Shortest paths in evolving graphs and imposed system workload imbalance

Nicolas Kourtellis, Ph.D.
Researcher
Telefonica Research, Barcelona

Brief intro

Now: @Telefonica Research, Barcelona

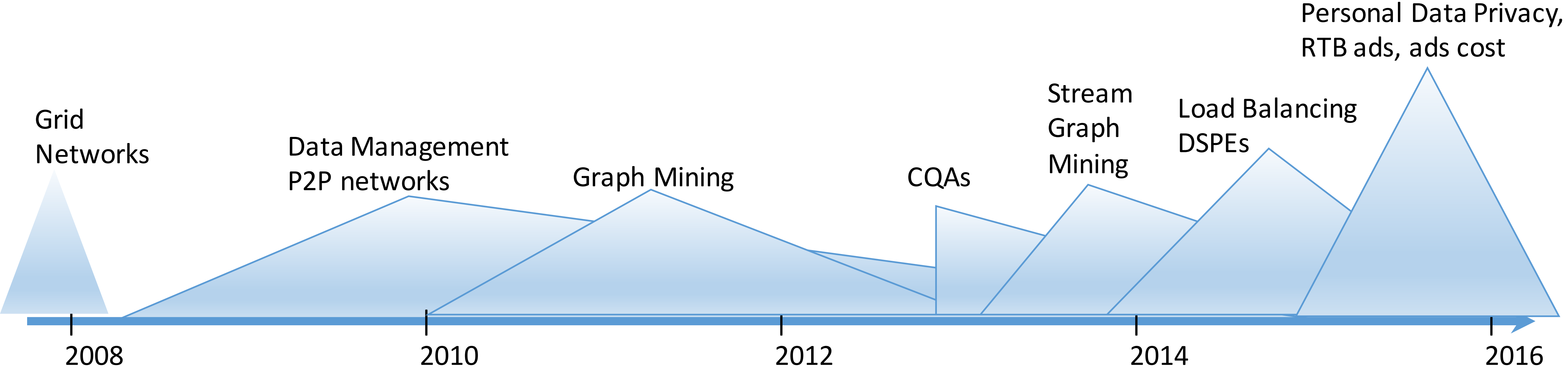
Past: @Yahoo Labs, Barcelona

@USF, Florida

- Distributed systems, P2P networks
- Social Network Analysis
- (Distributed) (Stream) Graph Mining
- Personal Data Privacy
- Reverse engineering RTB, targeted ads
- Hate speech/cyberbullying/fake news

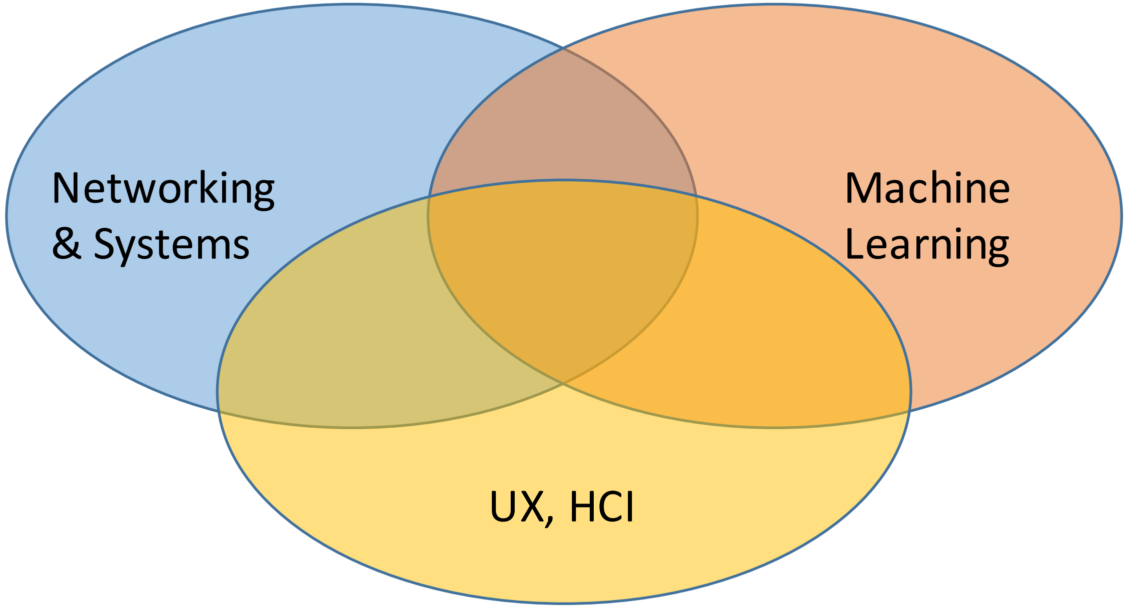
PROTASIS

Research areas of interest



Telefonica Research

- Scientific group created in 2006, located in Barcelona, Spain
- 14+ PhDs, PhD students, interns,…
- Publishing to academic venues + patents for IP of TEF
- Participating in EU and national projects
- Internal innovation projects



Outline

- Scalable Online Betweenness Centrality in Evolving Graphs
- Application: Minimum Wiener & Relaxed Connector Problem
- Load Balancing of Skewed Workloads: Partial Key Grouping

Scalable Online Betweenness Centrality in Evolving Graphs*

*N. Kourtellis, G. De Francisci Morales and F. Bonchi. Scalable Online Betweenness Centrality in Evolving Graphs. IEEE Transactions on Knowledge and Data Engineering, 27(9), Sep. 2015
*N. Kourtellis, G. De Francisci Morales and F. Bonchi. Scalable Online Betweenness Centrality in Evolving Graphs. IEEE ICDE 2016

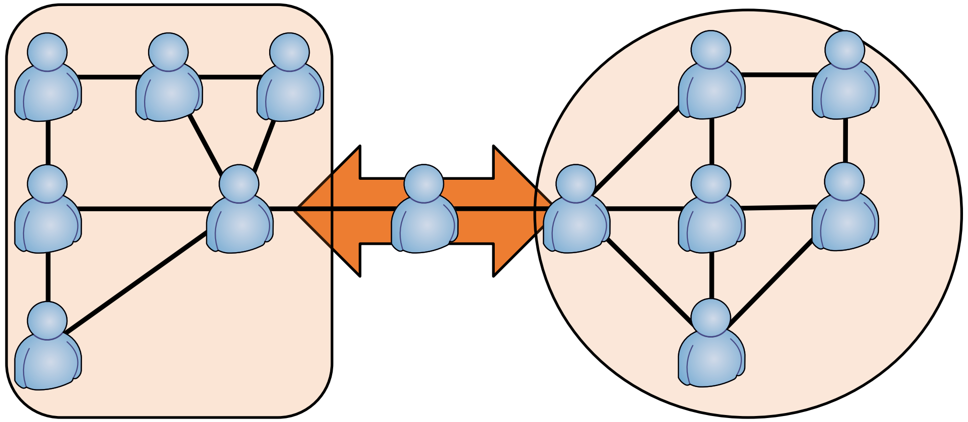
Graph Mining

- Graphs are everywhere!
  - Online social networks, mobile call networks, CQA networks, web networks...
- They change over time!
  - New vertices and edges added
  - Old vertices and edges removed
  - Weights changing
- Graph properties reveal potentials of network processes
  - Diffusion, search, important network elements, etc.
- Studying on dynamic graphs: challenging… depending on the metric

How to measure <u>**exact**</u> betweenness centrality <u>**online**</u> in <u>**dynamic**</u> graphs?

BC: Betweenness Centrality

- Measures how much a vertex lies on the *shortest paths* of other vertices
- O(nm) in unweighted graphs, O(n^2logn+nm) in weighted graphs
- High BC vertices (edges)
  - Control communication between distant vertices
  - Allocate resources for routing, content dissemination, malware detection



$$VBC(v) = \sum_{s,t \in V, s \neq t} \frac{\sigma(s,t \mid v)}{\sigma(s,t)}$$

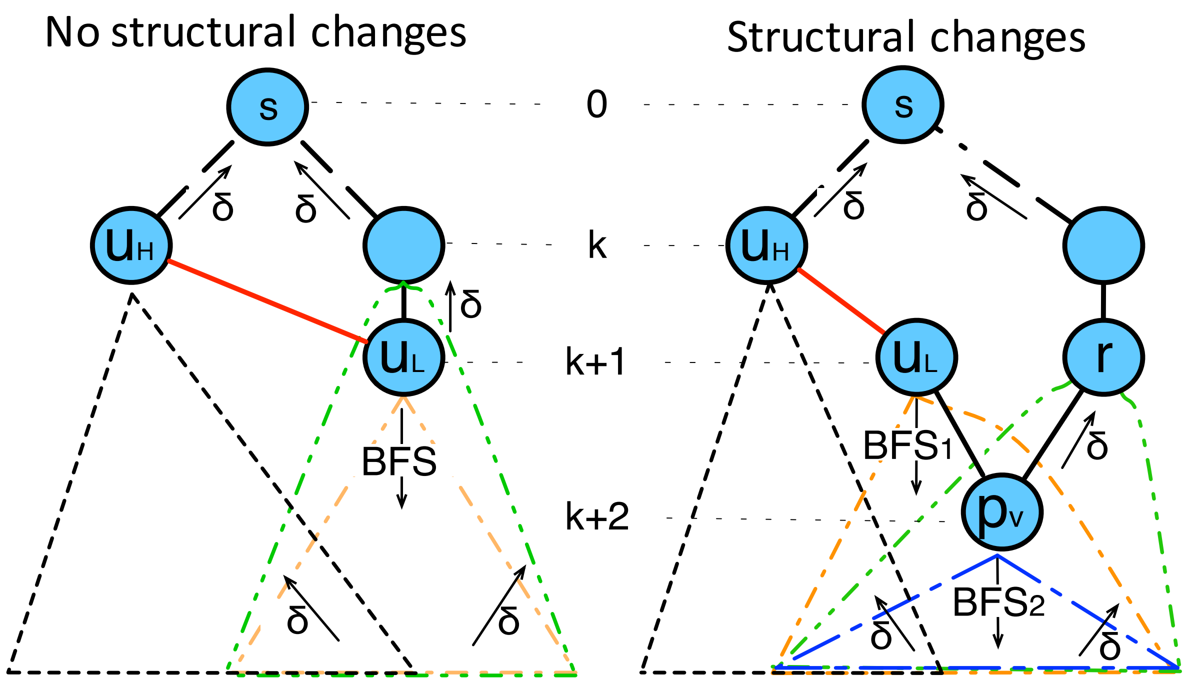
$$EBC(e) = \sum_{s,t \in V, s \neq t} \frac{\sigma(s,t \mid e)}{\sigma(s,t)}$$

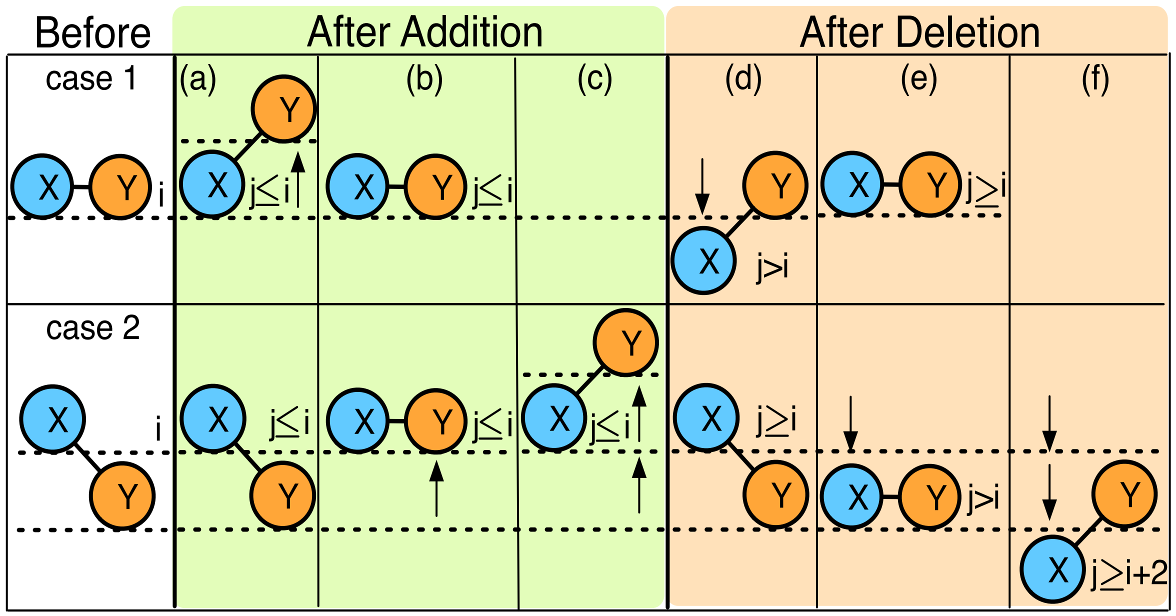
Scalable Online BC in Evolving Graphs

Framework proposed:

- maintains *both vertex & edge betweenness* up-to-date for same computational cost
- handles *both additions and removals* of vertices and edges in a unified approach
- has *reduced space overhead* and is truly *scalable* and amenable to real-world deployment
- can be parallelized and deployed on top of modern distributed, stream and parallel, processing engines

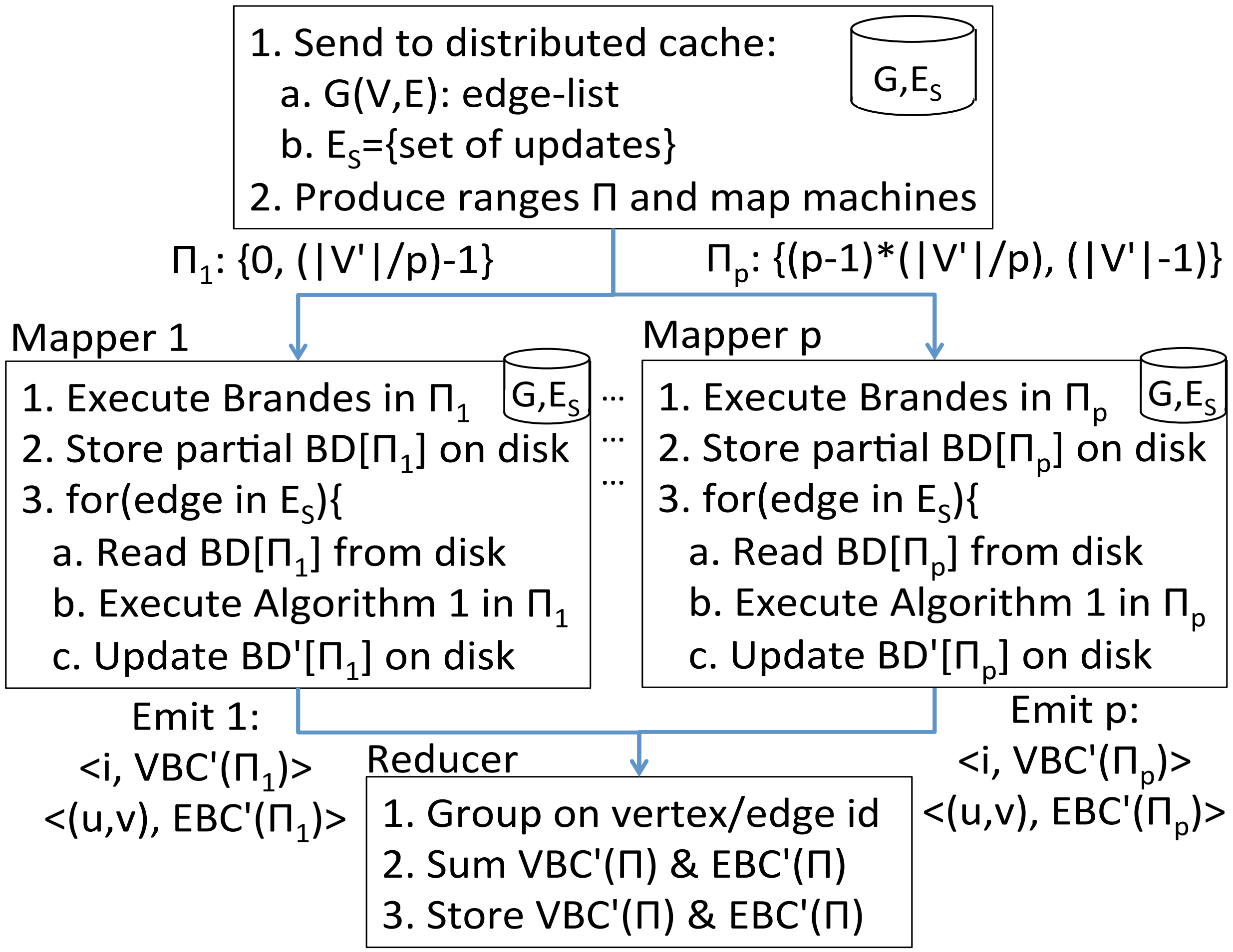
System design for BC measurement





$$\delta_s(v) = \sum_{w:v\in P_s(w)} \frac{\sigma(s,v)}{\sigma(s,w)}(1+\delta_s(w))$$

Parallelization on Hadoop cluster



$$\text{Map}: \quad \langle \Pi_i, \emptyset \rangle \rightarrow [\langle id, \text{PBC}_s(id) \rangle \ \forall id \in G, \forall s \in \Pi_i]$$

$$\text{Reduce}: \quad \langle id, [\text{PBC}_s(id), \ldots] \ \forall s \in V \rangle \rightarrow \langle id, BC(id) \rangle$$

Experimental Setup

	Dataset	$\lvert V\rvert(LCC)$	$\lvert E\rvert(LCC)$	AD	CC	ED
synthetic	1k	1000	5895	11.8	0.263	5.47
	10k	10 000	58 539	11.7	0.219	6.56
	100k	100 000	587 970	11.8	0.207	7.07
	1000k	1 000 000	5 896 878	11.8	0.204	7.76
real-world	wikielections	7066	100 780	8.3	0.126	3.78
	slashdot	51 082	117 377	51.1	0.006	5.23
	facebook	63 392	816 885	63.7	0.148	5.62
	epinions	119 130	704 571	12.8	0.081	5.49
	dblp	1 105 171	4 835 099	8.7	0.6483	8.18
	amazon	2 146 057	5 743 145	3.5	0.0004	7.46

- 3 implementations (MP, MO, DO)
- 100 random edges added/removed
- real and synthetic networks
- Single server tests
  - 8-core Intel Xeon @2.4GHz, 50GB RAM
- Hadoop cluster, 100s machines
  - 8-core Intel Xeon @2.4GHz, 24GB RAM
- Speedup comparison with Brandes' and 3 state-of-art
- Averaged over 10 executions for each setup

Key performance results

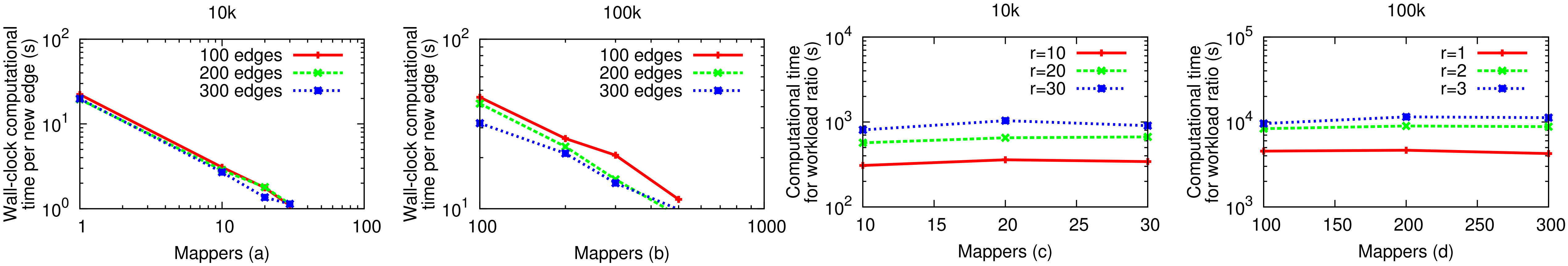
Dataset	$\|V\|$	MO avg (max)	[21]	[24]	[17]
wikivote	7k	75 (181)		3	
contact	10k	75 (153)		4	
UCI (fb-like)	2k	32 (90)	18		
ca-GrQc	4k	31 (378)	68	2	40
ca-HepTh	8k	42 (80)	358		40
adjnoun	.1k	48 (172)			20
ca-CondMat	19k	94 (395)			109
as-22july06	23k	70 (291)			61
slashdot (50GB)	51k	88 (178)			X

[17]: Green et al., [21]: Kas et al., [24]: Qube

Dataset	Addition			Removal		
	Min	Med	Max	Min	Med	Max
1k	3	12	23	2	10	19
10k	16	34	62	2	35	155
100k	21	49	96	4	45	134
1000k	5	10	20	1	12	78
wikielections	9	47	95	1	45	92
slashdot	15	25	121	8	24	127
facebook	10	66	462	1	102	243
epinions	24	56	138	2	45	90
dblp	3	8	15	3	8	429
amazon	2	4	15	2	3	5

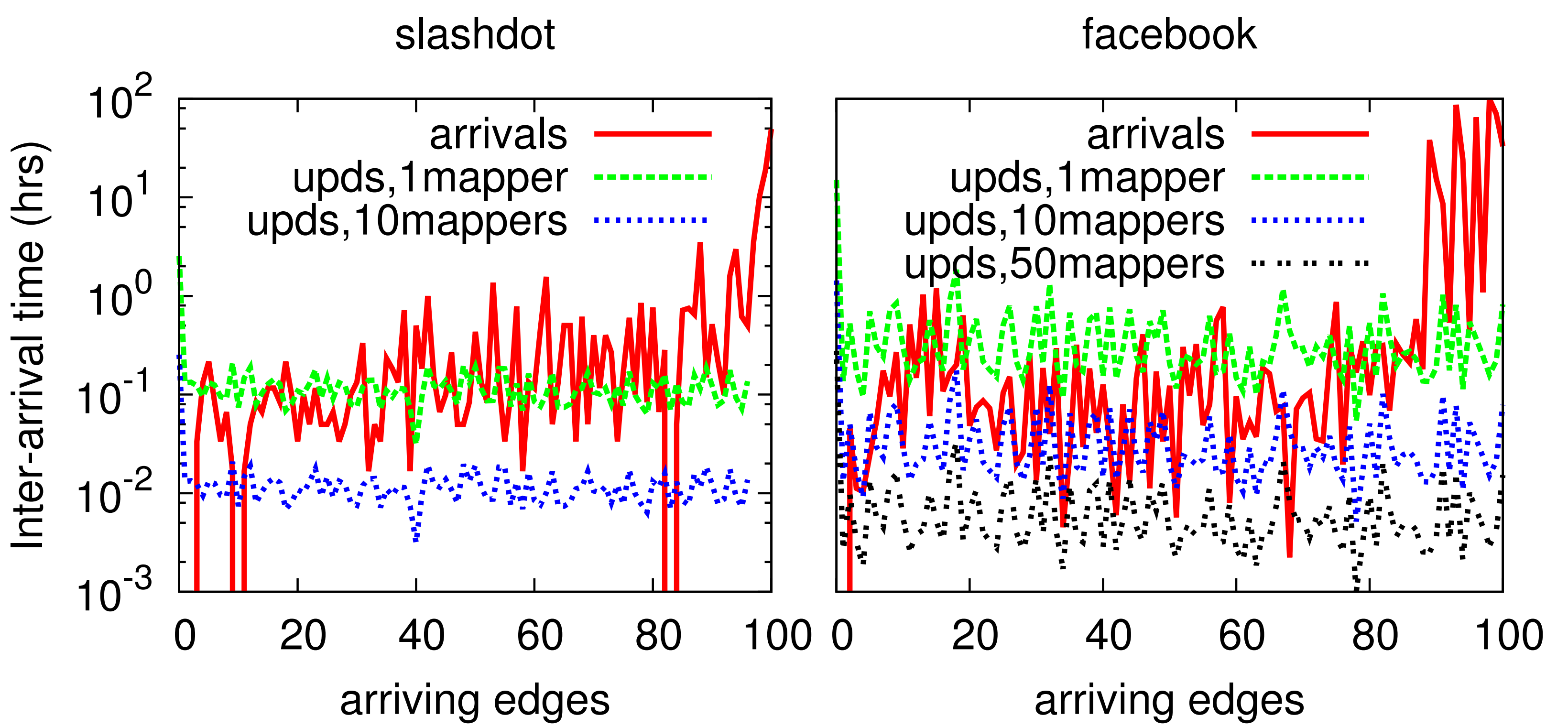
- Better at networks with high clustering
- Better at memory consumption
- Handles vertex & edge betweenness simultaneously

System Scalability



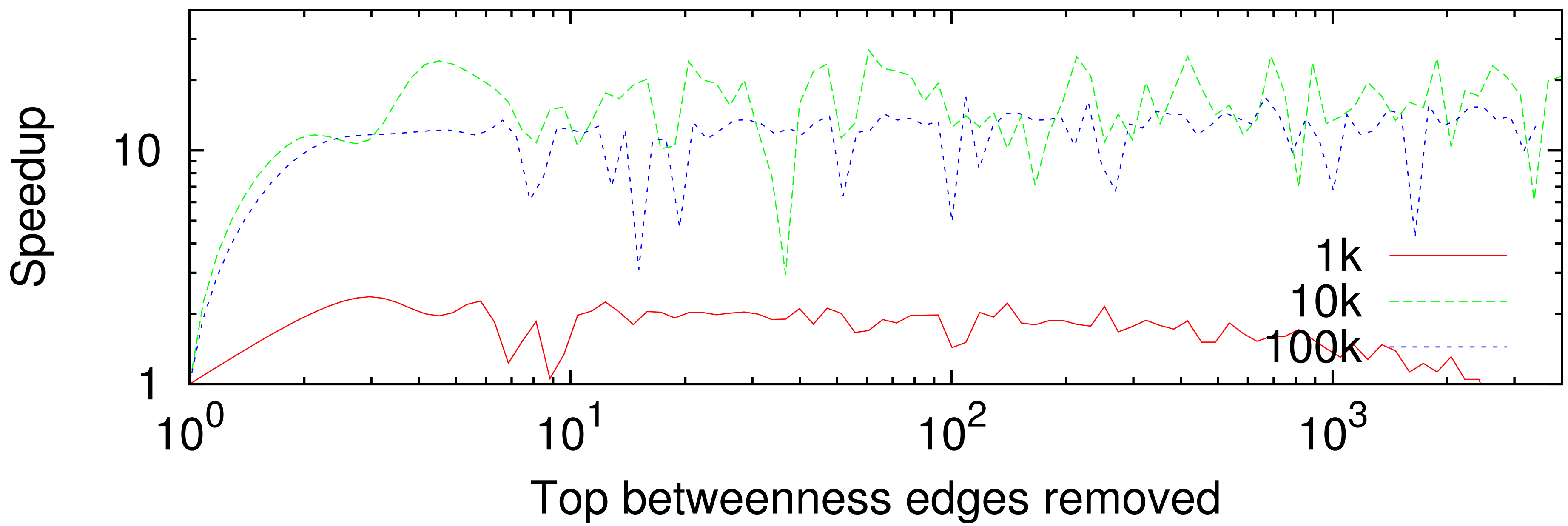
- Computation decreases almost linearly regardless of workload and graph size
- Workload/mappers ”static” -> computation time ”static”

Online updates of BC



- Online detection of top (or changing) BC vertices and edges for better system design

Application: Girvan-Newman Communities



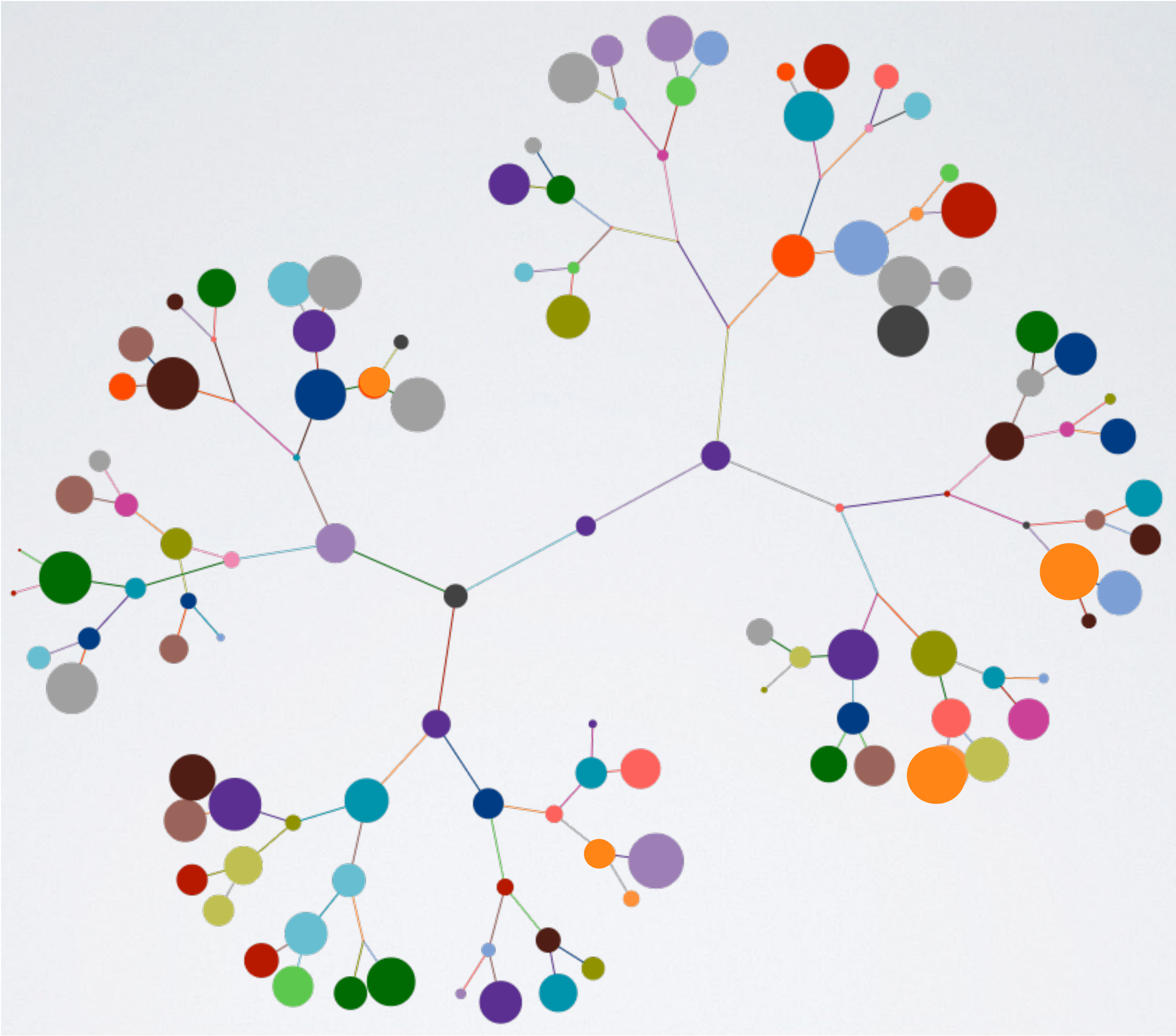
Research extensions & improvements

- Exact solutions
  - Skip unmodified parts of graph, different algorithmic constructions, …
- Approximations
  - Sampling, batches, hypergraph sketches, …
- Applications
  - Spam detection, attacks in bee colonies, …

Application:
Minimum Wiener & Relaxed Connector Problem*

* N. Ruchansky, F. Bonchi, D. García-Soriano, F. Gullo, N. Kourtellis, The minimum wiener connector problem. ACM SIGMOD 2015.
* N. Ruchansky, F. Bonchi, D. García-Soriano, F. Gullo, N. Kourtellis, To Be Connected, or Not to Be Connected: That is the Minimum Inefficiency Subgraph Problem. ACM CIKM 2017.

Application: Minimum Wiener Connector



Infected patients: how did it spread?

Proteins: what connects them?

Ads: Who else should be displayed to?

Terrorists: Who else was involved in the attack?

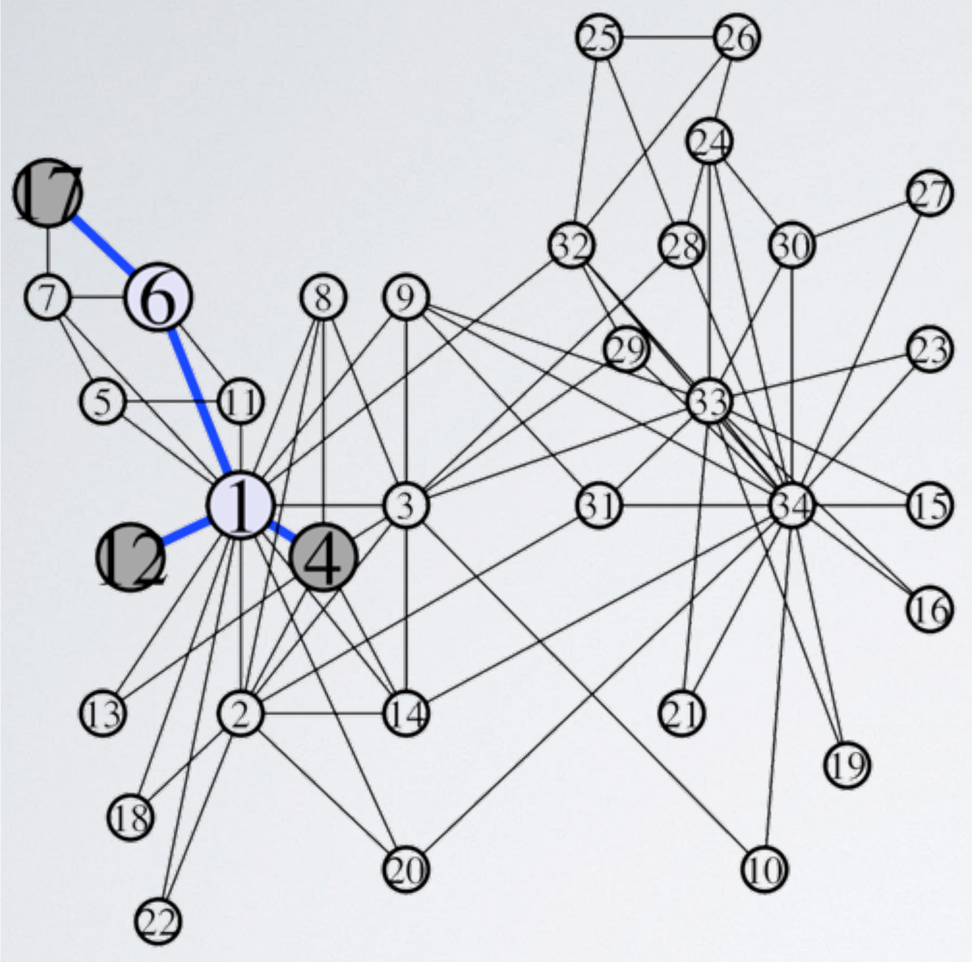
Minimum Wiener Connector Problem

- Find the connected subgraph containing and minimizing the **Wiener Index** (the sum of pairwise shortest distances) between a set of query vertices Q:

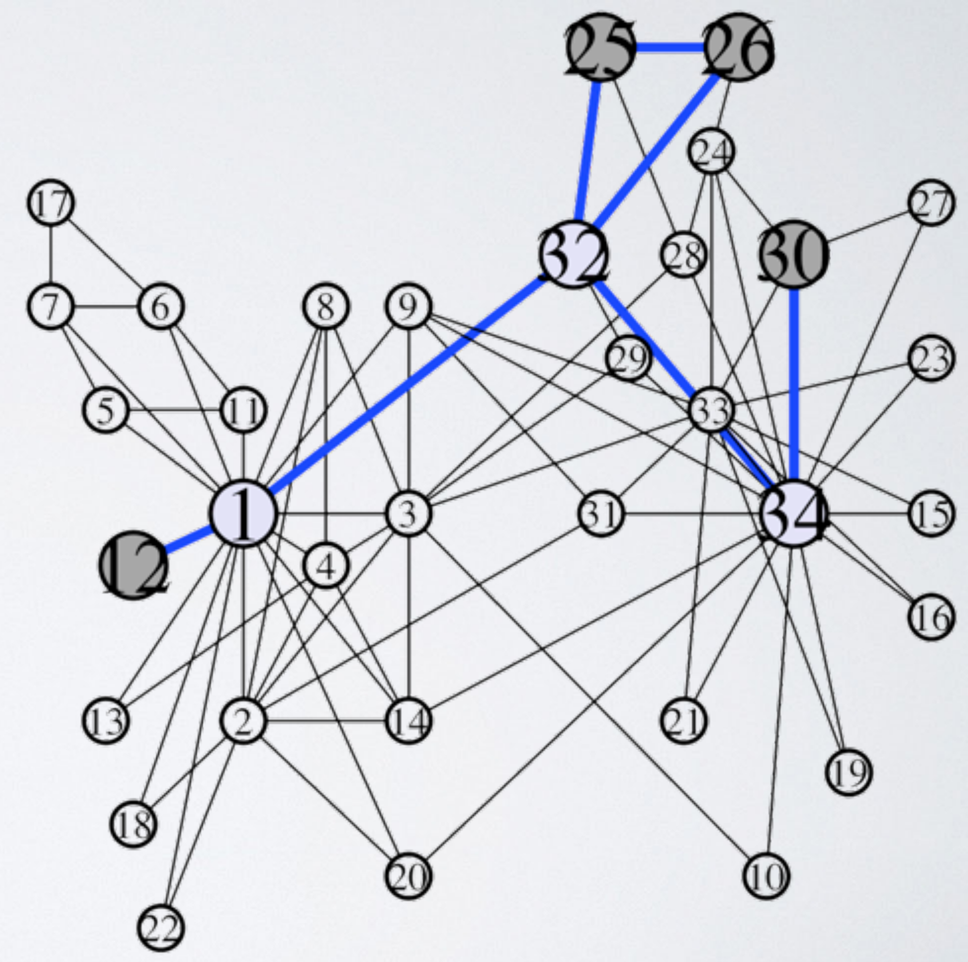
$$H^* = \operatorname*{arg\,min}_{G[S]:Q\subseteq S\subseteq V} \sum_{\{u,v\}\in S} d_{G[S]}(u,v)$$

- NP-hard to find the optimal such connector
- Devised a constant factor approximation algorithm
  - runs in $\tilde{O}(|Q||E|)$
  - parameter-free

Example: Karate Club



Query nodes in same community



Query nodes from different communities

Smaller, denser and more central vertices

		email	yeast	oregon	astro	dblp	youtube	
$\|V[H]\|$		671	819	9028	12758	11804	17865	CTP
		155	188	4556	1735	7349	5615	CPS
		137	100	1846	598	842	684	PPR
		26	**24**	26	26	25	19	ST
		24	**24**	**23**	**23**	**23**	**17**	WS-Q
$\delta(H)$		0.016	0.016	0.01	<0.01	<0.01	0.01	CTP
		0.047	0.028	0.02	0.019	0.01	<0.01	CPS
		0.029	0.039	0.02	0.07	0.01	0.02	PPR
		0.080	0.088	0.090	0.09	0.08	0.1	ST
		0.093	**0.091**	**0.106**	**0.13**	**0.11**	**0.13**	WS-Q
$bc(H)$		<0.01	<0.01	<0.01	<0.01	<0.01	<0.01	CTP
		0.03	0.02	<0.01	<0.01	<0.01	<0.01	CPS
		0.03	<0.01	<0.01	0.02	0.01	<0.01	PPR
		0.09	0.07	0.10	0.11	0.10	0.13	ST
		0.11	**0.11**	**0.12**	**0.14**	**0.12**	0.18	WS-Q
$\mathbf{W}(H)$		$\approx 750k$	$\approx 2M$	$\approx 137M$	$\approx 292M$	$\approx 400M$	$\approx 1.5G$	CTP
		54 598	69 296	$\approx 50M$	$\approx 8.3M$	$\approx 12.6M$	$\approx 561M$	CPS
		52 222	15 838	$\approx 7.5M$	40 079	$\approx 1.2M$	$\approx 1.3M$	PPR
		1 200	1 259	1 164	1 318	3 371	1 324	ST
		968	**931**	**923**	**1 007**	**2 043**	**956**	WS-Q

Summary of results

- Finding a **connector for a set of query nodes** in a graph is an interesting and relevant problem
- Wiener Index is the **sum of shortest-path distances**, which is intuitive graph measure of closeness
- Constant factor approximation algorithm that runs in $\tilde{O}(|Q||E|)$
  - Parameter-free
- Returns small and dense subgraphs
  - Easy to visualize and explain
  - Fast to compute
  - No matter whether the query nodes belong to the same community or not
  - Adds “important” nodes (high centrality) while optimizing distances between all added nodes
- For query nodes across different communities:
  - Connector contains nodes that span structural holes (bridges between communities)

Extension: Relaxed Connector Problem

- Previously, we considered the question of the form:
  - *Summarize the relationships that exist among among **all** query vertices*
- **Wiener Connector** gave us a small, informative answer
- Another type of question can be:
  - *Summarize the relationships that **may** exist among **all, or parts** of query vertices*

Desired properties of relaxed solution

- Parsimonious vertex addition
  - Vertices should be added iff they help form a more **cohesive** subgraph
- Outlier Tolerance
  - Query vertices which are far from others should remain disconnected
- Multi-community awareness
  - If the query vertices span multiple communities, connectedness should not be imposed among them

Minimizing Network Inefficiency

- Find the relaxed connected subgraph that minimizes the network inefficiency among a set of query vertices Q:

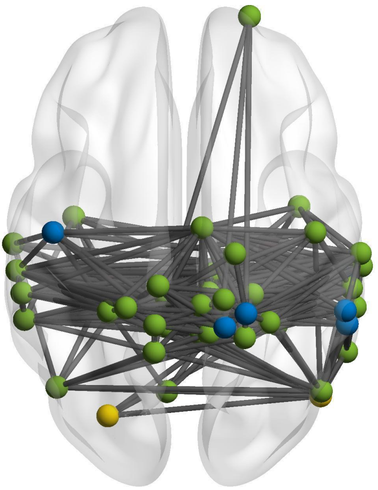
$$\mathcal{I}(G) = \sum_{\substack{u,v \in V \\ u \neq v}} 1 - \frac{1}{d_G(v,u)} \qquad H^* = \operatorname*{arg\,min}_{G[S]:Q \subseteq S \subseteq V} I(G[S])$$

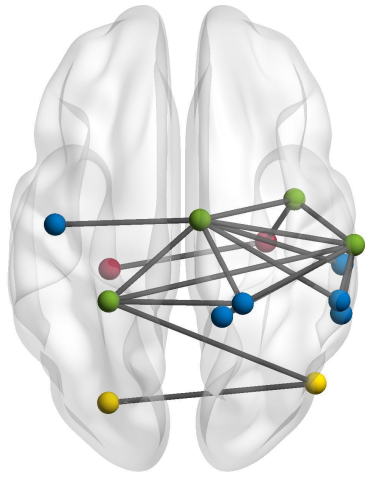
- NP-hard to find the optimal such relaxed connector
- Parameter-free

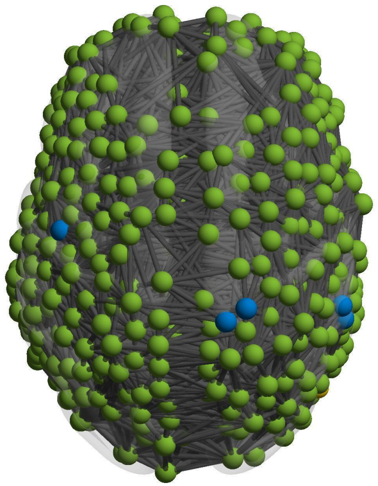
Proposed Greedy Algorithm

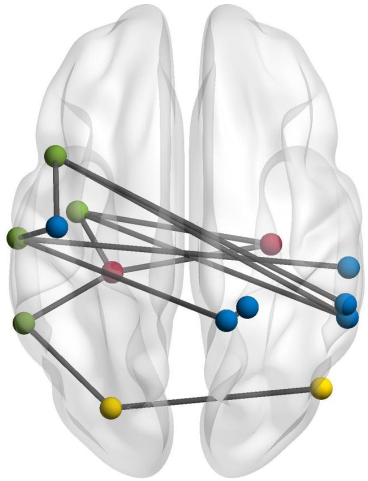
1. Start with a (Wiener?) connector for Q
2. Remove one vertex at a time until Q is disconnected
   - Note: Demands recomputation of pairwise shortest paths within intermediate solutions
3. Choose the intermediate solution that minimizes I(G)

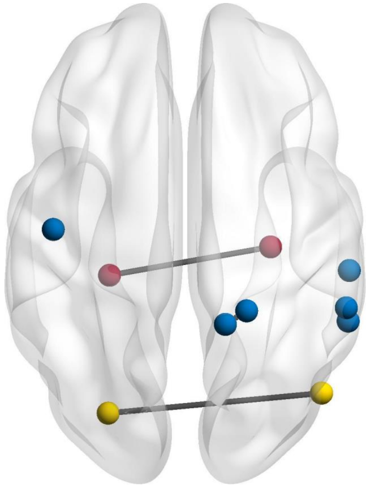
Example: Brain Co-Activation Network

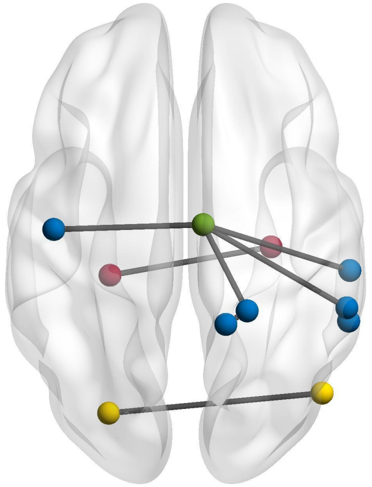

Centerpiece Subgraph


Minimum Wiener Connector


Cocktail Party


Bump Hunting


MDL-based


Minimum Inefficiency

- 638 vertices (cortical areas)
- 18625 edges (functional associations)

- memory and motor function (blue vertices)
- emotions (yellow vertices)
- visual processing (red vertices)
- green vertices are added to produce solution

Shortest-paths framework on evolving graphs

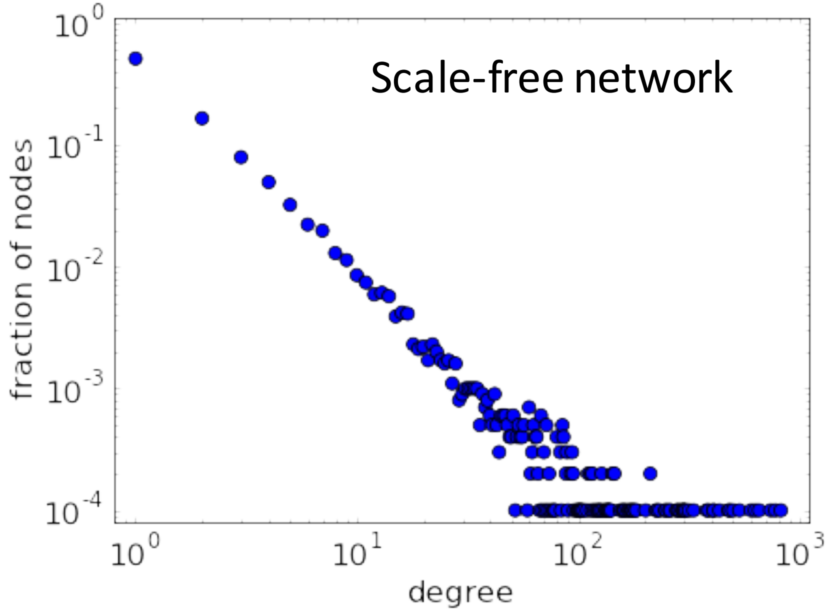
1. Inputted graphs are dynamic
   - Need to update shortest paths for re-computing full connector
2. Re-computation of shortest paths for the relaxed solution
   - Nodes sequentially removed -> ”dynamic” subgraph for study
3. Estimated shortest path distances (via oracles)
   - Oracles need to be updated (landmarks, etc.)
4. Parallelization of computation
   - Shortest path summaries computed faster and efficiently

Load Balancing of Skewed Workloads: PKG*

* MAU Nasir, G. De Francisci Morales, D. Garcia-Soriano, N. Kourtellis, M. Serafini. Partial Key Grouping: Load-Balanced Partitioning of Distributed Streams. IEEE ICDE 2015
* MAU Nasir, G. De Francisci Morales, N. Kourtellis, M. Serafini. When Two Choices Are not Enough: Balancing at Scale in Distributed Stream Processing. IEEE ICDE 2016

Data Distributions: Usually skewed!

- Example Domains
  - Social Networks
  - Web
  - Economy
  - Biology
- Example metrics
  - Centrality: degree, betweenness, closeness
- Skewed Distribution
  - Power-law
  - Zipf Distribution
  - Log Normal



Computing on Stream Processing Engines

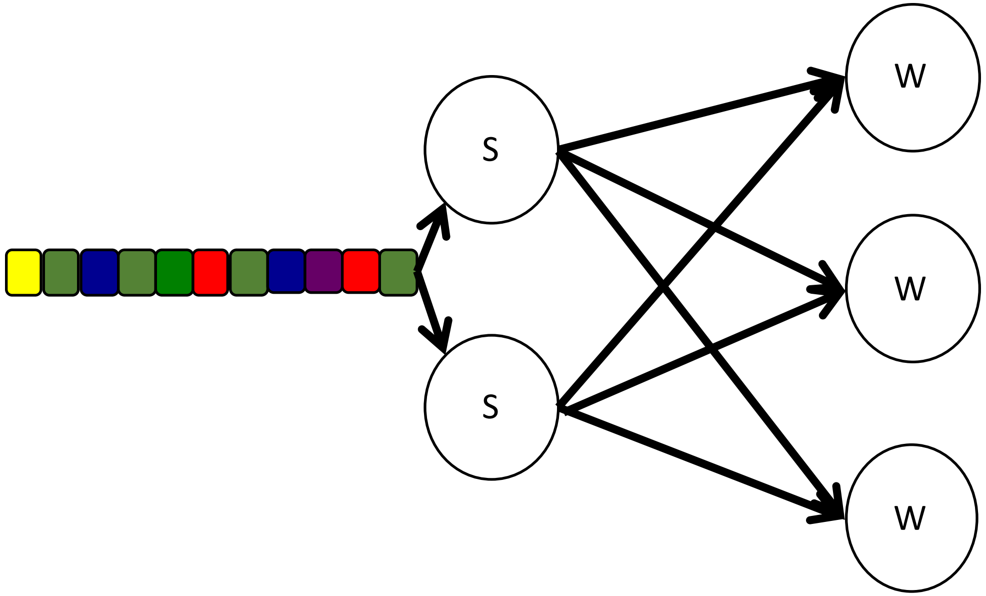
- Online Machine Learning
- Real Time Query Processing
- Graph Mining
- Continuous Computation

APACHE STORM

Flink

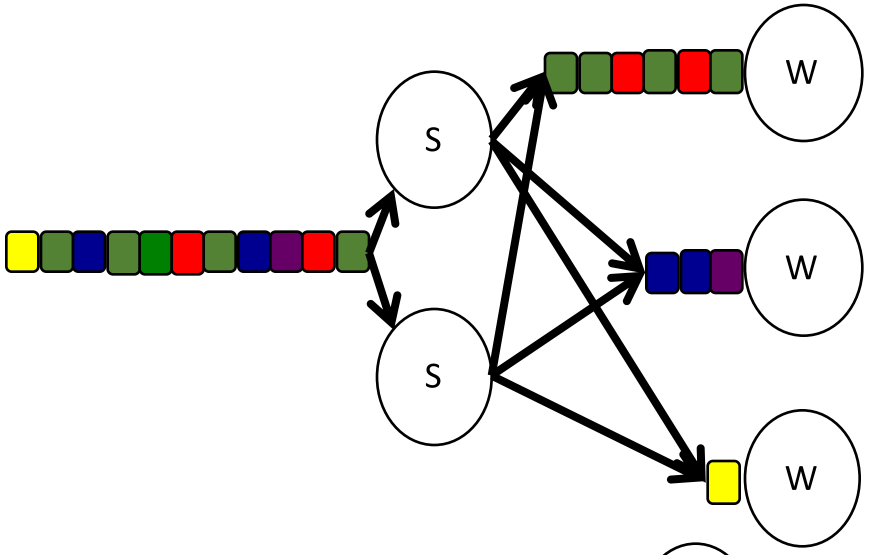
SAMZA

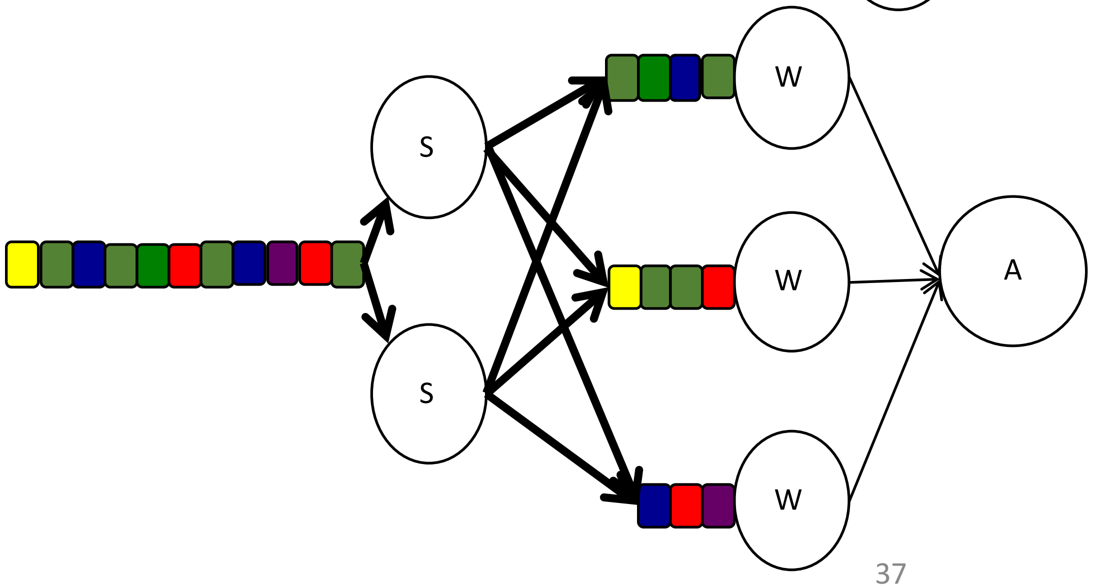
- Streaming Applications -> DAGs
  - Heavily depended on key distribution



Stream Grouping

- Key or Fields Grouping
  - Hash-based assignment
  - Stateful operations, e.g., page rank, degree count
  - Efficient Routing
  - Load Imbalance
- Shuffle Grouping
  - Round-robin assignment
  - Stateless operations, e.g., data logging, OLTP
  - Load Balance
  - Additional Memory
  - Additional Aggregation phase
- All Grouping





Possible solution to imbalance

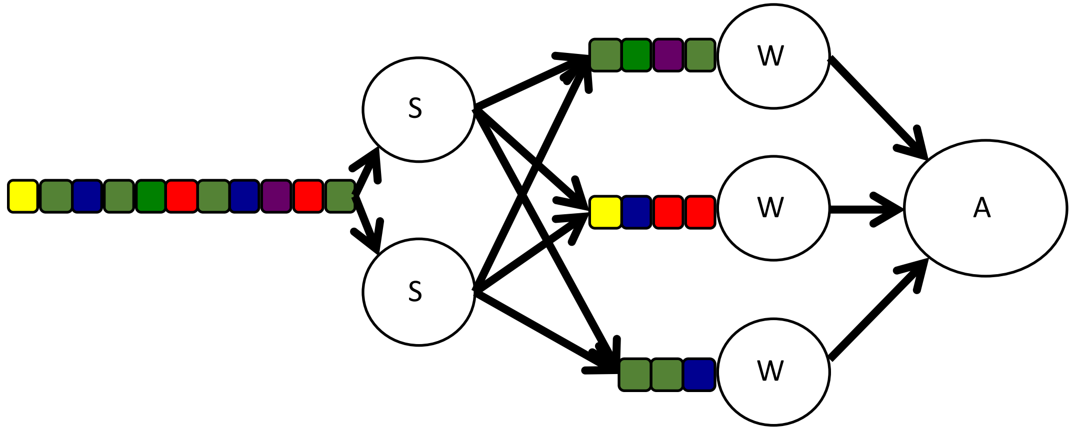
- Dynamic load rebalancing
  - detect load imbalance
  - perform data migration
- Challenges
  - How often to check the load imbalance
  - Migration: not directly supported in DSPEs + requires modifications
  - State management for stateful operation

Partial Key Grouping & Power of 2 choices (POTC)

- Balls-and-bins problem
  - For each ball, pick 2 bins **uniformly at random**
  - Assign the ball to **least loaded** of the 2 bins
  - Bounded imbalance in bins

$$I(t) = \max_i(L_i(t)) - \operatorname{avg}_i(L_i(t)), \text{ for } i \in \mathcal{W}$$

- PKG Algorithm
  - Bins=workers
  - Split each key into 2 workers
  - Estimate load on workers
- Benefits:
  - Decentralized
  - Stateless
  - Handles Skew well



Analysis

- Problem Formulation
  - n workers -> bins
  - keys k_i ε K -> colors
  - m messages -> colored balls
  - d -> number of options for each ball (or message)

- Minimize the difference between maximum and average workload

Analysis

- Key Distribution
  - We pick each key $k_i \in K$ with probability p_i from the distribution D, where $p_1 \geq p_2 \geq p_3$ ….
- Maximum load proportional to most frequent key (with p_1)
- If $p_1 > 2/n$ the expected imbalance will be lower bounded by

$$I(m) = (p_1/2 - 1/n)\, m$$

Analysis

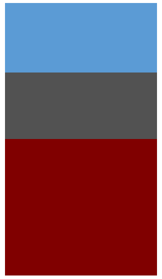
- Assume a key distribution *D* with maximum probability p1 ≤ 2/n. Then the imbalance after m steps of Greedy-d process satisfies, with probability at least 1 – 1/n,

$$I(m) = \begin{cases} O\left(\frac{m}{n} \cdot \frac{\ln n}{\ln \ln n}\right), & \mathit{if}\ d = 1 \\ O\left(\frac{m}{n}\right), & \mathit{if}\ d \geq 2 \end{cases}$$

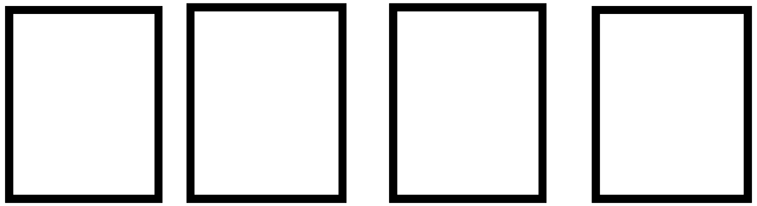
Analysis

- An example with four workers

- In ideal scenario, each worker should handle 25% of the keys

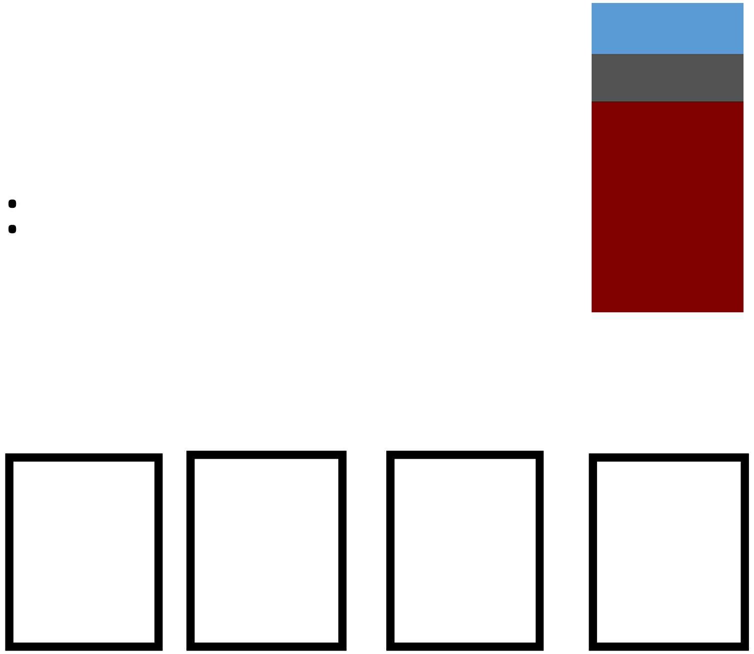


- We need to consider three cases:
  - When $p_1 = 2/4 = 0.5$



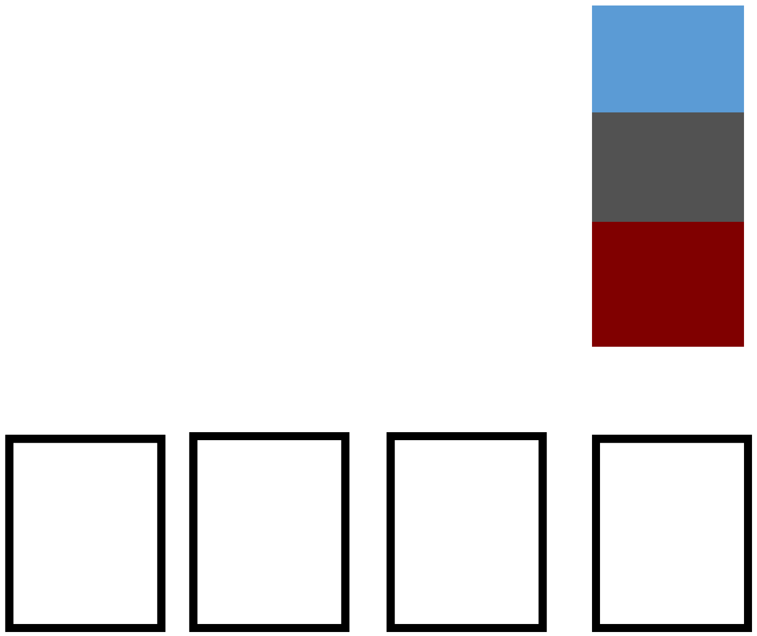
Analysis

- An example with four workers
- In ideal scenario, each worker should handle 25% of the keys
- We need to consider three cases:
  - When $p_1 = 2/4 = 0.5$
  - When $p_1 > 0.5$



Analysis

- An example with four workers
- In ideal scenario, each worker should handle 25% of the keys
- We need to consider three cases:
  - When $p_1 = 2/4 = 0.5$
  - When $p_1 > 0.5$
  - When $p_1 < 0.5$



Applications

- Most algorithms that use Shuffle Grouping can be expressed using Partial Key Grouping to reduce:
  - Memory footprint
  - Aggregation overhead

- Algorithms that use Key Grouping can be rewritten to achieve load balance

Examples

- Naïve Bayes Classifier

- Streaming Parallel Decision Trees

- Heavy Hitters and Space Saving

Naïve Bayes Classifier

- Counts co-occurrences of each feature and class value

- Key Grouping
  - Vertical Parallelism: each feature is tracked by single worker process

- Shuffle Grouping
  - Horizontal Parallelism: each feature is tracked by all worker processes

- Partial Key Grouping
  - Each feature is tracked by exactly two processes

Stream Groupings: A (new) summary

Stream Grouping	Pros	Cons
Key Grouping	- Scalable	- Load Imbalance
Shuffle Grouping	- Load Balance	- Memory Overhead - Aggregation O(W)
Partial Key Grouping	- Scalable - Load Balance - Memory Cost	- Aggregation O(1)

Experimental Questions

- How does **local estimation** compare to a global oracle?
- How **robust** is Partial Key Grouping in skew?
- How does PKG perform on a real deployment on **Apache Storm**?

Metric: Load Imbalance

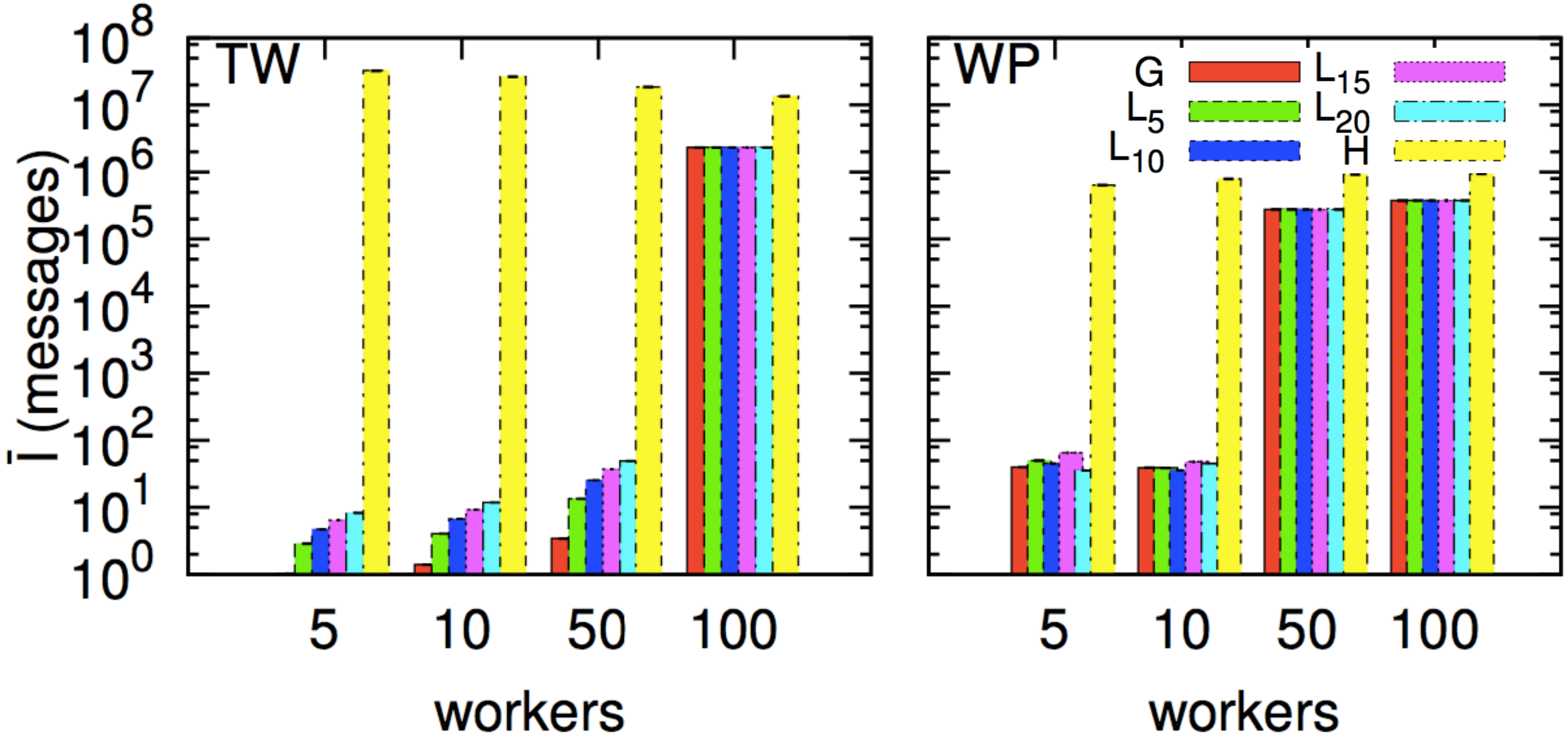
- The difference between the maximum and the average load of the workers at time t

$$I(t) = \max_i(L_i(t)) - \operatorname*{avg}_i(L_i(t)), \text{ for } i \in \mathcal{W}$$

Effect of Key Splitting

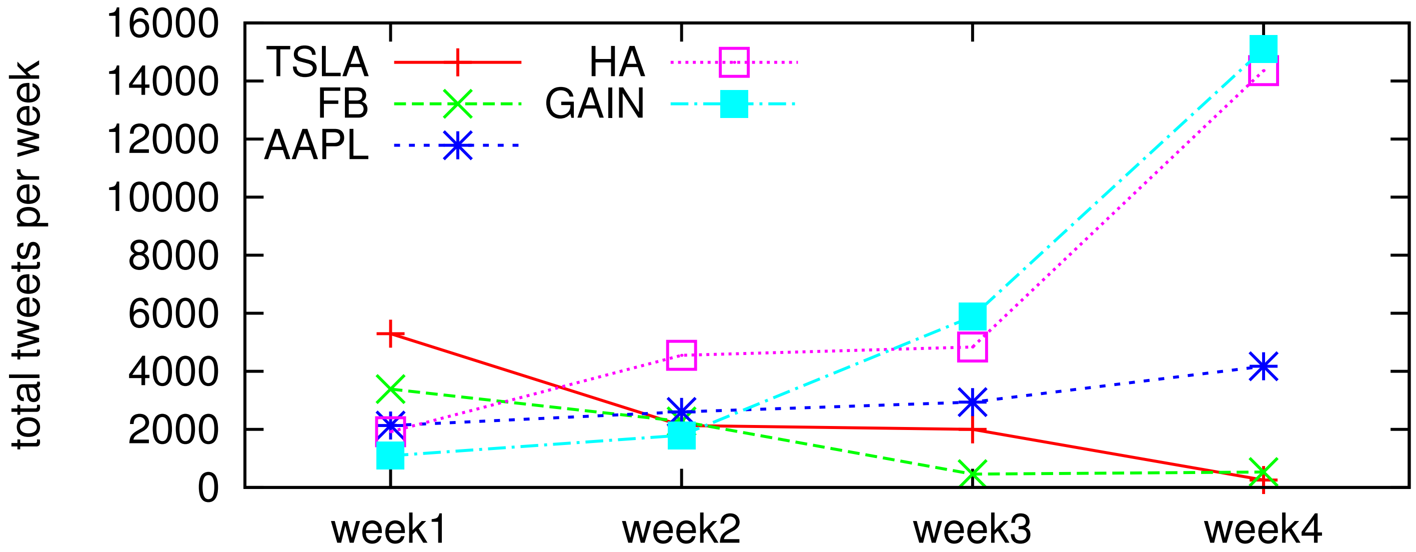
Dataset	Wikipedia (WP)				Twitter (TW)			
Workers	5	10	50	100	5	10	50	100
PKG	0.8	2.9	5.9e5	8.0e5	0.4	1.7	2.74	4.0e6
Off-Greedy	0.8	0.9	1.6e6	1.8e6	0.4	0.7	7.8e6	2.0e7
On-Greedy	7.8	1.4e5	1.6e6	1.8e6	8.4	92.7	1.2e7	2.0e7
PoTC	15.8	1.7e5	1.6e6	1.8e6	2.2e4	5.1e3	1.4e7	2.0e7
Hashing	1.4e6	1.7e6	2.0e6	2.0e6	4.1e7	3.7e7	2.4e7	3.3e7

Local Load Estimation

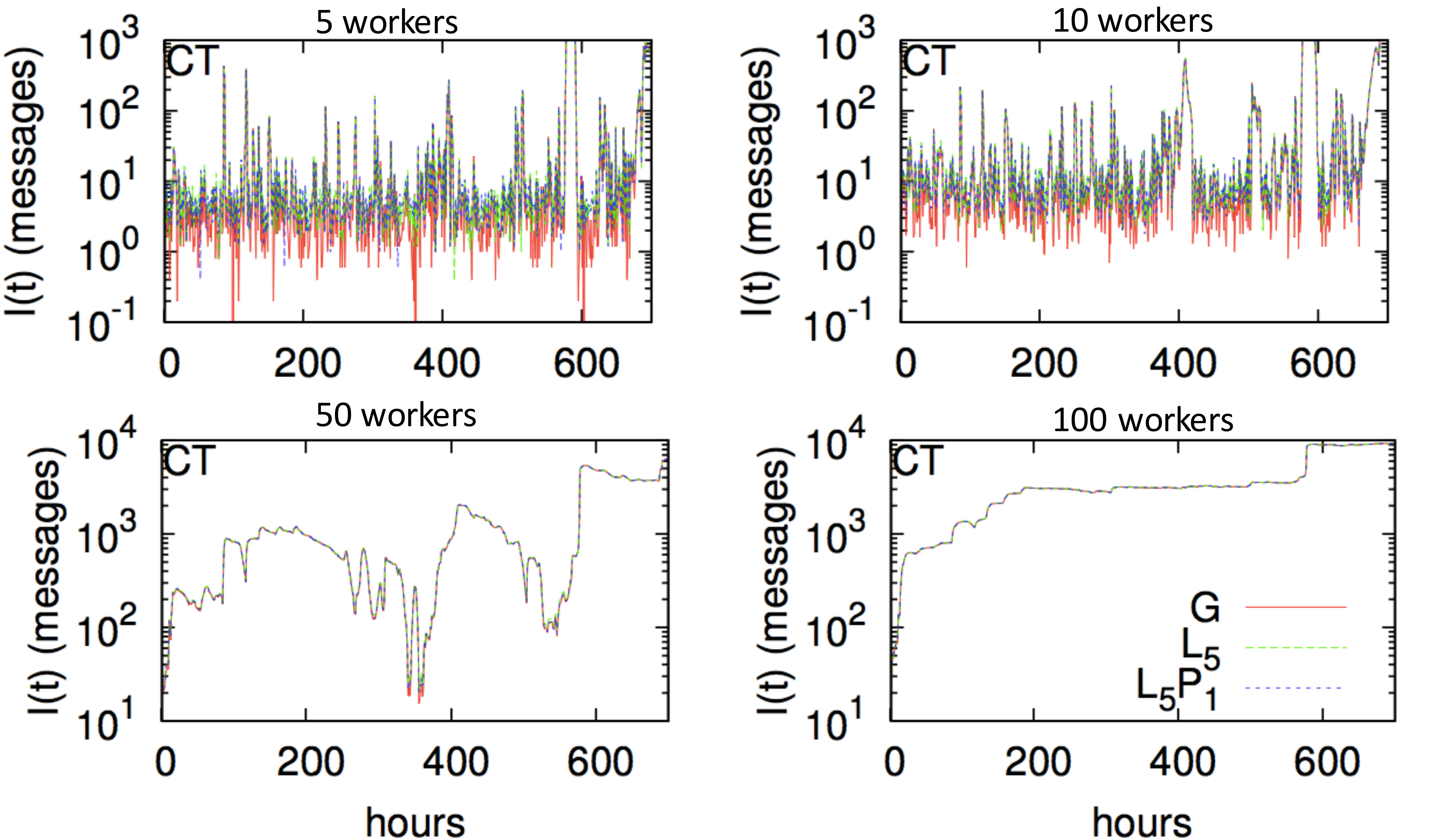


Robustness in drift

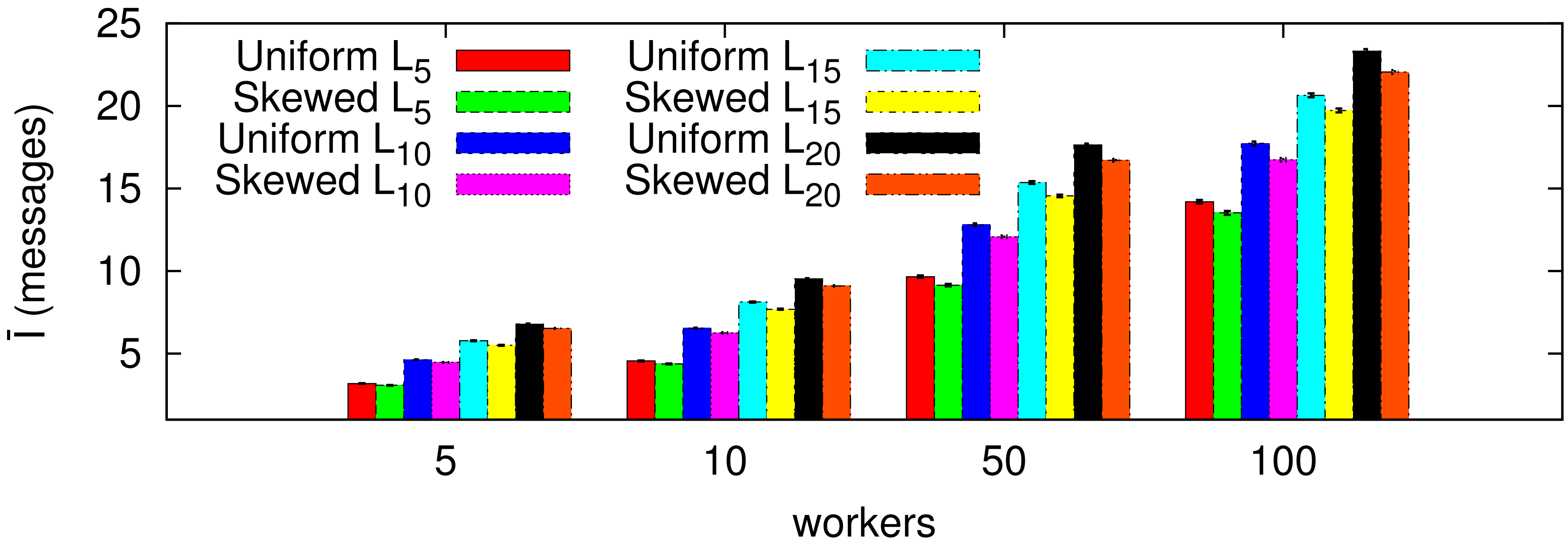
- Changing trends in data: cashtags
  - Used in the stock market to identify a publicly traded company: e.g., $AAPL for Apple
- Skewed load at source: graph metrics for social networks
  - Test different data distribution at the sources



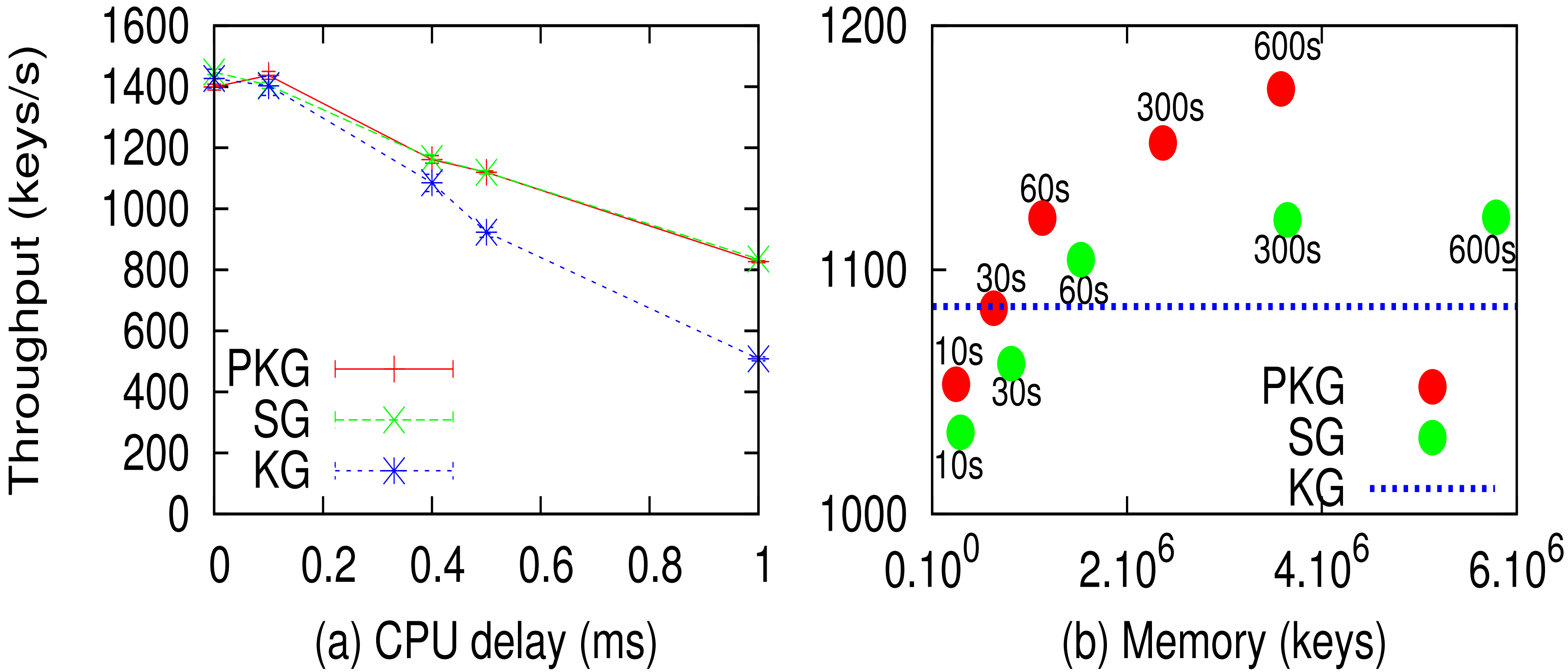
Robustness in drift



Robustness: Uniform vs. Skewed distribution



Real deployment: Apache Storm



What’s next?

- Novel metrics to capture graph dynamics
  - Speed / acceleration of change
  - Change: nodes/edges/weights
- Graph dynamics considered
  - Time granularity
  - Graph entity granularity (node/edge/weight level, community/cluster level, ...)
- Novel systems for graph mining
  - Distributed and parallel stream processing
  - Real-time constraints
- Applications
  - Time-critical constraints
  - Predictions
  - Recommendations

Shortest paths in evolving graphs and imposed system workload imbalance

Nicolas Kourtellis, Ph.D.

Telefonica Research, Barcelona

nicolas.kourtellis@telefonica.com

@kourtellis