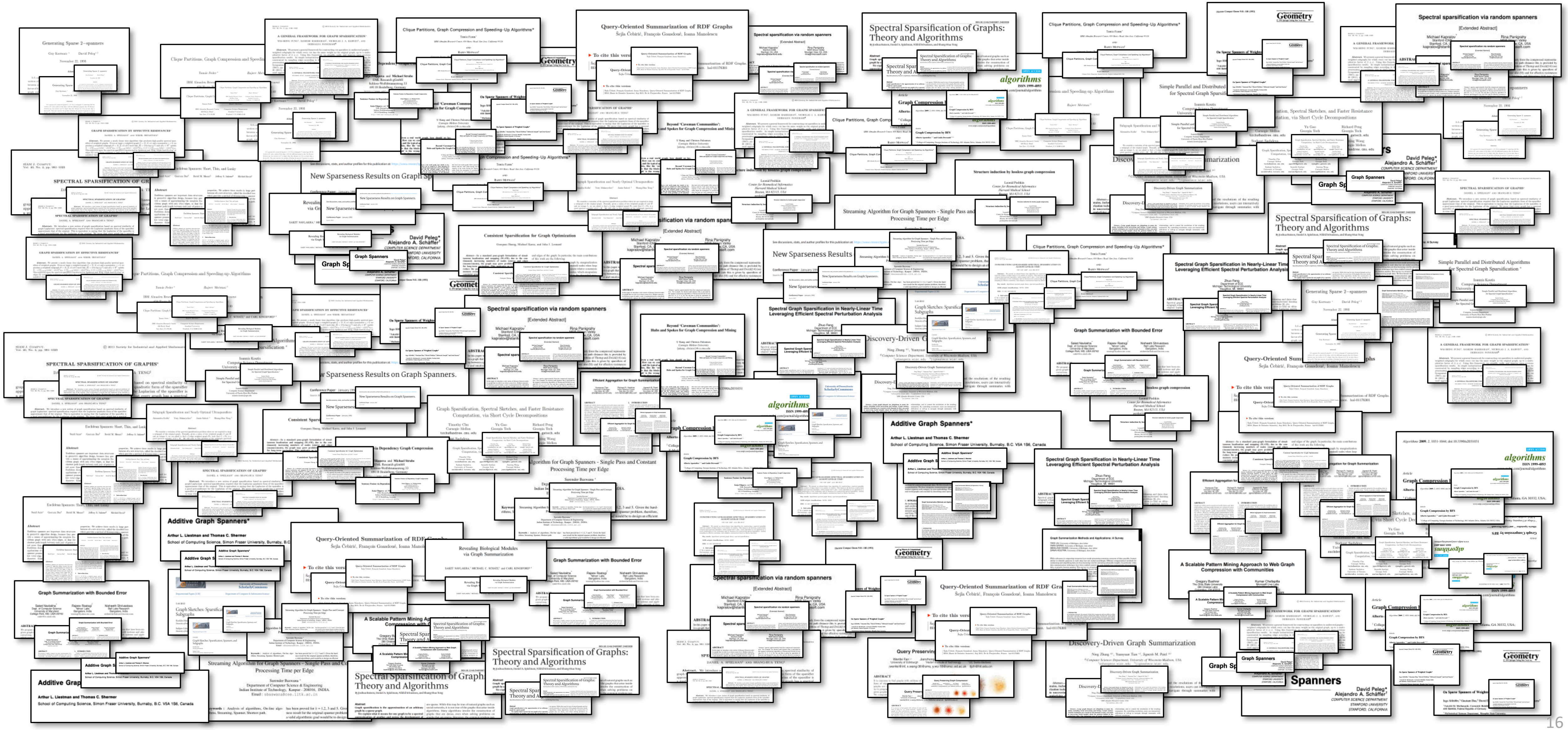
Details in the paper

```
/*********** Single-edge compression kernels (§ 4.2) *****************/
spectral_sparsify(E e) { //More details in § 4.2.1
  double Y = SG.connectivity_spectral_parameter();
  double edge_stays = min(1.0, Y / min(e.u.deg, e.v.deg));
  if(edge_stays < SG.rand(0,1)) atomic SG.del(e);
  else e.weight = 1/edge_stays;
```

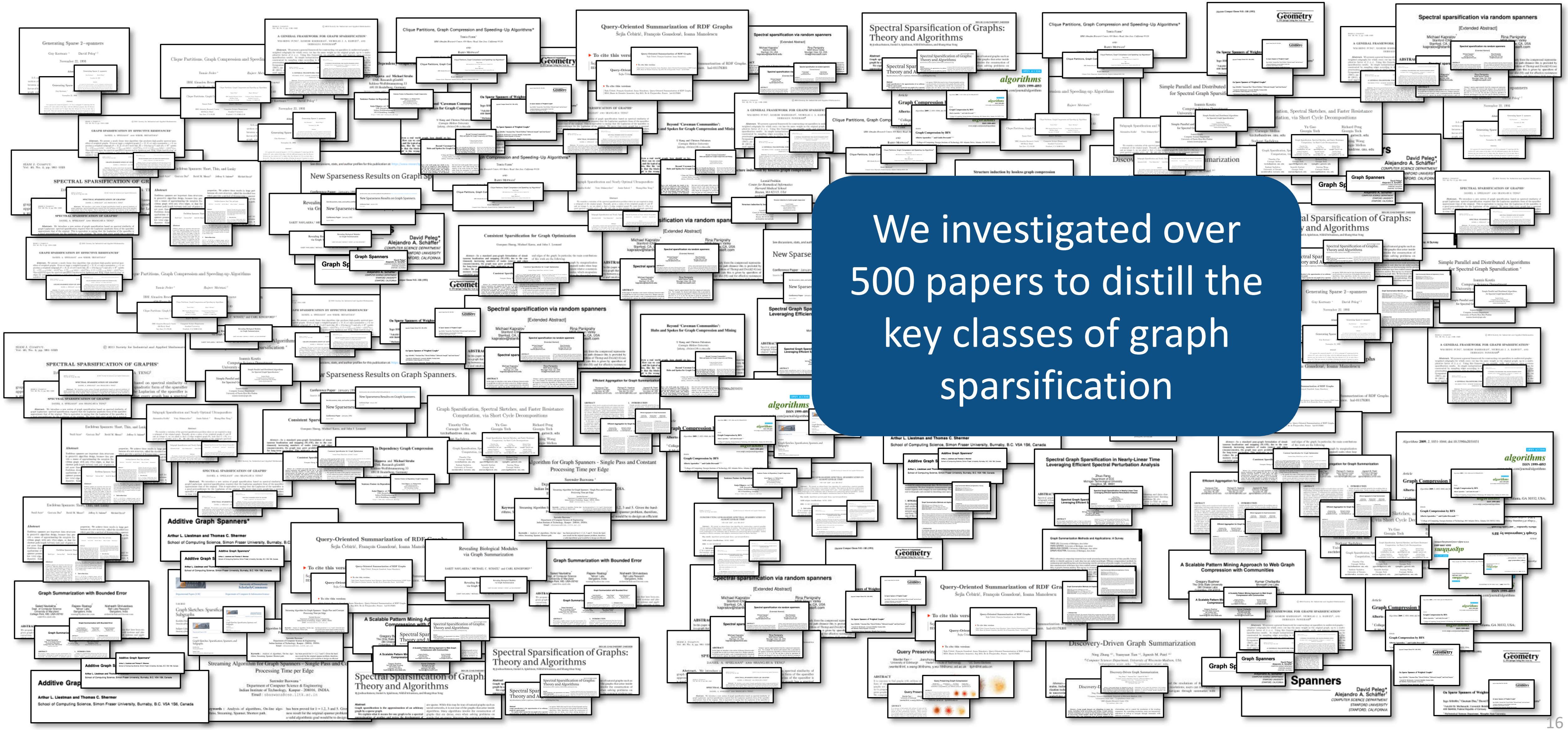
```
/*********** Single-vertex compression kernel (§ 4.4) ***************/
low_degree(V v) {
  if(v.deg==0 or v.deg==1) atomic SG.del(v); }
/************* Subgraph compression kernels (§ 4.5) ****************/
derive_spanner(vector<V> subgraph) { //Details in § 4.5.3
  //Replace "subgraph" with a spanning tree
  subgraph = derive_spanning_tree(subgraph);
  //Leave only one edge going to any other subgraph.
  vector<set<V>> subgraphs(SG.sgr_cnt);
  foreach(E e: SG.out_edges(subgraph)) {
    if(!subgraphs[e.v.elem_ID].empty()) atomic del(e);
} }
derive_summary(vector<V> cluster) { //Details in § 4.5.4
  //Create a supervertex "sv" out of a current cluster:
  V sv = SG.min_id(cluster);
  SG.summary.insert(sv); //Insert sv into a summary graph
  //Select edges (to preserve) within a current cluster:
  vector<E> intra = SG.summary_select(cluster, SG.ε);
  SG.corrections_plus.append(intra);
  //Iterate over all clusters connected to "cluster":
  foreach(vector<V> cl: SG.out_clusters(out_edges(cluster))) {
    [E, vector<E>] (se, inter) = SG.superedge(cluster,cl,SG.ε);
    SG.summary.insert(se);
    SG.corrections_minus.append(inter);
  }
  SG.update_convergence();
}
```



How expressive is the compression kernel abstraction?



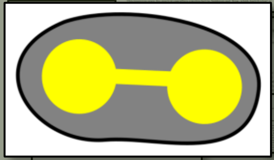
How expressive is the **compression kernel abstraction?**

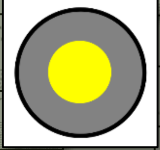


We investigated over 500 papers to distill the key classes of graph sparsification

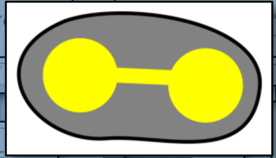
How expressive is the compression kernel abstraction?

Random uniform (and other forms of) sampling

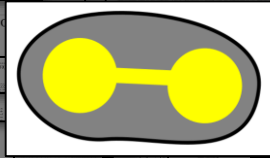




Spectral sparsifiers (preserve spectra)

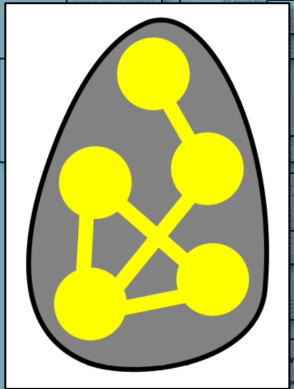


Cut sparsifiers (preserve cuts)



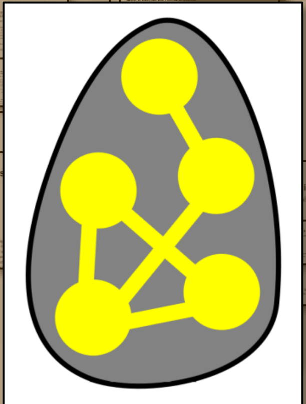
Compression kernels: an abstraction that enables expressing fundamental classes of sparsification

We investigated over 500 papers to distill the key classes of graph sparsification



Summarizations (preserve neighborhoods)

Spanners (preserve pairwise distances)



Others

Slim Graph delivers a simple, intuitive, versatile ...

1 ... **Abstraction & programming model** for easy development and rapid prototyping of lossy graph compression methods

2 ... **Compression method** that preserves different graph properties that are important for the practice of graph processing

3 ... **Criterion (criteria?)** to assess the accuracy of lossy graph compression methods

4 ... *High-performance* and *extensible* **system** for implementing and executing lossy graph compression

Solved

Number of ways [1] to sparsify (compress) a graph with *n* vertices

$$O\left(2^{\binom{n}{2}}\right)$$

[1] R. C. Entringer, P. Erdos. "On the Number of Unique Subgraphs of a Graph", Journal of Combinatorial Theory 1972

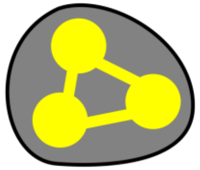
Slim Graph: A Novel Compression Method "Triangle Reduction"

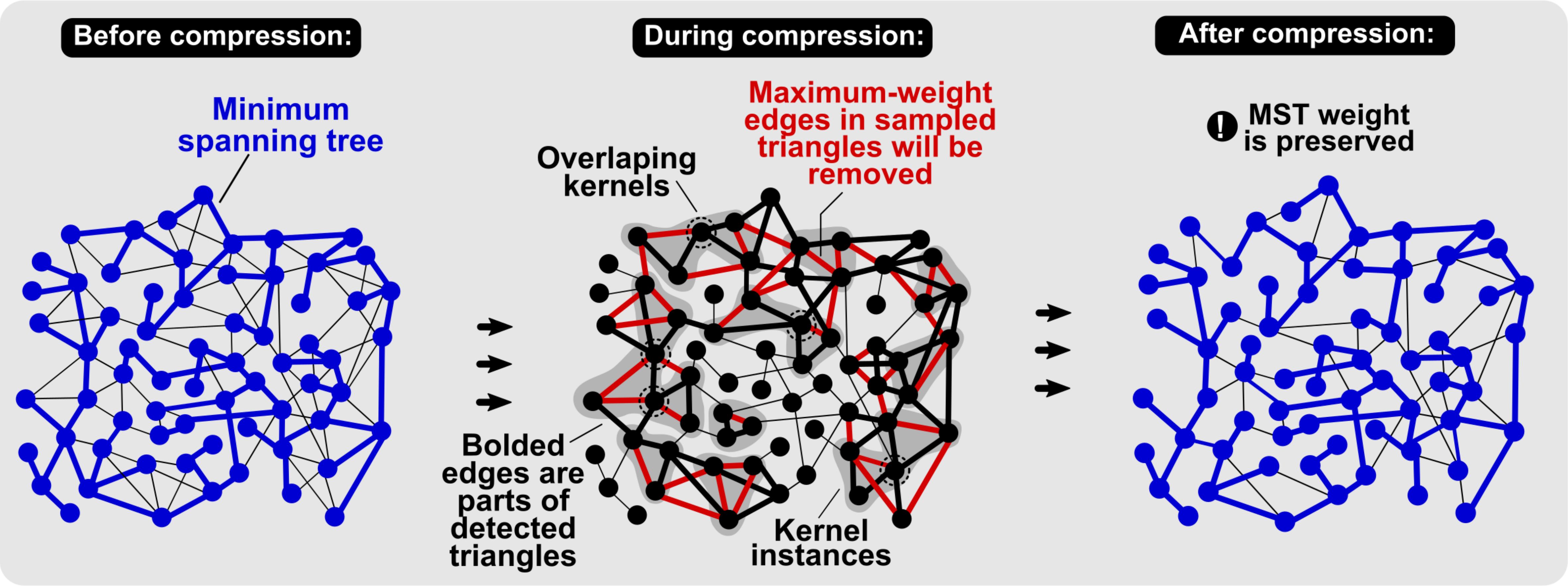
Each triangle, with a certain selected probability p, is „reduced" – some of its parts are removed.

Here, we consider one edge in a triangle

As we show later, it preserves different graph properties

Triangle kernels: Triangle Reduction





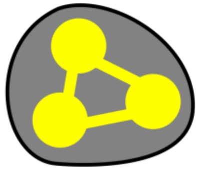
Slim Graph: A Novel Compression Method "Triangle Reduction”

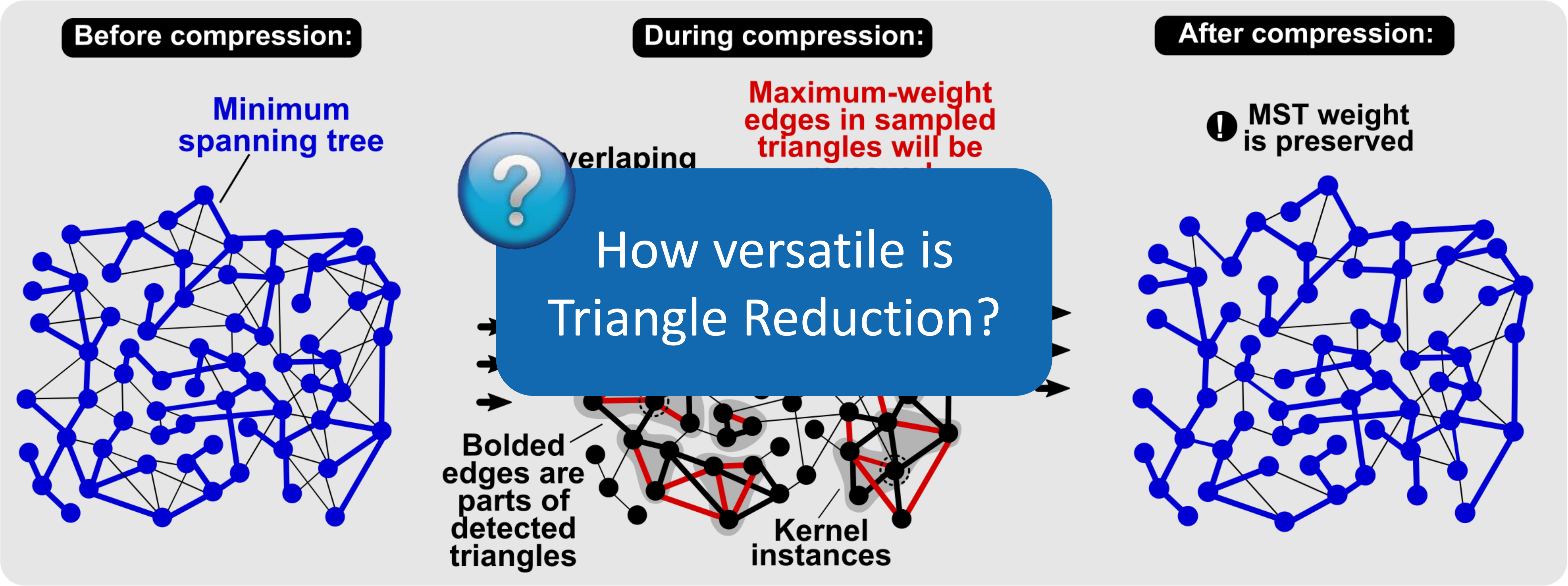
Each triangle, with a certain selected probability p, is „reduced” – some of its parts are removed.

Here, we consider one edge in a triangle

As we show later, it preserves different graph properties

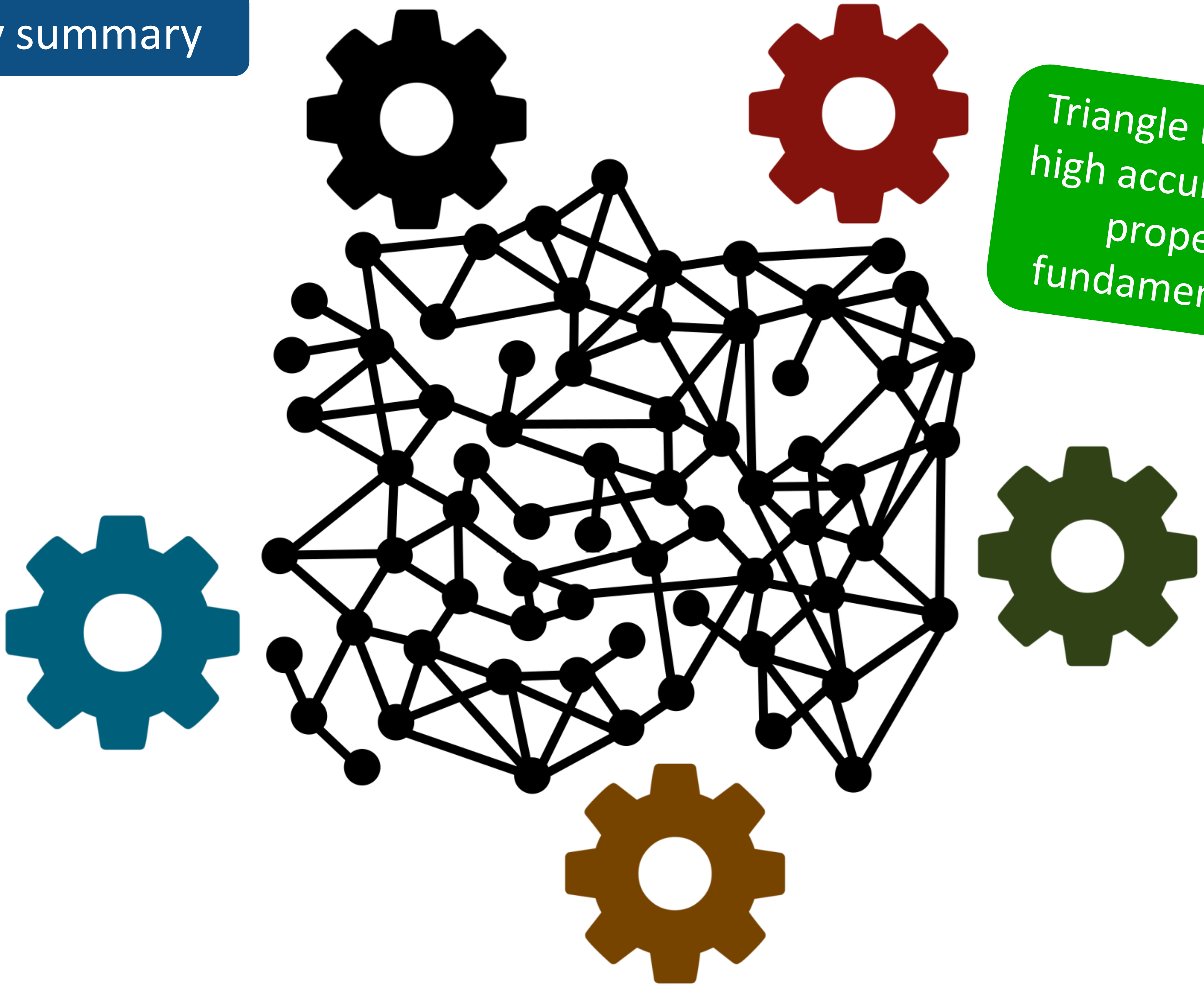
Triangle kernels: Triangle Reduction





How versatile is Triangle Reduction?

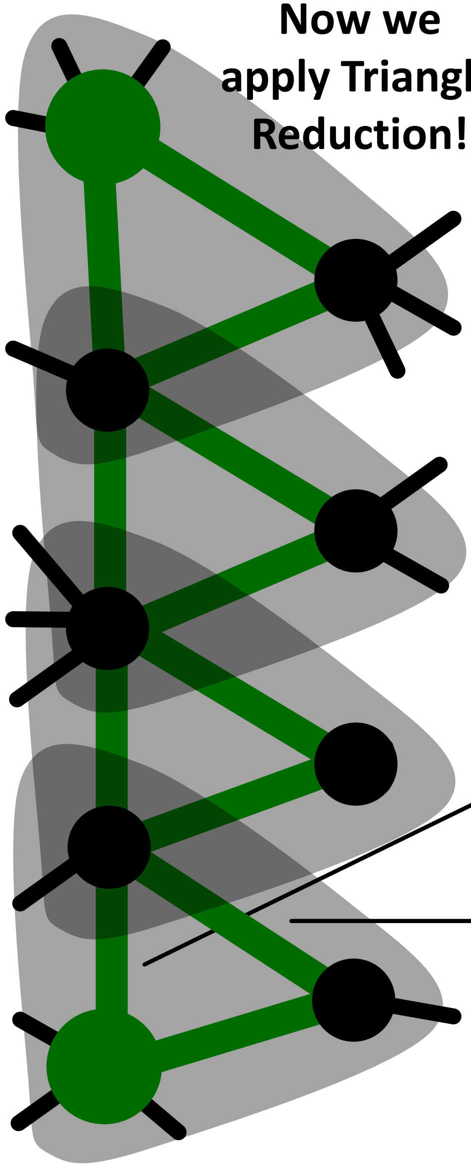
The key summary



Triangle Reduction preserves (with high accuracy) representative graph properties associated with fundamental classes of workloads

The key intuition behind preserving distances by Triangle Reduction

Now we apply Triangle Reduction!



Distances

Graph traversals (e.g., BFS, SSSP)

By how much can distances increase?

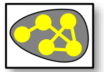
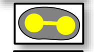
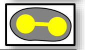
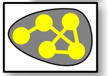
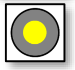
Distances increase by **at most 2x**

This is the shortest path between two green vertices in the uncompressed graph

In the worst-case, this will be a new shortest path in the compressed graph

We provide proofs, derivations, and discussions for many other properties…

Theoretical Analysis of Slim Graph

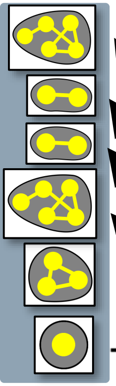
	$\|V\|$	$\|E\|$	Shortest s-t path length	Average path length	Diameter	Average degree	Maximum degree	#Triangles	#Connected components	Chromatic number	Max. indep. set size	Max. cardinal. matching size
												
												
												
												
												
												

Theoretical Analysis of Slim Graph

Different graph properties

(…+ some others ☺)

	$\lvert V \rvert$	$\lvert E \rvert$	Shortest s-t path length	Average path length	Diameter	Average degree	Maximum degree	#Triangles	#Connected components	Chromatic number	Max. indep. set size	Max. cardinal. matching size



Different compression methods

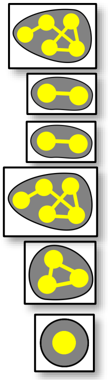
We analyzed the impact of 6 fundamental lossy graph compression methods (implemented with different compression kernels) on >12 different graph properties

Theoretical Analysis of Slim Graph

Different graph properties **(…+ some others ☺)**

We derive / provide 60+ bounds (it's actually close to 100 now)

	$\|V\|$	$\|E\|$	Shortest s-t path length	Average path length	Diameter	Average degree	Maximum degree	#Triangles	#Connected components	Chromatic number	Max. indep. set size	Max. cardinal. matching size



Different compression methods

Each field is a separate result

We analyzed the impact of 6 fundamental lossy graph compression methods (implemented with different compression kernels) on >12 different graph properties

Theoretical Analysis of Slim Graph

Different graph properties (...+ some others ☺)

We derive / provide 60+ bounds (it's actually close to 100 now)

	$\|V\|$	$\|E\|$	Shortest s-t path length	Average path length	Diameter	Average degree	Maximum degree	#Triangles	#Connected components	Chromatic number	Max. indep. set size	Max. cardinal. matching size
Original graph	n	m	$\mathcal{P}$	$\overline{\mathcal{P}}$	D	$\overline{d}$	d	T	$\mathcal{C}$	C_R	$\widehat{I}_S$	$\widehat{M}_C$
Summarization	n	$m \pm 2\epsilon m$	$1, \ldots, \infty$	$1, \ldots, \infty$	$1, \ldots, \infty$	$\overline{d} \pm \epsilon \overline{d}$	$d \pm \epsilon d$	$T \pm 2\epsilon m$	$\mathcal{C} \pm 2\epsilon m$	$C_R \pm 2\epsilon m$	$\widehat{I}_S \pm 2\epsilon m$	$\widehat{M}_C \pm 2\epsilon m$
Edge sampling	n	$(1-p)m$	∞	∞	∞	$(1-p)\overline{d}$	$(1-p)d$	$(1-p^3)T$	$\leq \mathcal{C} + pm$	$\geq C_R - pm$	$\leq \widehat{I}_S + pm$	$\geq \widehat{M}_C - pm$
Spectral sparsifiers	n	$\tilde{O}(n/\epsilon^2)$	$\leq n$	$\leq n$	$\leq n$	$\tilde{O}(1/\epsilon^2)$	$\geq d/2(1+\epsilon)$	$\tilde{O}(n^{3/2}/\epsilon^3)$	$\overset{w.h.p.}{=} \mathcal{C}$	$\leq d/2(1+\epsilon)$	$\geq 2(1+\epsilon)n/d$	≥ 0
Spanners	n	$O(n^{1+1/k})$	$O(k\mathcal{P})$	$O(k\overline{\mathcal{P}})$	$O(kD)$	$O(n^{1/k})$	$\leq d$	$O(n^{1+2/k})$	$\mathcal{C}$	$O(n^{1/k} \log n)$	$\Omega\left(\frac{n^{1-1/k}}{\log n}\right)$	≥ 0
Triangle Reduction	n	$\leq m - \frac{pT}{3d}$	$\overset{w.h.p.}{\leq} \mathcal{P} + p\mathcal{P}$	$\leq \overline{\mathcal{P}} + \frac{pT}{n(n-1)}$	$\overset{w.h.p.}{\leq} D + pD$	$\leq \overline{d} - \frac{pT}{dn}$	$\geq d/2$	$\leq (1 - \frac{p}{d})T$	$\mathcal{C}$	$\geq C_R - pT$	$\leq \widehat{I}_S + pT$	$\geq \widehat{M}_C/2$
Vertex Sampling	$n-k$	$m-k$	$\mathcal{P}$	$\geq \overline{\mathcal{P}} - \frac{kD}{n}$	$\geq D - 2$	$\geq \overline{d} - \frac{k}{n}$	d	T	$\mathcal{C}$	C_R	$\geq \widehat{I}_S - k$	$\geq \widehat{M}_C - k$

Different compression methods

Each field is a separate result

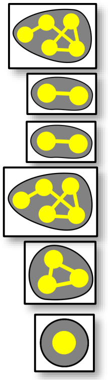
We analyzed the impact of 6 fundamental lossy graph compression methods (implemented with different compression kernels) on >12 different graph properties

Theoretical Analysis of Slim Graph

Different graph properties **(...+ some others ☺)**

We derive / provide 60+ bounds (it's actually close to 100 now)

	$\|V\|$	$\|E\|$	Shortest s-t path length	Average path length	Diameter	Average degree	Maximum degree	#Triangles	#Connected components	Chromatic number	Max. indep. set size	Max. cardinal. matching size
Original graph	n	m	$\mathcal{P}$	$\overline{\mathcal{P}}$	D	$\bar{d}$	d	T	$\mathcal{C}$	C_R	$\widehat{I}_S$	$\widehat{M}_C$
Summarization	n	$m \pm 2\epsilon m$	$1,\dots,\infty$	$1,\dots,\infty$	$1,\dots,\infty$	$\bar{d} \pm \epsilon\bar{d}$	[illegible]	$T \pm 2\epsilon m$	$\mathcal{C} \pm 2\epsilon m$	$C_R \pm 2\epsilon m$	$\widehat{I}_S \pm 2\epsilon m$	$\widehat{M}_C \pm 2\epsilon m$
Edge sampling	n	$(1-p)m$	∞	∞	[illegible]	[illegible]	[illegible]	$1-p^3)T$	$\leq \mathcal{C} + pm$	$\geq C_R - pm$	$\leq \widehat{I}_S + pm$	$\geq \widehat{M}_C - pm$
Spectral sparsifiers	n	$\tilde{O}(n/\epsilon^2)$	$\leq n$	[illegible]	[illegible]	[illegible]	[illegible]	$3/2/\epsilon^3)$	$\overset{w.h.p.}{=} \mathcal{C}$	$\leq d/2(1+\epsilon)$	$\geq 2(1+\epsilon)n/d$	≥ 0
Spanners	n	$O(n^{1+1/k})$	$O(k\mathcal{P})$	$O($	[illegible]	[illegible]	[illegible]	$1+2/k)$	$\mathcal{C}$	$O(n^{1/k}\log n)$	$\Omega\left(\frac{n^{1-1/k}}{\log n}\right)$	≥ 0
Triangle Reduction	n	$\leq m - \frac{pT}{3d}$	$\overset{w.h.p.}{\leq} \mathcal{P} + p\mathcal{P}$	$\leq \overline{\mathcal{P}} + \frac{pT}{n(n-}$	[illegible]	dn	$\geq d/2$	$\leq (1 - \frac{p}{d})T$	$\mathcal{C}$	$\geq C_R - pT$	$\leq \widehat{I}_S + pT$	$\geq \widehat{M}_C/2$
Vertex Sampling	$n-k$	$m-k$	$\mathcal{P}$	$\geq \overline{\mathcal{P}} - \frac{kD}{n}$	$\geq D-2$	$\geq \bar{d} - \frac{k}{n}$	d	T	$\mathcal{C}$	C_R	$\geq \widehat{I}_S - k$	$\geq \widehat{M}_C - k$



Looks complex - no worries, we will not go over it here ☺

Different compression methods

Each field is a separate result

We analyzed the impact of 6 fundamental lossy graph compression methods (implemented with different compression kernels) on >12 different graph properties

Slim Graph delivers a simple, intuitive, versatile ...

1 ... **Abstraction & programming model** for easy development and rapid prototyping of lossy graph compression methods

Solved

Number of ways [1] to sparsify (compress) a graph with *n* vertices

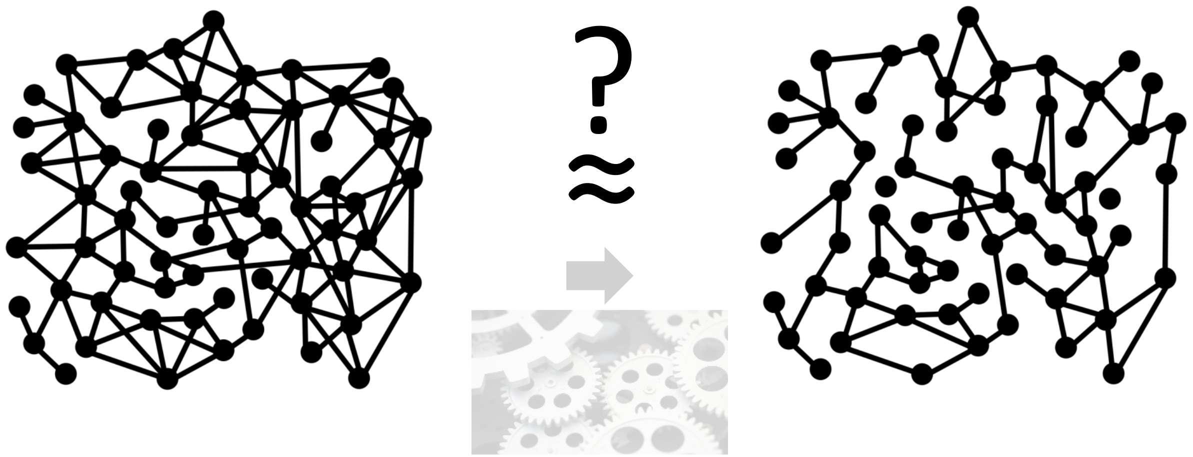
$$O\left(2^{\binom{n}{2}}\right)$$

[1] R. C. Entringer, P. Erdos. "On the Number of Unique Subgraphs of a Graph", Journal of Combinatorial Theory 1972

2 ... **Compression method** that preserves different graph properties that are important for the practice of graph processing

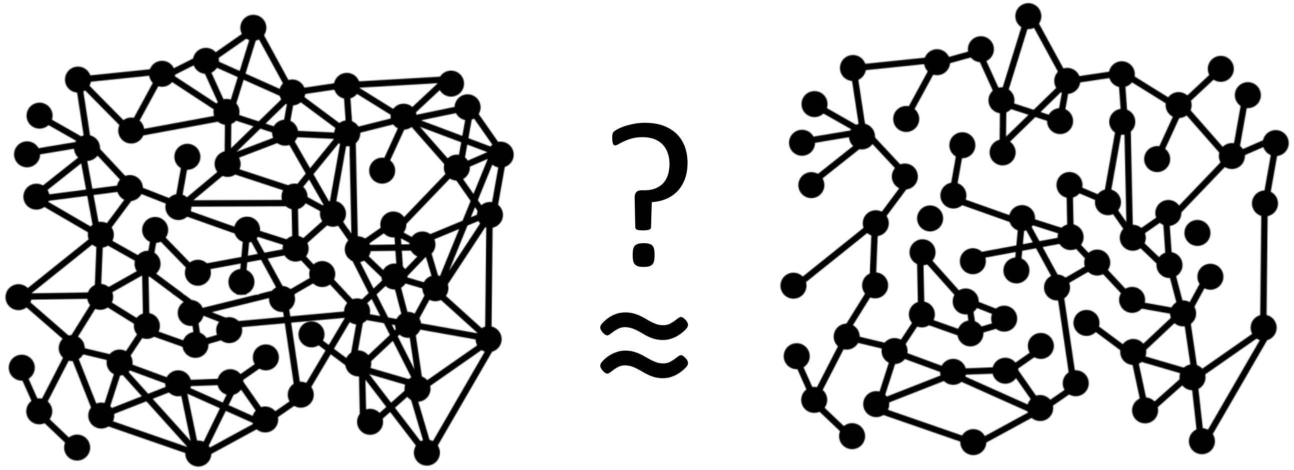
Solved

3 ... **Criterion (criteria?)** to assess the accuracy of lossy graph compression methods

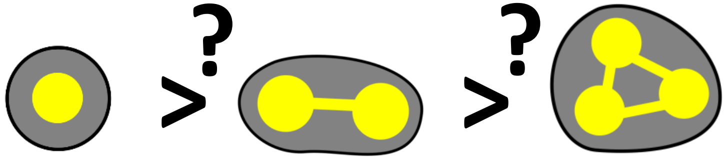


4 ... *High-performance* and *extensible* **system** for implementing and executing lossy graph compression

Slim Graph: Criteria for Compression Accuracy

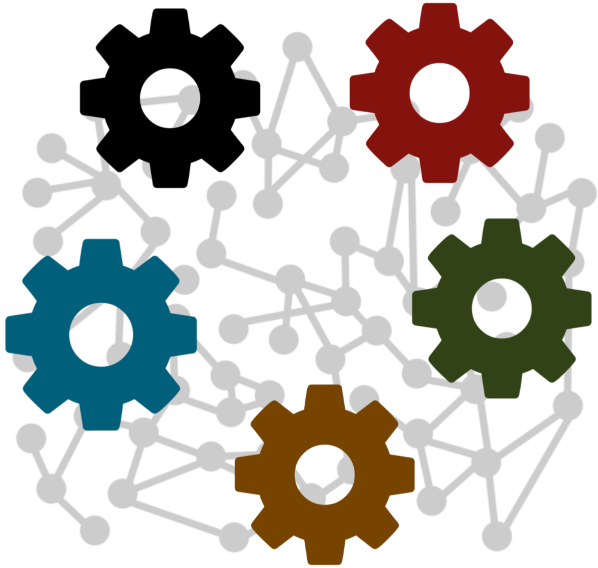


Which compression scheme is better (= more accuracy) for which graph property?



One can analyze this in theory. This gives fundamental insights. But… it may be very hard or impossible.

Slim Graph offers different metrics based on **the type of the outcome** of **specific graph processing workloads**

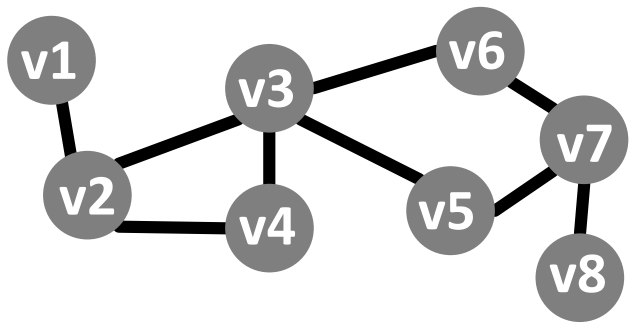


Slim Graph: Criteria for Compression Accuracy

Type of workload output: **vertex importance scores**

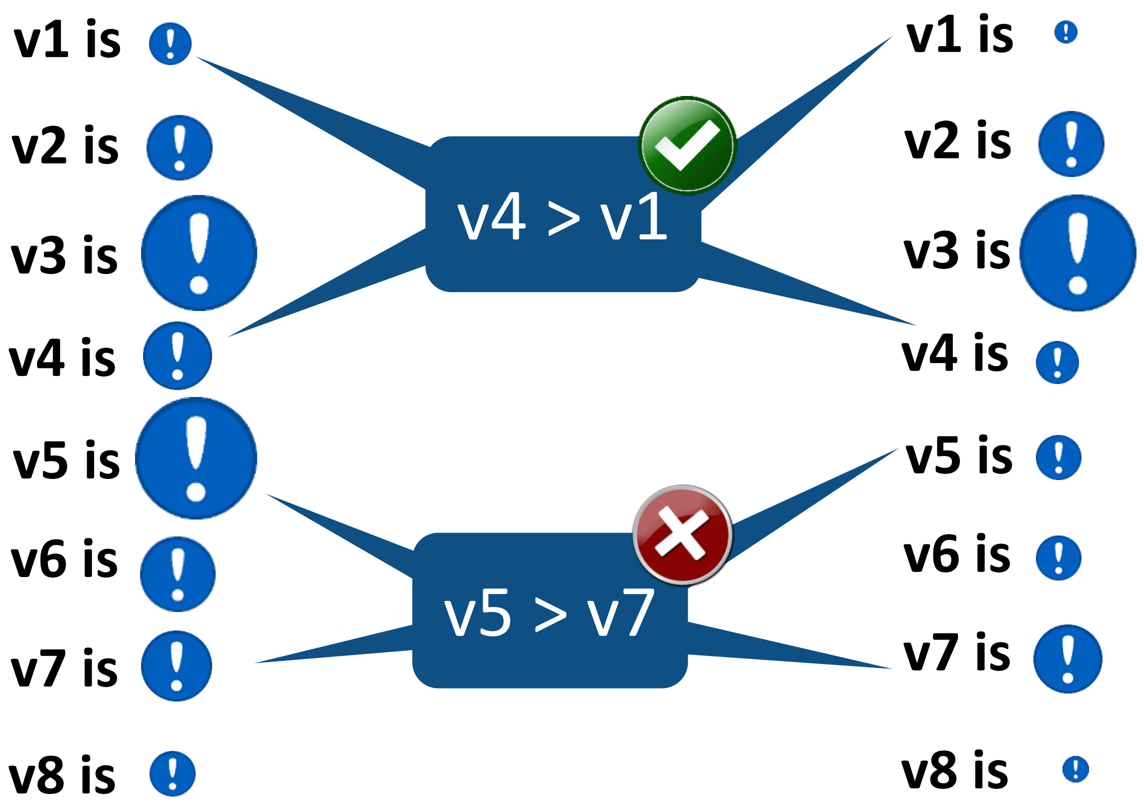
Examples: Degree Centrality, Betweenness Centrality, Katz Centrality

Metric: #reorderings, i.e., "How many pairs of vertices swapped their importance after compression?"



Let's use degree centrality

Time to compress! ☺



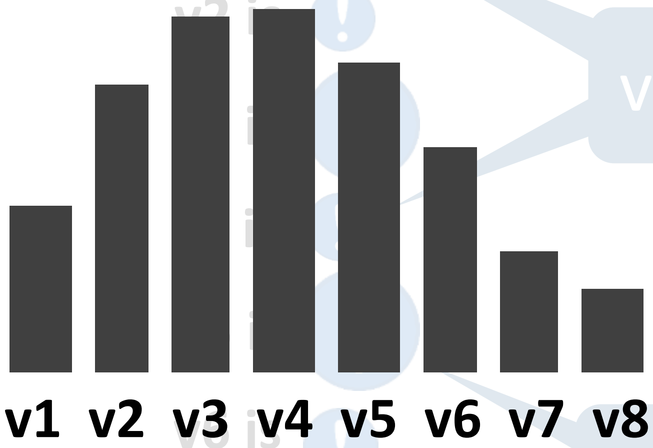
Slim Graph: Criteria for Com... Accu...

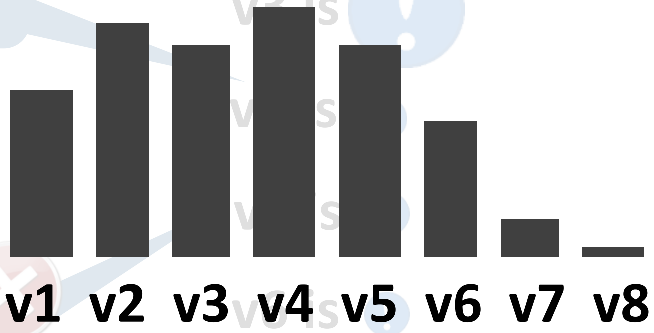
Please check the paper for details on the precise statistical formulation ☺

We have metrics for other classes of workloads, for example...

Metric: #reorderings, i.e., "How many pairs of vertices swapped their importance after compression"

Centrality, Katz Centrality





DIVERGENCE(P(x): Probability distribution before compression , Q(x): Probability distribution after compression)

Slim Graph delivers a simple, intuitive, versatile ...

1 ... **Abstraction & programming model** for easy development and rapid prototyping of lossy graph compression methods

Solved

Number of ways [1] to sparsify (compress) a graph with *n* vertices

$$O\left(2^{\binom{n}{2}}\right)$$

[1] R. C. Entringer, P. Erdos. "On the Number of Unique Subgraphs of a Graph", Journal of Combinatorial Theory 1972

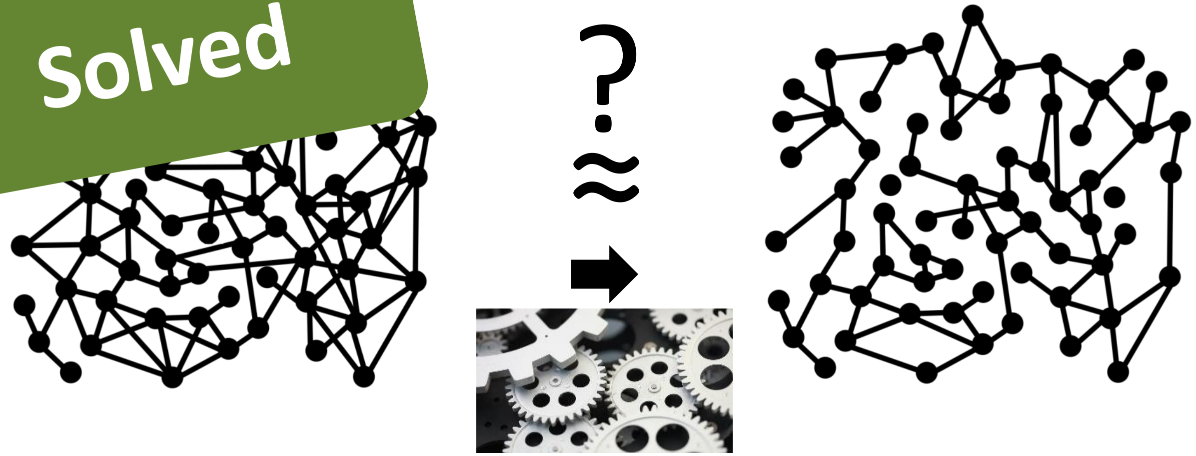
2 ... **Compression method** that preserves different graph properties that are important for the practice of graph processing

Solved

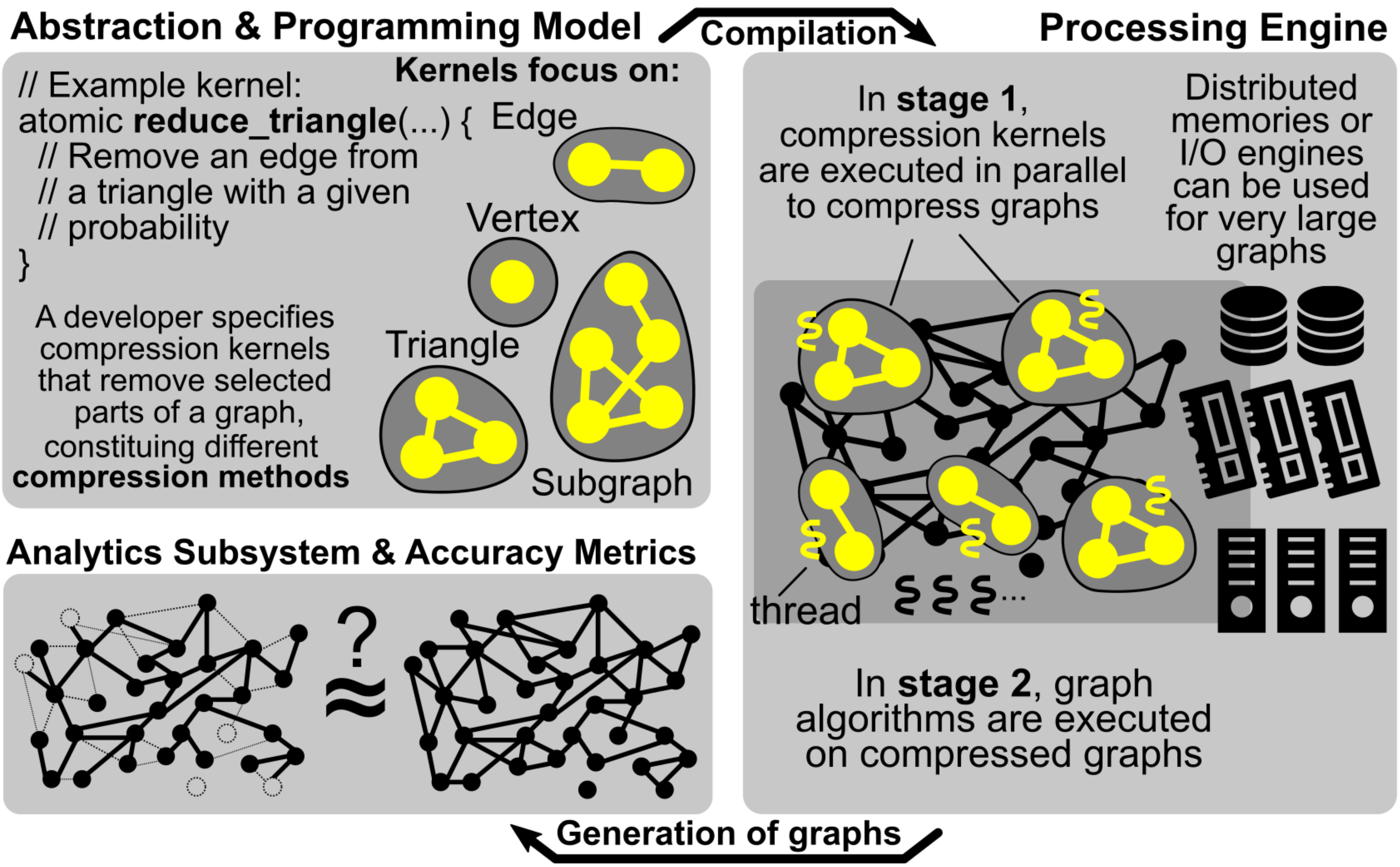
3 ... **Criterion (criteria?)** to assess the accuracy of lossy graph compression methods

Solved

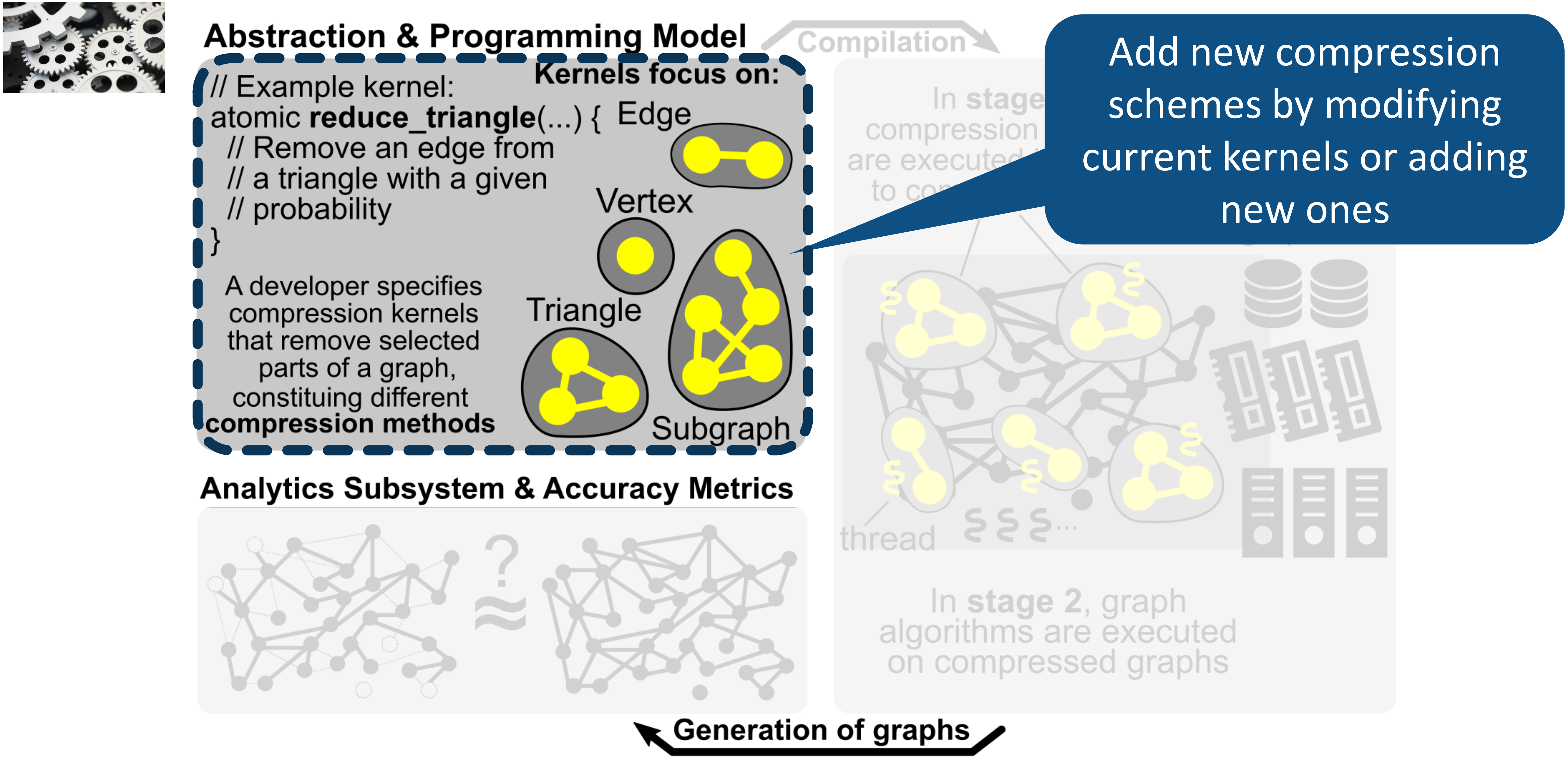
4 ... *High-performance* and *extensible* **system** for implementing and executing lossy graph compression



Slim Graph: High-Performance Extensible System

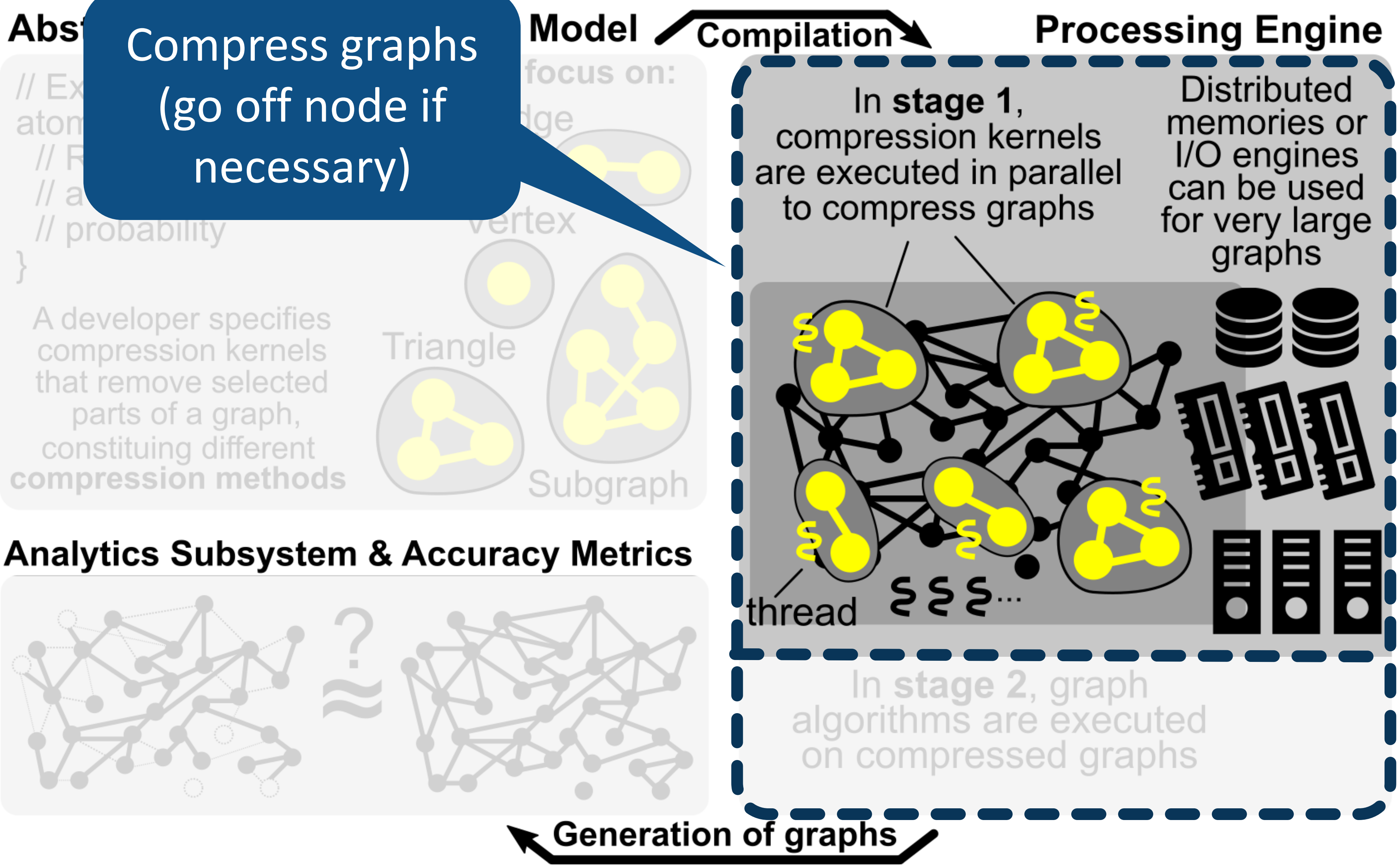


Slim Graph: High-Performance Extensible System

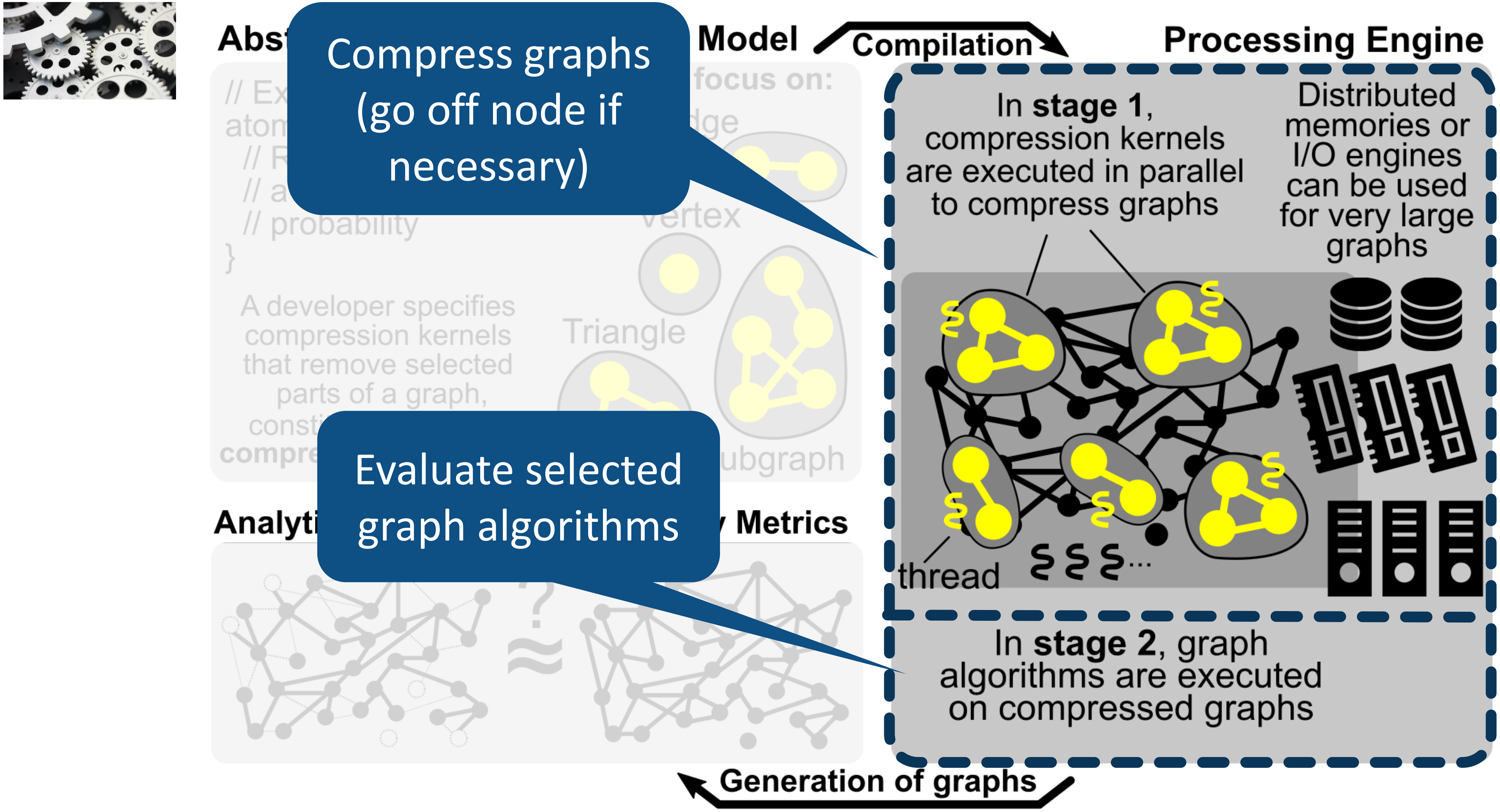


Slim Graph: High-Performance Extensible System

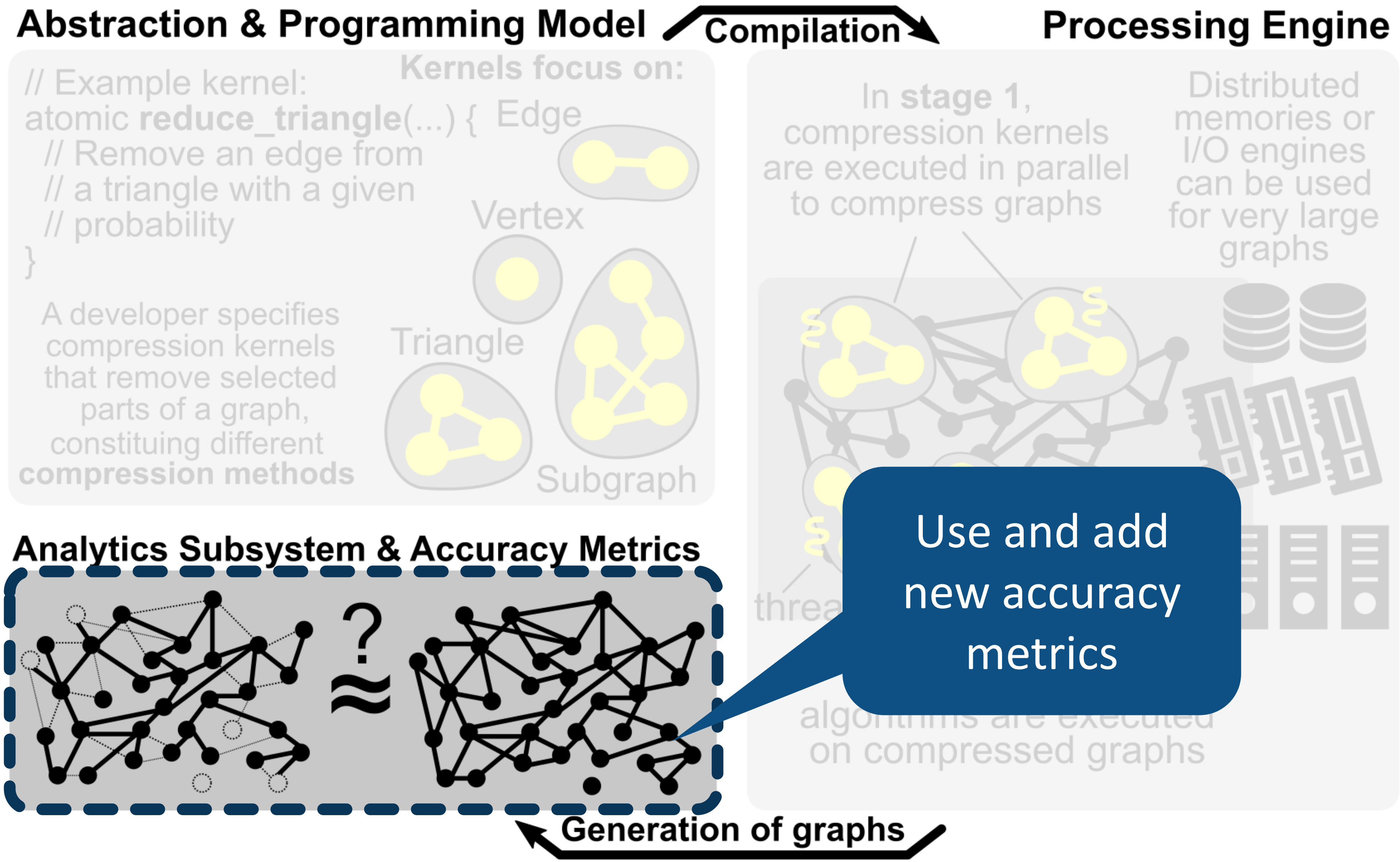
Compress graphs (go off node if necessary)



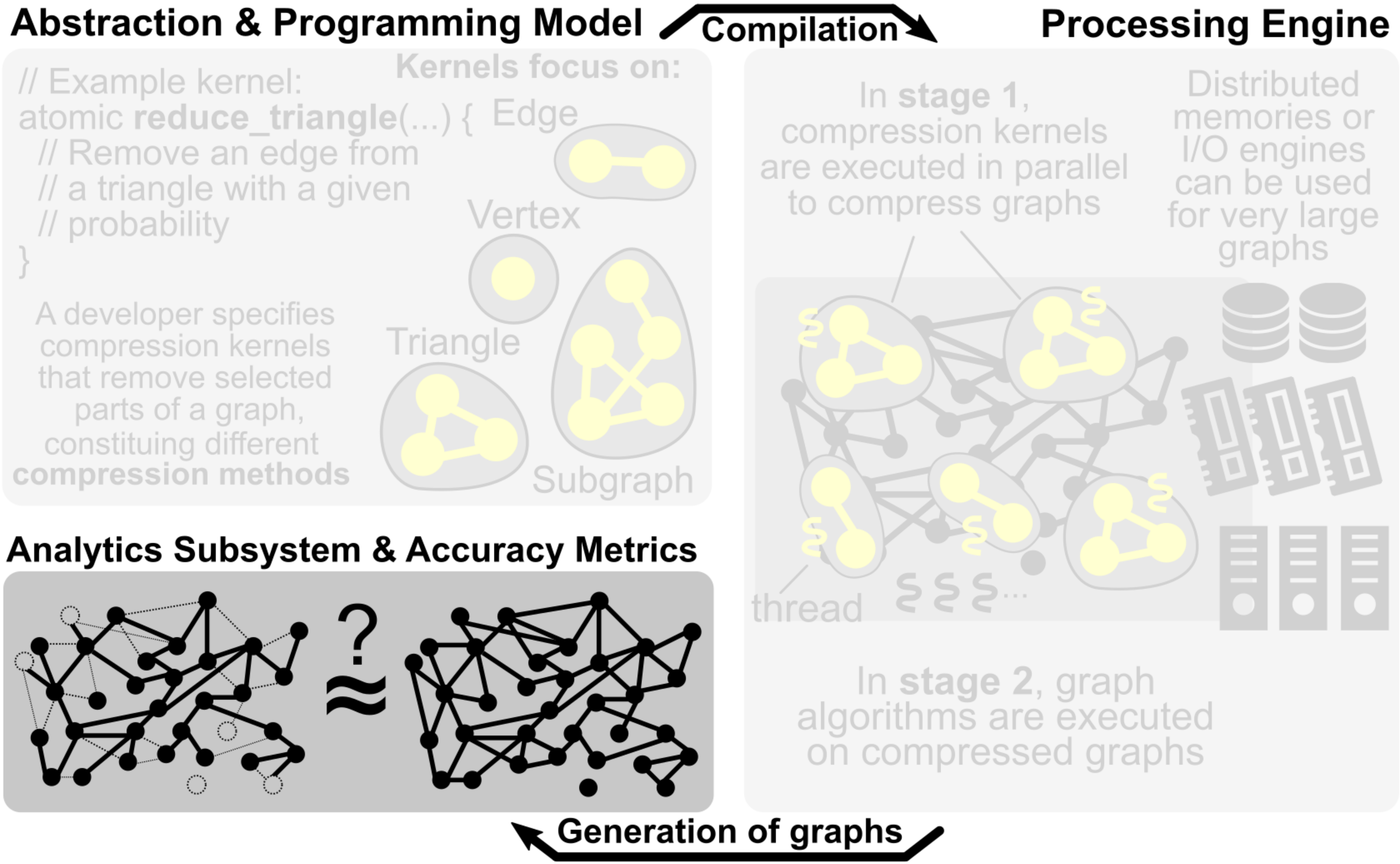
Slim Graph: High-Performance Extensible System



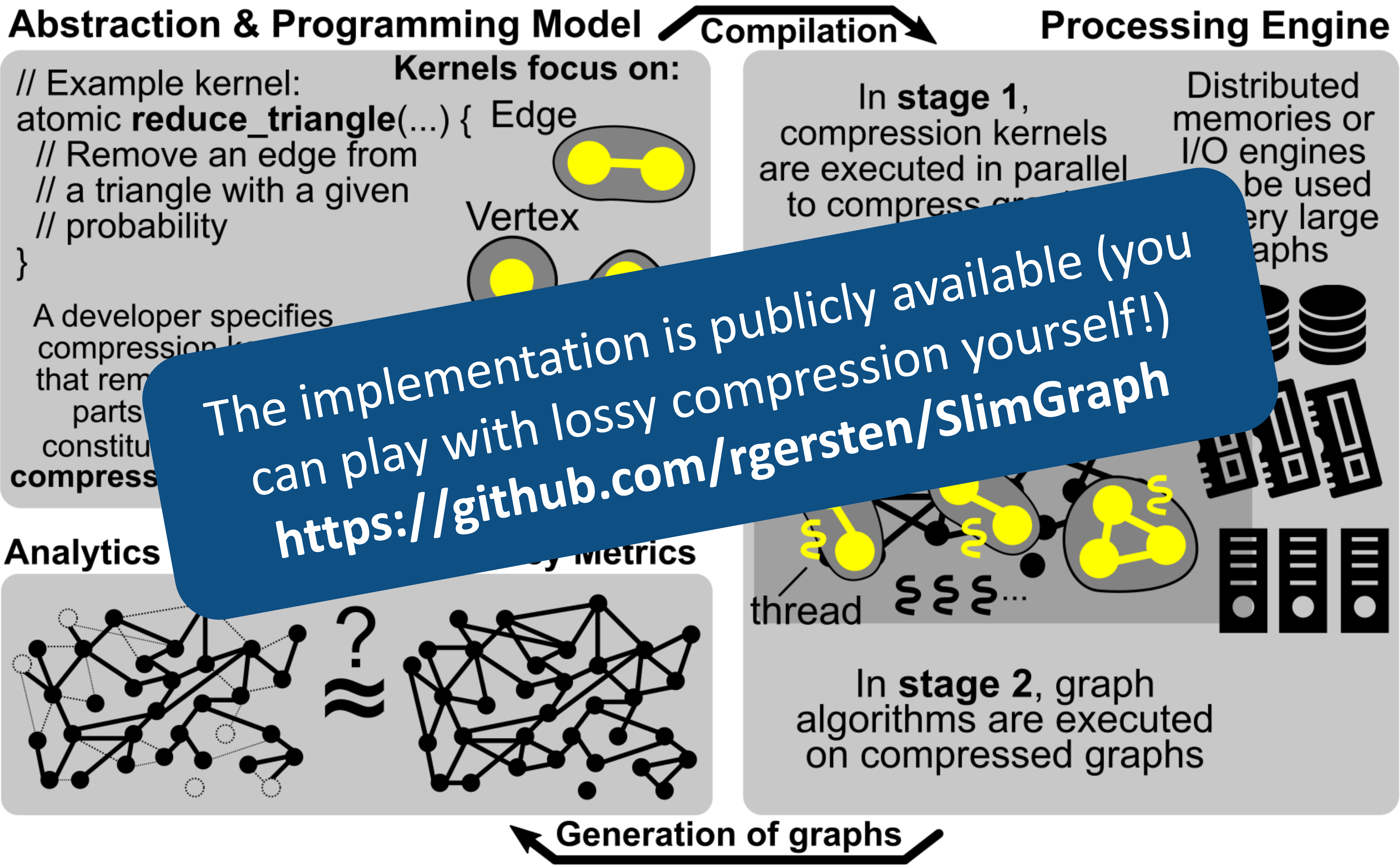
Slim Graph: High-Performance Extensible System



Slim Graph: High-Performance Extensible System



Slim Graph: High-Performance Extensible System



The implementation is publicly available (you can play with lossy compression yourself!)

https://github.com/rgersten/SlimGraph

Performance Analysis
Used machines & Goals

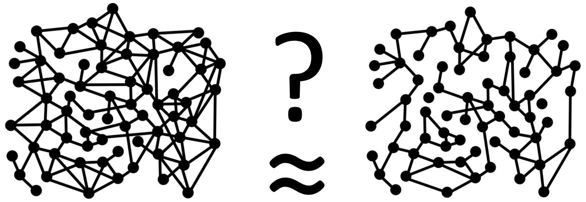
Goal 1: Enable scalable compression of large graphs

Goal 2: Enable comparison of different aspects of lossy graph compression

CSCS Cray Piz Daint,
64 GB per compute node

CSCS Ault server, 768 GB of DRAM

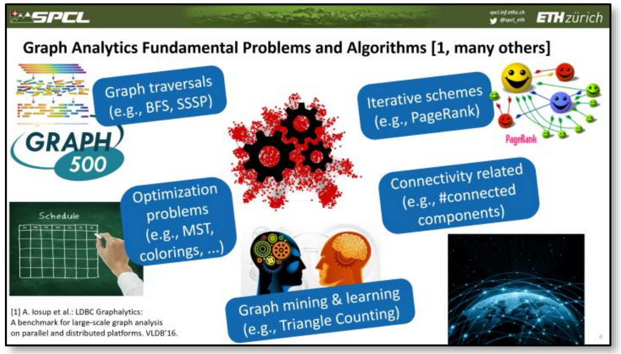
Storage Reductions vs. Speedups vs. Accuracy Loss [vs. Compression Overhead]



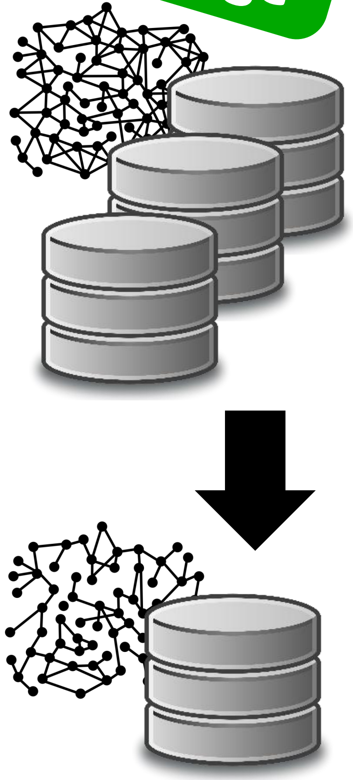
Accuracy loss

High accuracy

Various workloads and graphs



Less storage



Storage reductions

Compression overhead

Compression method B

Compression method A

Speedups of graph algorithms

Faster workloads

Slim Graph enables comparison and analysis of four aspects of lossy graph compression

Triangle Reduction Analysis

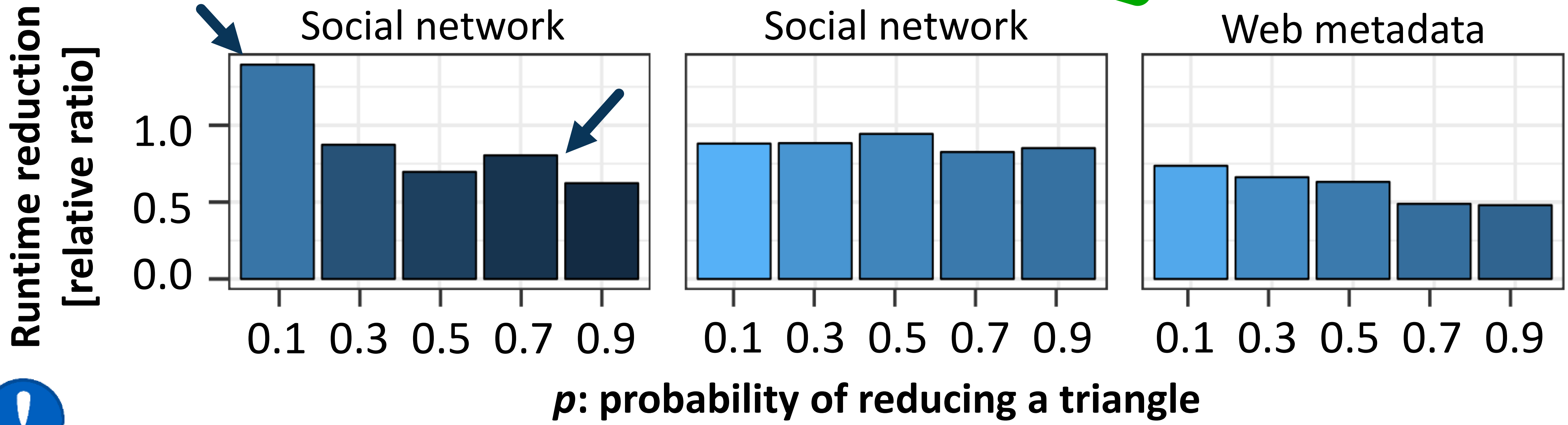
Workload: BFS traversal

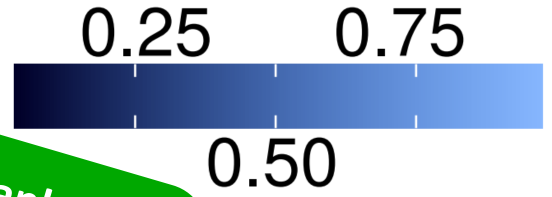
High Accuracy

Less storage

Faster workloads

Selected insights...





...compression ratio: ...er of edges in the ...n to the number of ...e original graph

Storage reduced even by **> 4x** (depends on the structure)

Runtime reduced even by **> 50%**

Surprising effects revealed: runtime can *increase* with *fewer* edges (synchronization!)

Use Slim Graph to navigate designing more efficient compression

Triangle Reduction Analysis

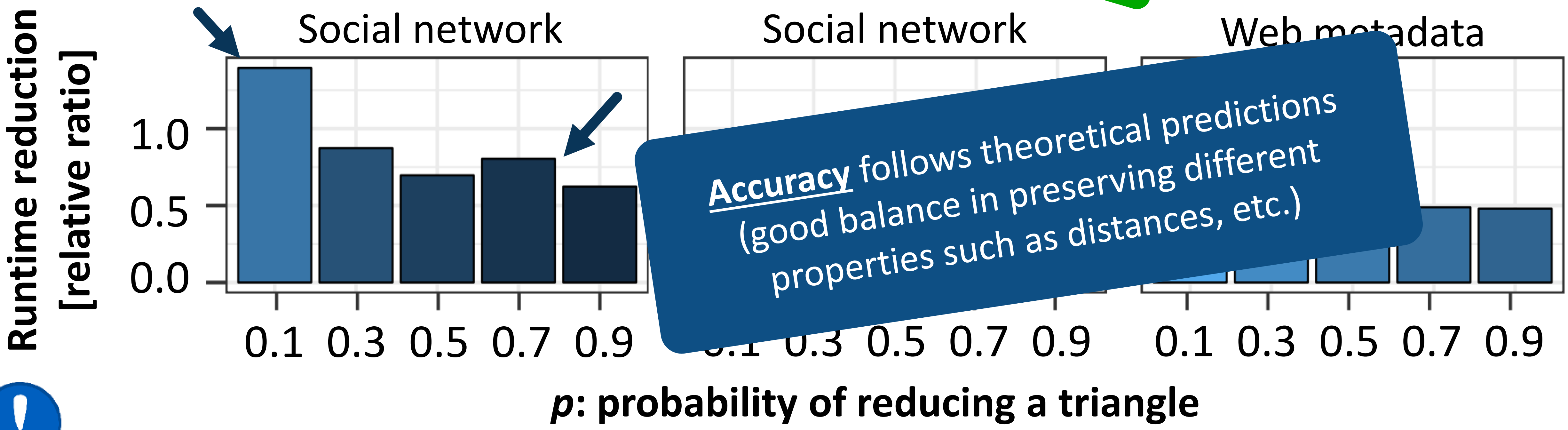
Workload: BFS traversal

High Accuracy

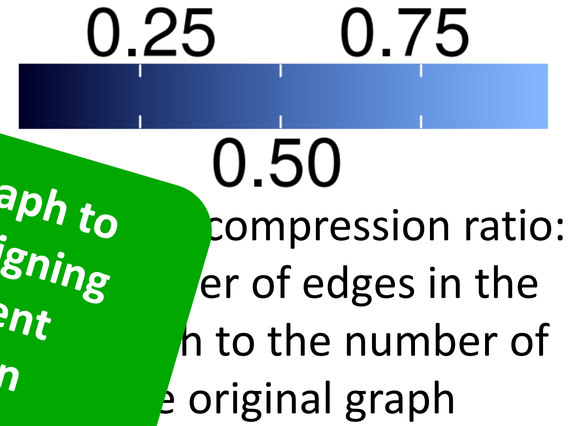
Less storage

Faster workloads

Selected insights…



Accuracy follows theoretical predictions (good balance in preserving different properties such as distances, etc.)



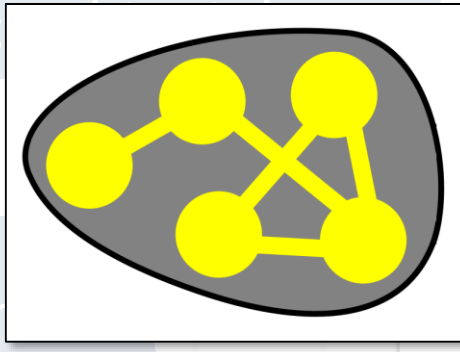
Storage reduced even by **> 4x** (depends on the structure)

Runtime reduced even by **> 50%**

Surprising effects revealed: runtime can *increase* with *fewer* edges (synchronization!)

Use Slim Graph to navigate designing more efficient compression

What if you want to go <u>extreme</u> in some respect?



<u>Storage</u> reduced **by > <u>10x</u>**

<u>Runtime</u> reduced by > 75%

Distances increase at most by 8x, but *proportionally* to the original ones

Metrics Analyses

Graph	EO 0.8-1-TR	EO 1.0-1-TR	Uniform ($p = 0.2$)	Uniform ($p = 0.5$)	Spanner ($k = 2$)	Spanner ($k = 16$)	Spanner ($k = 128$)
s-you	0.0121	0.0167	0.1932	0.6019	0.0054	0.2808	0.2993
h-hud	0.0187	0.0271	0.0477	0.1633	0.0340	0.2794	0.3247
il-dbl	0.0459	0.0674	0.0749	0.2929	0.0080	0.1980	0.2005
v-skt	0.0410	0.0643	0.0674	0.2695	0.0311	0.1101	0.2950
v-usa	0.0089	0.0100	0.1392	0.5945	0.0000	0.0074	0.0181

Graph	Original	0.2-1-TR									
s-you	11.38	1.544	0.037	0.091	1.416	5.825	7.626	0.071	0.000	0	0.007	0.426
s-flx	9.389	0.645	0.017	0.075	1.173	4.802	6.933	0.000	0.070	0	0.001	0.219
s-flc	1091	6.845	0.164	8.765	136.6	557.9	250.7	1.327	0.001	0	0.016	1.517
s-cds	3157	18.56	0.561	25.24	394.8	1615	844.5	45.392	0.001	0	0.015	4.821
s-lib	938.3	31.51	0.902	7.569	116.9	480.2	82.59	167.0	5.708	0	0.000	0.042
s-pok	59.82	10.25	0.280	0.480	7.494	30.58	41.27	0.362	0.000	0	0.005	1.962
h-dbp	6.299	1.158	0.072	0.051	0.822	3.218	2.295	0.440	0.002	0	0.020	1.981
h-hud	14.71	1.832	0.083	0.117	1.839	7.538	7.373	0.001	0.000	0	0.005	2.495
l-cit	5.973	1.994	0.091	0.048	0.747	3.059	5.128	0.240	0.000	0	0.007	1.931
l-dbl	45.57	6.144	0.257	0.365	5.671	23.33	22.64	0.033	0.004	0	0.066	8.572
v-ewk	235.2	14.13	0.422	1.886	29.33	120.3	110.0	0.034	0.000	0	0.008	2.436
v-skt	50.88	2.642	0.099	0.395	6.455	26.01	22.24	5.777	0.502	0	0.016	2.376

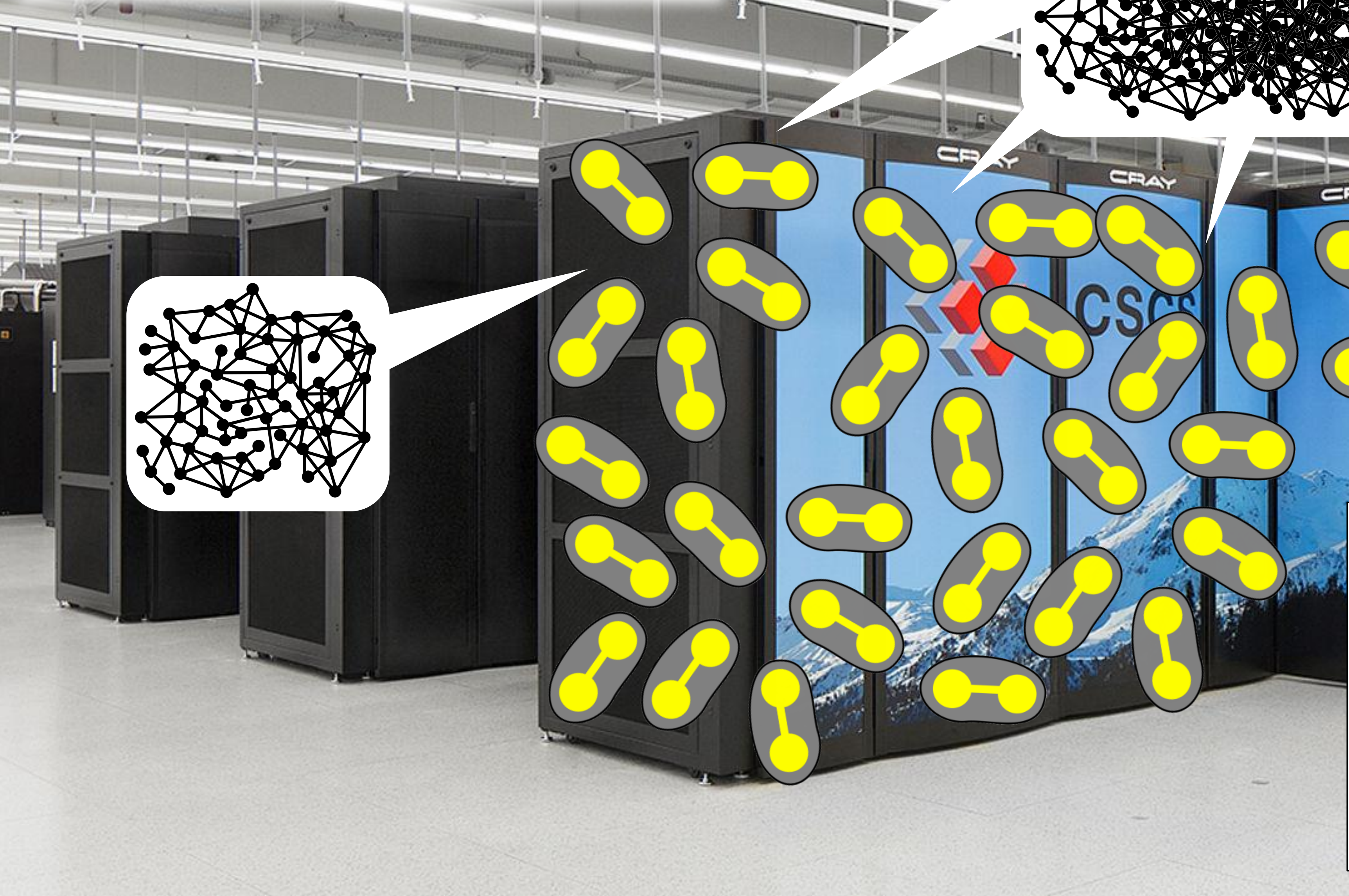
Metrics Analyses

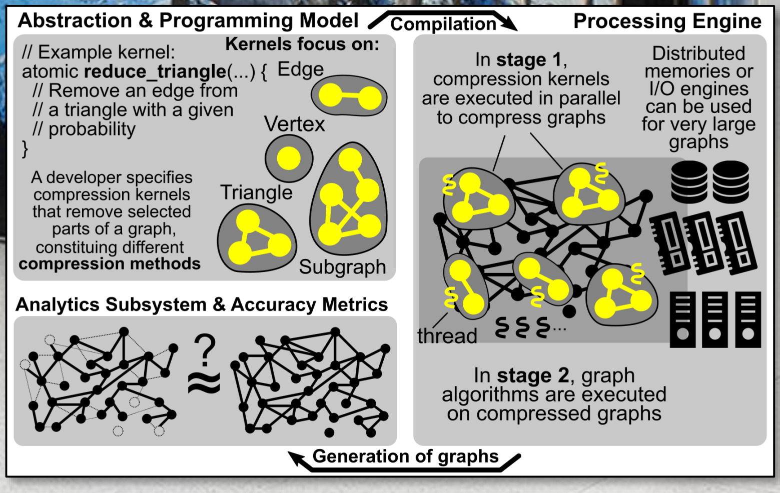
Graph	EO 0.8-1-TR	EO 1.0-1-TR	Uniform ($p = 0.2$)	Uniform ($p = 0.5$)	Spanner ($k = 2$)	Spanner ($k = 16$)	Spanner ($k = 128$)
s-you	0.0121	0.0167	0.1932	0.6019	0.0054	0.2808	0.2993
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0.3247
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0.2005
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0.2950
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0.0181

Graph	Original											
s-you	11.38	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
s-flx	9.389	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
s-flc	1091	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
s-cds	3157	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
s-lib	938.3	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
s-pok	59.82	10.25	0.280	0.480	7.494	30.58	41.27	0.362	0.000	0	0.005	1.962
h-dbp	6.299	1.158	0.072	0.051	0.822	3.218	2.295	0.440	0.002	0	0.020	1.981
h-hud	14.71	1.832	0.083	0.117	1.839	7.538	7.373	0.001	0.000	0	0.005	2.495
l-cit	5.973	1.994	0.091	0.048	0.747	3.059	5.128	0.240	0.000	0	0.007	1.931
l-dbl	45.57	6.144	0.257	0.365	5.671	23.33	22.64	0.033	0.004	0	0.066	8.572
v-ewk	235.2	14.13	0.422	1.886	29.33	120.3	110.0	0.034	0.000	0	0.008	2.436
v-skt	50.88	2.642	0.099	0.395	6.455	26.01	22.24	5.777	0.502	0	0.016	2.376

Both reordering and divergence based accuracy metrics' values **increase monotonically** (in all the cases) with the scope of compression.

Slim Graph enables distributed compression for large graphs

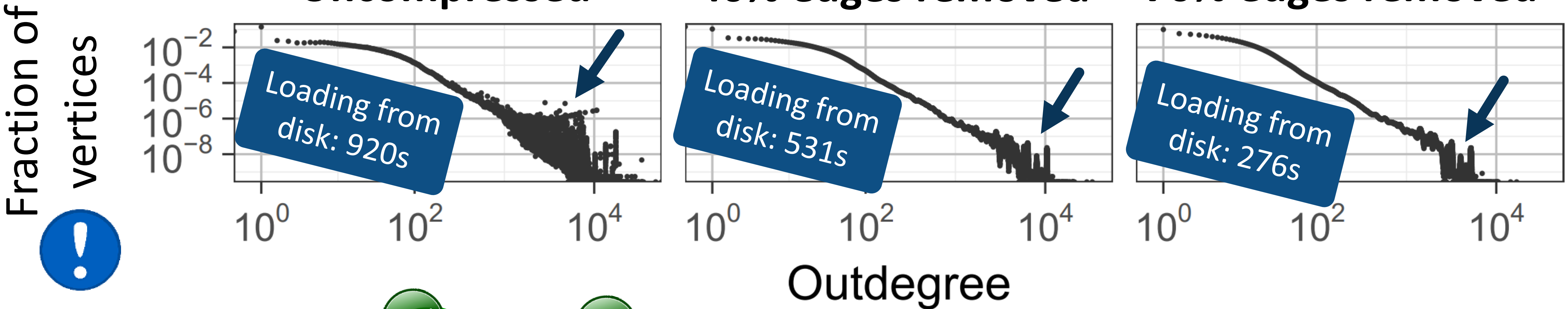




Distributed Large-Scale Lossy Compression with Slim Graph

5 **largest publicly available** real-world graphs

Results for the largest graph:



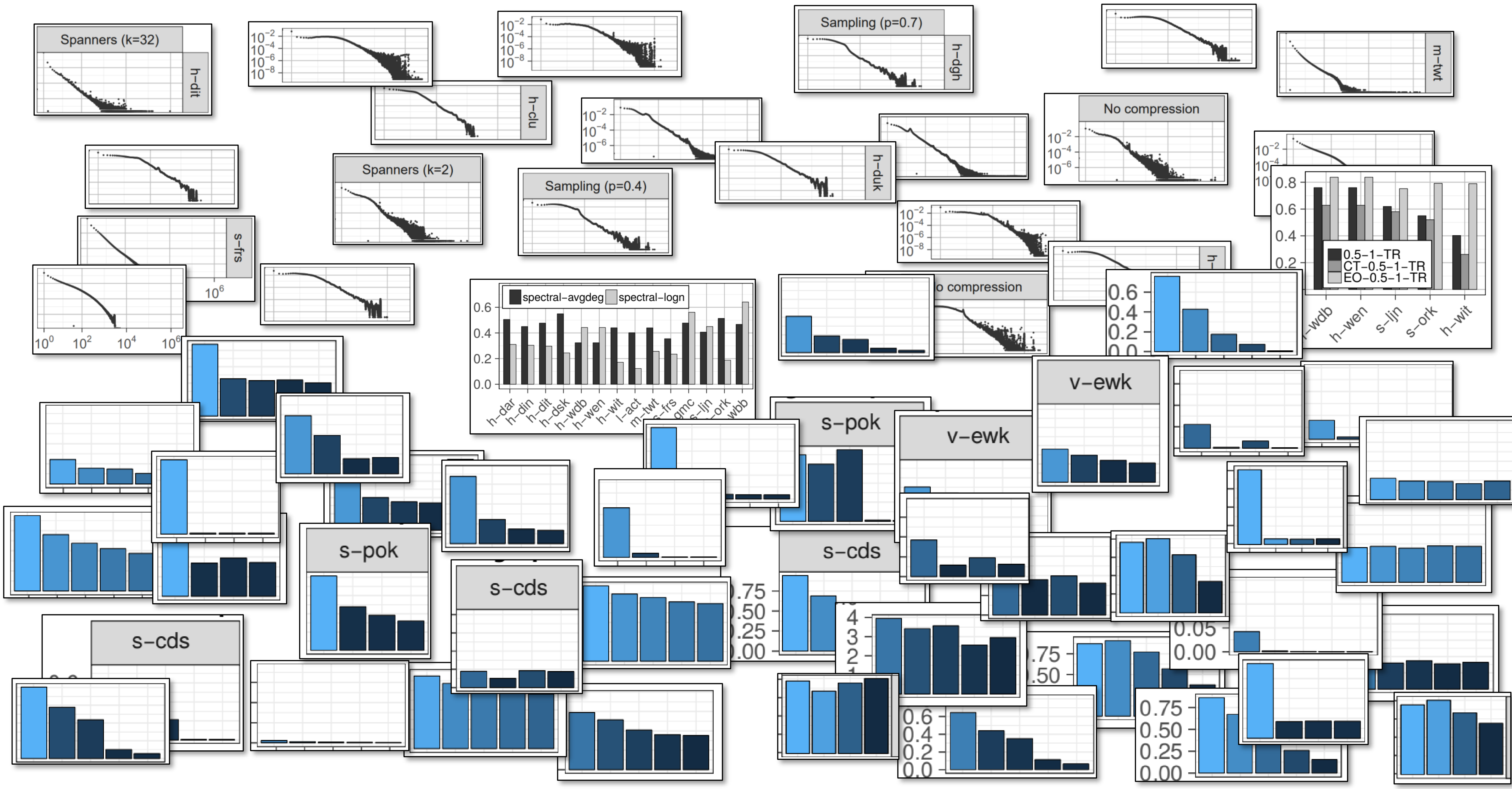
I/O time is reduced (benefits any computation)

Faster workloads

Less storage

Use Slim Graph to find **novel use cases** of lossy graph compression

Slim Graph enabled us to discover an interesting effect of "**removing the clutter**" – (mild) sampling could be used as preprocessing


Spanners (k=32)
h−dit
h−clu
Sampling (p=0.7)
h−dgh
m−twt
Spanners (k=2)
Sampling (p=0.4)
h−duk
No compression
s−frs
spectral−avgdeg
spectral−logn
h−dar
h−din
h−dit
h−dsk
h−wdb
h−wen
h−wit
l−act
m−twt
gmc
0.5−1−TR
CT−0.5−1−TR
EO−0.5−1−TR
h−wdb
h−wen
s−ljn
s−ork
h−wit
v−ewk
s−pok
s−cds

Slim Graph delivers a simple, intuitive, versatile ...

1 ... **Abstraction & programming model** for easy development and rapid prototyping of lossy graph compression methods

Solved

2 ... **Compression method** that preserves different graph properties that are important for the practice of graph processing

Solved

3 ... **Criterion (criteria?)** to assess the accuracy of lossy graph compression methods

Solved

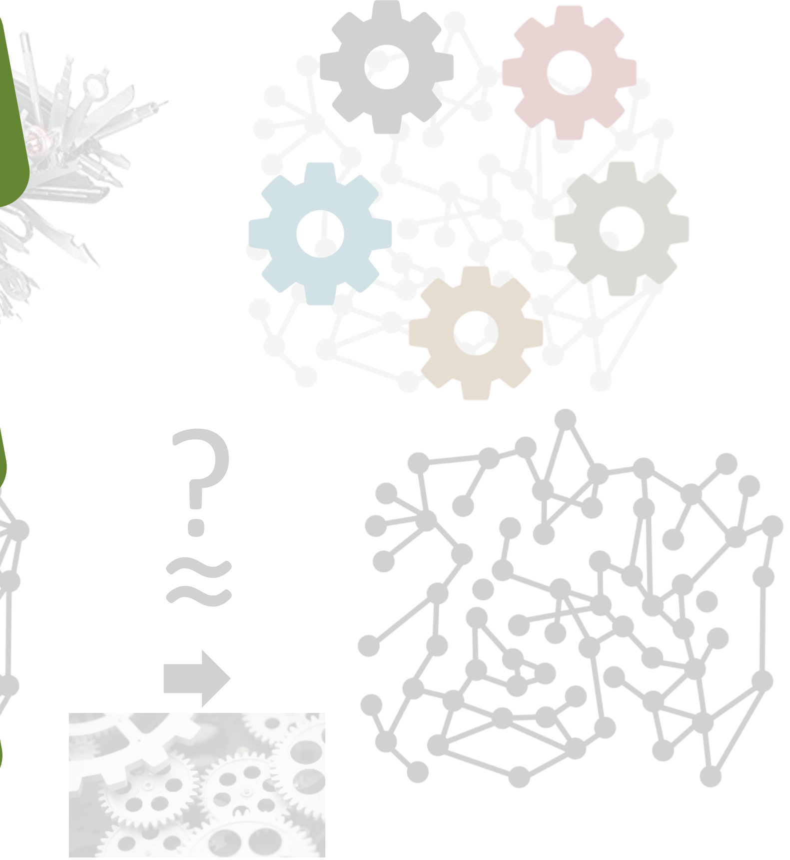
4 ... *High-performance* and *extensible* **system** for implementing and executing lossy graph compression

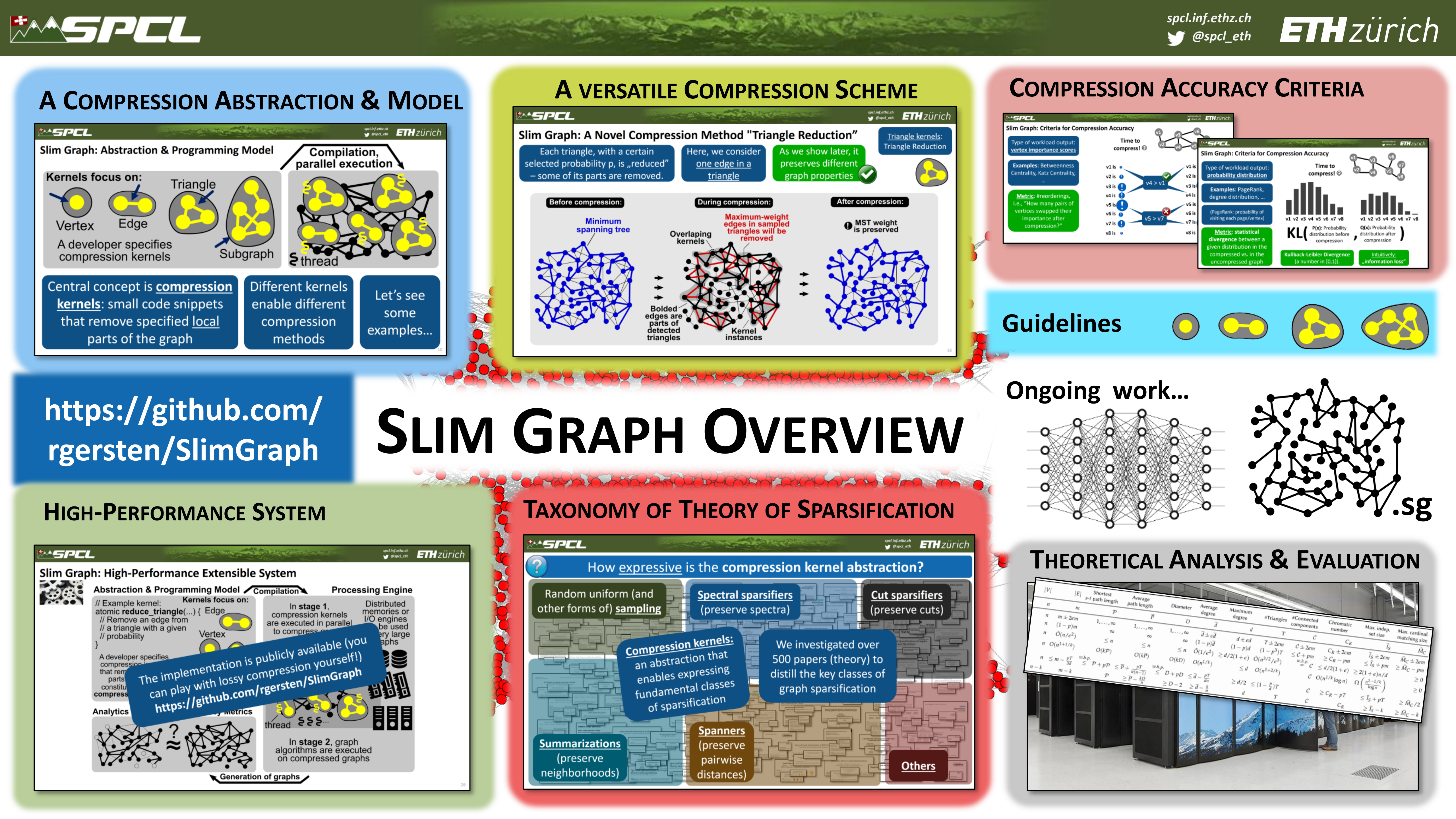
Solved

Number of ways [1] to sparsify (compress) a graph with *n* vertices

$$O\left(2^{\binom{n}{2}}\right)$$

[1] R. C. Entringer, P. Erdos. "On the Number of Unique Subgraphs of a Graph", Journal of Combinatorial Theory 1972




spcl.inf.ethz.ch
@spcl_eth
ETH zürich
A Compression Abstraction & Model
Slim Graph: Abstraction & Programming Model
Compilation, parallel execution
Kernels focus on:
Vertex
Edge
Triangle
Subgraph
thread
A developer specifies compression kernels
Central concept is compression kernels: small code snippets that remove specified local parts of the graph
Different kernels enable different compression methods
Let's see some examples...
A versatile Compression Scheme
Slim Graph: A Novel Compression Method "Triangle Reduction"
Each triangle, with a certain selected probability p, is „reduced" – some of its parts are removed.
Here, we consider one edge in a triangle
As we show later, it preserves different graph properties
Triangle kernels: Triangle Reduction
Before compression:
During compression:
After compression:
Minimum spanning tree
Overlapping kernels
Maximum-weight edges in sampled triangles will be removed
MST weight is preserved
Bolded edges are parts of detected triangles
Kernel instances
Compression Accuracy Criteria
Slim Graph: Criteria for Compression Accuracy
Type of workload output: vertex importance scores
Examples: Betweenness Centrality, Katz Centrality, ...
Metric: #reorderings, i.e., "How many pairs of vertices swapped their importance after compression?"
Time to compress!
Type of workload output: probability distribution
Examples: PageRank, degree distribution, ...
Metric: statistical divergence between a given distribution in the compressed vs. in the uncompressed graph
Kullback-Leibler Divergence
Intuitively: „information loss"
Guidelines
Ongoing work...
.sg
https://github.com/rgersten/SlimGraph
Slim Graph Overview
High-Performance System
Slim Graph: High-Performance Extensible System
Abstraction & Programming Model
Compilation
Processing Engine
// Example kernel:
atomic reduce_triangle(...) {
// Remove an edge from
// a triangle with a given
// probability
}
Kernels focus on:
In stage 1, compression kernels are executed in parallel to compress
Distributed memories or I/O engines can be used
The implementation is publicly available (you can play with lossy compression yourself!) https://github.com/rgersten/SlimGraph
Analytics
Metrics
thread
In stage 2, graph algorithms are executed on compressed graphs
Generation of graphs
Taxonomy of Theory of Sparsification
How expressive is the compression kernel abstraction?
Random uniform (and other forms of) sampling
Spectral sparsifiers (preserve spectra)
Cut sparsifiers (preserve cuts)
Compression kernels: an abstraction that enables expressing fundamental classes of sparsification
We investigated over 500 papers (theory) to distill the key classes of graph sparsification
Summarizations (preserve neighborhoods)
Spanners (preserve pairwise distances)
Others
Theoretical Analysis & Evaluation

Backup Slides and Slides' Variants

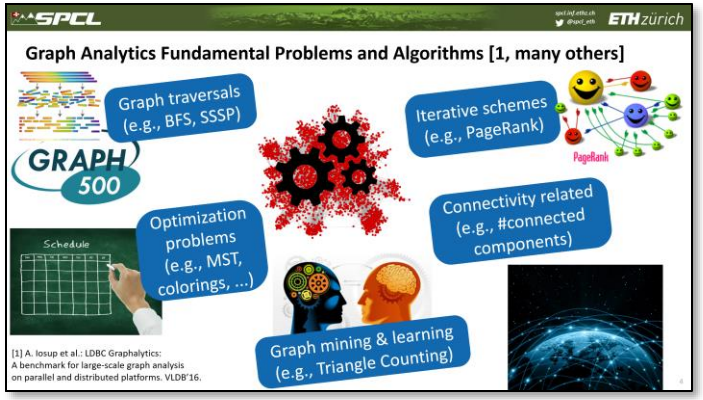
High Accuracy

Less storage

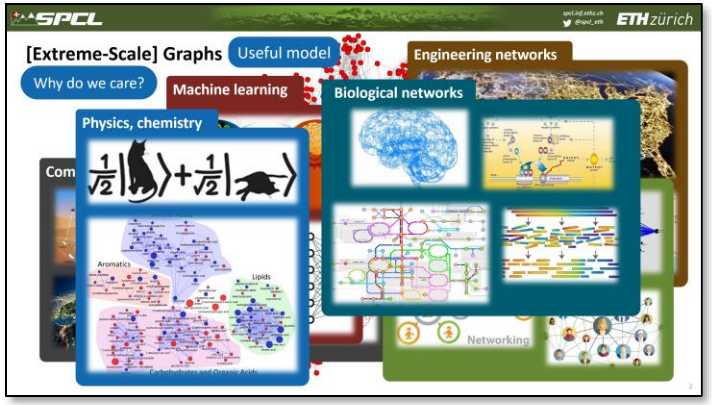
Faster workloads

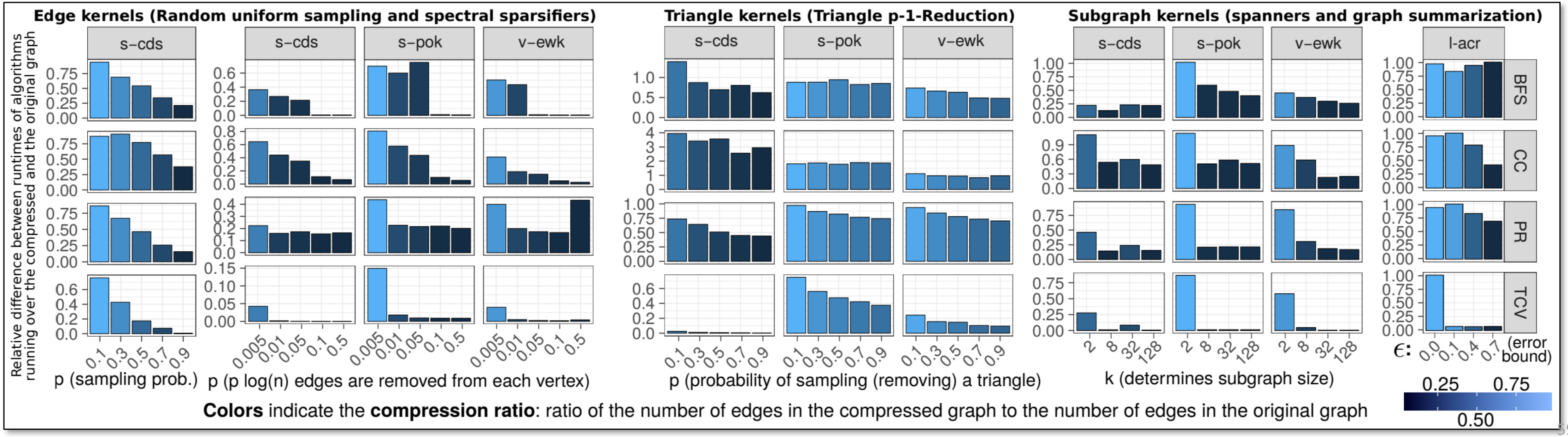
Selected insights…

Various workloads are considered

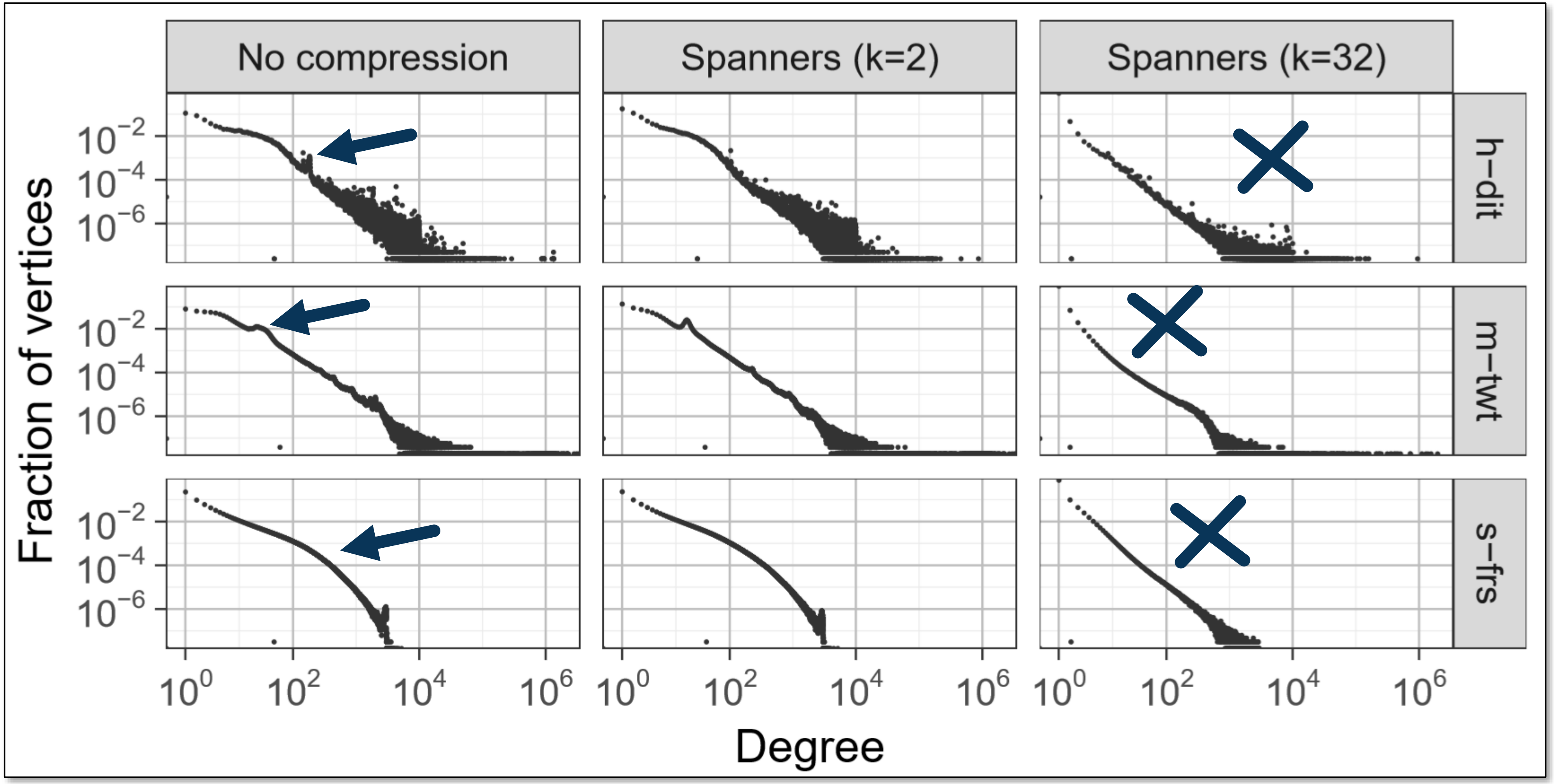


Various real-world graphs are used





Compressing Largest-Scale Graphs with Slim Graph



The first analysis of the impact of spanners on degree distribution

An interesting "leveling" effect

How large are extreme-scale graphs today?

Laboratory for Web Algorithmics datasets [1]

Graph	Crawl date	Nodes	Arcs
uk-2014	2014	787 801 471	47 614 527 250
eu-2015	2015	1 070 557 254	91 792 261 600
gsh-2015	2015	988 490 691	33 877 399 152

> 875 GB

> 1.7 TB

> 625 GB

Sogou 搜狗

> 233 TB

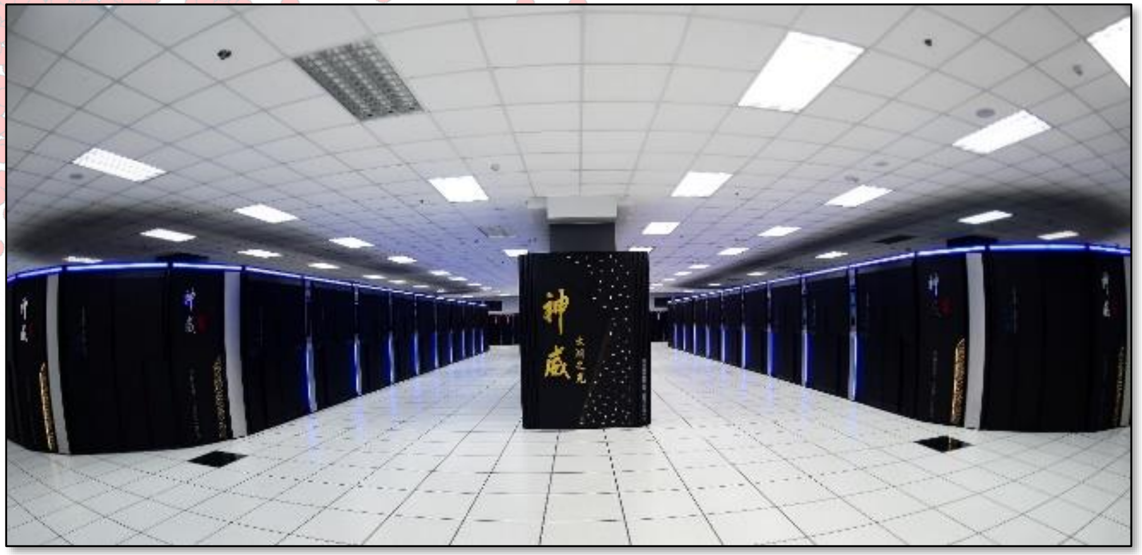
271 billion vertices, 12 trillion edges [4]

The runs used nearly all memory on compute nodes of TaihuLight!

Web data commons datasets [2]

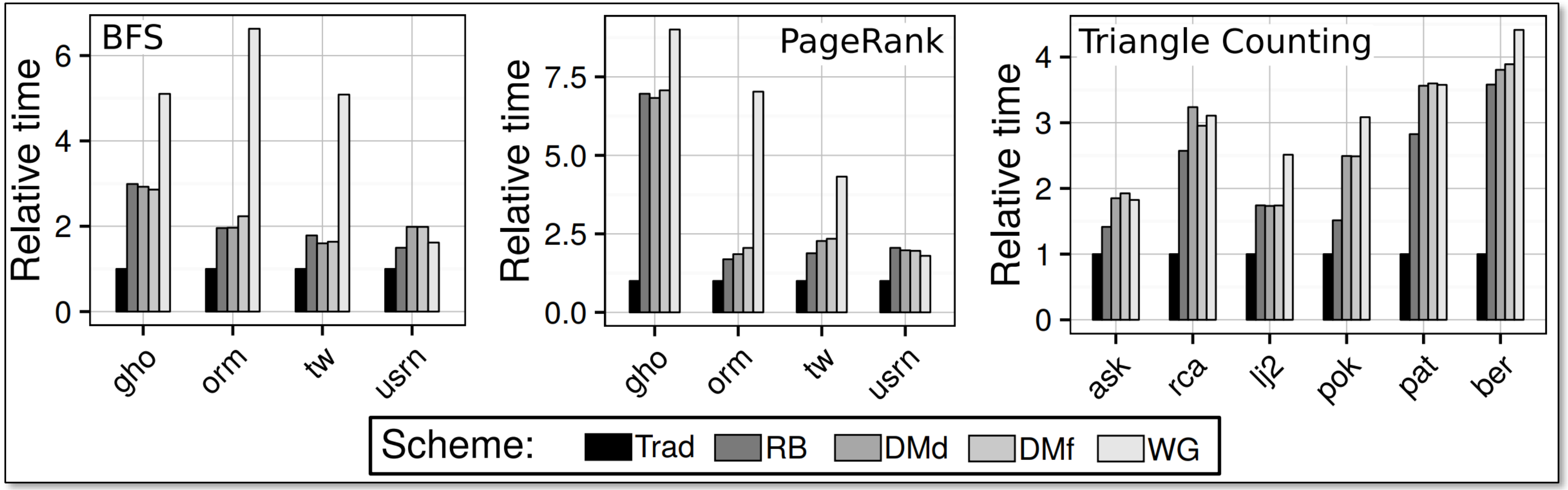
> 2.5 TB

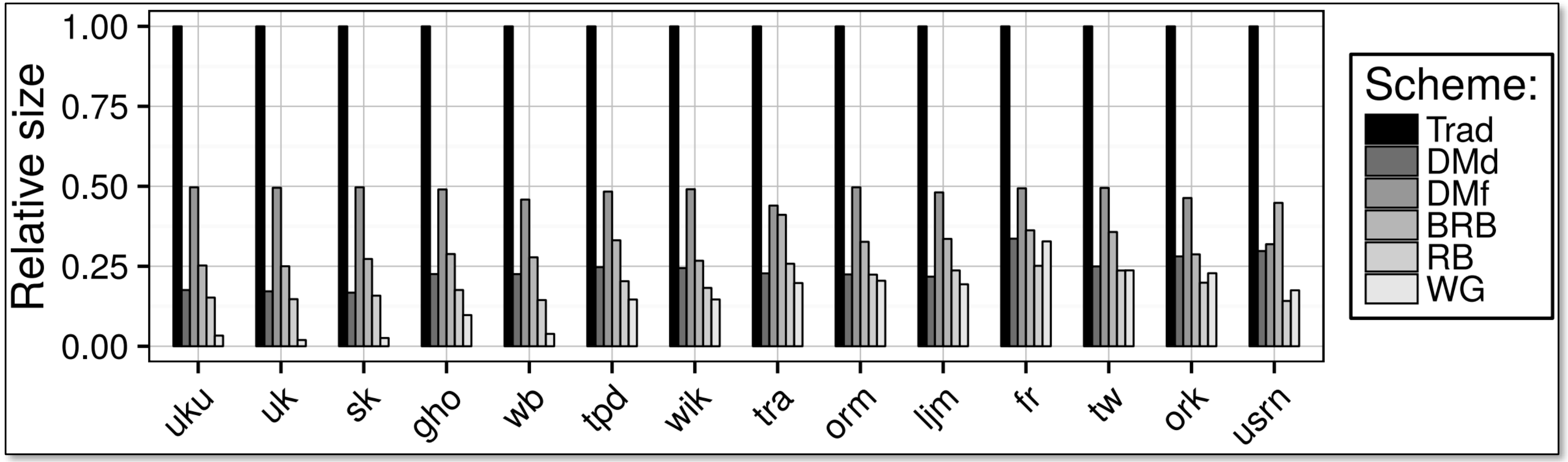
Granularity	#Nodes	#Arcs
Page	3,563 million	128,736 million
Host	101 million	2,043 million



[1] http://law.di.unimi.it/datasets.php
[2] http://webdatacommons.org/hyperlinkgraph/2012-08/download.html
[4] Heng Lin et al.: ShenTu: Processing Multi-Trillion Edge Graphs on Millions of Cores in Seconds, SC18, **Gordon Bell Finalist**

How about lossless compression?





[Traditional] compression incurs expensive decompression [1,2]

[1] M. Besta et al.: "Log(Graph): A Near-Optimal High-Performance Graph Representation", PACT'18
[2] M. Besta, T. Hoefler. "Survey and taxonomy of lossless graph compression and space-efficient graph representations", arXiv'19

spcl.inf.ethz.ch
@spcl_eth

Problems!

Huge size

But… we show [1,2] that ≈20-30% less storage is really as good as you can get due to fundamental storage lower bounds.

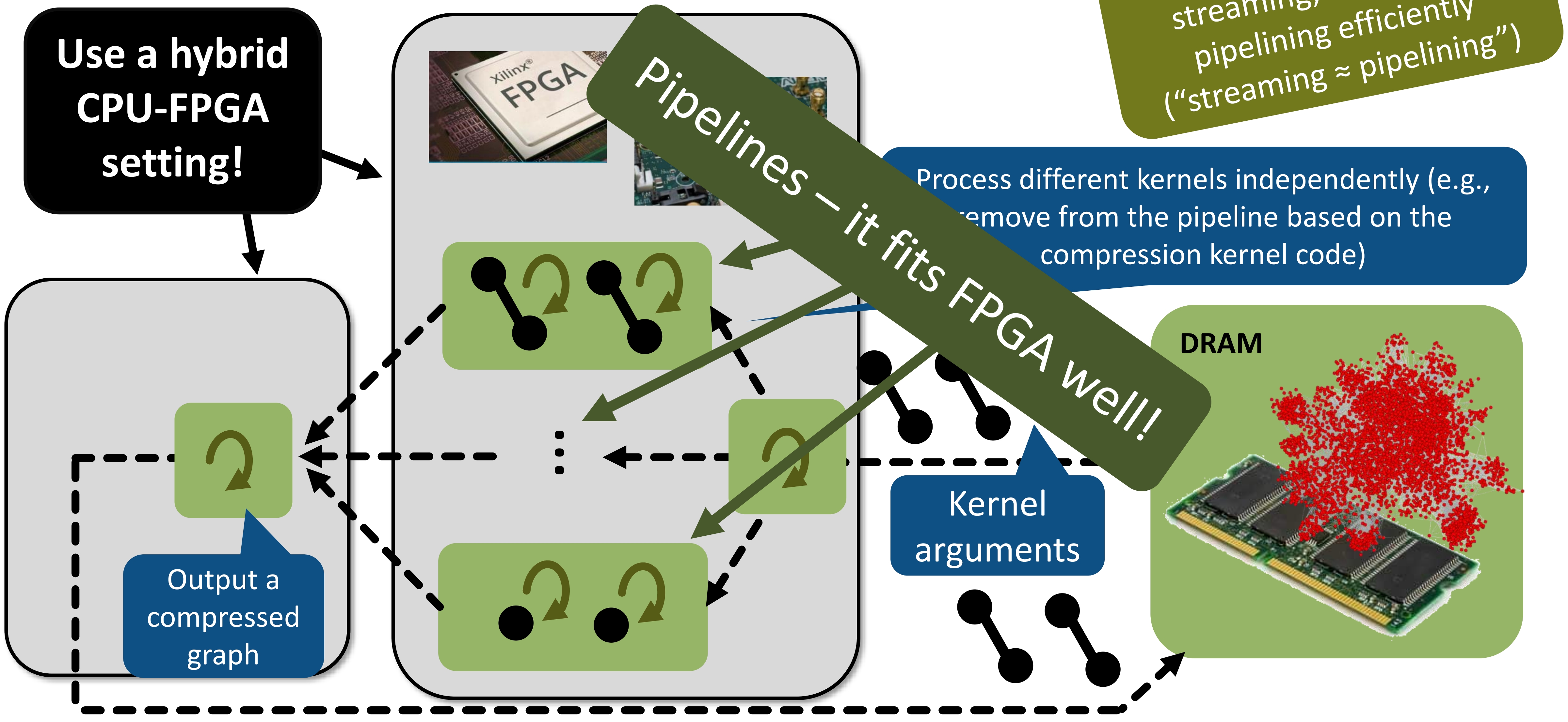
How about lossless compression?

p (edge probability)

[1] M. Besta et al.: "Log(Graph): A Near-Optimal High-Performance Graph Representation", PACT'18
[2] M. Besta, T. Hoefler. "Survey and taxonomy of lossless graph compression and space-efficient graph representations", arXiv'18

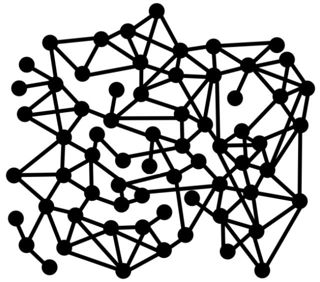
Substream-Centric Slim Graph

Use some form of streaming; we can use pipelining efficiently ("streaming ≈ pipelining")

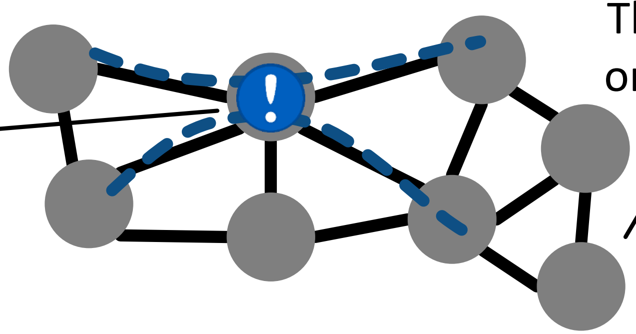


Slim Graph: Abstraction & Programming Model

Input:



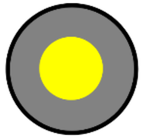
At least two paths (this one is relevant!)



This poor one has 0

Vertex kernels: removing degree-1 vertices

Betweenness Centrality [1] relative scores are preserved [2]



Betweenness centrality of a vertex determines the vertex importance (#shortest paths)

[1] M. Barthelemy. “Betweenness Centrality in large complex networks”, The European physical journal B, 2004
[2] J. Matta. “Comparing the speed and accuracy of approaches to Betweenness Centrality approximation”, Computational Social Networks 2019

[Extreme-Scale] Graphs

Useful model

Why do we care?

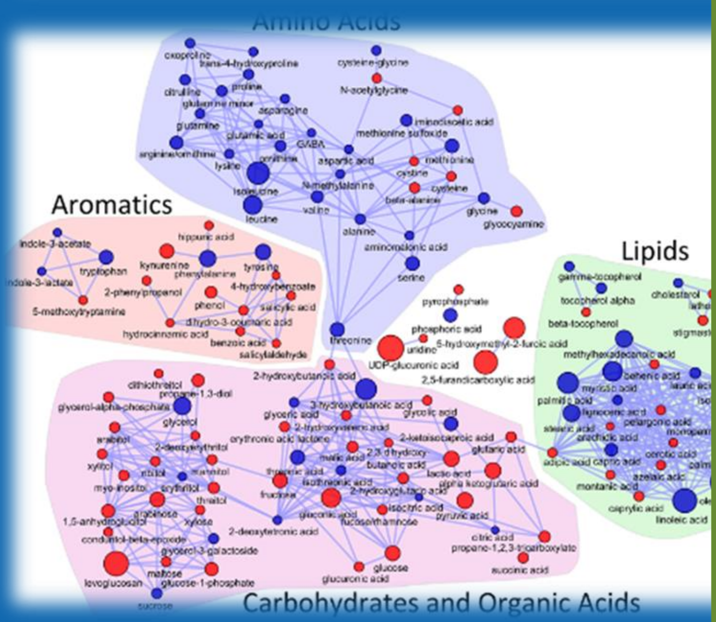
Engineering networks

...even philosophy ☺

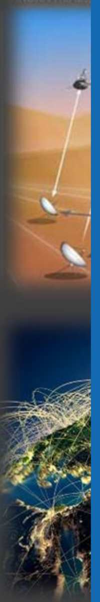
Machine learnin

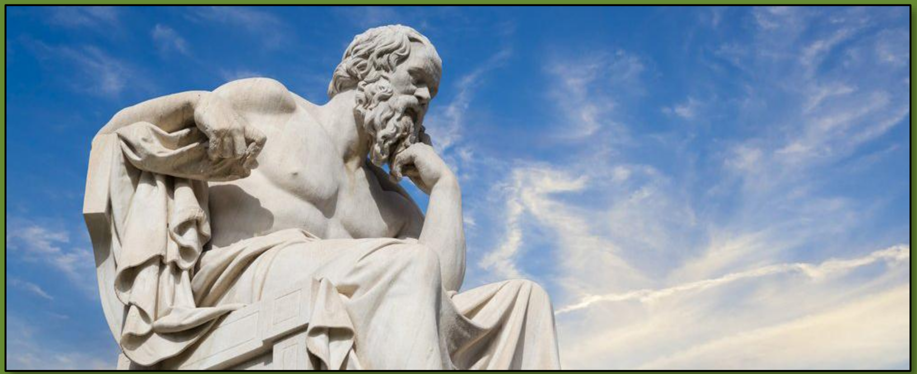
orks

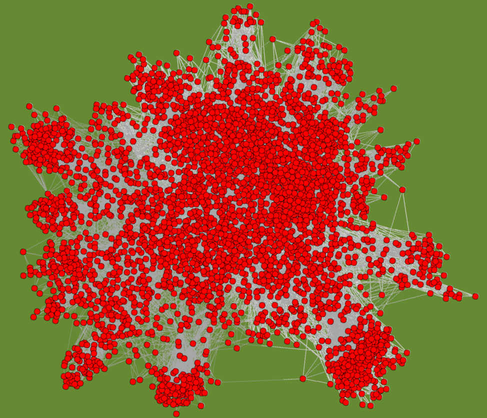
Physics, chemistry



Com







FOSDEM 2016 / Schedule / Events / Developer rooms / Graph Processing / Modeling a Philosophical Inquiry: from MySQL to a graph database

Modeling a Philosophical Inquiry: from MySQL to a graph database

The short story of a long refactoring process

Track: Graph Processing devroom
Room: AW1.126
Day: Saturday
Start: 12:45
End: 13:35

Bruno Latour wrote a book about philosophy (an inquiry into modes of existence). He decided that the paper book was no place for the numerous footnotes, documentation or glossary, instead giving access to all this information surrounding the book through a web application which would present itself as a reading companion. He also offered to the community of readers to submit their contributions to his inquiry by writing new documents to be added to the platform. The first version

Slim Graph: Abstraction and Programming Model

Edge kernels: implementing spectral sparsification and sampling

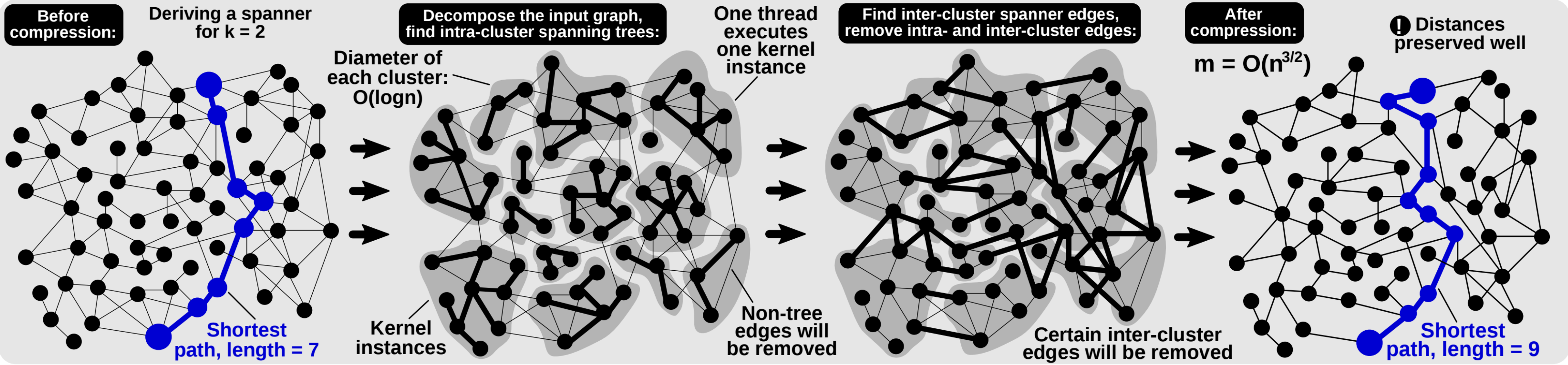
```
/********** Single-edge compression kernels (§ 4.2) ****************/
spectral_sparsify(E e) { //More details in § 4.2.1
  double Υ = SG.connectivity_spectral_parameter();
  double edge_stays = min(1.0, Υ / min(e.u.deg, e.v.deg));
  if(edge_stays < SG.rand(0,1)) atomic SG.del(e);
  else e.weight = 1/edge_stays;
}
random_uniform(E e) { //More details in § 4.2.2
  double edge_stays = SG.p;
  if(edge_stays < SG.rand(0,1)) atomic SG.del(e);
}
```

Slim Graph: Abstraction and Programming Model

Subgraph kernels: spanners

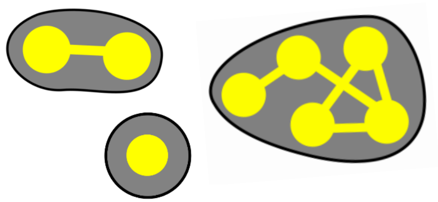
Slim Graph: Abstraction and Programming Model

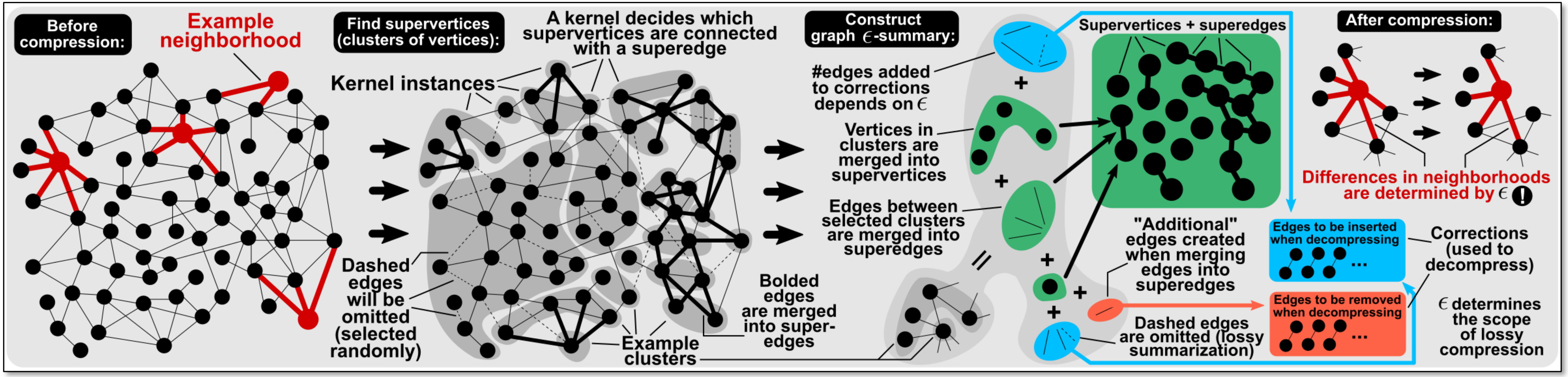
Subgraph kernels: spanners



Slim Graph: Abstraction & Programming Model

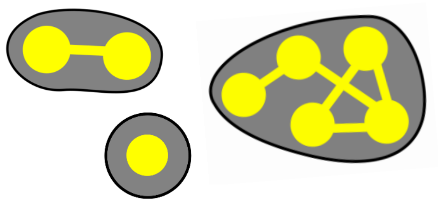
More kernels





Slim Graph: Abstraction & Programming Model

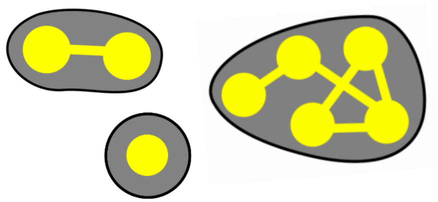
More kernels



```
/********** Single-vertex compression kernel (§ 4.4) ***************/
low_degree(V v) {
  if(v.deg==0 or v.deg==1) atomic SG.del(v); }
/************* Subgraph compression kernels (§ 4.5) ****************/
derive_spanner(vector<V> subgraph) { //Details in § 4.5.3
  //Replace "subgraph" with a spanning tree
  subgraph = derive_spanning_tree(subgraph);
  //Leave only one edge going to any other subgraph.
  vector<set<V>> subgraphs(SG.sgr_cnt);
  foreach(E e: SG.out_edges(subgraph)) {
    if(!subgraphs[e.v.elem_ID].empty()) atomic del(e);
} }
derive_summary(vector<V> cluster) { //Details in § 4.5.4
  //Create a supervertex "sv" out of a current cluster:
  V sv = SG.min_id(cluster);
  SG.summary.insert(sv); //Insert sv into a summary graph
  //Select edges (to preserve) within a current cluster:
  vector<E> intra = SG.summary_select(cluster, SG.ε);
  SG.corrections_plus.append(intra);
  //Iterate over all clusters connected to "cluster":
  foreach(vector<V> cl: SG.out_clusters(out_edges(cluster))) {
    [E, vector<E>] (se, inter) = SG.superedge(cluster,cl,SG.ε);
    SG.summary.insert(se);
    SG.corrections_minus.append(inter);
  }
  SG.update_convergence();
}
```

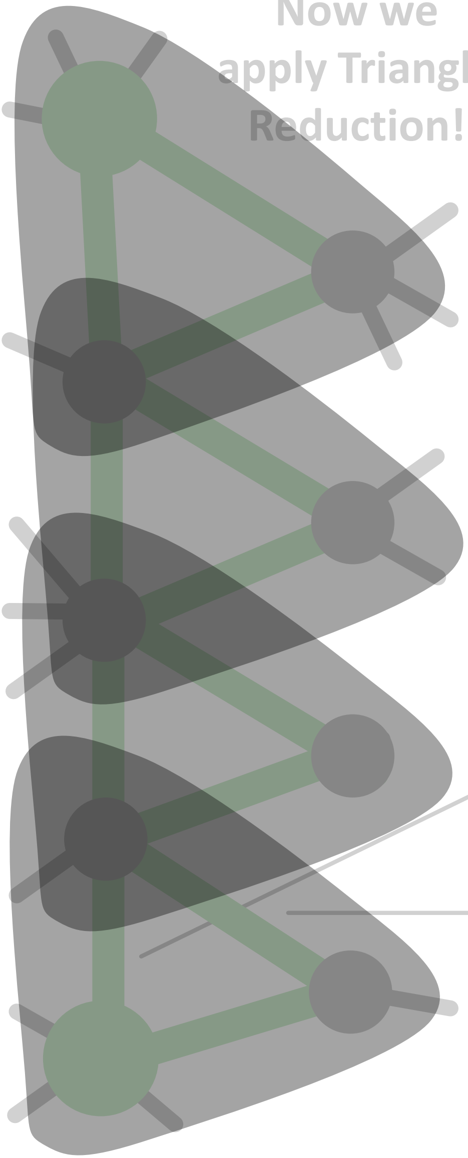
Slim Graph: Abstraction & Programming Model

More kernels



```
/********** Single-edge compression kernels (§ 4.2) ****************/
spectral_sparsify(E e) { //More details in § 4.2.1
  double Y = SG.connectivity_spectral_parameter();
  double edge_stays = min(1.0, Y / min(e.u.deg, e.v.deg));
  if(edge_stays < SG.rand(0,1)) atomic SG.del(e);
  else e.weight = 1/edge_stays;
}
random_uniform(E e) { //More details in § 4.2.2
  double edge_stays = SG.p;
  if(edge_stays < SG.rand(0,1)) atomic SG.del(e);
}
/************* Triangle compression kernels (§ 4.3) ****************/
p-1-reduction(vector<E> triangle) {
  double tr_stays = SG.p;
  if(tr_stays < SG.rand(0,1))
    atomic SG.del(rand(triangle)); }
p-1-reduction-EO(vector<E> triangle) {
  double tr_stays = SG.p;
  if(tr_stays < SG.rand(0,1)) {
    E e = rand(triangle);
    atomic {if(!e.considered) SG.del(e);
            else e.considered  = true; } } }
```

The key intuition behind some derivations for Triangle Reduction



Distances

Graph traversals (e.g., BFS, SSSP)

By how much can distances increase?

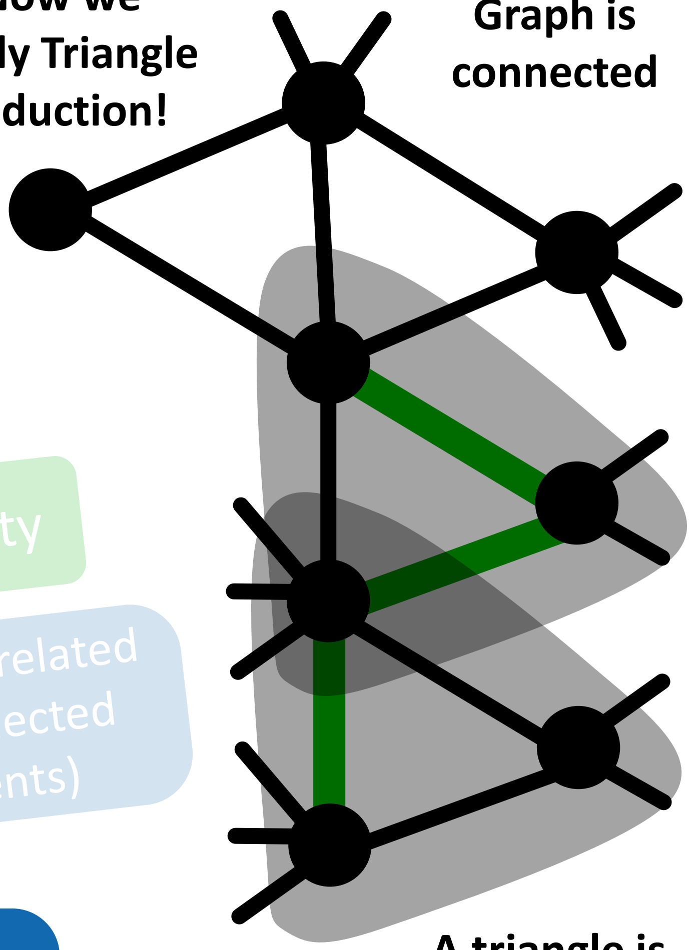
Distances increase by **at most 2x**

Connectivity

Connectivity related (e.g., #connected components)

Can we disconnect a graph?

No ☺



Challenge 2: Theoretical schemes are complex and hard to code and use – how to simplify?

$\widetilde{G} = \texttt{Sparsify2}(G, \epsilon, p)$, where $G = (V, E, w)$ has al

4a. Let ${}^{\delta}V$ be the set of vertices in V with degr
vertices attached to edges in E^i. Let ${}^{\delta}V^i$ be t

4b. For each δ, let ${}^{\delta}C^i_1, \ldots, {}^{\delta}C^i_{k^{\delta}_i}$ be the sets of for
have an edge of E^i on their boundary. Let
has more than $2^{\delta+2}$ edges of E^i on its boun
has between 2^{δ} and $2^{\delta+2}$ edges on its bounda
subdivision in the paragraph immediately aft
the resulting collection of sets.

4c. Let π be the map of partition of W^i by the sets
of (W^i, E^i) under π.

4e. Let $\widetilde{H}^i = \texttt{BoundedSparsify}(H^i, \hat{\epsilon}, p/(c_8 nl \log$
under π whose edges are a subset of E.

$\widetilde{G} = \texttt{Sparsify}(G, \epsilon, p)$, where $G = (V, E, w)$ and $w(e) \leq 1$ for

0. Set $Q = \lceil 6/\epsilon \rceil$, $b = 6/\epsilon$, $c = 6/\epsilon$, $\hat{\epsilon} = \epsilon/6$, and $l = \lceil \log_2$

1. For each edge $e \in E$,
   a. choose r_e so that $Q \leq 2^{r_e} w_e < 2Q$,
   b. let q_e be the largest integer such that $q_e 2^{-r_e} \leq w_e$
   c. set $z_e = q_e 2^{-r_e}$.

2. Let $\widehat{G} = (V, E, z)$, and express
$$\widehat{G} = \sum_{i \geq 0} 2^{-i} G^i,$$
where in each graph G^i all edges have weight 1, and
$\lceil \log_2 2Q \rceil$ of these graphs.

3. Let E^i be the edge set of G^i. Let $E^{\leq i} = \cup_{j \leq i} E^j$. For e
connected components of V under $E^{\leq i}$. For $i = 0$, set

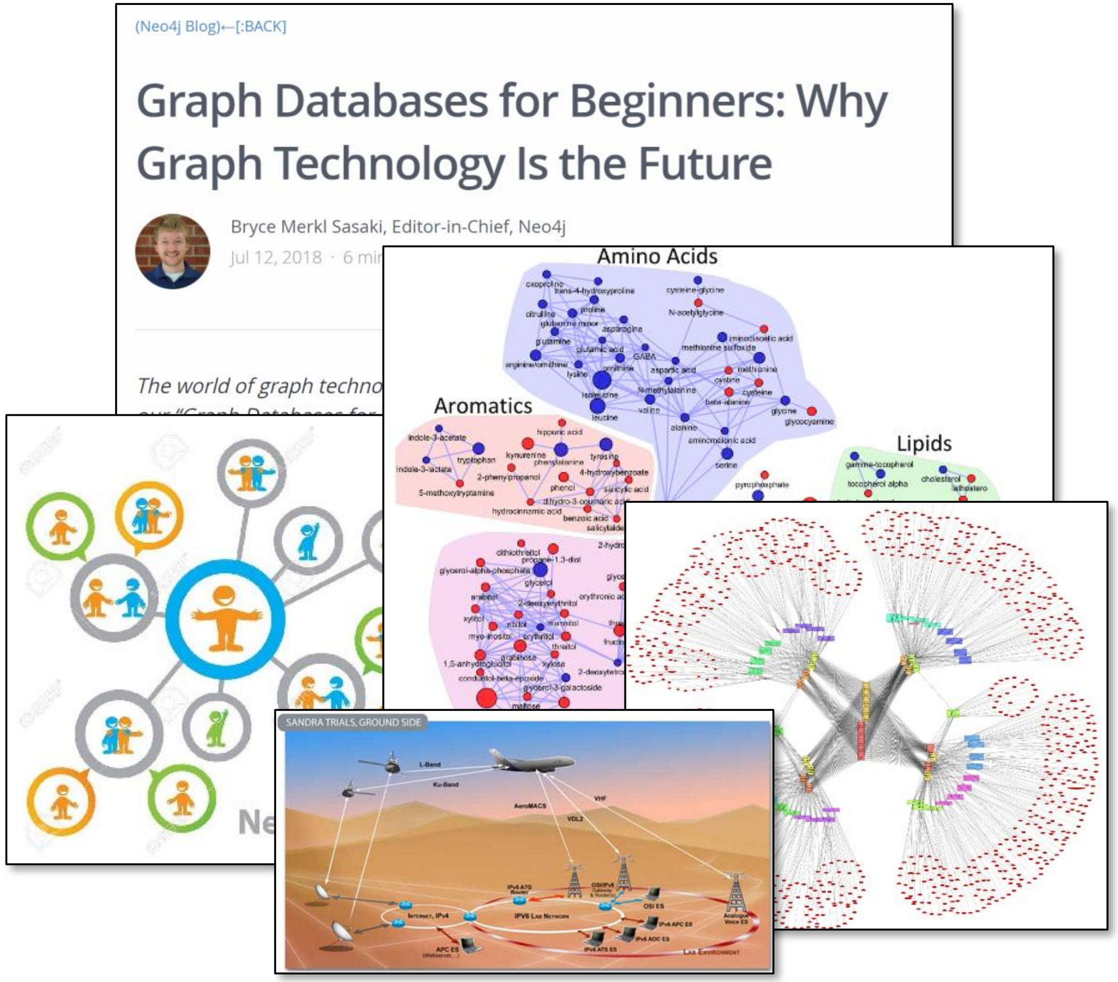
4. For each i for which E^i is non-empty,
   a. Let V^i be the set of vertices attached to edges in
   b. Let $C^i_1, \ldots, C^i_{k_i}$ be the sets of form $D^{\leq i-l}_j \cap V^i$ tha
   edge of E^i on their boundary, (that is, the inter
   contracting edges in $E^{\leq i-l}$). Let $W^i = \cup_j C^i_j$.
   c. Let π be the map of partition $C^i_1, \ldots, C^i_{k_i}$, and
   (W^i, E^i) under π.
   d. $\widetilde{H}^i = \texttt{BoundedSparsify}(H^i, \hat{\epsilon}, p/(2nl))$.
   e. Let $\widetilde{G}^i$ be a pullback of $\widetilde{H}^i$ under π whose edges a

5. Return $\widetilde{G} = \sum_i 2^{-i} \widetilde{G}^i$.

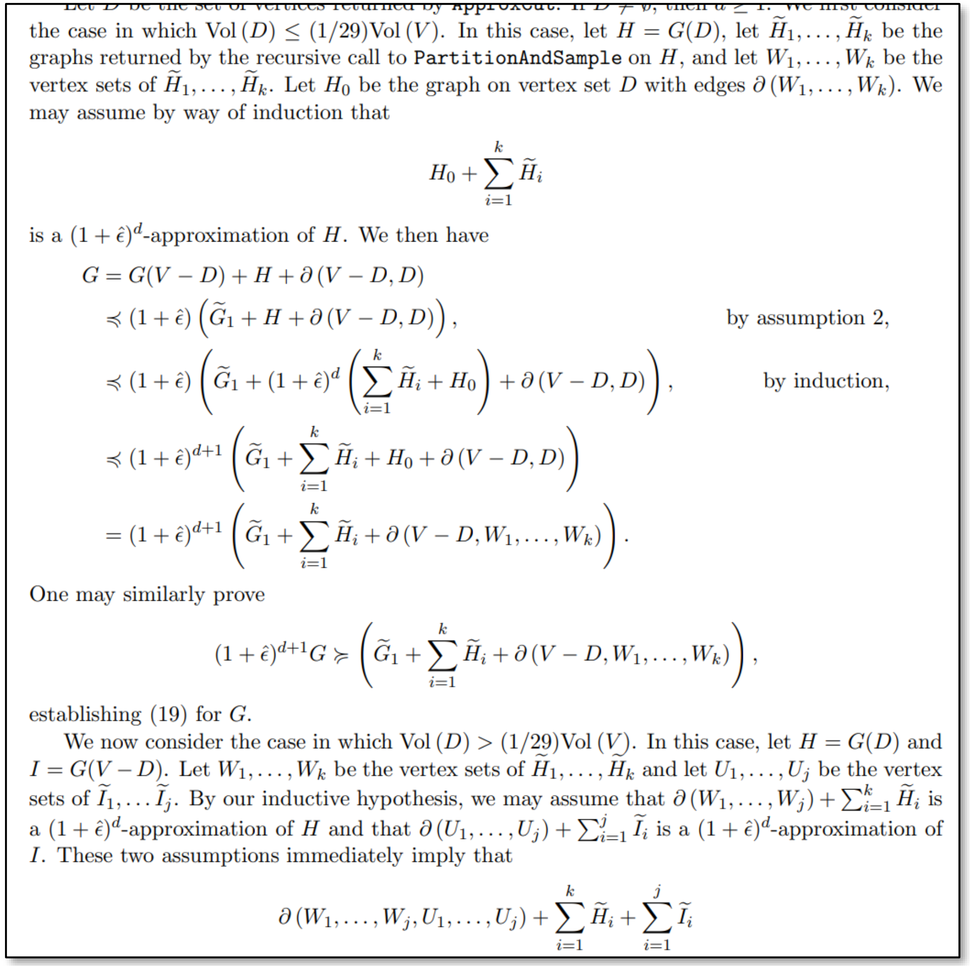
$= G(D)$, let $\widetilde{H}_1, \ldots, \widetilde{H}_k$ be the
on H, and let $W_1, \ldots, W_k$ be the
vertex sets of $H_1, \ldots, H_k$. Let H_0 be the graph on vertex set D with edges $\partial(W_1, \ldots, W_k)$. We may assume by way of induction that
$$H_0 + \sum_{i=1}^{k} \widetilde{H}_i$$
is a $(1+\hat{\epsilon})^d$-approximation of H. We then have
$$\begin{aligned}
G &= G(V-D) + H + \partial(V-D, D) \\
&\preccurlyeq (1+\hat{\epsilon})\left(\widetilde{G}_1 + H + \partial(V-D, D)\right), && \text{by assumption 2,} \\
&\preccurlyeq (1+\hat{\epsilon})\left(\widetilde{G}_1 + (1+\hat{\epsilon})^d \left(\sum_{i=1}^{k} \widetilde{H}_i + H_0\right) + \partial(V-D, D)\right), && \text{by induction,} \\
&\preccurlyeq (1+\hat{\epsilon})^{d+1}\left(\widetilde{G}_1 + \sum_{i=1}^{k} \widetilde{H}_i + H_0 + \partial(V-D, D)\right) \\
&= (1+\hat{\epsilon})^{d+1}\left(\widetilde{G}_1 + \sum_{i=1}^{k} \widetilde{H}_i + \partial(V-D, W_1, \ldots, W_k)\right).
\end{aligned}$$
One may similarly prove
$$(1+\hat{\epsilon})^{d+1} G \succcurlyeq \left(\widetilde{G}_1 + \sum_{i=1}^{k} \widetilde{H}_i + \partial(V-D, W_1, \ldots, W_k)\right),$$
establishing (19) for G.

We now consider the case in which $\mathrm{Vol}(D) > (1/29)\mathrm{Vol}(V)$. In this case, let $H = G(D)$ and $I = G(V-D)$. Let $W_1, \ldots, W_k$ be the vertex sets of $\widetilde{H}_1, \ldots, \widetilde{H}_k$ and let $U_1, \ldots, U_j$ be the vertex sets of $\widetilde{I}_1, \ldots \widetilde{I}_j$. By our inductive hypothesis, we may assume that $\partial(W_1, \ldots, W_j) + \sum_{i=1}^{k} \widetilde{H}_i$ is a $(1+\hat{\epsilon})^d$-approximation of H and that $\partial(U_1, \ldots, U_j) + \sum_{i=1}^{j} \widetilde{I}_i$ is a $(1+\hat{\epsilon})^d$-approximation of I. These two assumptions immediately imply that
$$\partial(W_1, \ldots, W_j, U_1, \ldots, U_j) + \sum_{i=1}^{k} \widetilde{H}_i + \sum_{i=1}^{j} \widetilde{I}_i$$

Challenge 3: What schemes matter in practice?



?



the case in which $\mathrm{Vol}(D) \leq (1/29)\mathrm{Vol}(V)$. In this case, let $H = G(D)$, let $\widetilde{H}_1, \ldots, \widetilde{H}_k$ be the graphs returned by the recursive call to PartitionAndSample on H, and let $W_1, \ldots, W_k$ be the vertex sets of $\widetilde{H}_1, \ldots, \widetilde{H}_k$. Let H_0 be the graph on vertex set D with edges $\partial(W_1, \ldots, W_k)$. We may assume by way of induction that

$$H_0 + \sum_{i=1}^{k} \widetilde{H}_i$$

is a $(1+\hat{\epsilon})^d$-approximation of H. We then have

$$\begin{aligned}
G &= G(V-D) + H + \partial(V-D, D) \\
&\preccurlyeq (1+\hat{\epsilon})\left(\widetilde{G}_1 + H + \partial(V-D, D)\right), && \text{by assumption 2,} \\
&\preccurlyeq (1+\hat{\epsilon})\left(\widetilde{G}_1 + (1+\hat{\epsilon})^d\left(\sum_{i=1}^{k} \widetilde{H}_i + H_0\right) + \partial(V-D, D)\right), && \text{by induction,} \\
&\preccurlyeq (1+\hat{\epsilon})^{d+1}\left(\widetilde{G}_1 + \sum_{i=1}^{k} \widetilde{H}_i + H_0 + \partial(V-D, D)\right) \\
&= (1+\hat{\epsilon})^{d+1}\left(\widetilde{G}_1 + \sum_{i=1}^{k} \widetilde{H}_i + \partial(V-D, W_1, \ldots, W_k)\right).
\end{aligned}$$

One may similarly prove

$$(1+\hat{\epsilon})^{d+1} G \succcurlyeq \left(\widetilde{G}_1 + \sum_{i=1}^{k} \widetilde{H}_i + \partial(V-D, W_1, \ldots, W_k)\right),$$

establishing (19) for G.

We now consider the case in which $\mathrm{Vol}(D) > (1/29)\mathrm{Vol}(V)$. In this case, let $H = G(D)$ and $I = G(V-D)$. Let $W_1, \ldots, W_k$ be the vertex sets of $\widetilde{H}_1, \ldots, \widetilde{H}_k$ and let $U_1, \ldots, U_j$ be the vertex sets of $\widetilde{I}_1, \ldots \widetilde{I}_j$. By our inductive hypothesis, we may assume that $\partial(W_1, \ldots, W_j) + \sum_{i=1}^{k} \widetilde{H}_i$ is a $(1+\hat{\epsilon})^d$-approximation of H and that $\partial(U_1, \ldots, U_j) + \sum_{i=1}^{j} \widetilde{I}_i$ is a $(1+\hat{\epsilon})^d$-approximation of I. These two assumptions immediately imply that

$$\partial(W_1, \ldots, W_j, U_1, \ldots, U_j) + \sum_{i=1}^{k} \widetilde{H}_i + \sum_{i=1}^{j} \widetilde{I}_i$$

Theoretical Analysis

12 graph properties

Insights?

60+ bounds

6 compression schemes

	$\|V\|$	$\|E\|$	Shortest s-t path length	Average path length	Diameter	Average degree	Maximum degree	#Triangles	#Connected components	Chromatic number	Max. indep. set size	Max. cardinal. matching size
Original graph	n	m	$\mathcal{P}$	$\overline{\mathcal{P}}$	D	$\bar{d}$	d	T	$\mathcal{C}$	C_R	$\widehat{I}_S$	$\widehat{M}_C$
Lossy ϵ-summary	n	$m \pm 2\epsilon m$	$1,\ldots,\infty$	$1,\ldots,\infty$	$1,\ldots,\infty$	$\bar{d} \pm \epsilon\bar{d}$	$d \pm \epsilon d$	$T \pm 2\epsilon m$	$\mathcal{C} \pm 2\epsilon m$	$C_R \pm 2\epsilon m$	$\widehat{I}_S \pm 2\epsilon m$	$\widehat{M}_C \pm 2\epsilon m$
Simple p-sampling	n	$(1-p)m$	∞	∞	∞	$(1-p)\bar{d}$	$(1-p)d$	$(1-p^3)T$	$\leq \mathcal{C} + pm$	$\geq C_R - pm$	$\leq \widehat{I}_S + pm$	$\geq \widehat{M}_C - pm$
Spectral ϵ-sparsifier	n	$\tilde{O}(n/\epsilon^2)$	$\leq n$	$\leq n$	$\leq n$	$\tilde{O}(1/\epsilon^2)$	$\geq d/2(1+\epsilon)$	$\tilde{O}(n^{3/2}/\epsilon^3)$	$\overset{w.h.p.}{=} \mathcal{C}$	$\leq d/2(1+\epsilon)$	$\geq 2(1+\epsilon)n/d$	≥ 0
$O(k)$–spanner	n	$O(n^{1+1/k})$	$O(k\mathcal{P})$	$O(k\overline{\mathcal{P}})$	$O(kD)$	$O(n^{1/k})$	$\leq d$	$O(n^{1+2/k})$	$\mathcal{C}$	$O(n^{1/k}\log n)$	$\Omega\left(\frac{n^{1-1/k}}{\log n}\right)$	≥ 0
EO p–1–Triangle Red.	n	$\leq m - \frac{pT}{3d}$	$\overset{w.h.p.}{\leq} \mathcal{P} + p\mathcal{P}$	$\leq \overline{\mathcal{P}} + \frac{pT}{n(n-1)}$	$\overset{w.h.p.}{\leq} D + pD$	$\leq \bar{d} - \frac{pT}{dn}$	$\geq d/2$	$\leq (1 - \frac{p}{d})T$	$\mathcal{C}$	$\geq C_R - pT$	$\leq \widehat{I}_S + pT$	$\geq \widehat{M}_C/2$
remove k deg-1 vertices	$n-k$	$m-k$	$\mathcal{P}$	$\geq \overline{\mathcal{P}} - \frac{kD}{n}$	$\geq D-2$	$\geq \bar{d} - \frac{k}{n}$	d	T	$\mathcal{C}$	C_R	$\geq \widehat{I}_S - k$	$\geq \widehat{M}_C - k$

… comp… h algorithms. Bounds that do not include inequalities hold det… nistically. If … ld w.h.p. (if the involved quantities are large enough). Note … ce the listed … ph of the original graph, m, C_R, $\bar{d}$, d, T, and $\widehat{M}_C$ never incr… over, $\mathcal{P}$, $\overline{\mathcal{P}}$, D, … sparsifier approximates the original graph spectrum.

<u>Summarizations</u> are **not accurate** (graphs can get arbitrarily disconnected)

<u>Sampling</u> is **accurate only in expectation (or w.h.p.) and when not many edges go** (all depends on whether a graph gets disconnected)

<u>Spanners</u> and <u>spectral sparsifiers</u> preserve well their associated properties

Some are new and non-trivial, for example we prove *constructively* a lower bound on the maximum cardinality matching (MCM) size *that depends only on the original MCM size*

Theoretical Analysis

12 graph properties

Insights?

60+ bounds

6 compression schemes

	$\|V\|$	$\|E\|$	Shortest s-t path length	Average path length	Diameter	Average [illegible]	[illegible]imum [illegible]ee	#Triangles	#Connected components	Chromatic number	Max. indep. set size	Max. cardinal. matching size
Original graph	n	m	$\mathcal{P}$	$\overline{\mathcal{P}}$	[illegible]	[illegible]	d	T	$\mathcal{C}$	C_R	$\widehat{I}_S$	$\widehat{M}_C$
Lossy ϵ-summary	n	$m \pm 2\epsilon m$	$1, \ldots$ [illegible]	[illegible]	[illegible]	[illegible]	ϵd	$T \pm 2\epsilon m$	$\mathcal{C} \pm 2\epsilon m$	$C_R \pm 2\epsilon m$	$\widehat{I}_S \pm 2\epsilon m$	$\widehat{M}_C \pm 2\epsilon m$
Simple p–sampling	n	$(1-p)m$	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]$)d$	$(1-p^3)T$	$\leq \mathcal{C} + pm$	$\geq C_R - pm$	$\leq \widehat{I}_S + pm$	$\geq \widehat{M}_C - pm$
Spectral ϵ-sparsifier	n	$\tilde{O}(n/\epsilon^2)$	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]$\epsilon)$	$\tilde{O}(n^{3/2}/\epsilon^3)$	$\overset{w.h.p.}{=} \mathcal{C}$	$\leq d/2(1+\epsilon)$	$\geq 2(1+\epsilon)n/d$	≥ 0
$O(k)$–spanner	n	$O(n^{1+1/k})$	[illegible]	[illegible]	[illegible]	[illegible]	$\leq d$	$O(n^{1+2/k})$	$\mathcal{C}$	$O(n^{1/k} \log n)$	$\Omega\left(\frac{n^{1-1/k}}{\log n}\right)$	≥ 0
EO p–1–Triangle Red.	n	$\leq m - \frac{pT}{3d}$	$\overset{w.h.p.}{\leq} \mathcal{P} +$ [illegible]	[illegible]	[illegible] $\leq D + pD$	$\leq \overline{d} - \frac{pT}{dn}$	$\geq d/2$	$\leq (1 - \frac{p}{d})T$	$\mathcal{C}$	$\geq C_R - pT$	$\leq \widehat{I}_S + pT$	$\geq \widehat{M}_C/2$
remove k deg-1 vertices	$n-k$	$m-k$	$\mathcal{P}$	$\geq \overline{\mathcal{P}} - \frac{kD}{n}$	$\geq D-2$	$\geq \overline{d} - \frac{k}{n}$	d	T	$\mathcal{C}$	C_R	$\geq \widehat{I}_S - k$	$\geq \widehat{M}_C - k$

Table 3: The impact of various compression schemes on the outcome of selected graph algorithms. Bounds that do not include inequalities hold deterministically. If not otherwise stated, the other bounds hold in expectation. Bounds annotated with w.h.p. hold w.h.p. (if the involved quantities are large enough). Note that since the listed compression schemes (except the scheme where we remove the degree 1 vertices) return a subgraph of the original graph, m, C_R, $\overline{d}$, d, T, and $\widehat{M}_C$ never increase. Moreover, $\mathcal{P}$, $\overline{\mathcal{P}}$,D, $\mathcal{C}$, and $\widehat{I}_S$ never decrease during compression. ϵ is a parameter that controls how well a spectral sparsifier approximates the original graph spectrum.

For other analyses and many more [detailed] insights, see the paper

Triangle Reduction is versatile; it also has properties of 2-spanners (or – w.h.p. – O(log n) spanners), cut sparsifiers (and is thus a special case of spectral sparsifiers)

Preserves exactly connectivity and the MST weight

Preserves provably well distances, cuts, and the degree distribution

A "By Product" of Our Work

The first survey on lossy graph compression

Properties of compression classes

8 classes of schemes

Compression scheme	#remaining edges	Work	W…		Preserves best...
Lossy compression schemes that are a part…					
(§ 4.2.1) **Spectral sparsification** ("High-conductance" sampling [1…					…spectra
(§ 4.2.2) **Edge sampling** (simple random-unif…					…e count
(§ 4.3) **Triangle reduction** …				…n)	Several (§ 6)
(§ 4.5.3) **Spanners** ($O(k)$-…			👍, 👎	$O(m+n)$	Distances
(§ 4.5.4) **Lossy summariz…**			👎, 👎*	$O(m+n)$	Count of common neighbors
…(some might be integrated with Slim Graph in future versions):					
(§ 4.6) **Lossy summarizati… …principle** (ApxMdl [101])	$\epsilon m^{\ddagger}$	$O(C^2 \log n + n m_S)^{\ddagger}$	👎, 👍	$O(m+n)$	Unknown
(§ 4.6) **Lossy linearization** [9…	$2kn^{*}$	$O(mdIT)^{*}$	👎, 👎	$O(m+n)$	Unknown
(§ 4.6) **Low-rank approximation** (clustered SVD [119, 132])	—	$O(n_c^3)^{\ddagger}$	👍, 👍	$O(n_c^2)$	[High error rates]
(§ 4.6) **Cut sparsification** (Benczúr–Karger [15])	$O(n \log n\, \epsilon^2)$	$O(m \log^3 n + m \log n / \epsilon^2)^{\ddagger}$	👍, 👎	$O(n+m)$	Cut sizes

Table 2: (§ 4) **Considered lossy compression schemes.** †**W,D** indicate support for weighted or directed graphs, respectively. Symbols used in Slim Graph schemes (p, k) are explained in corresponding sections. $Storage needed to conduct compression. In the **SWeG lossy summarization** [125], ϵ controls the approximation ratio while I is the number of iterations (originally set to 80 [125]). *SWeG covers undirected graphs but uses a compression metric for directed graphs. In **ApxMdl** [101], ϵ controls the approximation ratio, $C \in O(m)$ is the number of "corrections", $m_S \in O(m)$ is the number of "corrected" edges. In **lossy linearization** [95], $k \in O(n)$ is a user parameter, I is the number of iterations of a "re-allocation process" (details in Section V.C.3 in the original work [95]), while T is a number of iterations for the overall algorithm convergence. In **clustered SVD approximation** [119, 132], $n_c \leq n$ is the number of vertices in the largest cluster in low-rank approximation. In **cut sparsifiers** [15], ϵ controls the approximation ratio of the cuts.

No time for this – check the paper ☺ (and stay tuned for the full survey paper coming soon!)

Theoretical Analysis

Triangle Reduction is versatile; it also has properties of:
2-spanners
O(log n) spanners (w.h.p.),
cut sparsifiers
(and is thus a special case of spectral sparsifiers)

Preserves exactly connectivity and the MST weight

Preserves provably well distances, cuts, matchings, the degree distribution, and others..

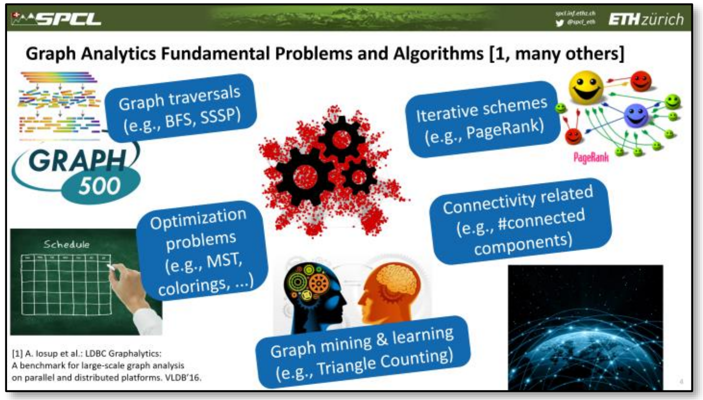
High Accuracy

Less storage

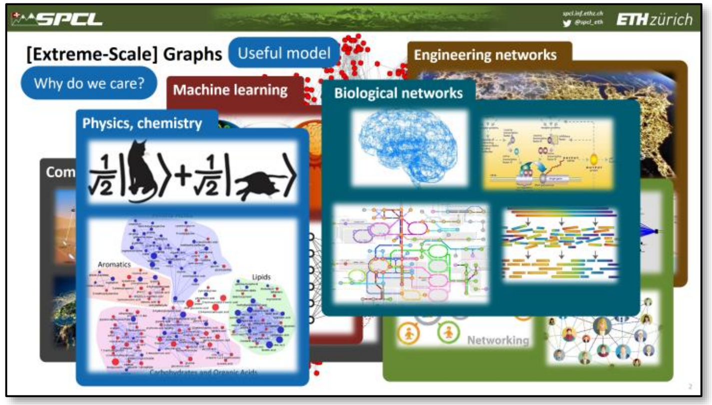
Faster workloads

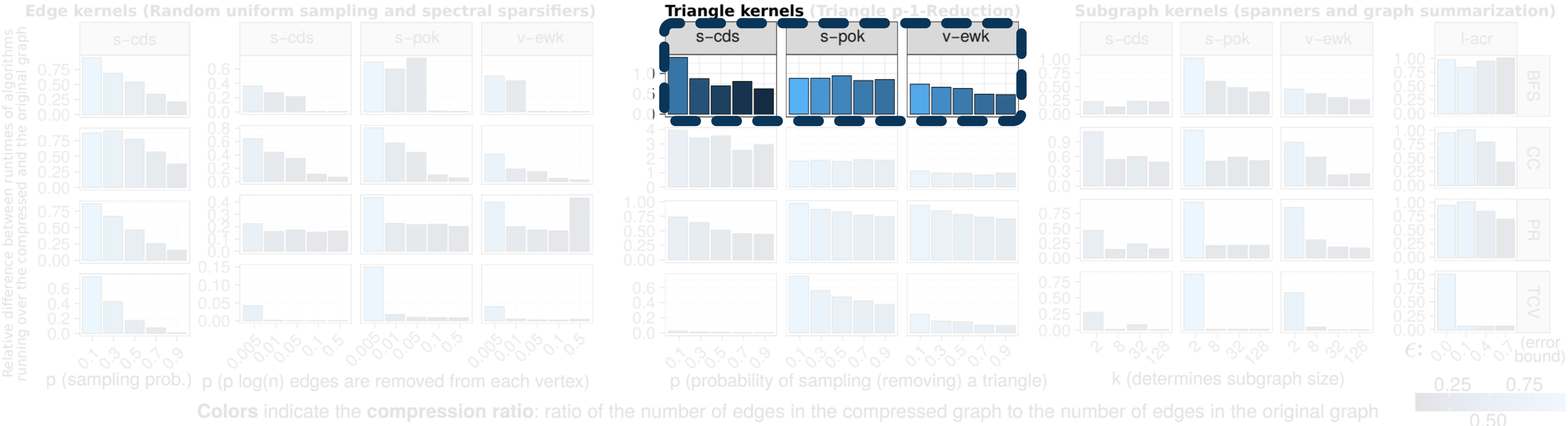
Selected insights…

Various workloads are considered



Various real-world graphs are used





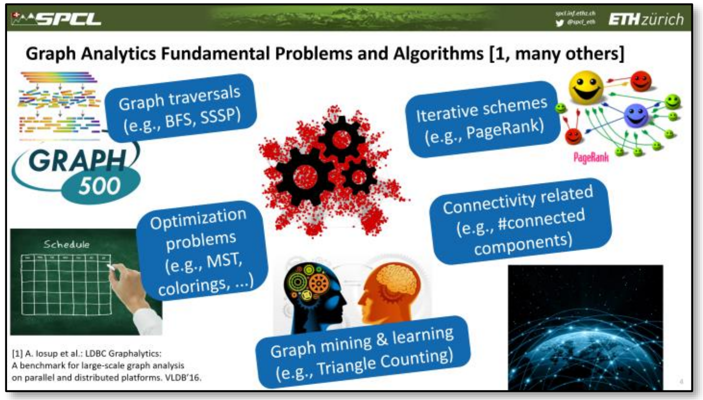
High Accuracy

Less storage

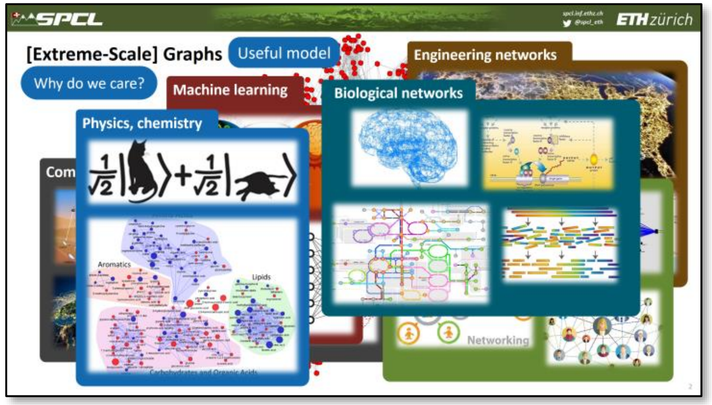
Faster workloads

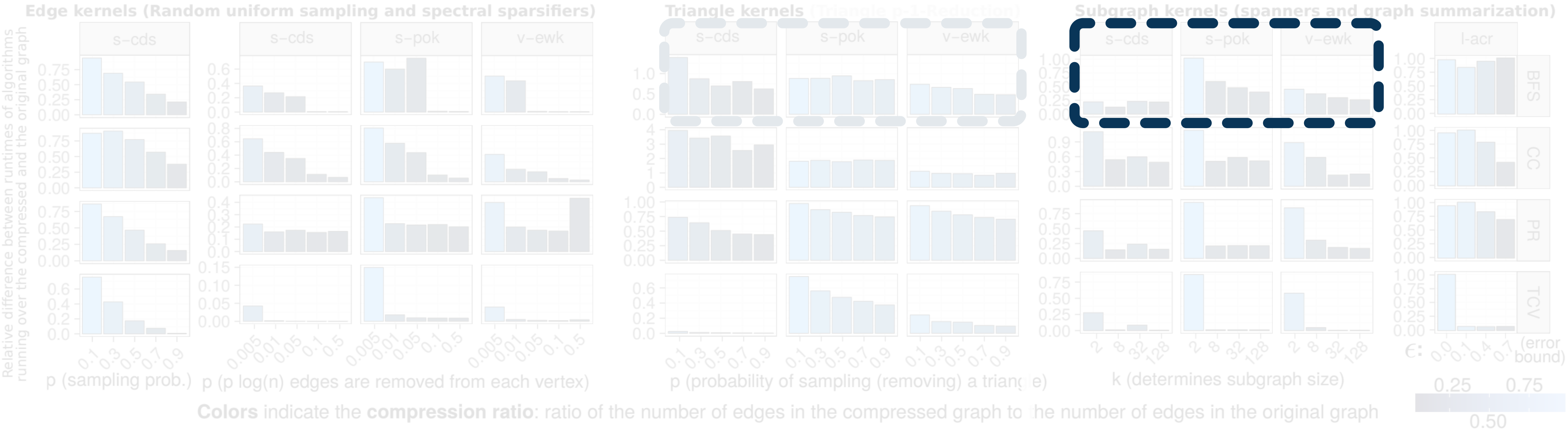
Selected insights...

Various workloads are considered



Various real-world graphs are used



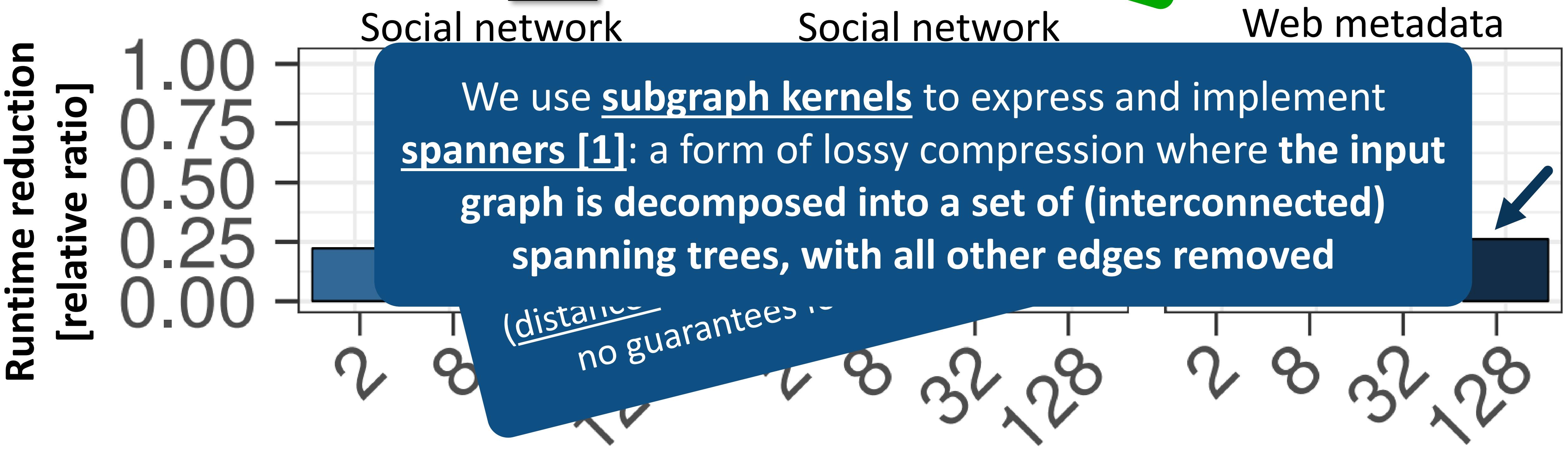


Subgraph Kernels Analysis

Workload: BFS traversal

High Accuracy · Less storage · Faster workloads

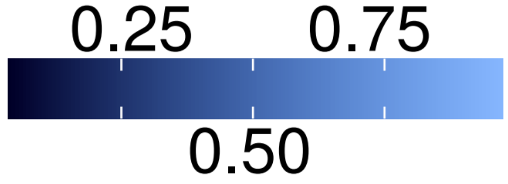
Selected insights…



We use **subgraph kernels** to express and implement **spanners [1]**: a form of lossy compression where **the input graph is decomposed into a set of (interconnected) spanning trees, with all other edges removed**

∝ Diameter *D* of subgraph kernels

(higher *D* → more edges are removed)



Colors indicate the compression ratio: ratio of the number of edges in the compressed graph to the number of edges in the original graph

Storage reduced **even by > 10x** (depends on the structure)

Runtime reduced even by > 75%

[1] D. Peleg. "Graph spanners", The Journal of Graph Theory, 1989

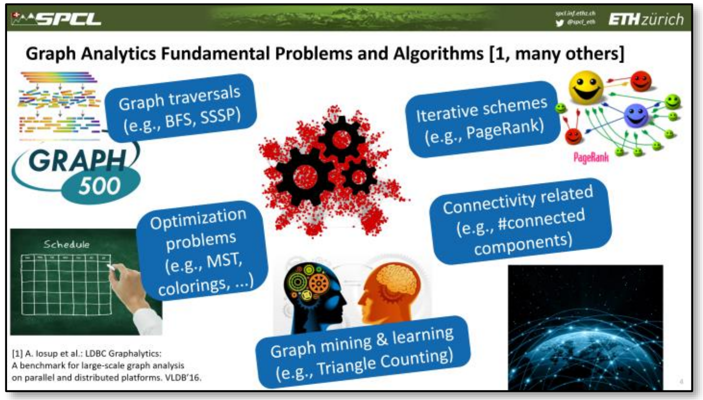
High Accuracy

Less storage

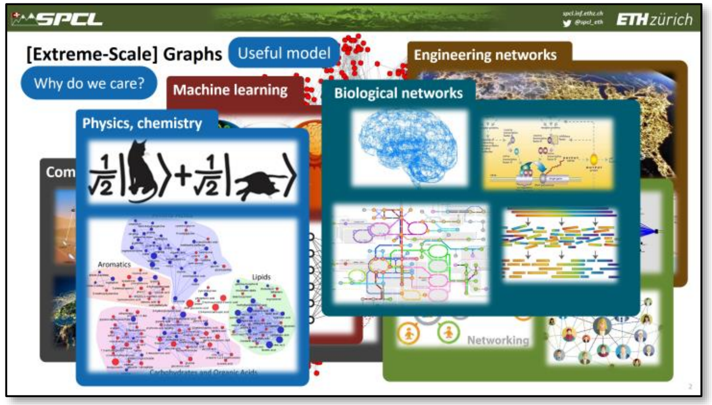
Faster workloads

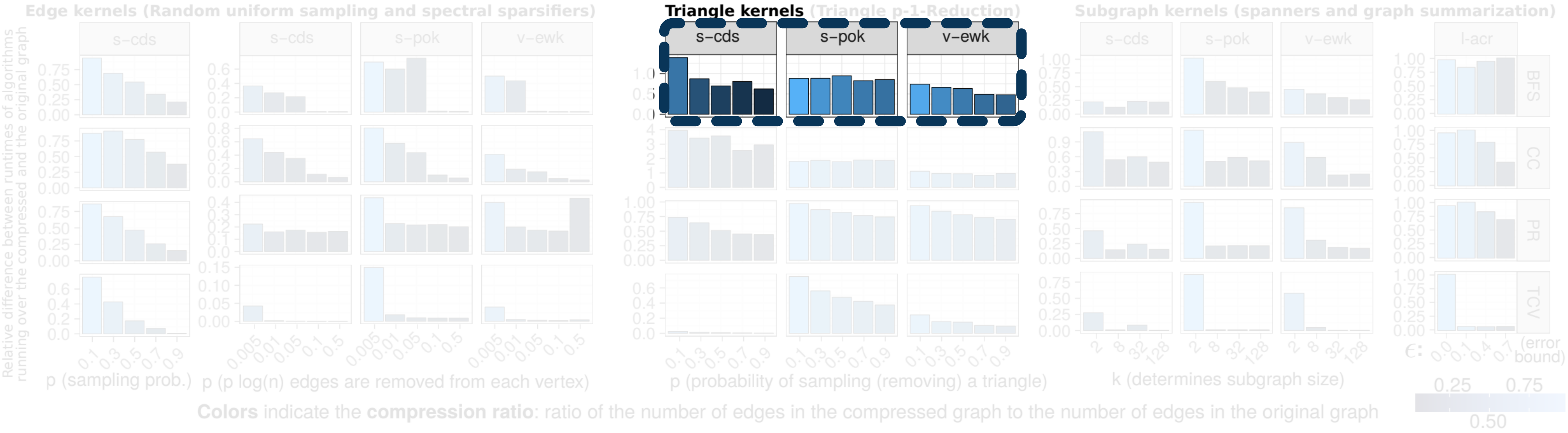
Selected insights…

Various workloads are considered

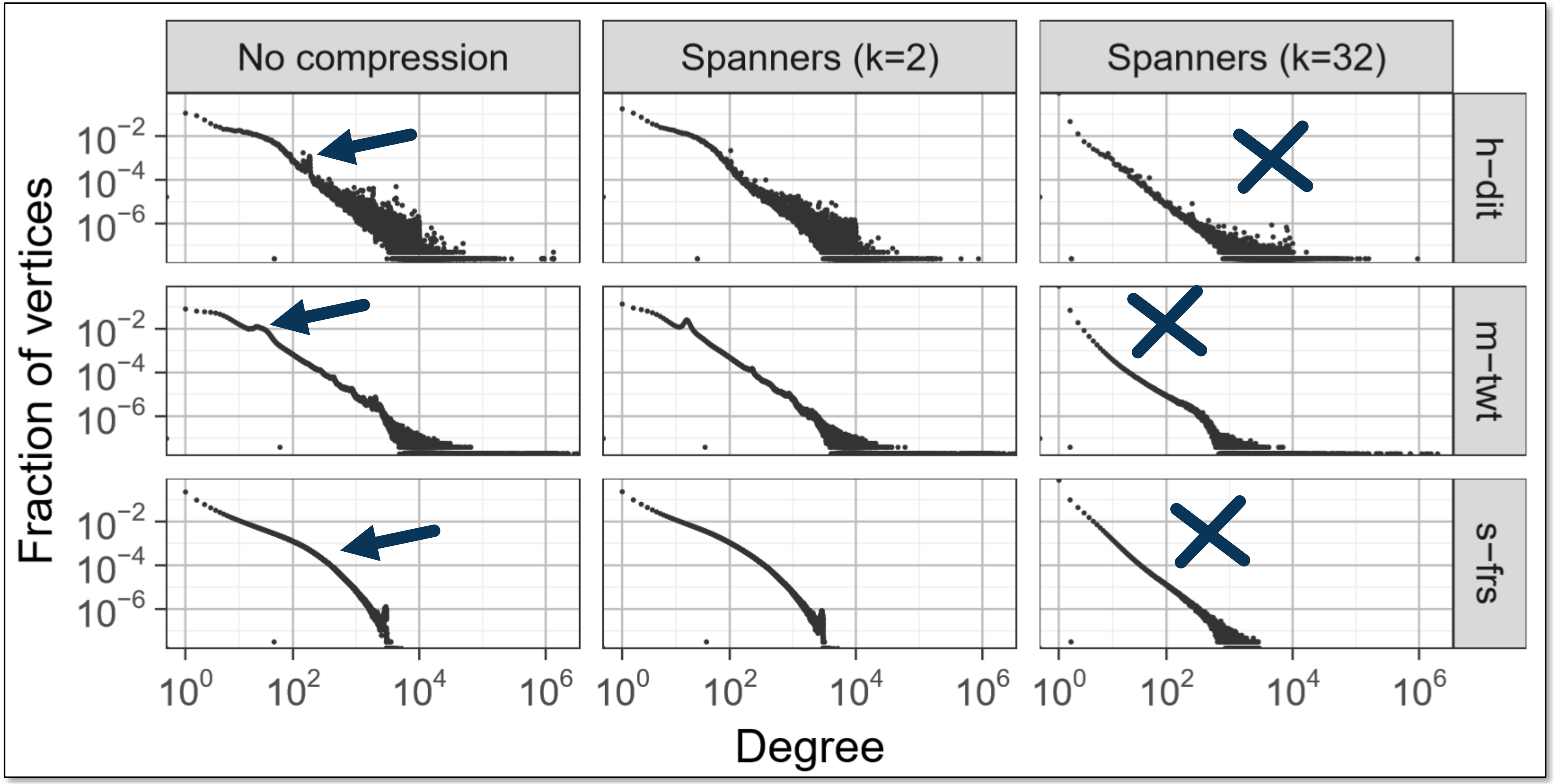


Various real-world graphs are used





Compressing Largest-Scale Graphs with Slim Graph



The first analysis of the impact of spanners on degree distribution

An interesting "leveling" effect

Accuracy Analysis: Compressing Largest-Scale Graphs with Slim Graph

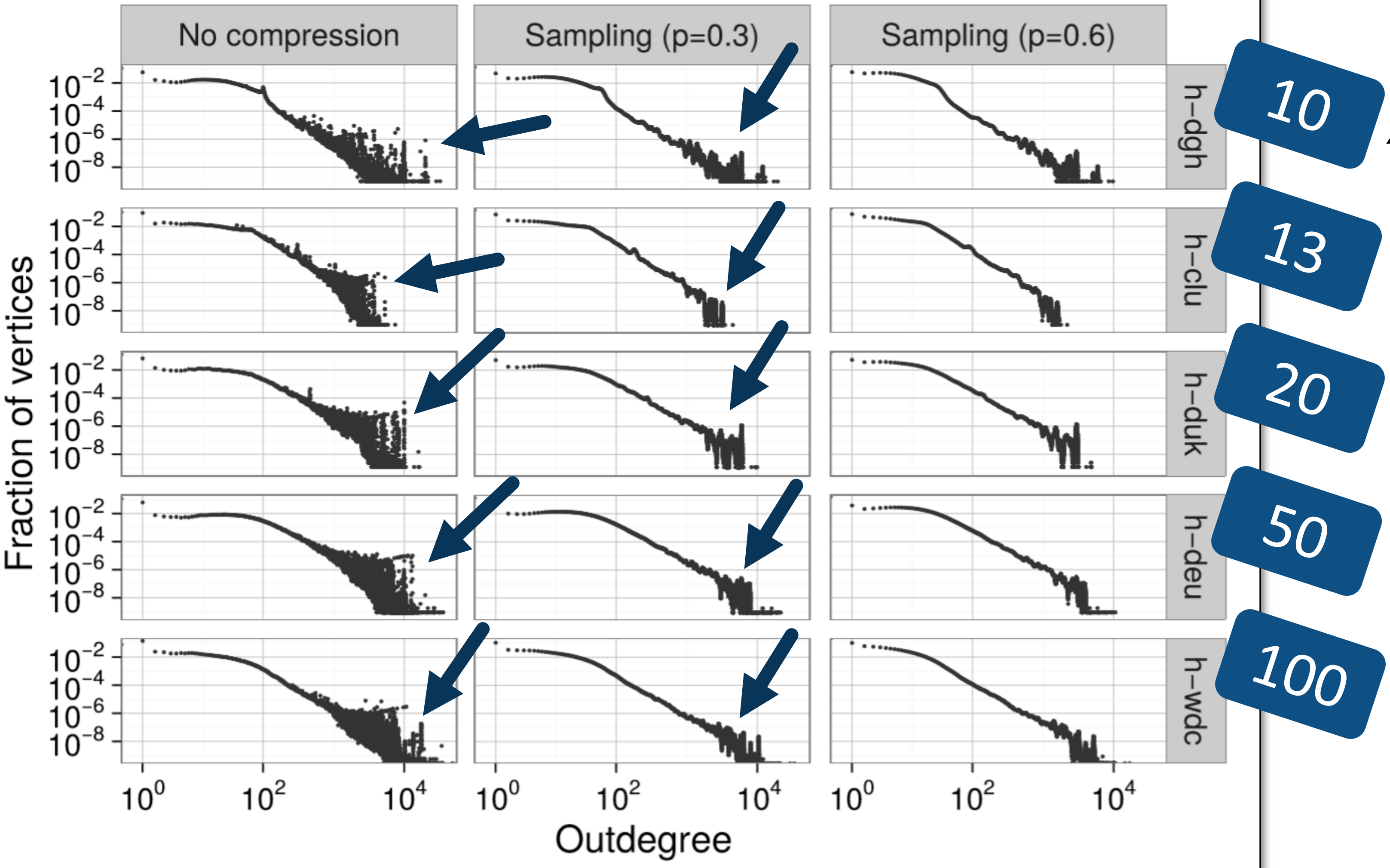


Figure 7: (Accuracy) Impact of random uniform sampling on the degree distribution of large graphs (the largest, h-wdc, has ≈128B edges).

Counts of compute nodes used to compress respective graphs

Largest-scale graph compression so far

5 largest publicly available real-world graphs

"removing the clutter" – (mild) sampling could be used as preprocessing?

Storage Reductions vs. vs. Accuracy Loss

Various real-world graphs are used

Kullback-Leibler divergence values between PageRank probability distributions in the original vs. the compressed graph

Graph	Triangle Reduction		Edge sampling		Spanners		
s-you	0.0121	0.0167	0.1932	0.6019	0.0054	0.2808	0.2993
h-hud	0.0187	0.0271	0.0477	0.1633	0.0340	0.2794	0.3247
il-dbl	0.0459	0.0674	0.0749	0.2929	0.0080	0.1980	0.2005
v-skt	0.0410	0.0643	0.0674	0.2695	0.0311	0.1101	0.2950
v-usa	0.0089	0.0100	0.1392	0.5945	0.0000	0.0074	0.0181

In each category, columns to the right indicate more edges removed

The KL divergence is always larger when more edges are removed