Mining Large Dynamic Graphs and Tensors

Kijung Shin

Ph.D. Candidate

Carnegie Mellon University

(kijungs@cs.cmu.edu)

Thesis Committee

- Prof. Christos Faloutsos (Chair)
- Prof. Tom M. Mitchell
- Prof. Leman Akoglu
- Prof. Philip S. Yu




What Do Real Graphs Look Like?

- Part 1 (Chapters 3 - 8)

How to Spot Anomalies?

- Part 2 (Chapters 9 - 13)

How to Model Behavior?

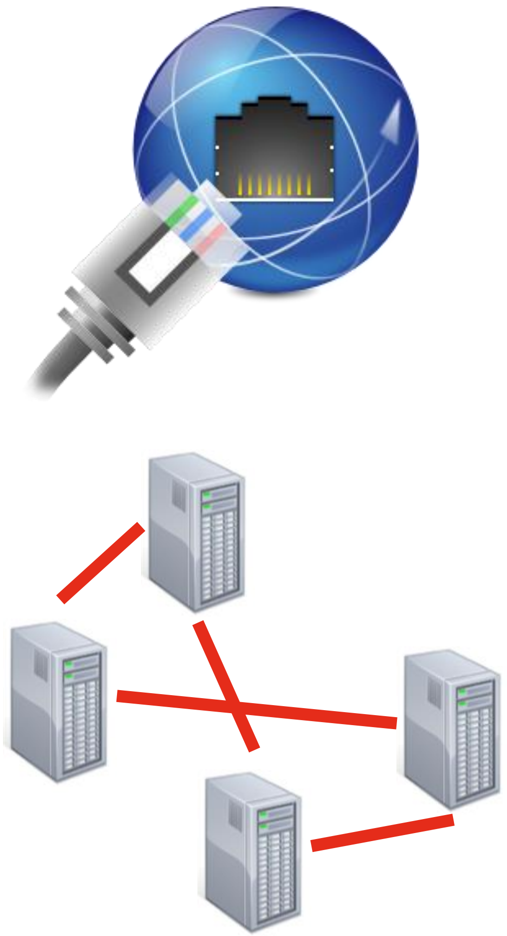
- Part 3 (Chapters 14 - 15)

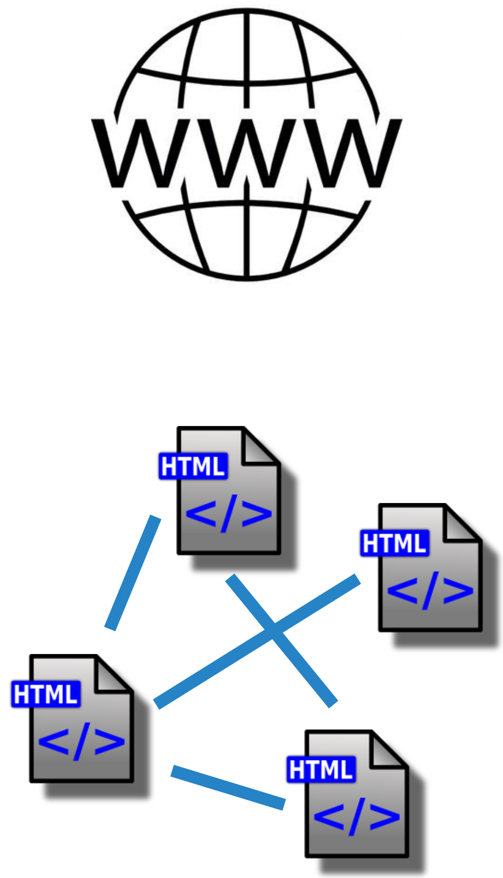
amazon

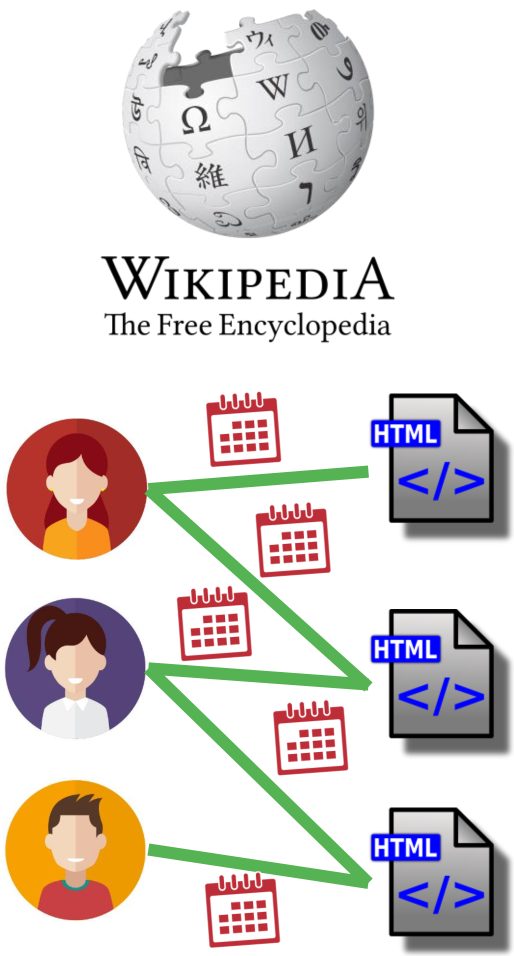
ebay

Alibaba.com

Graphs are Everywhere!







Graphs are Large and Dynamic

- **Large**: many nodes, more edges

40B+ web pages

2B+ active users

500M+ products

5M+ articles

- **Dynamic**: additions/deletions of nodes and edges

Follow

Unfollow

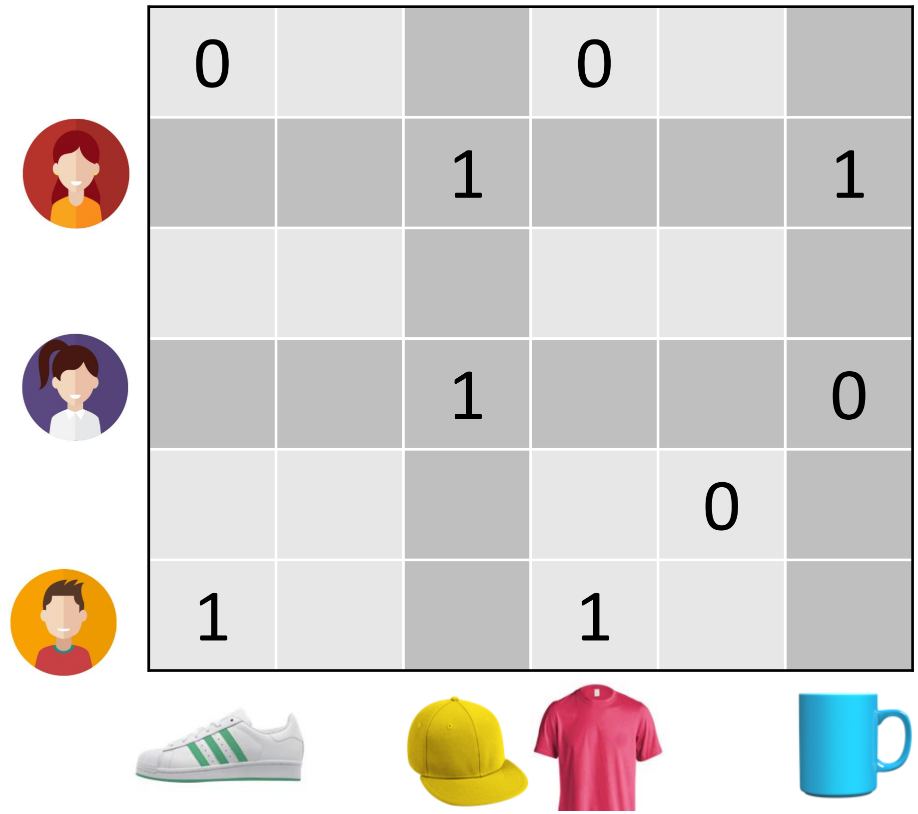
.. and with Rich Side Information

- **Rich**: timestamps, scores, text, etc.

Simple Graphs are Matrices

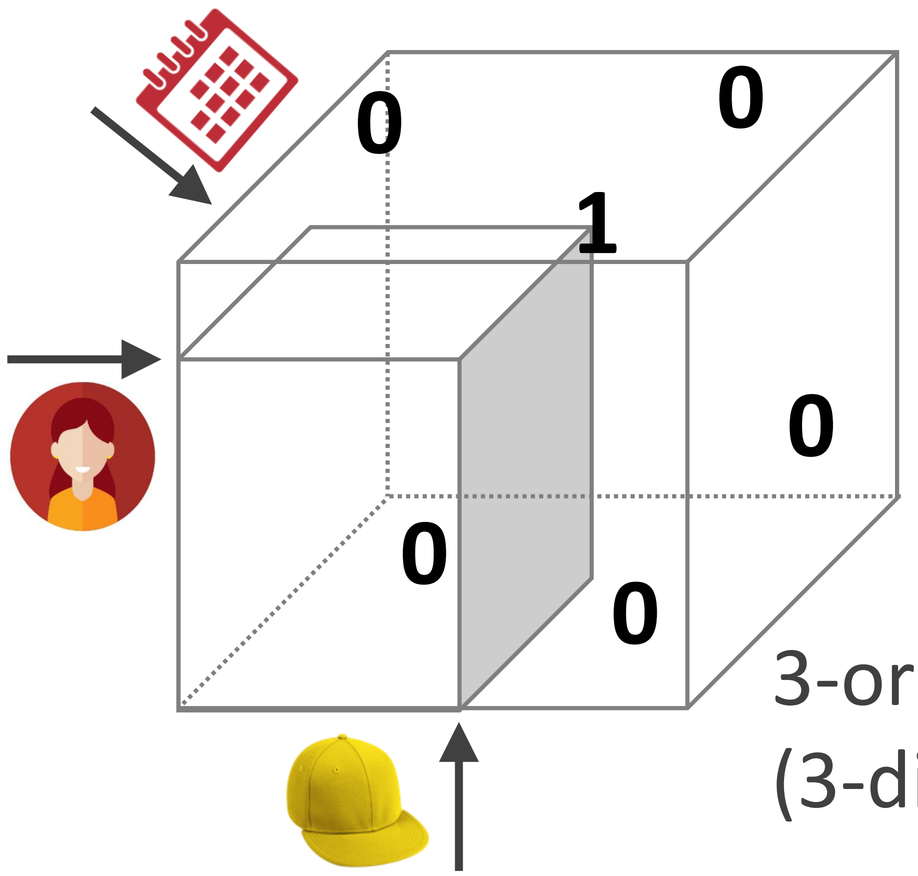
Graph

Adjacency Matrix



Rich Graphs are Tensors

- **Tensors**: multi-dimensional array



+ Stars ⭐⭐⭐
  (4-order tensor)
+ Text "Satisfied ☺"
  (5-order tensor)

3-order tensor
(3-dimensional array)

Thesis Goal and Focus

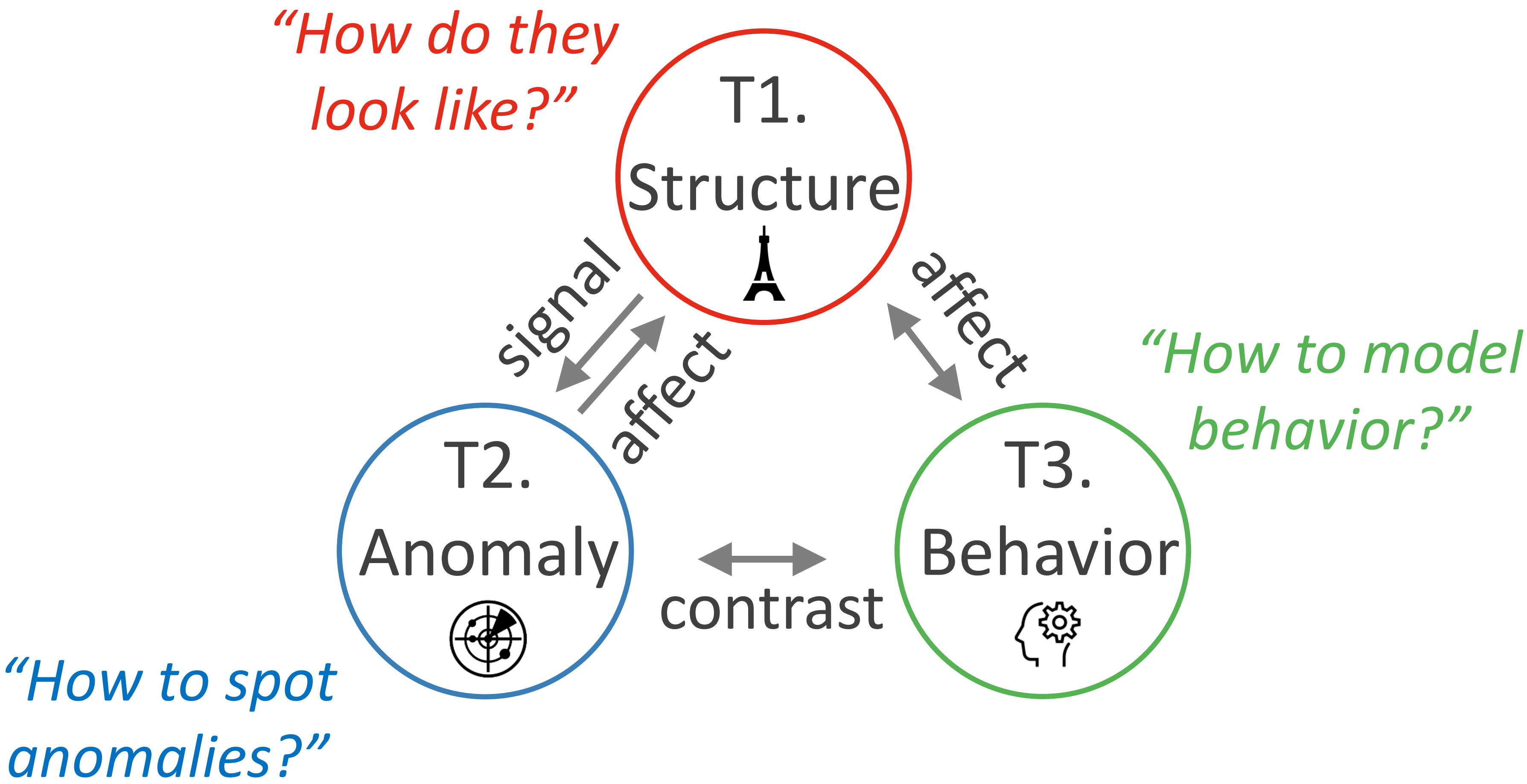
- **Goal:**

> To Fully Understand and Utilize
> Large Dynamic ***Graphs*** and ***Tensors***
> on ***User Behavior***

- **Our Focus**: To Develop ***Scalable Algorithms*** for
  - T1. Structure Analysis (Part 1) 
  - T2. Anomaly Detection (Part 2) 
  - T3. Behavior Modeling (Part 3) 

Tasks and Their Relation

- Given large dynamic ***graphs*** or ***tensors***,



Our Tools for Scalability

- We design **(sub) linear algorithms**

Approx. Sampling Streaming Out-of-core Parallel

- Running on **big data platforms**

- Exploiting **empirical patterns in data**
  - locality, power-laws, etc.

Organization of the Thesis

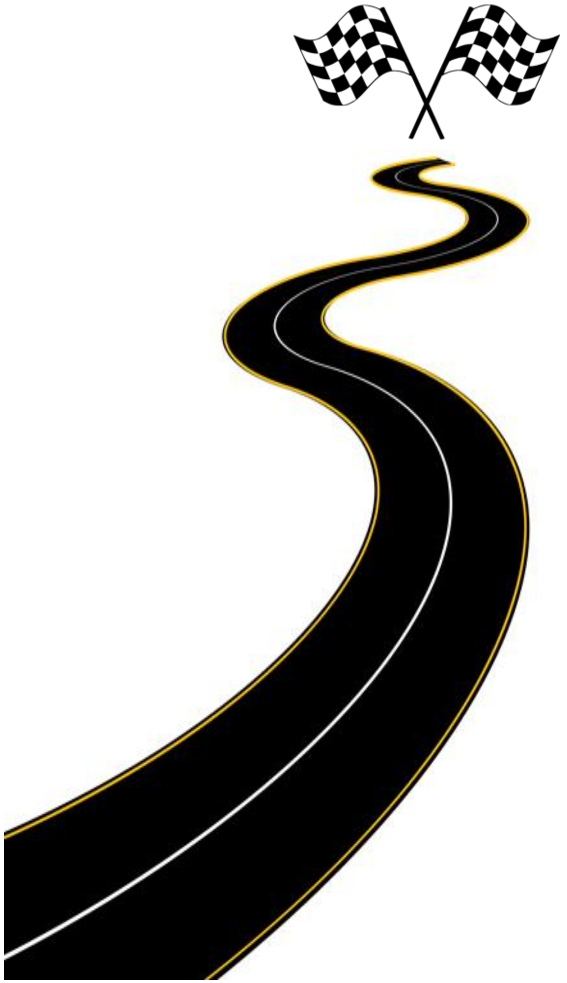
	Part1. Structure Analysis	**Part2.** Anomaly Detection	**Part3.** Behavior Modeling
Graphs	Triangle Count (§§ 3-6) Summarization (§ 7) Chapter	Anomalous Subgraph (§ 9)	Purchase Behavior (§ 14)
Tensors	Summarization (§ 8)	Dense Subtensors (§§ 10-13) Chapters	Progression (§ 15)

Focuses of This Presentation

	Part1. Structure Analysis	**Part2.** Anomaly Detection	**Part3.** Behavior Modeling
Graphs	**Triangle Count (§§ 3-6)** **Summarization (§ 7)**	Anomalous Subgraph (§ 9)	**Purchase Behavior (§ 14)**
Tensors	Summarization (§ 8)	Dense Subtensors (§§ 10-13)	Progression (§ 15)

Roadmap

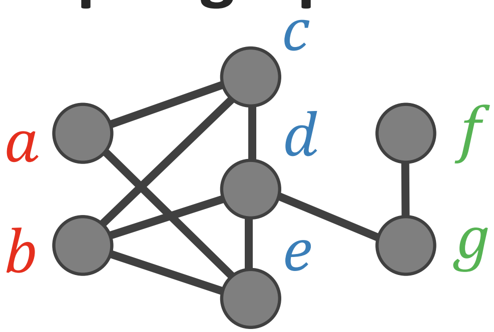
- **T1. Structure Analysis (Part 1) <<**
- T2. Anomaly Detection (Part 2)
- T3. Behavior Modeling (Part 3)
- Future Directions
- Conclusion



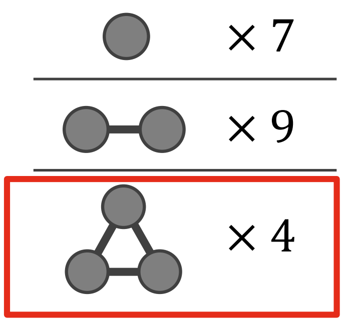
T1. Structure Analysis (Part 1)

"Given a large graph (or tensor), how can we analyze its structure?

Input graph



Basic statistics

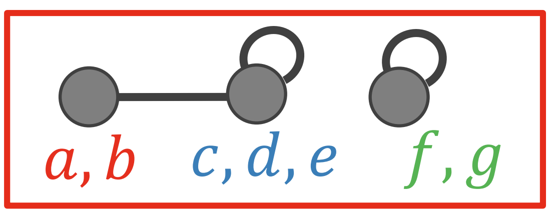


Structure measures

- density
- clustering coefficients
- transitivity ratio
- triangle connectivity

T1-1. Triangle Counting

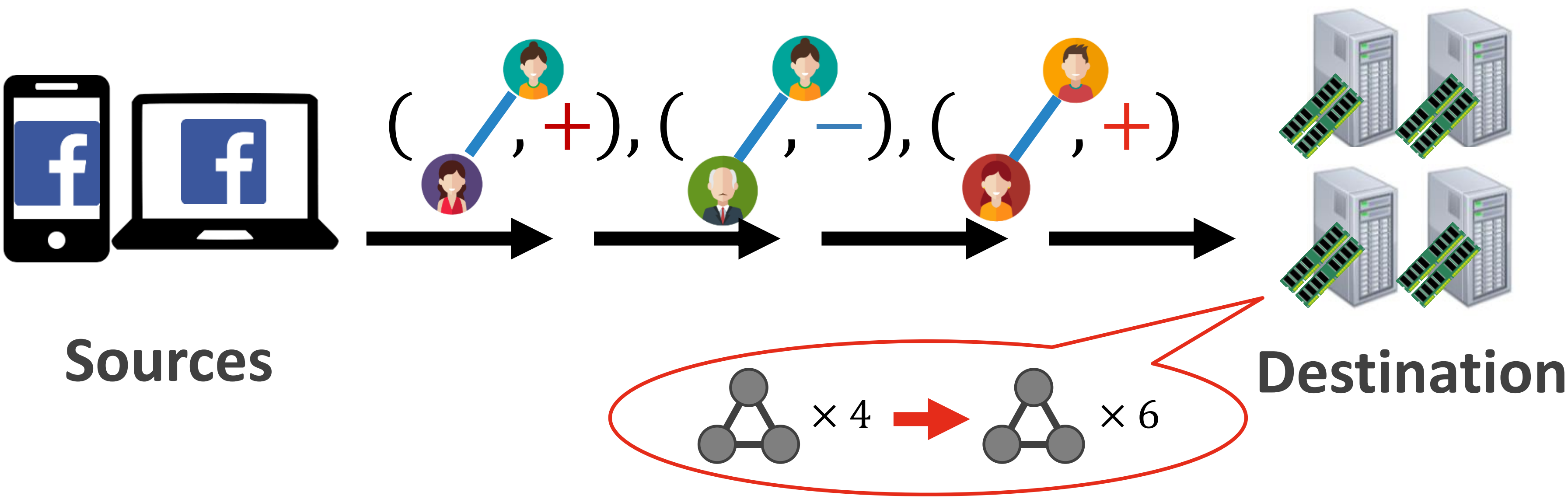
Summary graph



T1-2. Summarization

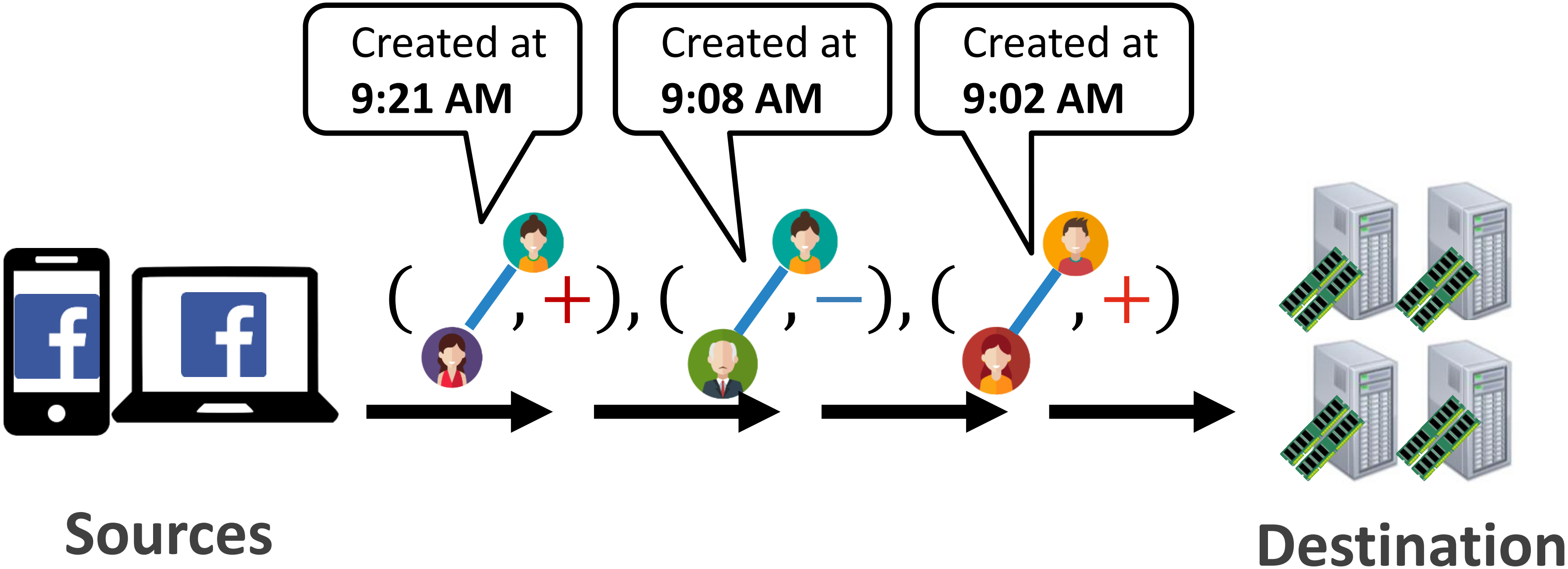
T1-1. Triangle Counting (§§ 4-6)

*"Given a **large dynamic graph**,
how can we **track the count of triangles**
accurately with **sub-linear memory**?"*



T1-1. Triangle Counting (§§ 4-6)

How can we exploit **temporal patterns**? (§4)

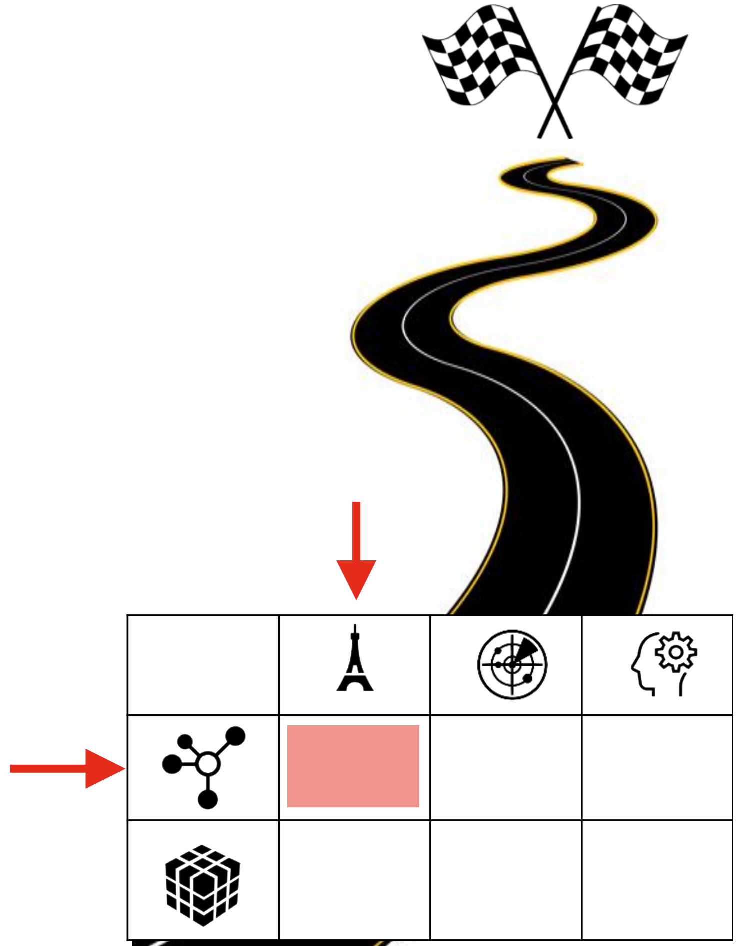


How can we handle **removed edges**? (§6)

How can we make good use of **multiple machines**? (§5)

Roadmap

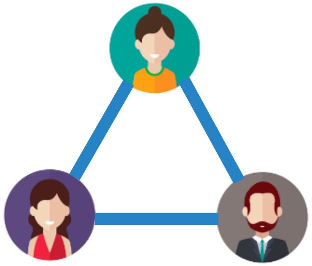
- T1. Structure Analysis (Part 1)
  - T1.1 Triangle Counting
    - **Handling Deletions (§6) <<**
    - …
  - …
- T2. Anomaly Detection (Part 2)
- T3. Behavior Modeling (Part 3)
- Future Directions
- Conclusions

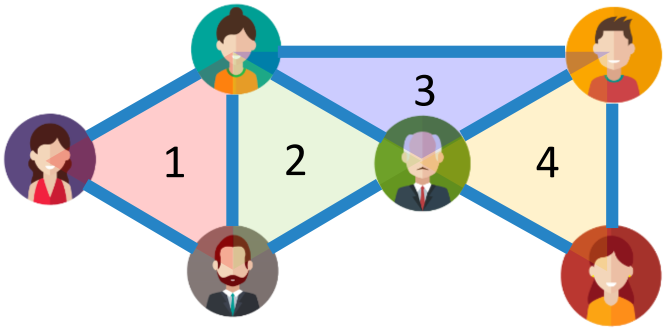


K. Shin, J. Kim B. Hooi, C. Faloutsos, "Think before You Discard: Accurate Triangle Counting in Graph Streams with Deletions", **ECML/PKDD 2018**

Triangles in a Graph

- **A triangle** is 3 nodes connected to each other
- **The count of triangles** is an important primitive
  - *Applications*:
    - community detection, spam detection, link prediction
  - *Structure measures*:
    - transitivity ratio, clustering coefficients, trussness





Remaining Challenge

- Counting triangles in *real-world graphs*
  - **Large**: not fitting in main memory
  - **Fully dynamic**: both *growing* and *shrinking*

Online social networks

Web

Citation networks

Call networks

Previous Work

- **Given**: a *large* and *fully-dynamic* graph
- **To estimate**: the count of triangles accurately

	Large Graph	Fully dynamic Graph		Accurate
		Growing	Shrinking	
MASCOT [LJK18]	✓	✓		✓
Triest-IMPR [DERU17]	✓	✓		✓
WRS [Shi17]	✓	✓		✓
ESD [HS17]		✓	✓	
Triest-FD [DERU17]	✓	✓	✓	
ThinkD (Proposed)	✓	✓	✓	✓

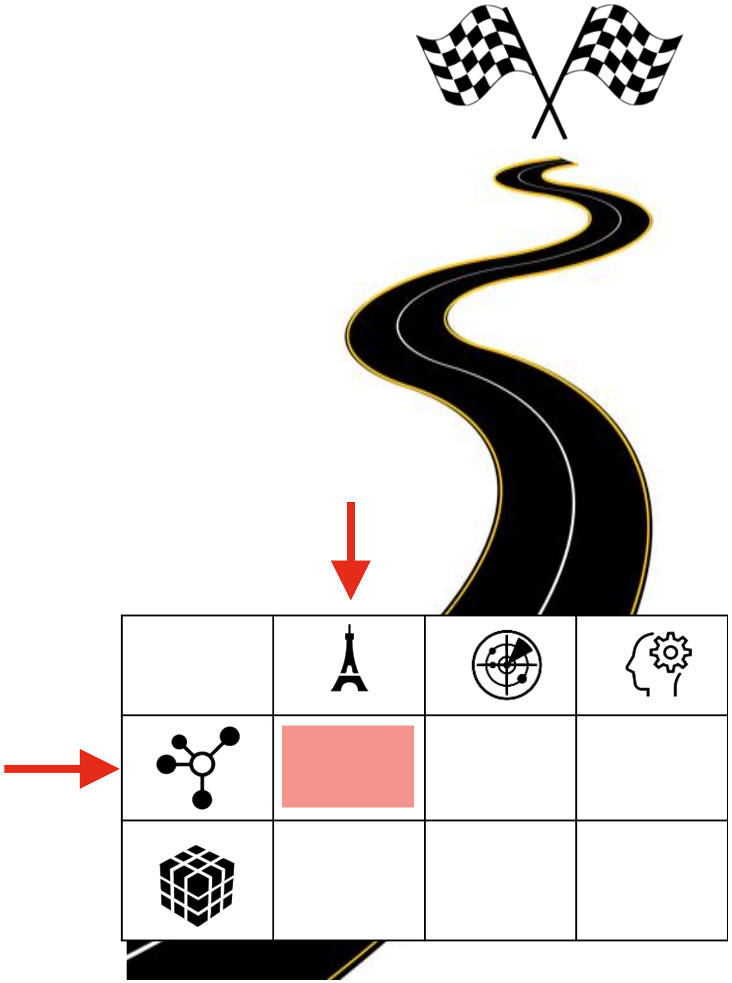
Our Contribution: ThinkD

- We develop **ThinkD** (**Think** before You **D**iscard):

- ☐ **Fast & Accurate:** outperforming competitors
- ☐ **Scalable:** linear data scalability
- ☐ **Theoretically Sound:** unbiased estimates

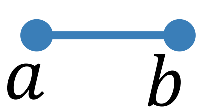
Roadmap

- T1. Structure Analysis (Part 1)
  - T1.1 Triangle Counting
    - Handling Deletions (§6)
      - **Problem Definition <<**
      - Proposed Method: ThinkD
      - Experiments
    - …
- T2. Anomaly Detection (Part 2)
- T3. Behavior Modeling (Part 3)
- …



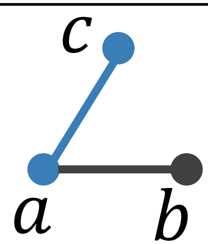
Fully Dynamic Graph Stream

- Our model for a large and fully-dynamic graph
- Discrete time t, starting from 1 and ever increasing
- At each time t, a ***change*** in the input graph arrives
  - ***change***: either an *insertion* or *deletion* of an edge

Time t	1					
Change (given)	$+(a,b)$					
Graph (unmate -rialized)						

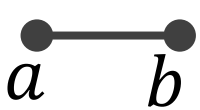
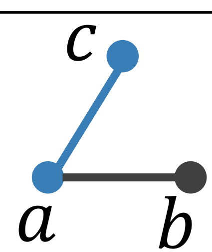
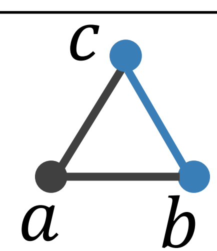
Fully Dynamic Graph Stream

- Our model for a large and fully-dynamic graph
- Discrete time t, starting from 1 and ever increasing
- At each time t, a *change* in the input graph arrives
  - *change*: either an *insertion* or *deletion* of an edge

Time t	1	2				
Change (given)	$+(a,b)$	$+(a,c)$				
Graph (unmate-rialized)						

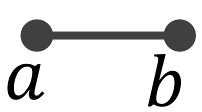
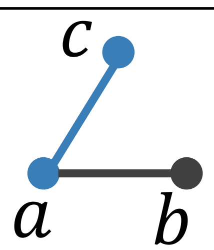
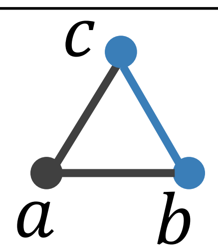
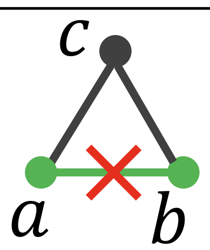
Fully Dynamic Graph Stream

- Our model for a large and fully-dynamic graph
- Discrete time t, starting from 1 and ever increasing
- At each time t, a *change* in the input graph arrives
  - *change*: either an *insertion* or *deletion* of an edge

Time t	1	2	3			
Change (given)	$+(a, b)$	$+(a, c)$	$+(b, c)$			
Graph (unmaterialized)						

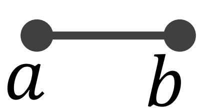
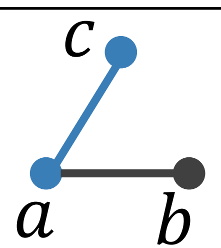
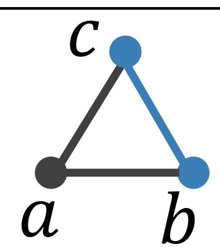
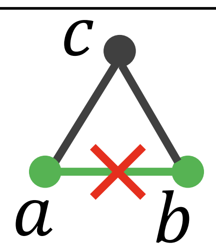
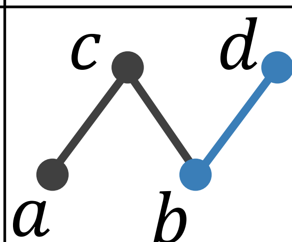
Fully Dynamic Graph Stream

- Our model for a large and fully-dynamic graph
- Discrete time t, starting from 1 and ever increasing
- At each time t, a *change* in the input graph arrives
  - *change*: either an *insertion* or *deletion* of an edge

Time t	1	2	3	4		
Change (given)	$+(a,b)$	$+(a,c)$	$+(b,c)$	$-(a,b)$		
Graph (unmate-rialized)						

Fully Dynamic Graph Stream

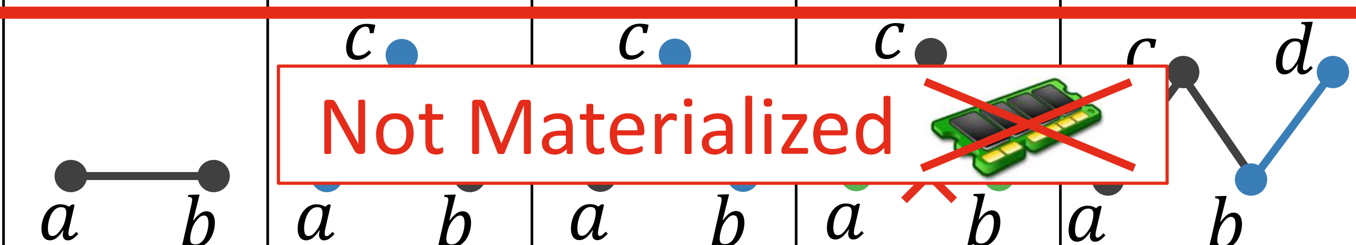
- Our model for a large and fully-dynamic graph
- Discrete time t, starting from 1 and ever increasing
- At each time t, a *change* in the input graph arrives
  - *change*: either an *insertion* or *deletion* of an edge

Time t	1	2	3	4	5	…
Change (given)	$+(a, b)$	$+(a, c)$	$+(b, c)$	$-(a, b)$	$+(b, d)$	…
Graph (unmate -rialized)						…

Fully Dynamic Graph Stream

- Our model for a large and fully-dynamic graph
- Discrete time t, starting from 1 and ever increasing
- At each time t, a *change* in the input graph arrives
  - *change*: either an *insertion* or *deletion* of an edge

Time t	1	2	3	4	5	…
Change (given)	$+(a,b)$	$+(a,c)$	$+(b,c)$	$-(a,b)$	$+(b,d)$	…
Graph (unmaterialized)						…



Problem Definition

- **Given**:
  - a fully-dynamic graph stream (possibly infinite)
  - memory space (finite)
- **Estimate:** the count of triangles
- **To Minimize:** estimation error

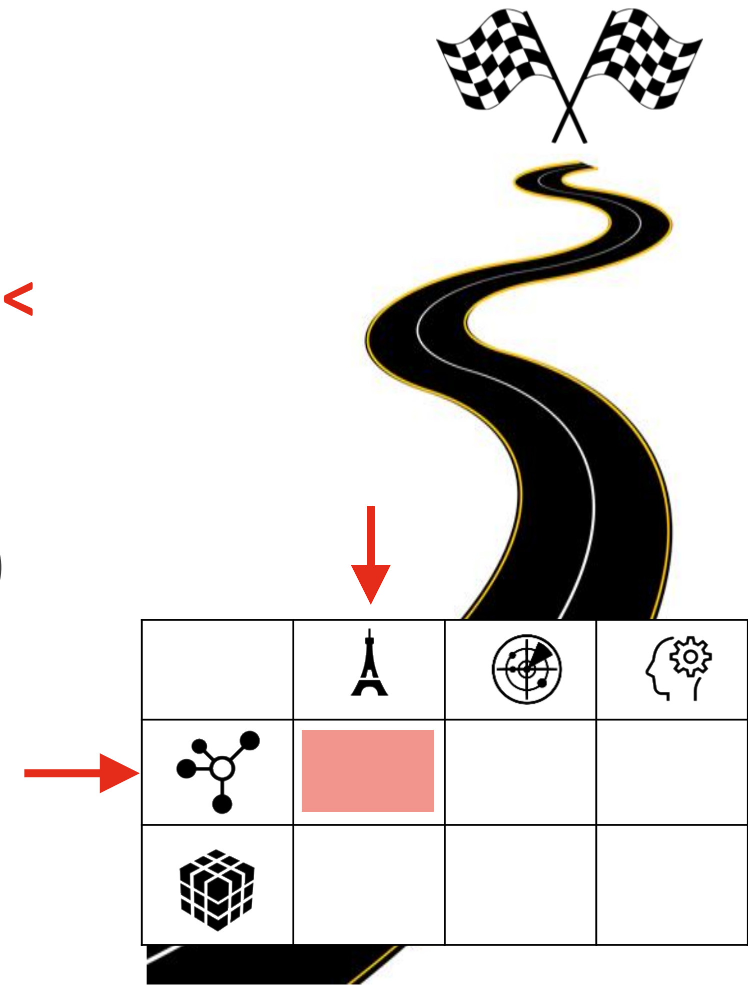
Time t	1	2	3	4	5	…
Changes	$+(a,b)$	$+(a,c)$	$+(b,c)$	$-(a,b)$	$+(b,d)$	…
# Triangles						…

Given (input)

Estimate (output)

Roadmap

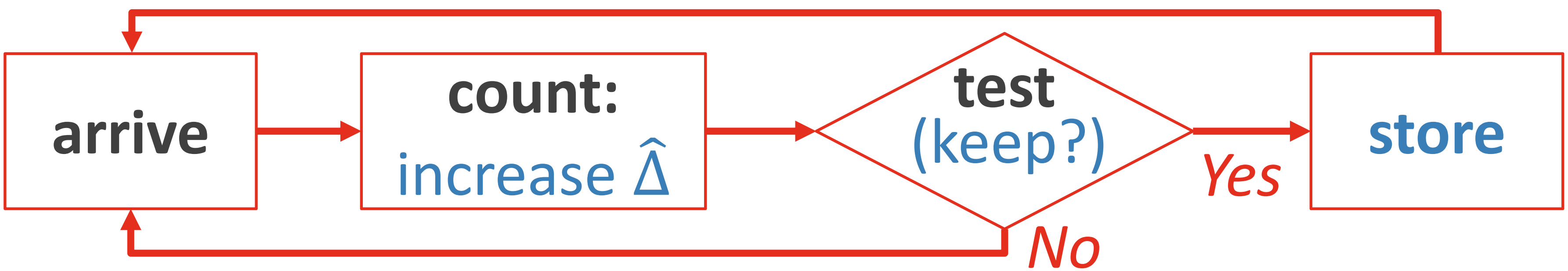
- T1. Structure Analysis (Part 1)
  - T1.1 Triangle Counting
    - Handling Deletions (§6)
      - Problem Definition
      - **Proposed Method: ThinkD <<**
      - Experiments
    - …
- T2. Anomaly Detection (Part 2)
- T3. Behavior Modeling (Part 3)
- Future Directions
- Conclusions



Overview of ThinkD

- Maintains and updates $\hat{\Delta}$
  - Number of (non-deleted) triangles that it has observed

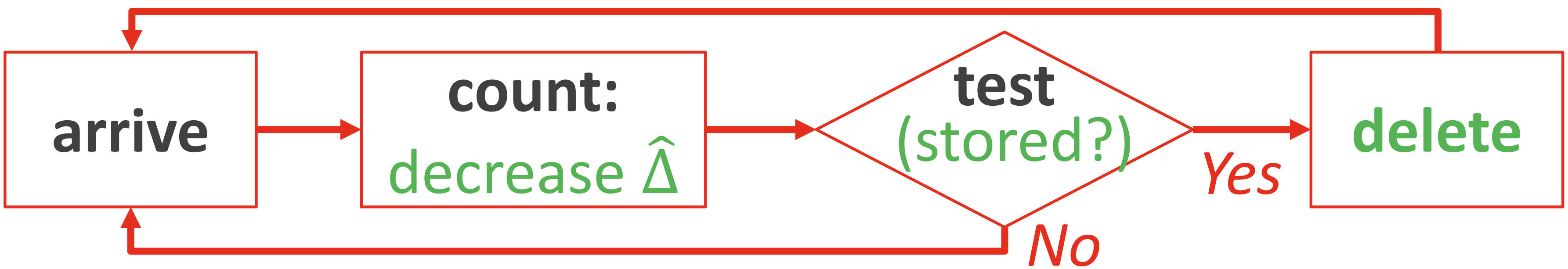
- How it processes an **insertion**:



- **arrive**: an insertion of an edge arrives
- **count:** count new triangles and increase $\hat{\Delta}$
- **test**: toss a coin
- **store**: store the edge in memory

Overview of ThinkD (cont.)

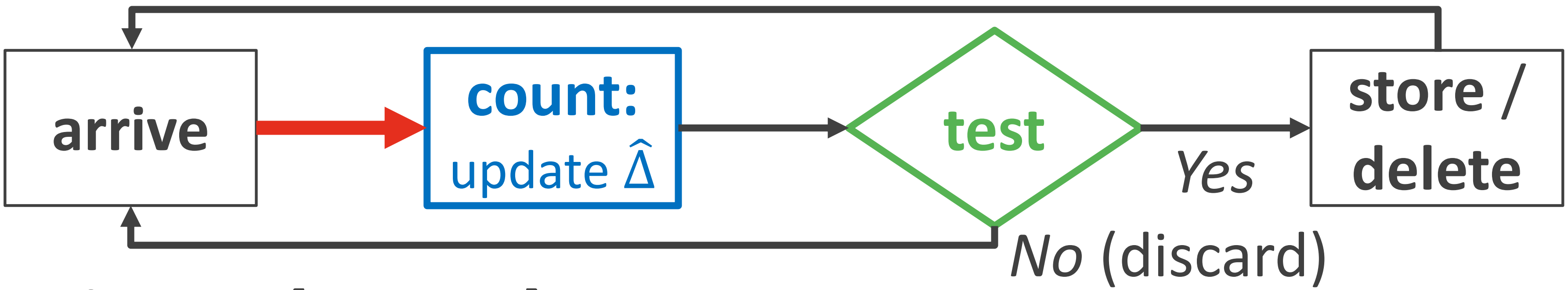
- Maintains and updates $\hat{\Delta}$
  - Number of (non-deleted) triangles that it has observed
- How it processes an **deletion**:



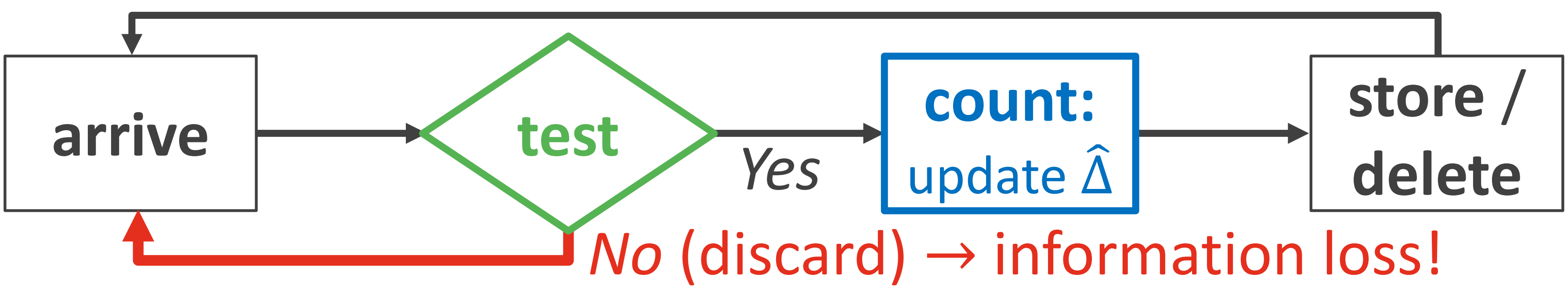
- **arrive**: a deletion of an edge arrives
- **count:** count deleted triangles and decrease $\hat{\Delta}$
- **test**: test whether the edge is stored in memory
- **delete**: delete the edge in memory

Why is ThinkD Accurate?

- **ThinkD (Think** before You **Discard):**
  - *every* arrived change is used to update $\hat{\Delta}$



- **Triest-FD** [DERU17]:
  - *some* changes are discarded without being used to update $\hat{\Delta}$



Two Versions of ThinkD

Q1: How to ***test*** *in the test step*
Q2: How to ***estimate*** *the count of all triangles from* $\hat{\Delta}$

- **ThinkD-FAST**: simple and fast
  - independent Bernoulli trials with probability p

- **ThinkD-ACC**: accurate and parameter-free
  - random pairing [GLH08]

Details

Unbiasedness of ThinkD-FAST

- $\frac{\hat{\Delta}}{p^2}$: estimated count of ***all triangles***
- Δ: true count of ***all triangles***

[Theorem 1] ***At any time t,***

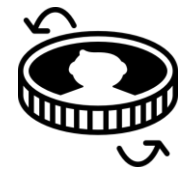
$$\mathbb{E}\left[\boxed{\frac{\hat{\Delta}}{p^2}}\right] = \boldsymbol{\Delta}$$

Unbiased estimate of ***Δ***

- Proof and a variance of $\hat{\Delta}/\mathrm{p}^2$: see the thesis

ThinkD-ACC: More Accurate

- **Disadvantage of ThinkD-FAST:**
  - setting the parameter p is not trivial
    - small p → underutilize memory
      → *inaccurate* estimation
    - large p → *out-of-memory* error



- **ThinkD-ACC** uses Random Pairing [RLH08]
  - always utilizes memory as fully as possible
  - gives more accurate estimation

Details

Scalability of ThinkD

- Let k be the size of memory
- For processing t changes in the input stream,

[Theorem 2] The time complexity of **ThinkD-ACC** is

$$O(k \cdot t)$$ linear in data size

[Theorem 3] If $p = O\left(\frac{k}{t}\right)$,

the time complexity **ThinkD-FAST** is

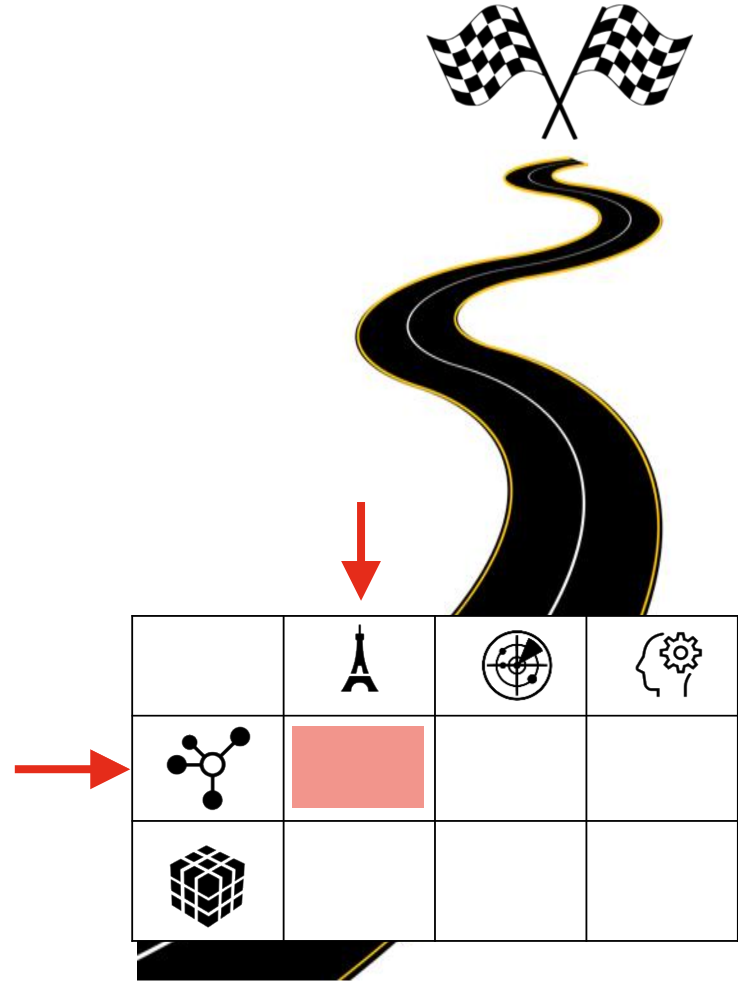
$$O(k \cdot t)$$

Advantages of ThinkD

- ☐ **Fast & Accurate:** outperforming competitors
- ☑ **Scalable:** linear data scalability (Theorems 2 & 3)
- ☑ **Theoretically Sound:** unbiased estimates (Theorem 1)

Roadmap

- T1. Structure Analysis (Part 1)
  - T1.1 Triangle Counting
    - Handling Deletions (§6)
      - Problem Definition
      - Proposed Method: ThinkD
      - **Experiments <<**
    - …
- T2. Anomaly Detection (Part 2)
- T3. Behavior Modeling (Part 3)
- …



Experimental Settings

- **Competitors**: *Triest-FD* [DERU17] & *ESD* [HS17]
  - triangle counting in fully-dynamic graph streams
- **Implementations**: Java 8
- **Datasets:**
  - insertions (edges in graphs) + deletions (random 20%)

Synthetic (**100B** edges)

Social Networks (**1.8B+** edges, …)

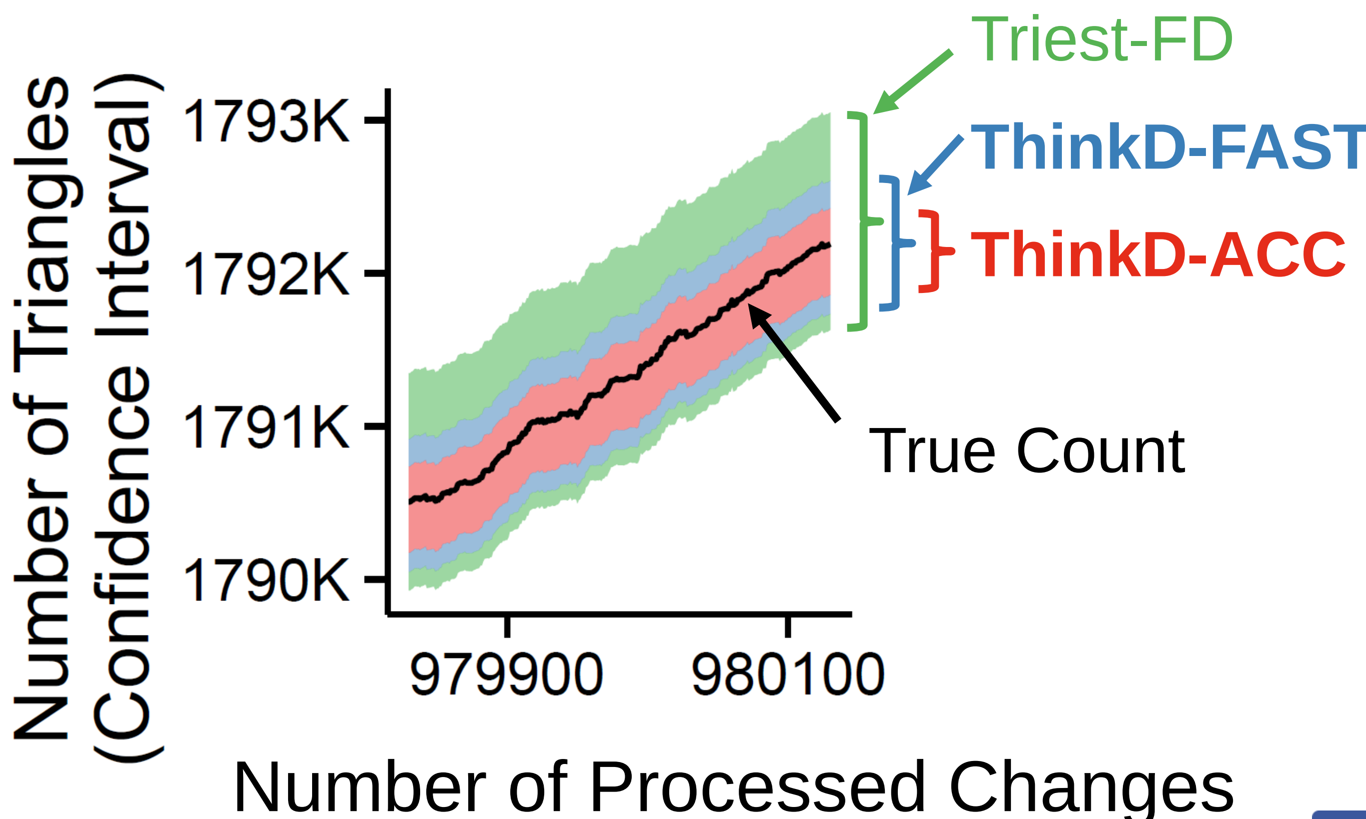
Citation (16M+)

Web (6M+)

Trust (0.7M+)

EXP1. Variance Analysis

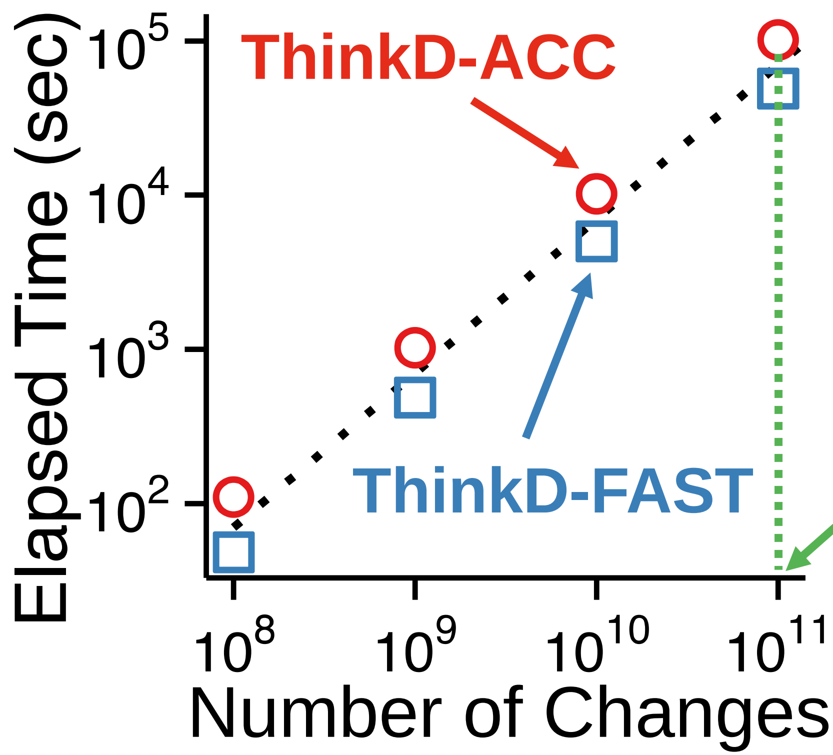
ThinkD is accurate with small variance



- dataset:

EXP2. Scalability [THM 2 & 3]

ThinkD is scalable

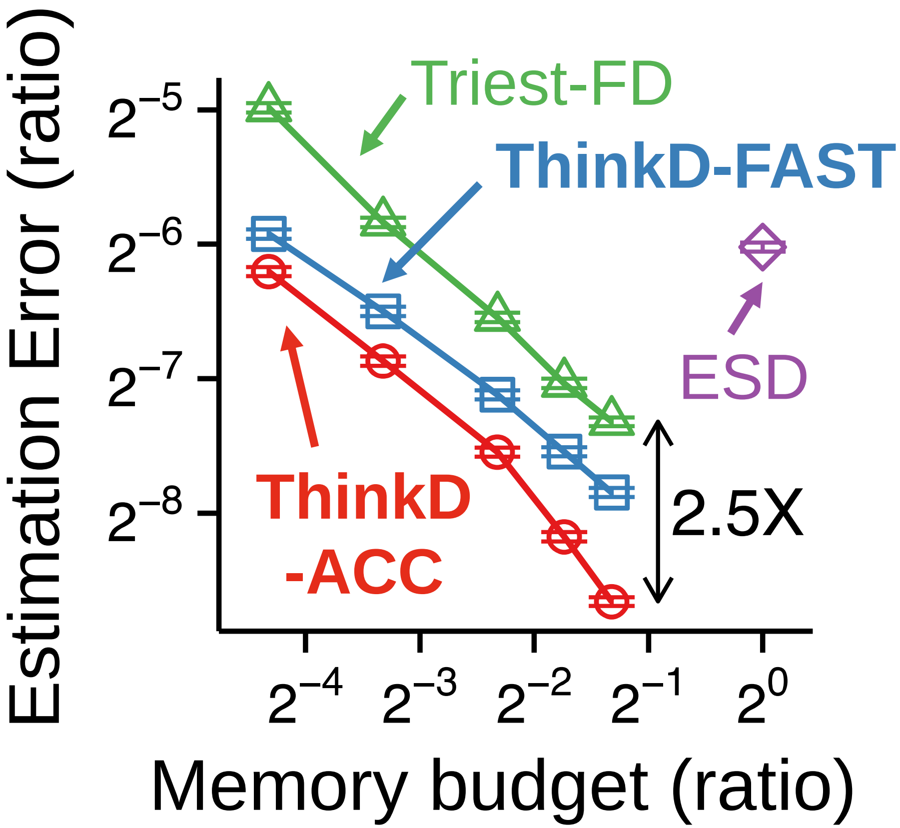


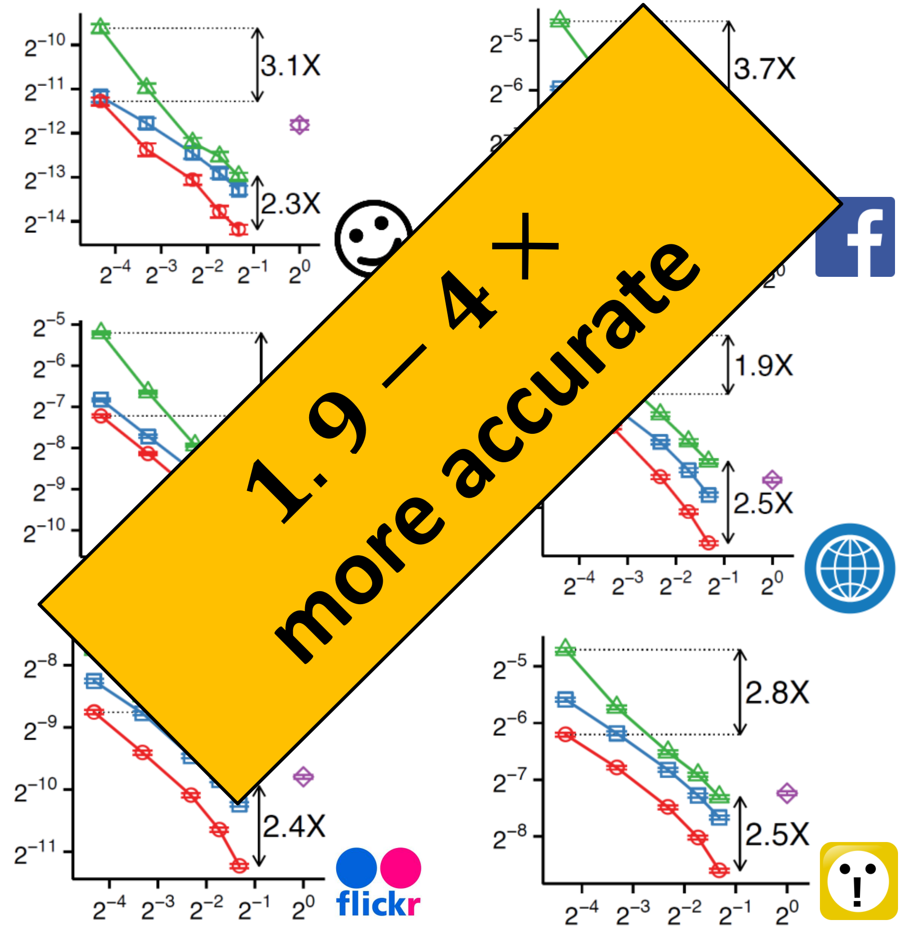
100 billion changes

- dataset: 

EXP3. Space & Accuracy

ThinkD outperforms its best competitors

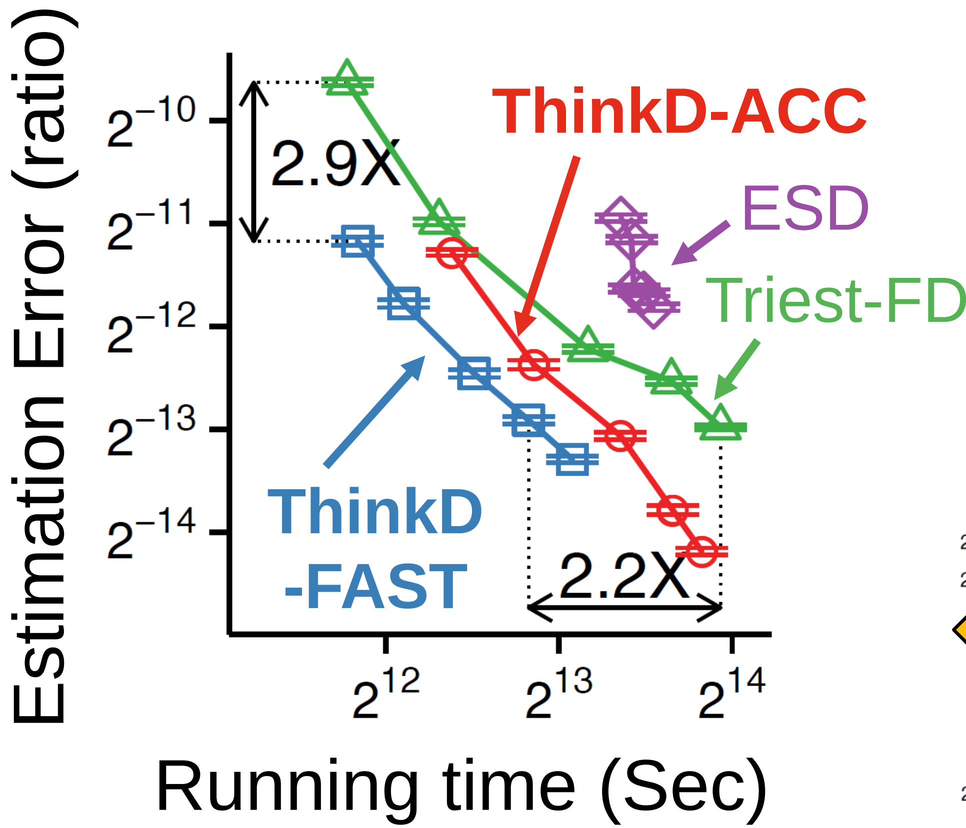


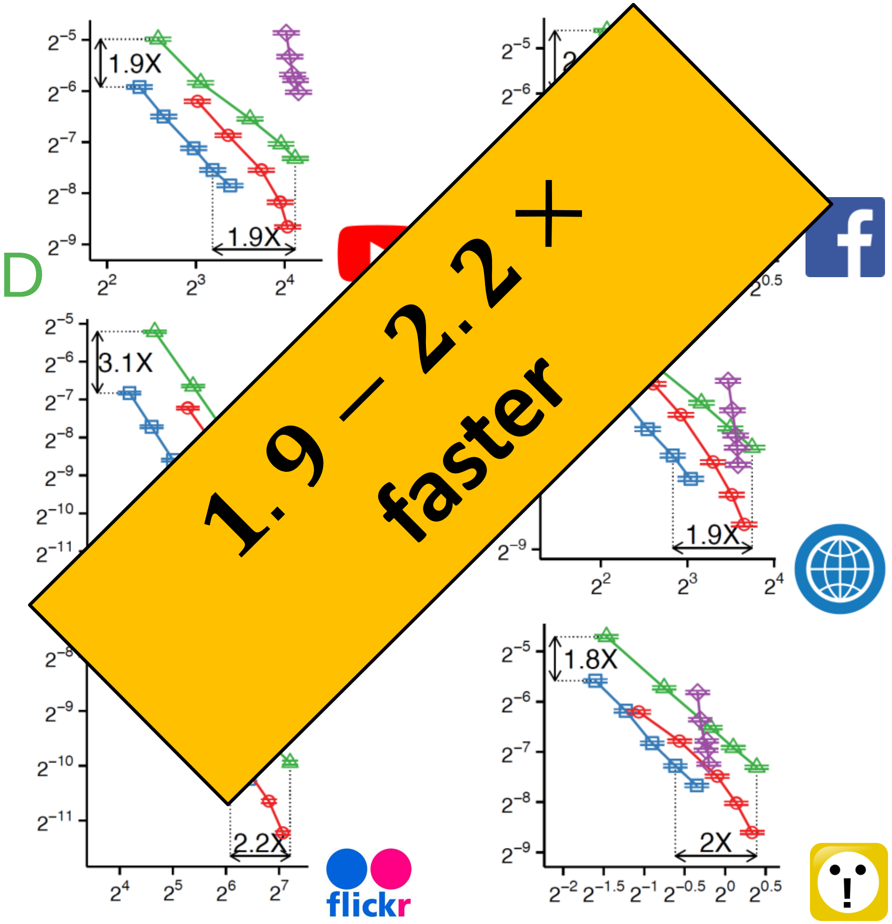


- dataset:

EXP4. Speed & Accuracy

ThinkD outperforms its best competitors





- dataset:

Advantages of ThinkD

- ☑ **Fast & Accurate:** outperforming competitors
- ☑ **Scalable:** linear data scalability
- ☑ **Theoretically Sound:** unbiased estimates

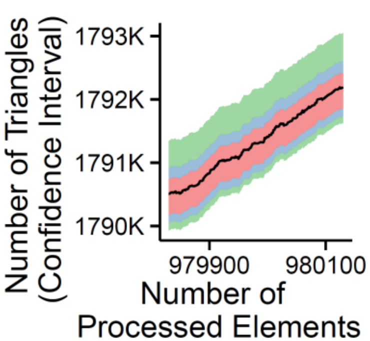
Summary of §6

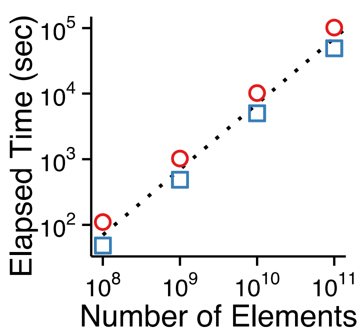
- We propose **ThinkD** (**Think** Before you **D**iscard)
  - for accurate ***triangle counting***
  - in ***large*** and ***fully-dynamic*** graphs

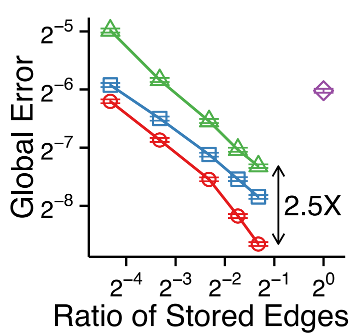
☑ **Fast & Accurate:** outperforming competitors

☑ **Scalable:** linear data scalability

☑ **Theoretically Sound:** unbiased estimates







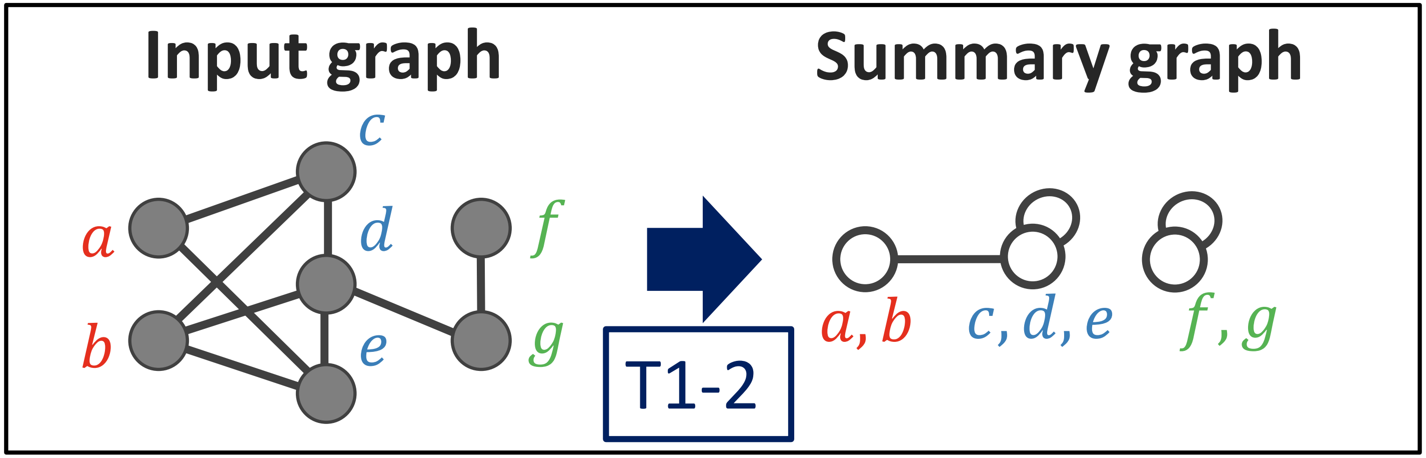
Download ThinkD

Organization of the Thesis (Recall)

	Part1. Structure Analysis	Part2. Anomaly Detection	Part3. Behavior Modeling
Graphs	Triangle Count (§§ 3-6) Summarization (§ 7)	Anomalous Subgraph (§ 9)	Purchase Behavior (§ 14)
Tensors	Summarization (§ 8)	Dense Subtensors (§§ 10-13)	Progression (§ 15)

T1.2 Summarization

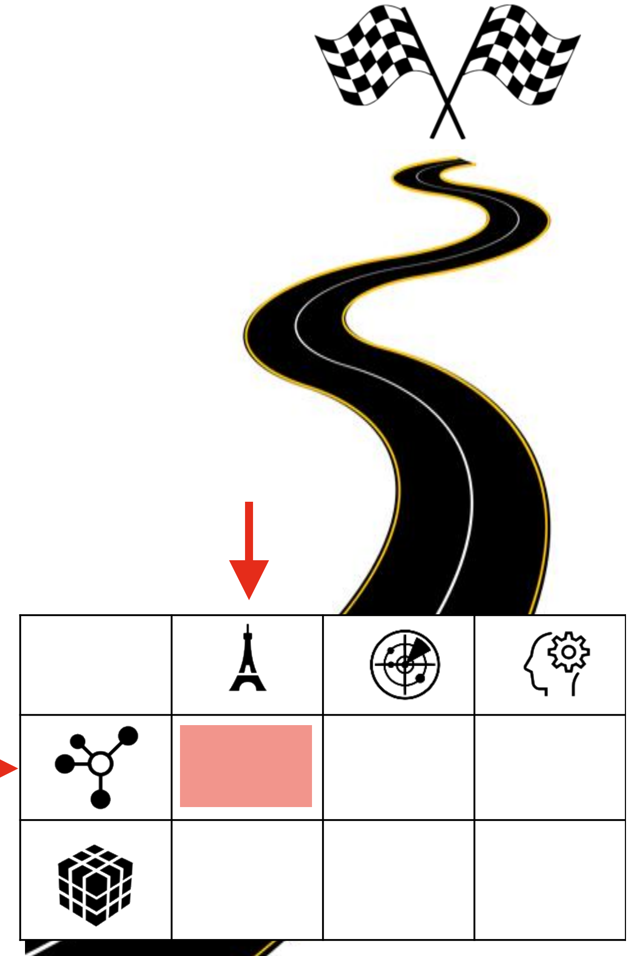
"Given a web-scale graph or tensor,
how can we ***succinctly represent*** *it?"*



- §7: Summarizing **Graphs**
- §8: Summarizing **Tensors** (via Tucker Decomposition)
  - External-memory algorithm with *1,000× improved scalability*

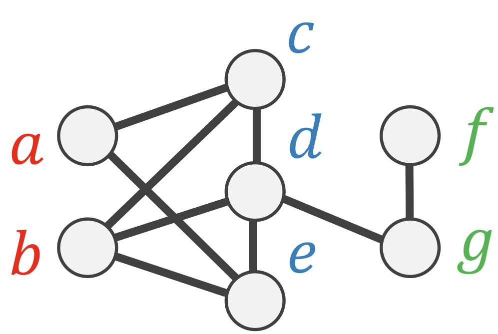
Roadmap

- T1. Structure Analysis (Part 1)
  - …
  - T1-2. Summarization (§§ 7-8)
    - Summarizing Graphs (§ 7)
      - **Problem Definition <<**
      - Proposed Method: SWeG
      - Experiments
    - …
- T2. Anomaly Detection (Part 2)
- T3. Behavior Modeling (Part 3)
- …

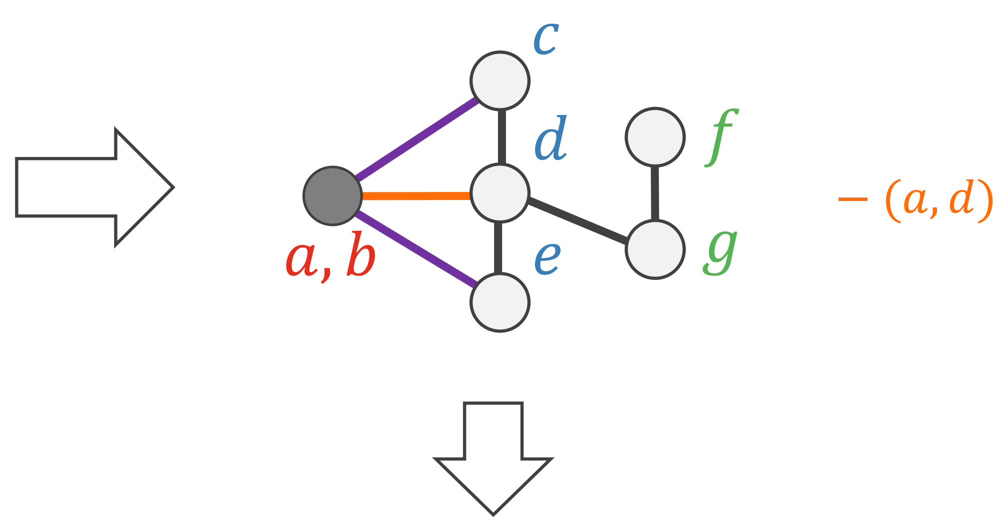


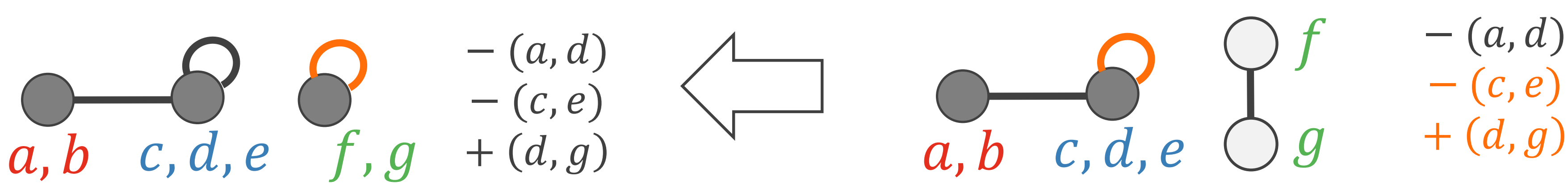
K. Shin, A. Ghoting, M. Kim, and H. Raghavan, “SWeG: Lossless and Lossy Summarization of Web-Scale Graphs”, **WWW 2019**

Graph Summarization: Example



Input Graph (w/ 9 edges)

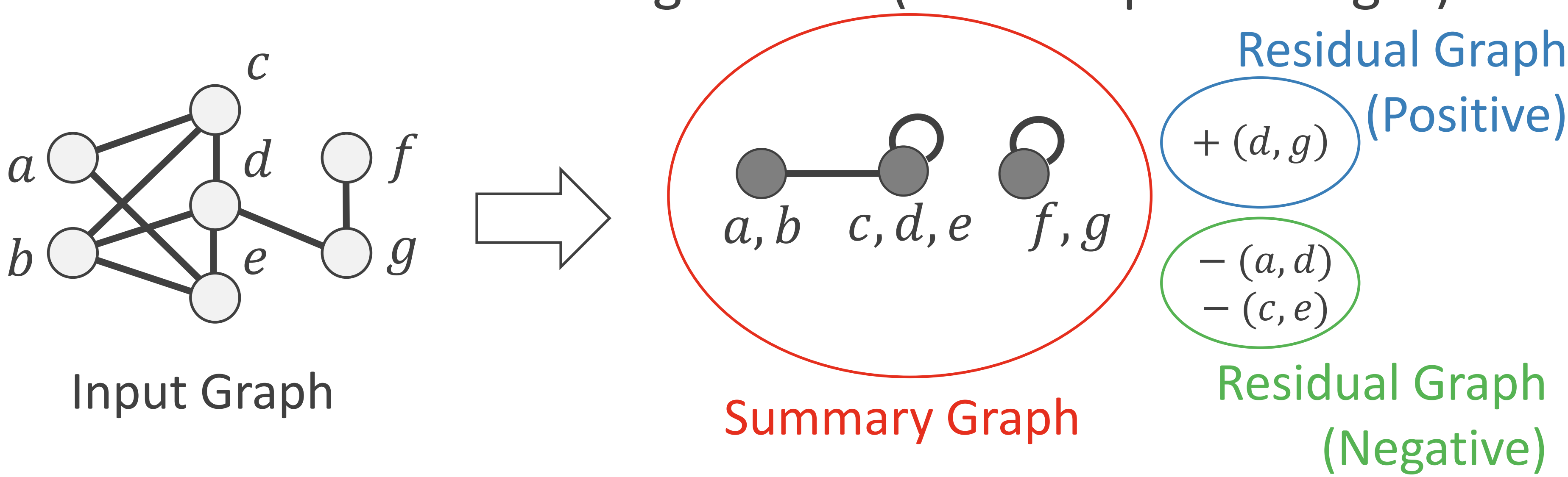




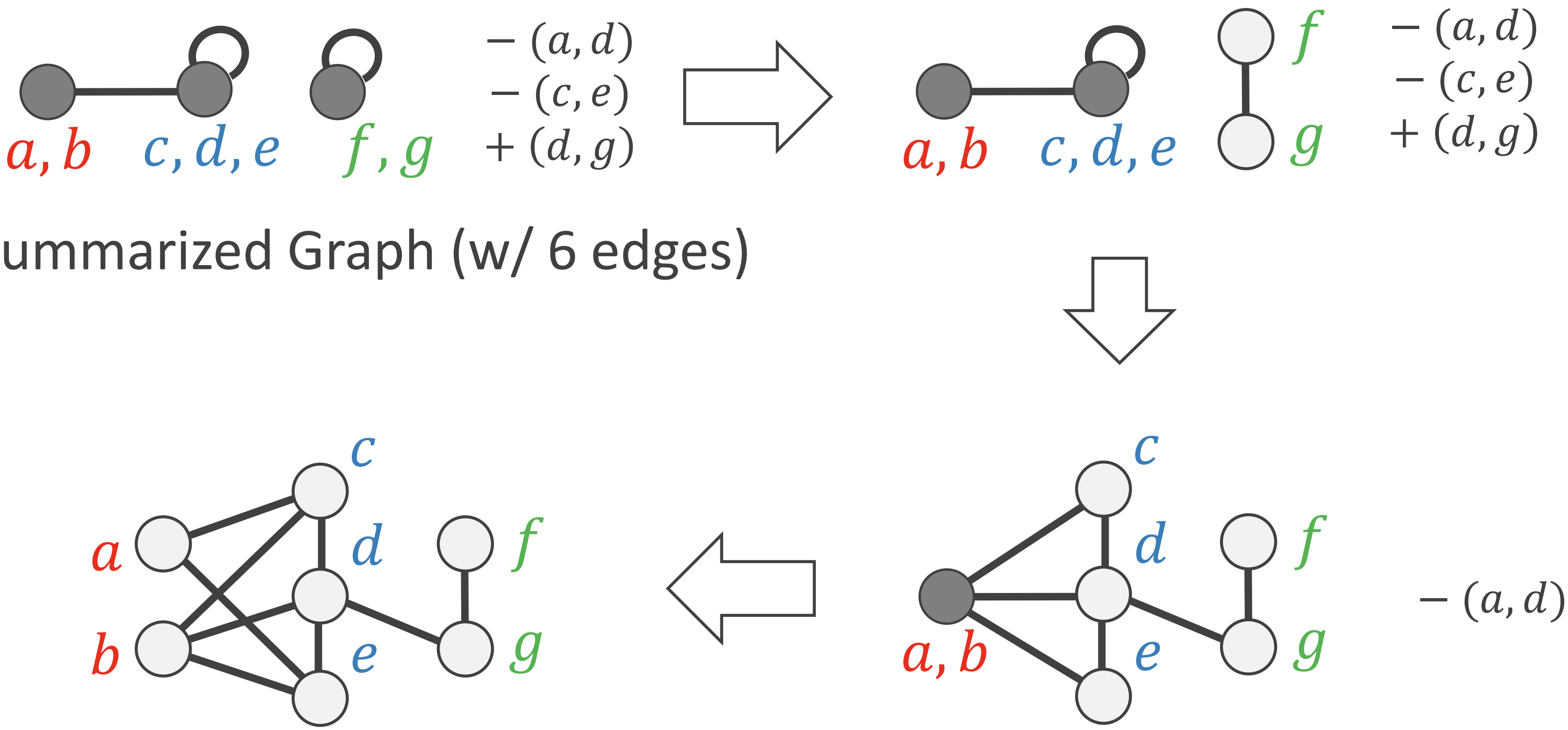
Output (w/ 6 edges)

Graph Summarization [NRS08]

- **Given:** an input graph
- **Find:**
  - a *summary graph*
  - positive and negative *residual graphs*
- **To Minimize**: the edge count (≈ description length)



Restoration: Example



Summarized Graph (w/ 6 edges)

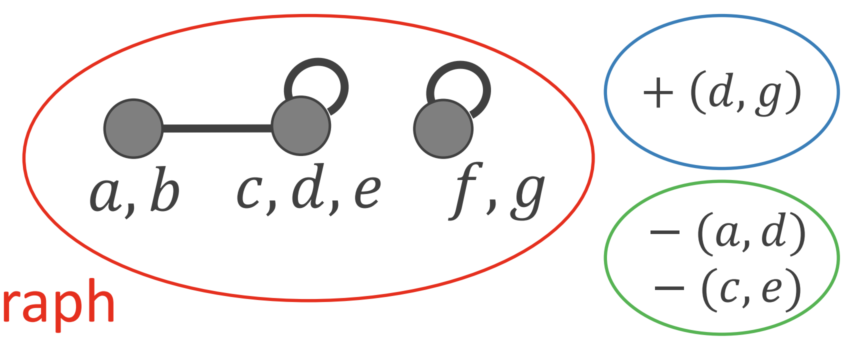
Restored Graph (w/ 9 edges)

Why Graph Summarization?

- Summarization:
  - the summary graph is easy to **visualize** and **interpret**
- Compression:
  - support **efficient neighbor queries**
  - applicable to **lossy compression**

  discussed in the thesis

  - **combinable** with other graph compression techniques
    - the outputs are also graphs

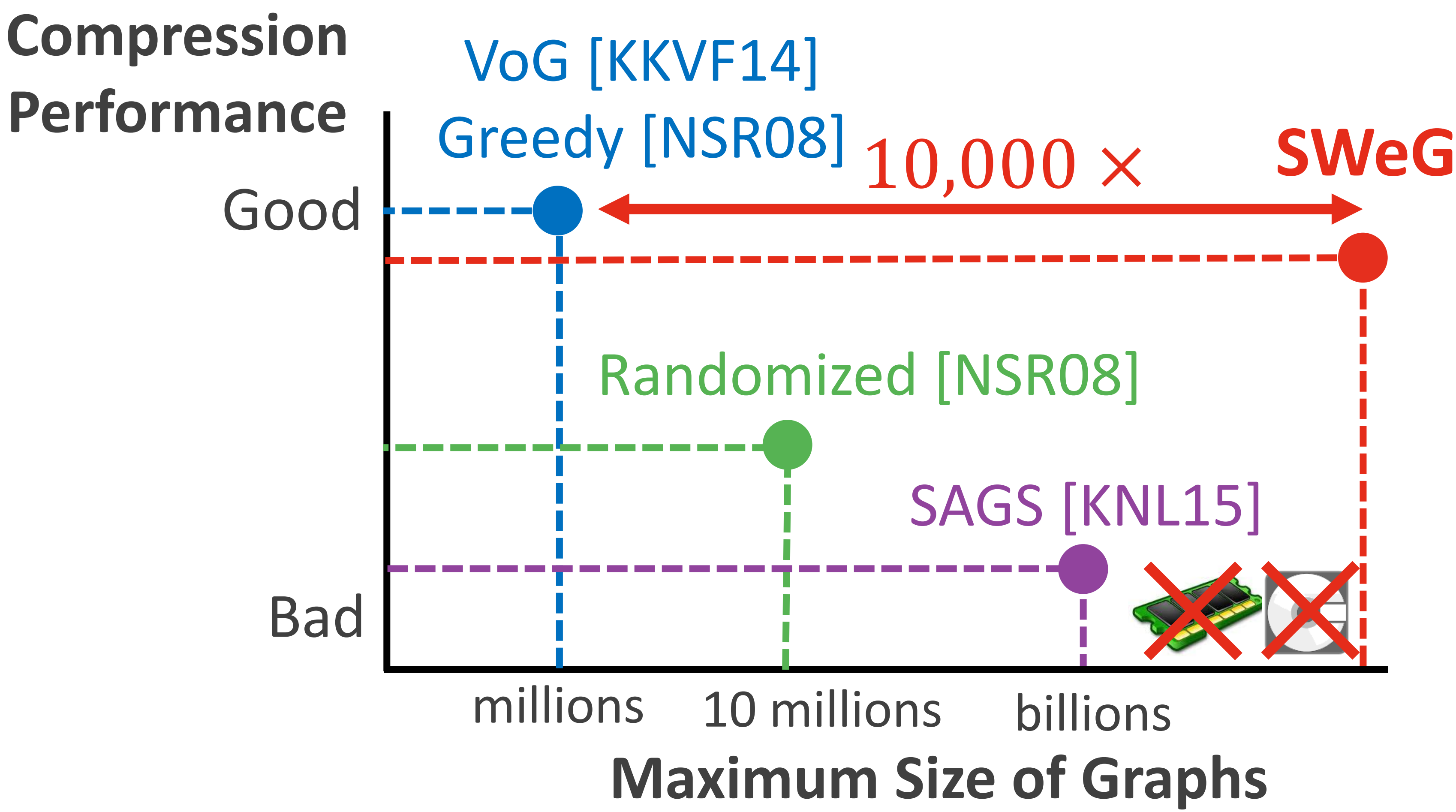


Residual Graph (Positive)

Residual Graph (Negative)

Summary Graph

Challenge: Scalability!



Our Contribution: SWeG

- We develop **SWeG** (**S**ummarizing **We**b-scale **G**raphs):

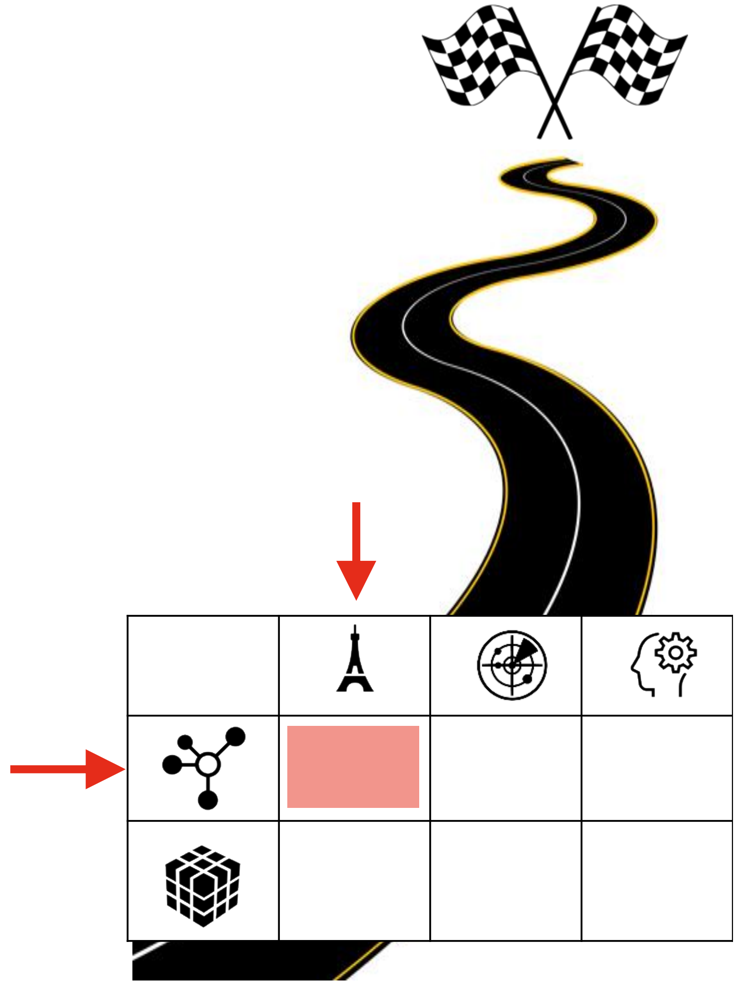
☐ **Fast with Concise Outputs**

☐ **Memory Efficient**

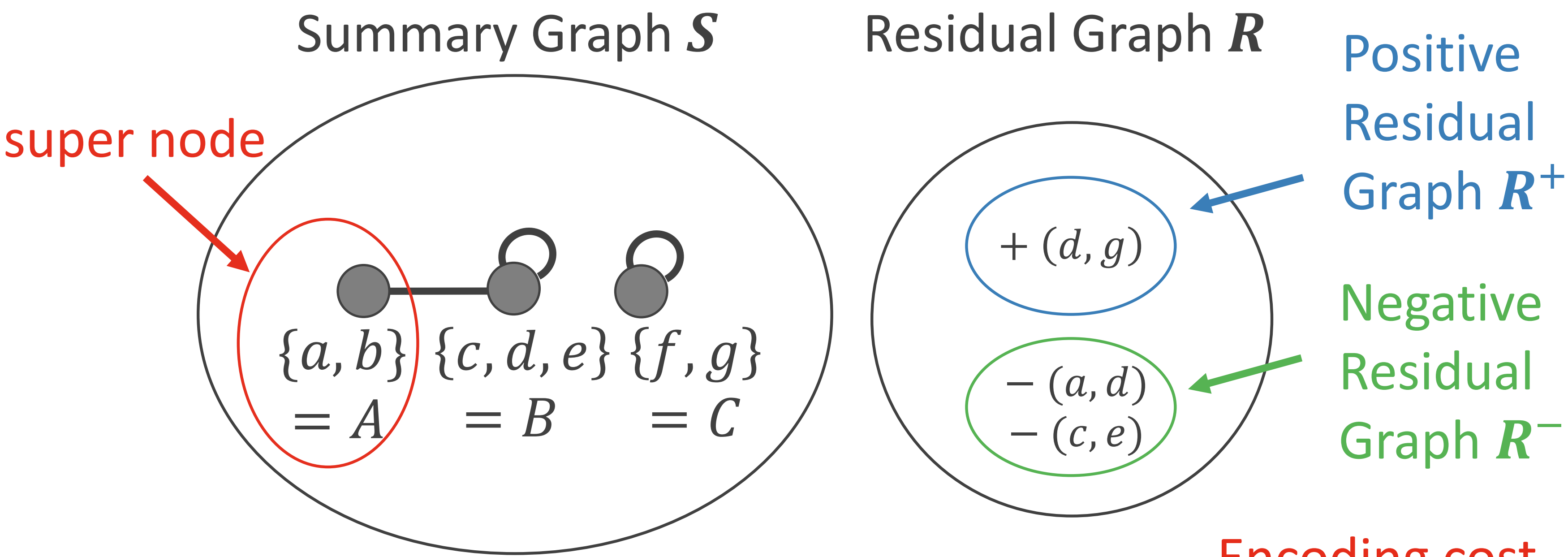
☐ **Scalable**

Roadmap

- T1. Structure Analysis (Part 1)
  - …
  - T1-2. Summarization (§§ 7-8)
    - Summarizing Graphs (§ 7)
      - Problem Definition
      - **Proposed Method: SWeG <<**
      - Experiments
    - …
- T2. Anomaly Detection (Part 2)
- T3. Behavior Modeling (Part 3)
- …



Terminologies



Encoding cost when A and B are **merged**

$$\boldsymbol{Saving}(\boldsymbol{A}, \boldsymbol{B}) := 1 - \frac{Cost(A \cup B)}{Cost(A) + Cost(B)}$$

Encoding cost of A

Encoding cost of B

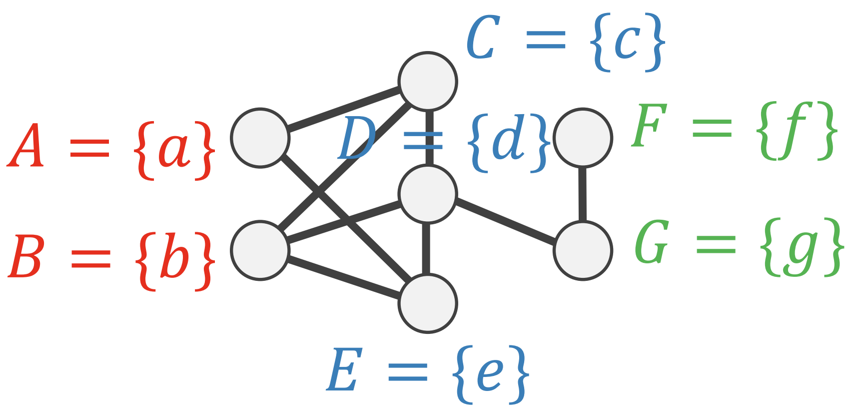
Overview of SWeG

- **Inputs:** - input graph $\boldsymbol{G}$
  - number of iterations $\boldsymbol{T}$
- **Outputs:** - summary graph $\boldsymbol{S}$
  - residual graph $\boldsymbol{R}$ (or $\boldsymbol{R}^{+}$ and $\boldsymbol{R}^{-}$)
- **Procedure:**

- S0: Initializing Step
- repeat T times
  - S1-1: Dividing Step
  - S1-2: Merging Step
- S2: Compressing Step (optional)

Overview: Initializing Step

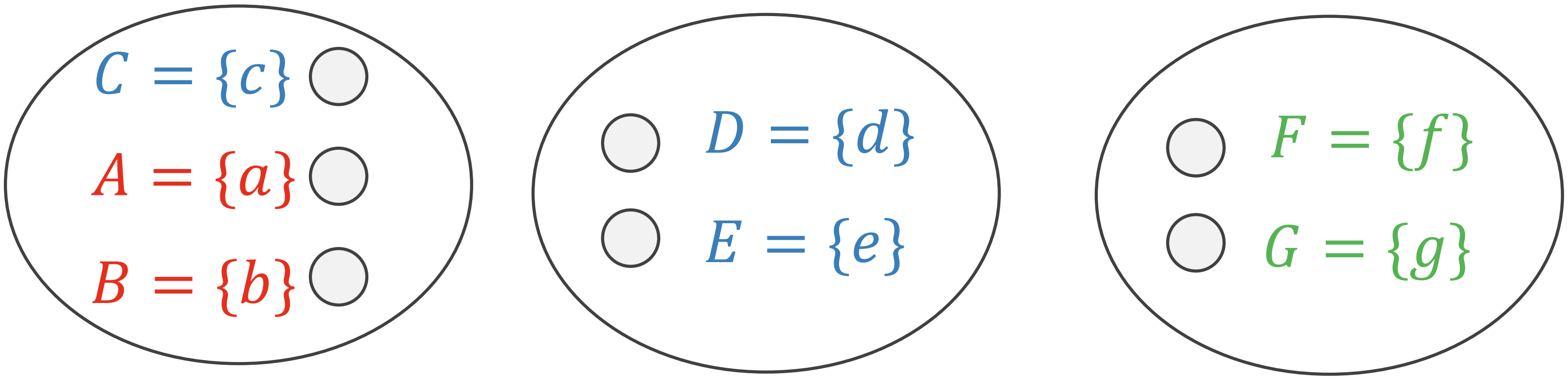
Summary Graph $\boldsymbol{S} = \boldsymbol{G}$ Residual Graph $\boldsymbol{R} = \emptyset$



- **S0: Initializing Step <<**
- repeat T times
  - S1-1: Dividing Step
  - S1-2: Merging Step
- S2: Compressing Step (optional)

Overview: Dividing Step

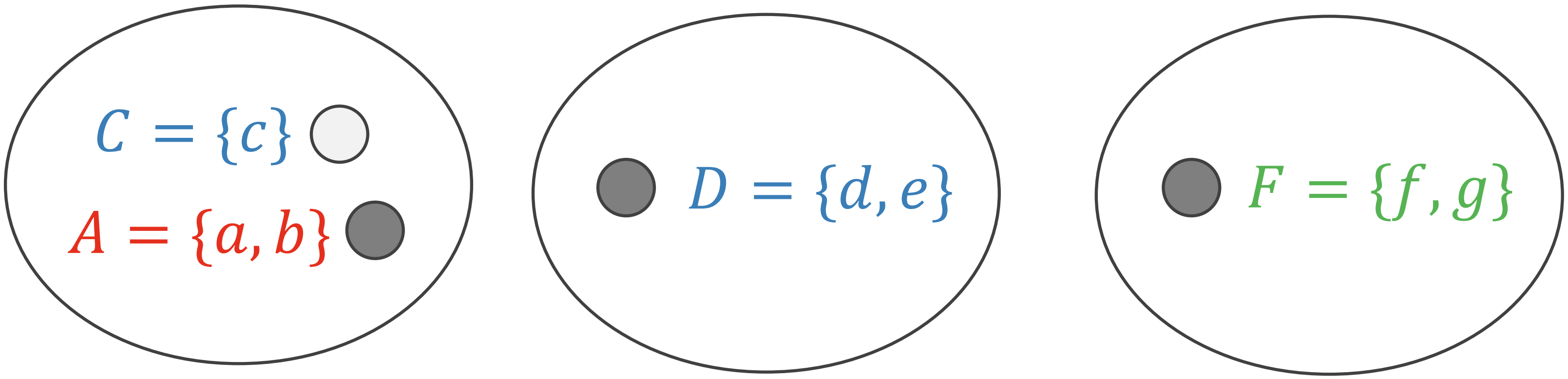
- Divides super nodes into groups
  - **MinHashing (used)**, EigenSpoke, Min-Cut, etc.



- S0: Initializing Step
- repeat T times
  - **S1-1: Dividing Step <<**
  - S1-2: Merging Step
- S2: Compressing Step (optional)

Overview: Merging Step

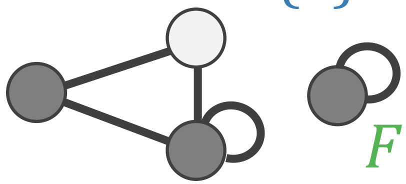
- Merge some supernodes within each group if $Saving > \theta^{(t)}$



- S0: Initializing Step
- repeat T times
  - S1-1: Dividing Step
  - **S1-2: Merging Step <<**
- S2: Compressing Step (optional)

Overview: Merging Step (cont.)

Summary Graph $\boldsymbol{S}$



Residual Graph $\boldsymbol{R}$

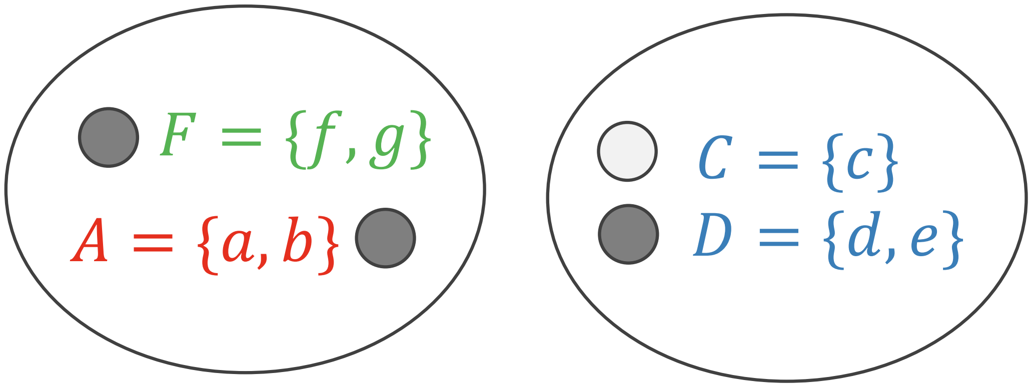
$+ (d, g)$

$- (a, d)$

- S0: Initializing Step
- repeat T times
  - S1-1: Dividing Step
  - S1-2: Merging Step
- S2: Compressing Step (optional)

Overview: Dividing Step

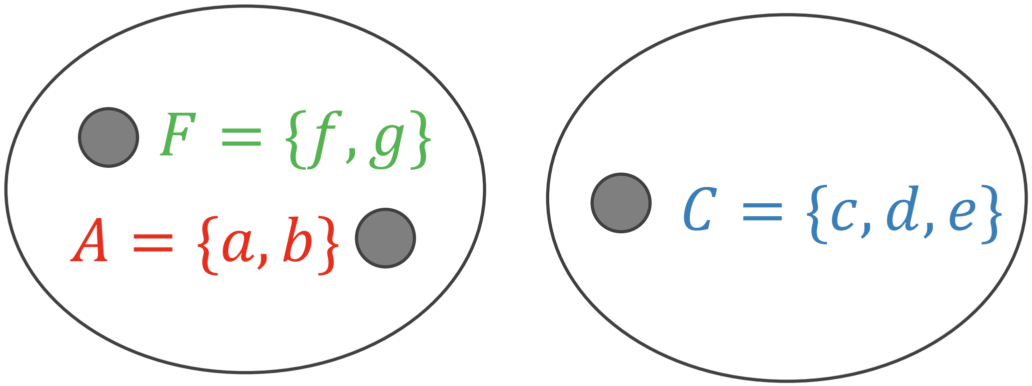
- Divides super nodes into groups



- S0: Initializing Step
- repeat T times
  - **S1-1: Dividing Step <<**
  - S1-2: Merging Step
- S2: Compressing Step (optional)

Overview: Merging Step

- Merge some supernodes within each group if $Saving > \theta^{(t)}$

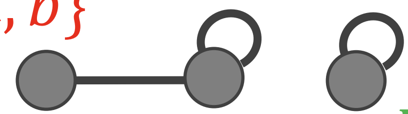


- S0: Initializing Step
- repeat T times
  - S1-1: Dividing Step
  - **S1-2: Merging Step <<**
- S2: Compressing Step (optional)

Overview: Merging Step (cont.)

Summary Graph $\boldsymbol{S}$

$A = \{a, b\}$



$F = \{f, g\}$

$B = \{c, d, e\}$

Residual Graph $\boldsymbol{R}$

$+ (d, g)$

$- (a, d)$
$- (c, e)$

- S0: Initializing Step
- repeat T times
  - S1-1: Dividing Step
  - S1-2: Merging Step
- S2: Compressing Step (optional)

Overview: Merging Step (cont.)

- Merge some supernodes within each group if $Saving > \theta^{(t)}$
- Decreasing $\theta^{(t)} = (1+t)^{-1}$
  - ***exploration*** of other groups

    → ***exploitation*** within each group
  - $\sim$ 30% better compression than $\theta^{(t)} = 0$

- S0: Initializing Step
- repeat T times
  - S1-1: Dividing Step
  - **S1-2: Merging Step <<**
- S2: Compressing Step (optional)

Overview: Compressing Step

- Compress each output graph ($\boldsymbol{S}$, $\boldsymbol{R}^{+}$ and $\boldsymbol{R}^{-}$)
- Use any off-the-shelf graph-compression algorithm
  - Boldi-Vigna [BV04]
  - VNMiner [BC08]
  - Graph Bisection [DKKO+16]

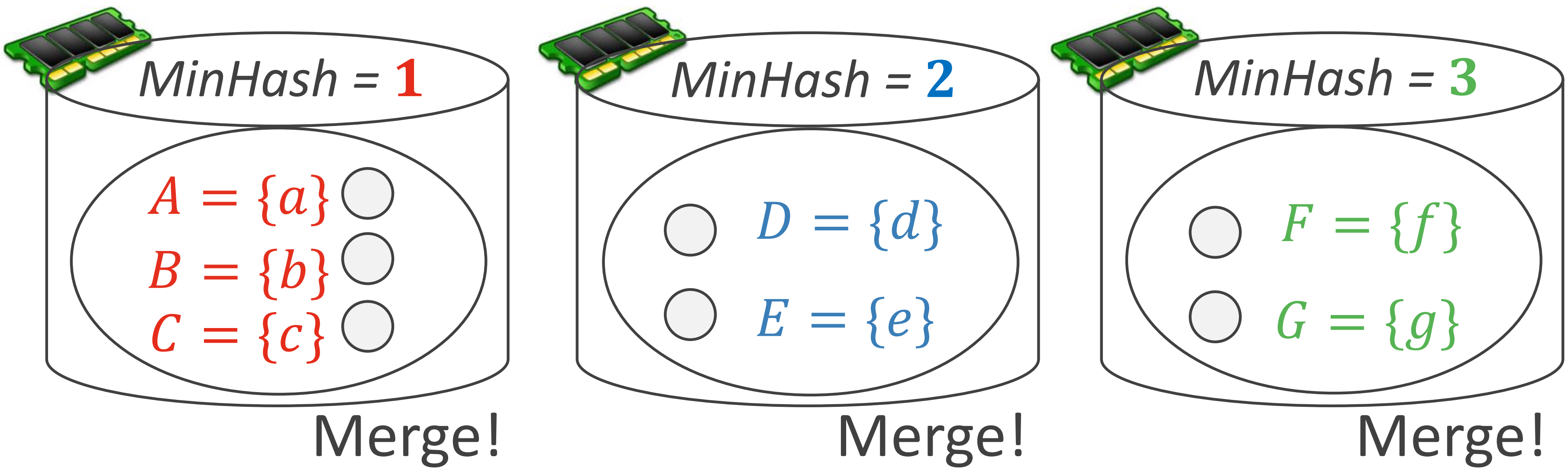
- S0: Initializing Step
- repeat T times
  - S1-1: Dividing Step
  - S1-2: Merging Step
- **S2: Compressing Step (optional) <<**

Details

Parallel & Distributed Processing

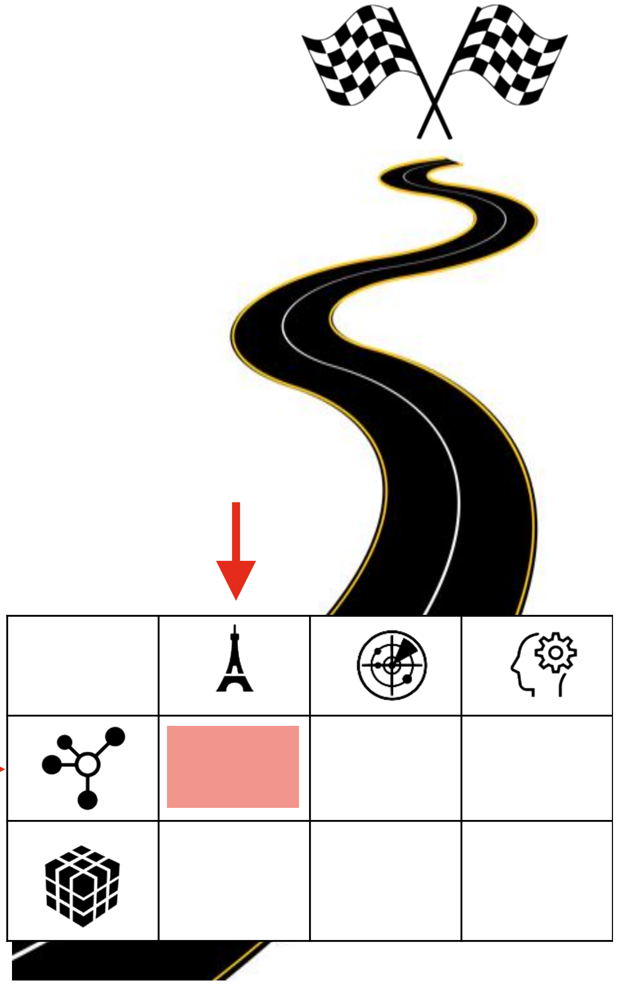
- **Map** stage: compute ***min hashes*** in ***parallel***
- **Shuffle** stage: divide super nodes using ***min hashes***
- **Reduce** stage: process groups independently in ***parallel***

No need to load the entire graph in memory!



Roadmap

- T1. Structure Analysis (Part 1)
  - …
  - T1-2. Summarization (§§ 7-8)
    - Summarizing Graphs (§ 7)
      - Problem Definition
      - Proposed Method: SWeG
      - **Experiments <<**
    - …
- T2. Anomaly Detection (Part 2)
- T3. Behavior Modeling (Part 3)
- …



Experimental Settings

- 13 real-world graphs (10K - **20B** edges)

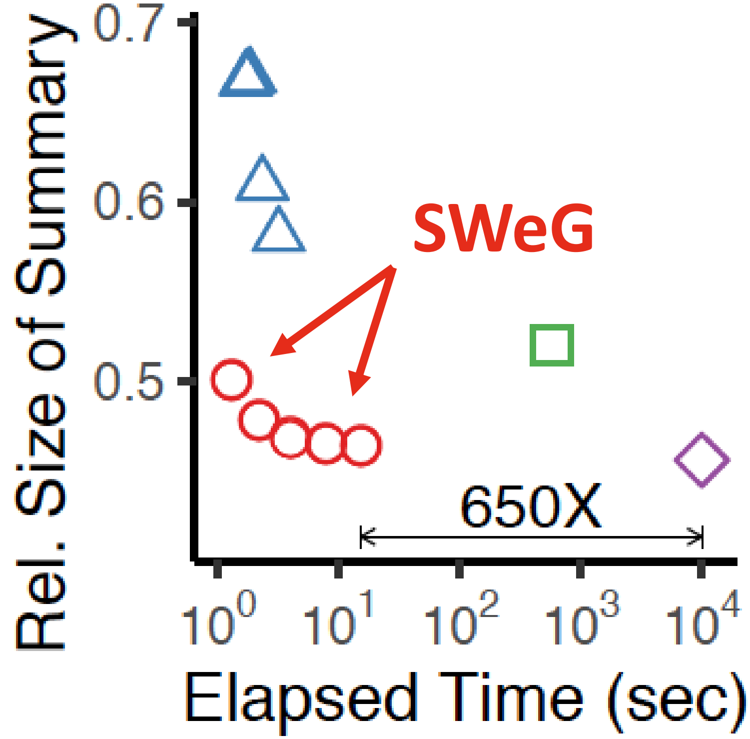
Social | Collaboration | Citation | Web | ...

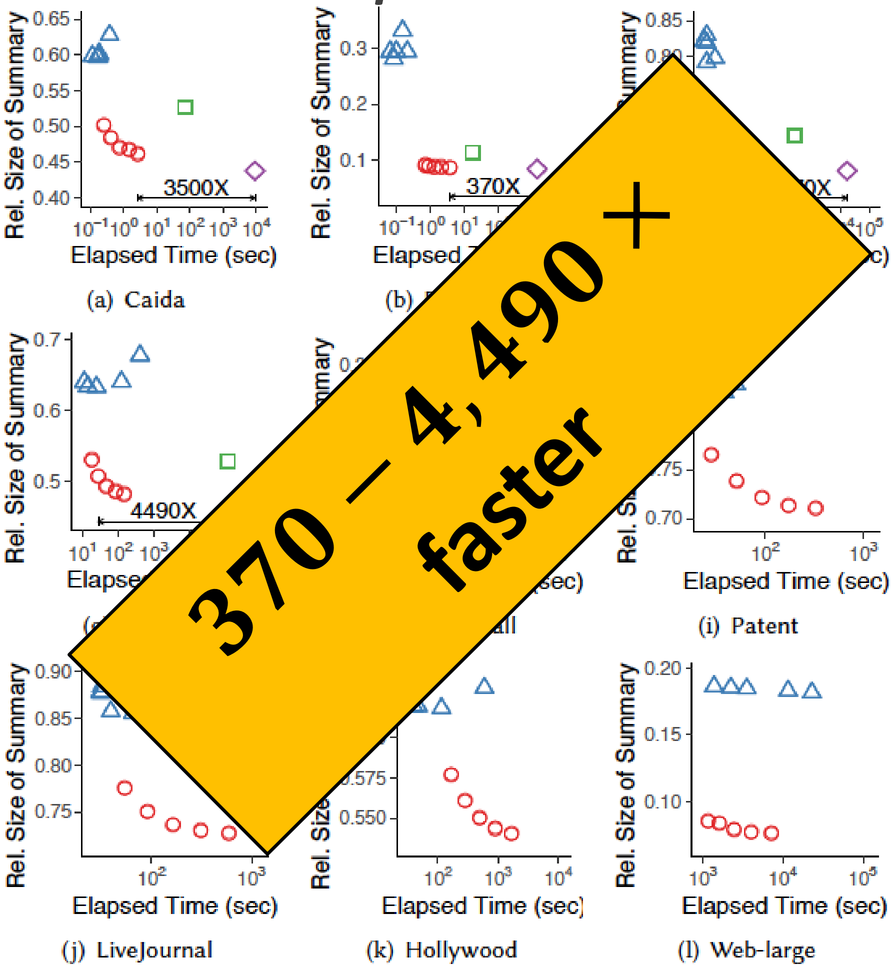
...

- Graph summarization algorithms:
  - *Greedy* [NRS08], *Randomized* [NSR08], *SAGS* [KNL15]
- Implementations: Java & Hadoop

EXP1. Speed and Compression

SWeG outperforms its competitors





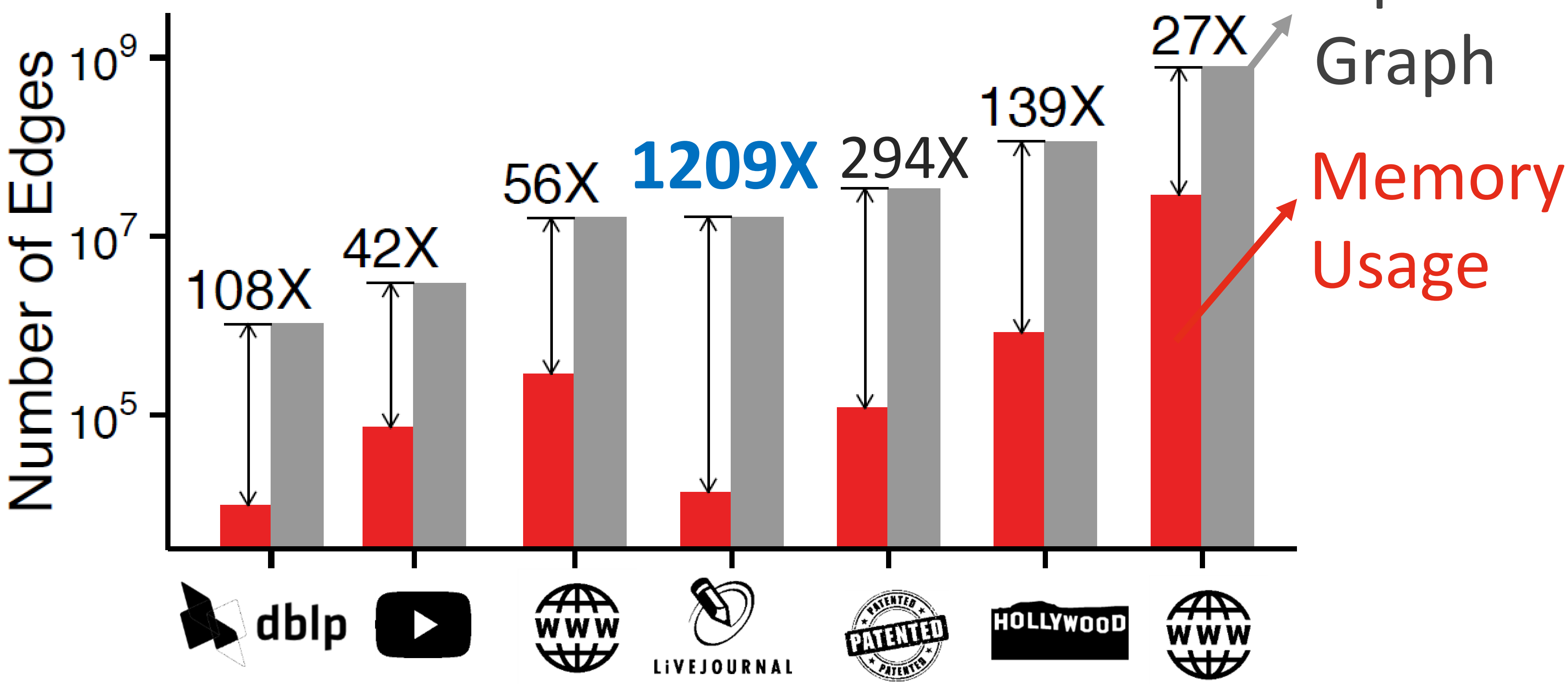
- dataset: dblp

Advantages of SWeG (Recall)

- [x] **Fast with Concise Outputs**
- [ ] Memory Efficient
- [ ] Scalable

EXP2. Memory Efficiency

*SWeG loads **≤0.1−4%** of edges in main memory at once*

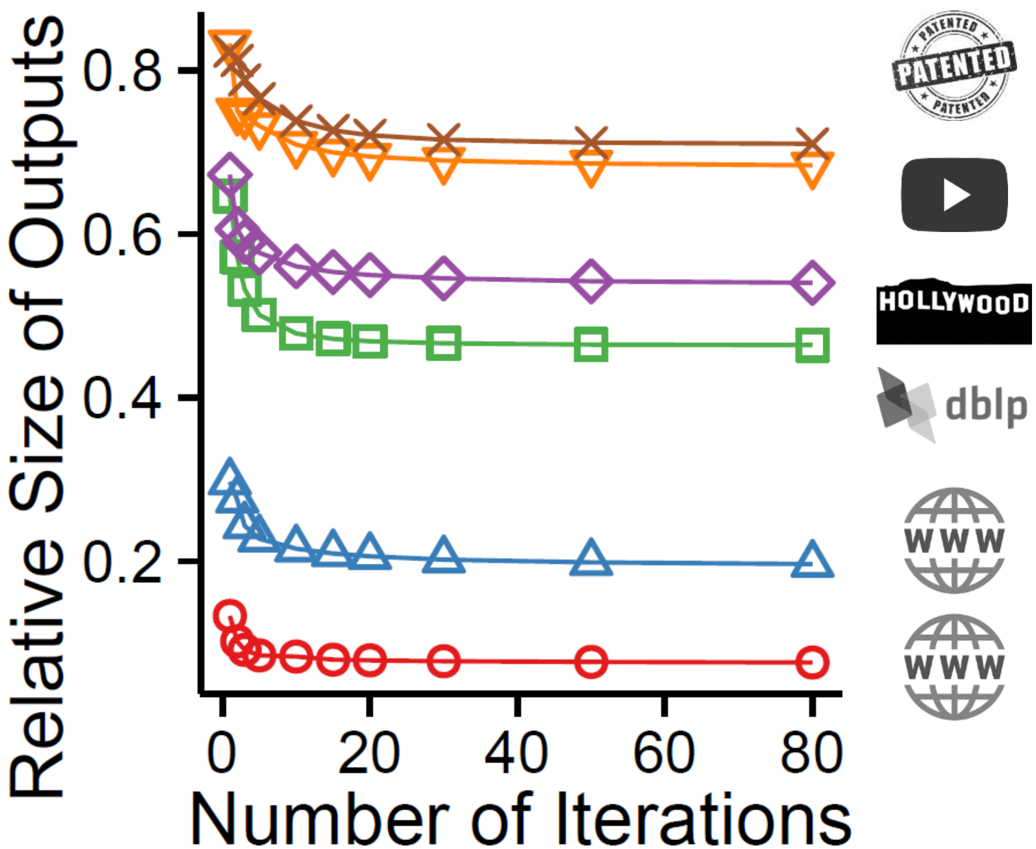


Advantages of SWeG (Recall)

- [x] Fast with Concise Outputs
- [x] Memory Efficient
- [ ] Scalable

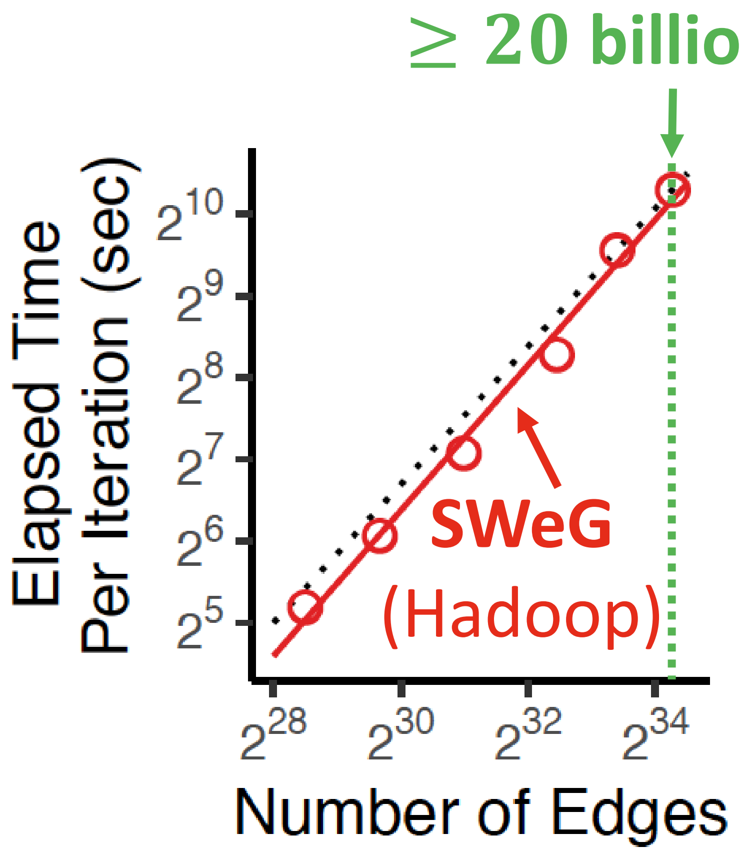
EXP3. Effect of Iterations

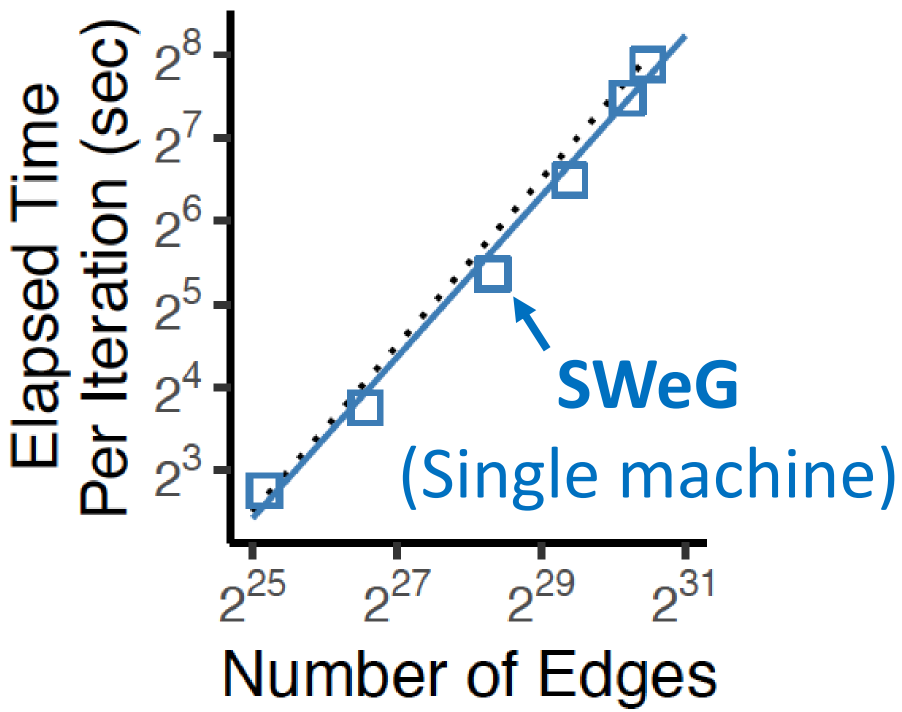
About 20 iterations are enough



EXP4. Data Scalability

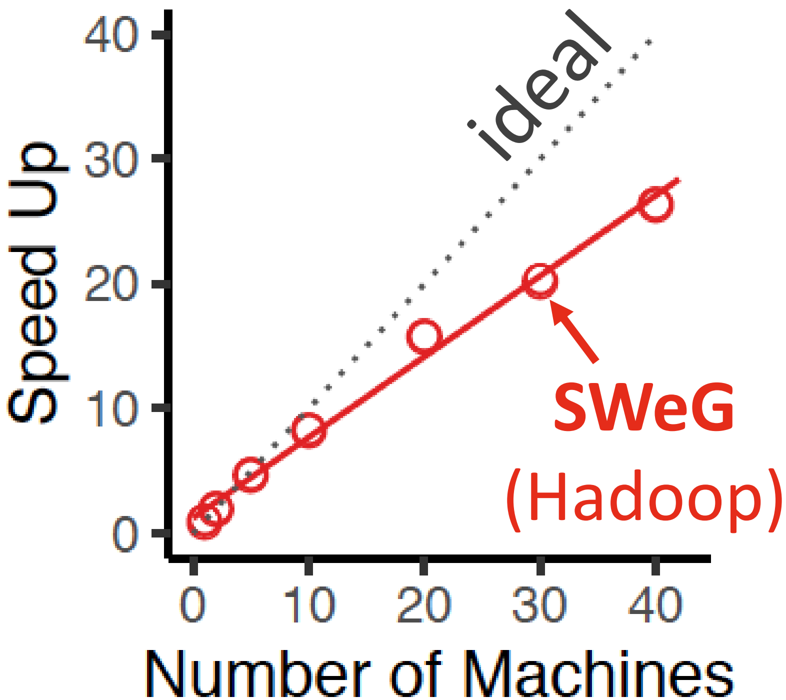
SWeG is linear in the number of edges

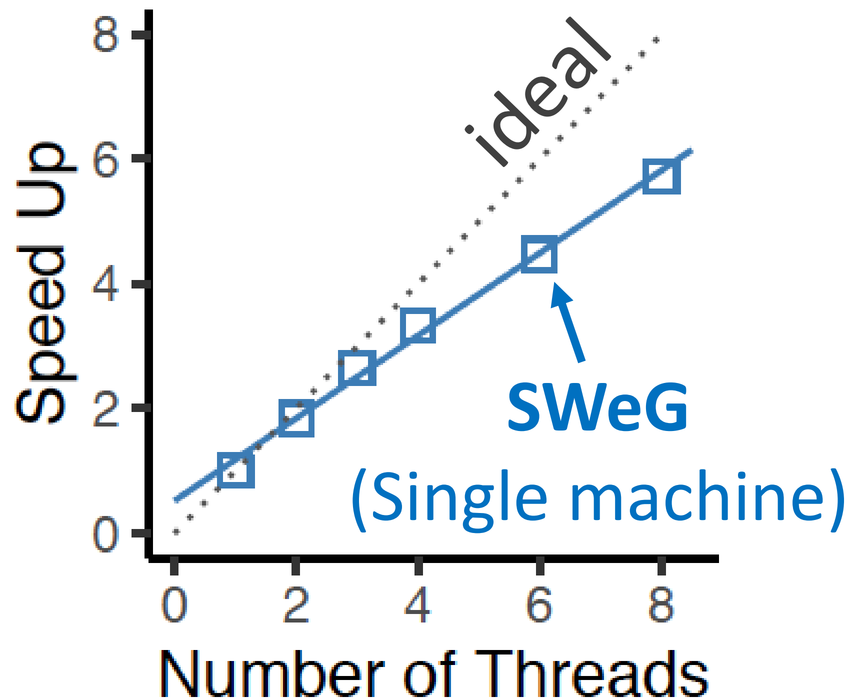




EXP5. Machine Scalability

SWeG scales up





Advantages of SWeG (Recall)

- ☑ Fast with Concise Outputs
- ☑ Memory Efficient
- ☑ Scalable

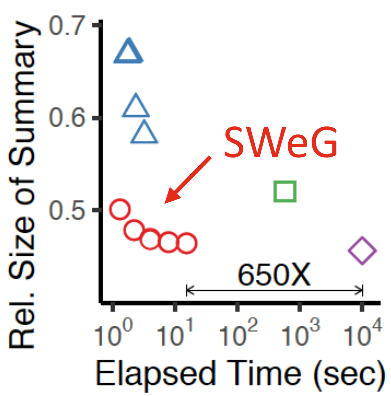
Summary of §7

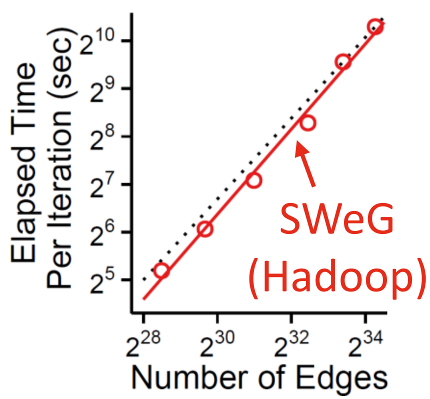
- We propose **SWeG (Summarizing Web Graphs)**
  - for summarizing large-scale graphs

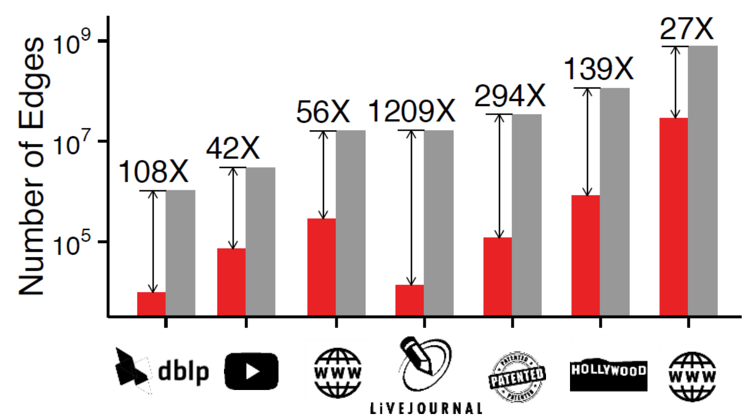
☑ **Fast with Concise Outputs**

☑ **Memory Efficient**

☑ **Scalable**

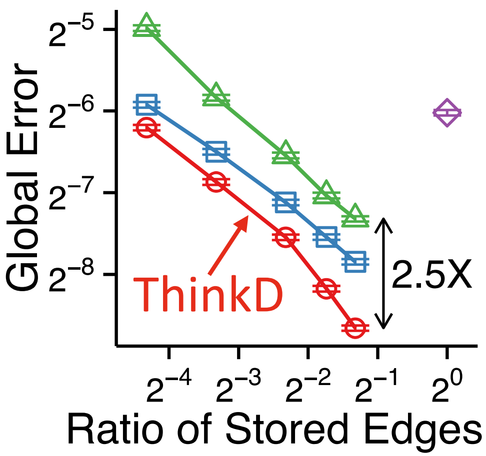


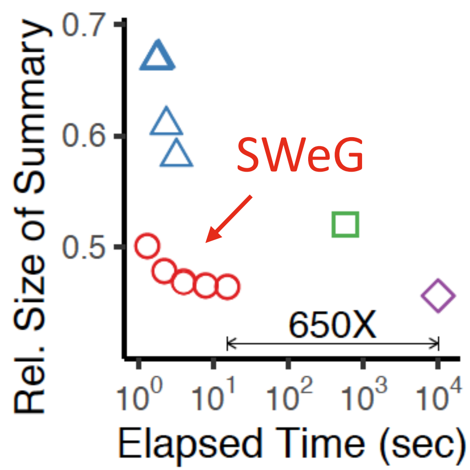




Contributions and Impact (Part 1)

- **Triangle counting algorithms** [ICDM17, PKDD18, PAKDD18]
- **Summarization algorithms** [WSDM17, WWW19]
- **Patent** on *SWeG*: filed by LinkedIn Inc.
- **Open-source software**: downloaded ***82 times***





github.com
/kijungs

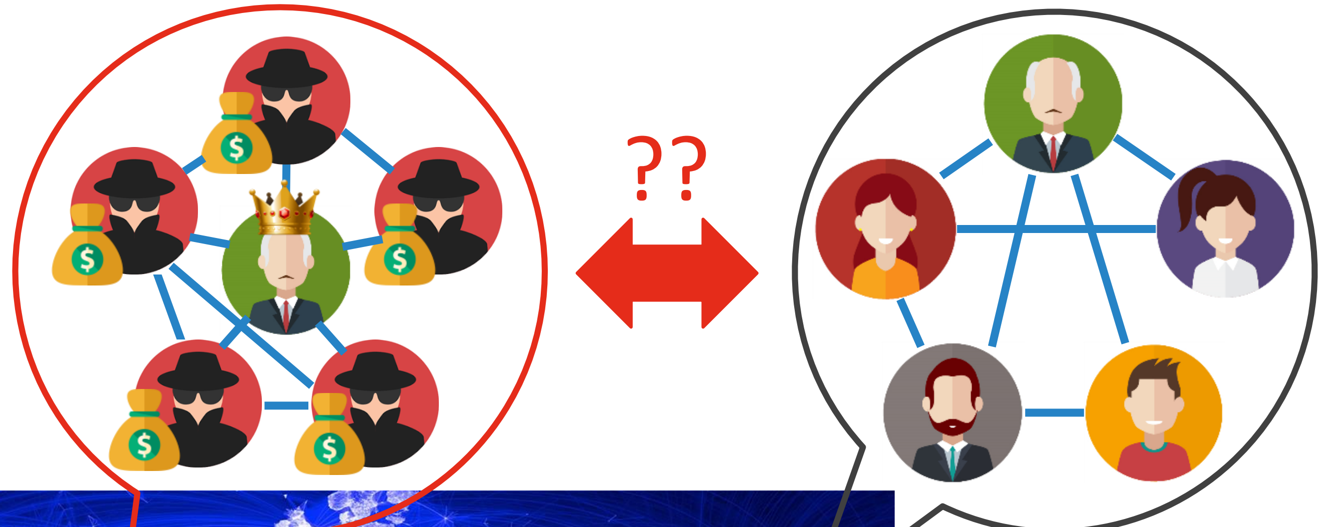
Organization of the Thesis (Recall)

	Part1. Structure Analysis	**Part2. Anomaly Detection**	Part3. Behavior Modeling
Graphs	Triangle Count (§§ 3-6) ✔ Summarization (§ 7) ✔	Anomalous Subgraph (§ 9)	Purchase Behavior (§ 14)
Tensors	Summarization (§ 8) ✔	Dense Subtensors (§§ 10-13)	Progression (§ 15)

T2. Anomaly Detection (Part 2)

*"How can we detect **anomalies** or **fraudsters** in large dynamic graphs (or tensors)?"*

Hint: *fraudsters* tend to form *dense subgraphs*

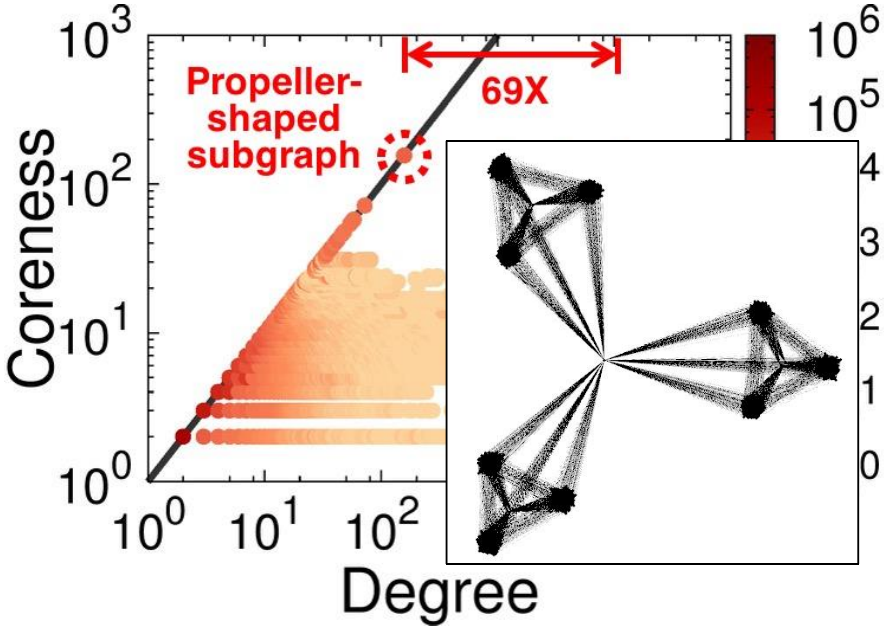


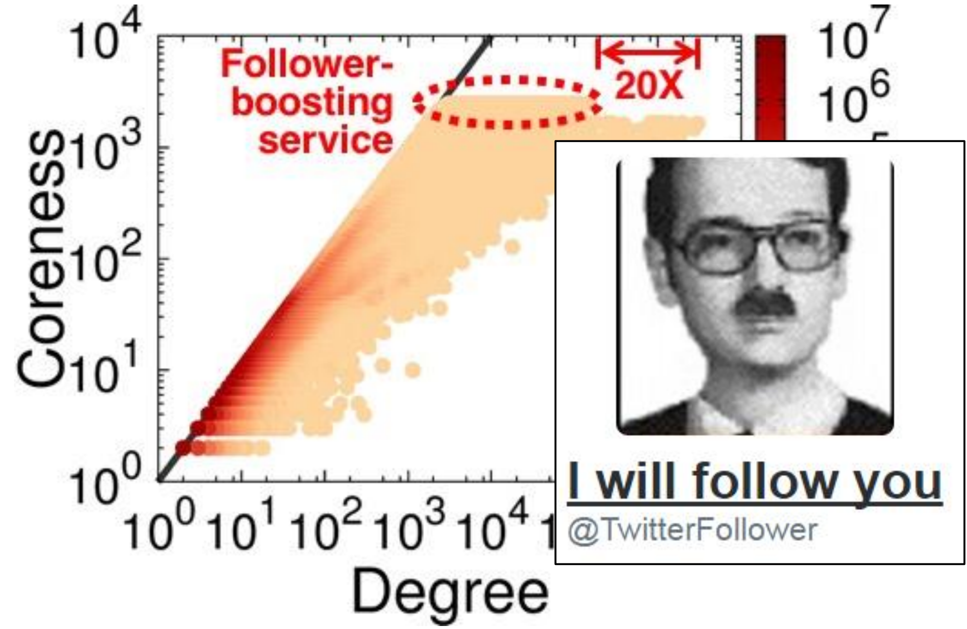
benign dense subgraphs

T2-1. Utilizing Patterns

- T2-1. Patterns and Anomalies in **Dense Subgraphs** (§ 9)

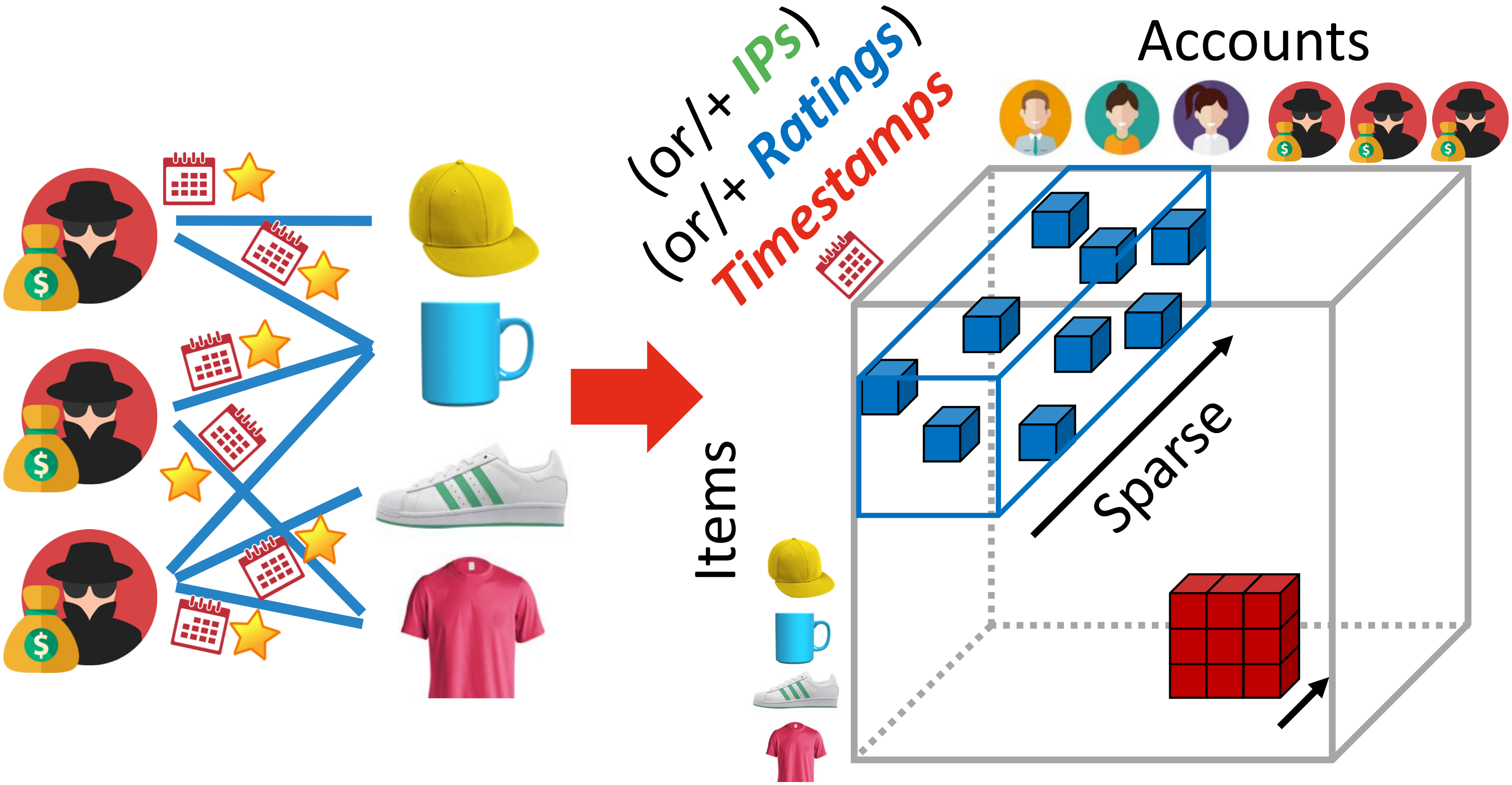
*"What are **patterns** in dense subgraphs?"*
*"What are **anomalies** deviating from the patterns?"*





K. Shin, T. Eliassi-Rad, C. Faloutsos, "Patterns and Anomalies in k-Cores of Real-world Graphs with Applications", **KAIS 2018** (formerly, **ICDM 2016**)

T2-2. Utilizing Side Information

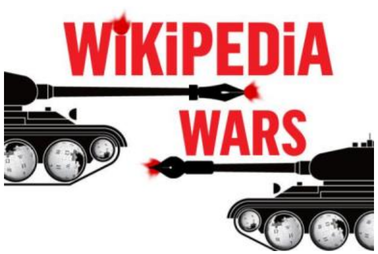


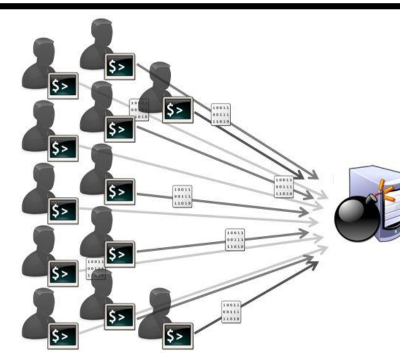
T2-2. Utilizing Side Information

*"How can we detect **dense subtensors** in large dynamic data?"*

- T2-2. Detecting **Dense Subtensors** (§§ 11-13)
  - In-memory Algorithm (§ 11)
  - Distribute Algorithm for Web-scale Tensors (§ 12)
  - Incremental Algorithms for Dynamic Tensors (§ 13)

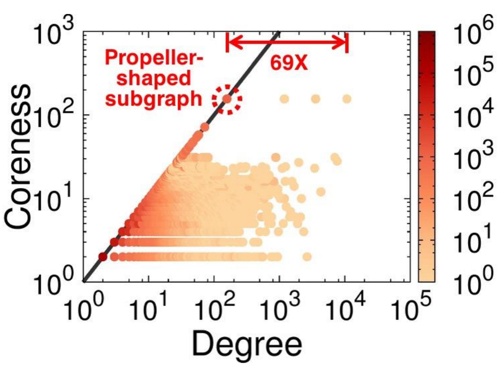
WIKIPEDIA The Free Encyclopedia | Download on the App Store | TCP/IP

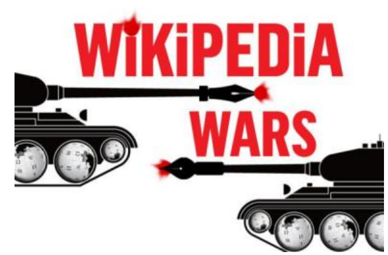




Contributions and Impact (Part 2)

- **Patterns** in dense subgraphs [ICDM16]
  - Award: best paper candidate at ICDM 2016
  - Class: Massachusetts Institute of Technology
- **Algorithms** for dense subtensors [PKDD16, WSDM17, KDD17]
  - Real-world usage: NAVER
- **Open-source software**: downloaded ***257 times***





github.com
/kijungs

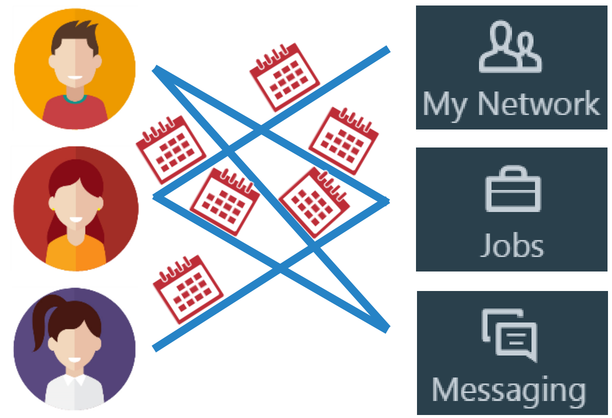
Organization of the Thesis (Recall)

	Part1. **Structure Analysis**	**Part2.** Anomaly Detection	**Part3.** Behavior Modeling
Graphs	Triangle Count (§§ 3-6) Summarization (§ 7)	Anomalous Subgraph (§ 9)	Purchase Behavior (§ 14)
Tensors	Summarization (§ 8)	Dense Subtensors (§§ 10-13)	Progression (§ 15)

T3. Behavior Modeling (Part 3)

*“How can we **model** the **behavior** of **individuals** in graph and tensor data?”*

Social Network



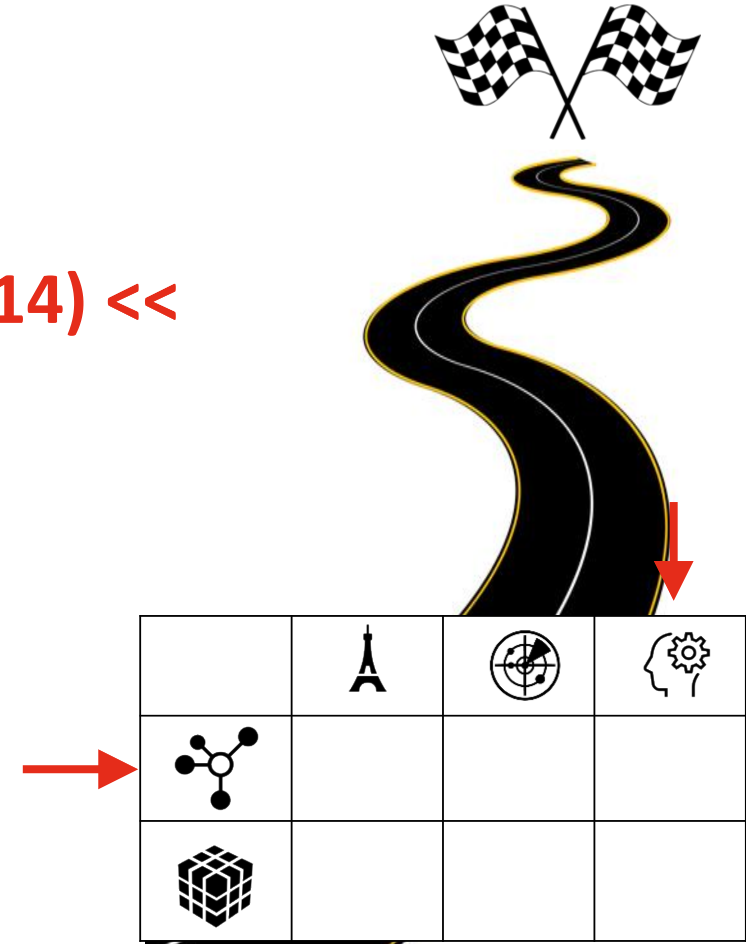
Behavior Log on Social Media

- T3-1. Modeling Purchase Behavior in a Social Network (§14)
- T3-2. Modeling Progression of Users of Social Media (§15)

*“How do users **evolve over time** on social media?”*

Roadmap

- T1. Structure Analysis (Part 1)
- T2. Anomaly Detection (Part 2)
- T3. Behavior Modeling (Part 3)
  - **T3-1. Modeling Purchases (§14) <<**
  - …
- Future Directions
- Conclusions



K Shin, E Lee, D Eswaran, AD Procaccia, "Why You Should Charge Your Friends for Borrowing Your Stuff", **IJCAI 2017**

Sharable Goods: Question

Portable crib

IKEA toolkit

DVDs

“What do they have in common?”

Sharable Goods: Properties

- Used occasionally
- Share with **friends**
- Do not share with **strangers**

Motivation: Social Inefficiency

Popular

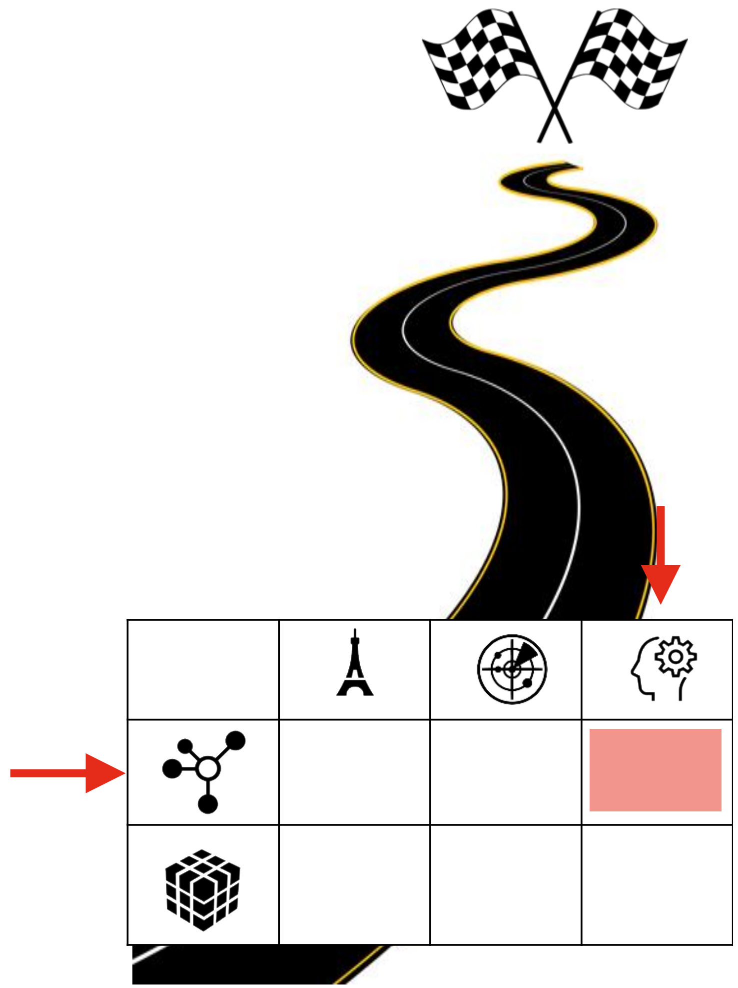
Lonely

Efficiency of Purchase	**High** (share with many)	**Low** (share with few)
Likelihood of Purchase	can be **Low** (likely to borrow)	can be **High** (likely to buy)

*Q1 "How large can **social inefficiency** be?"*
*Q2 "How to **nudge** people towards efficiency?"*

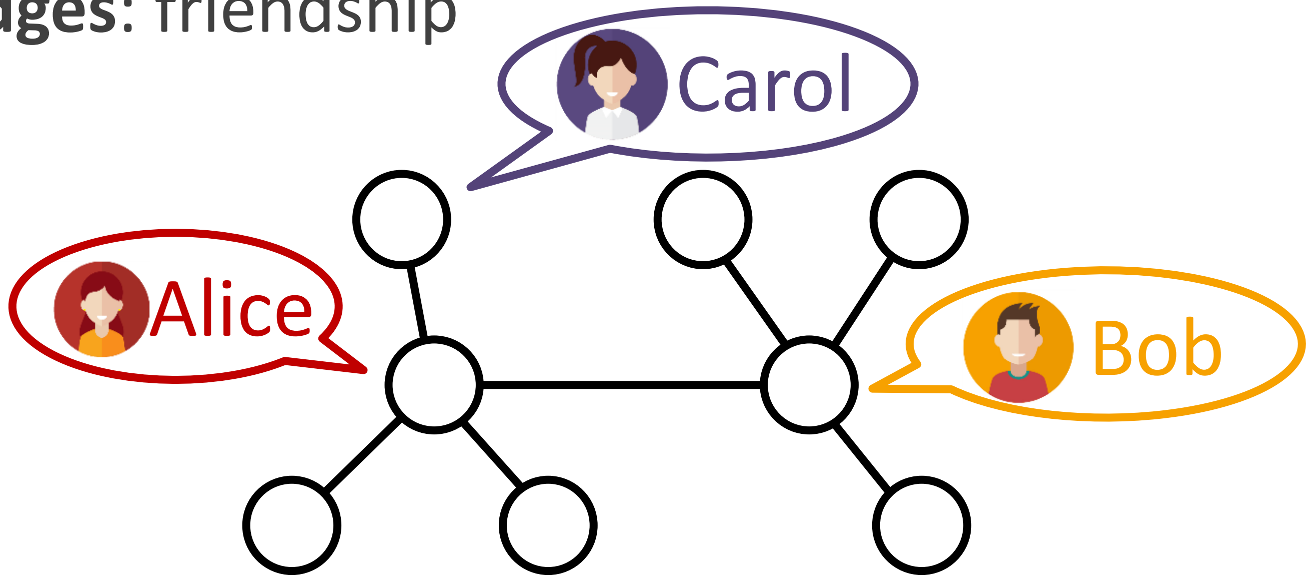
Roadmap

- T1. Structure Analysis (Part 1)
- T2. Anomaly Detection (Part 2)
- T3. Behavior Modeling (Part 3)
  - T3-1. Modeling Purchases (§14)
    - **Toy Example <<**
    - Game-theoretic Model
    - Best Rental-fee Search
  - …
- Future Directions
- Conclusions



Social Network

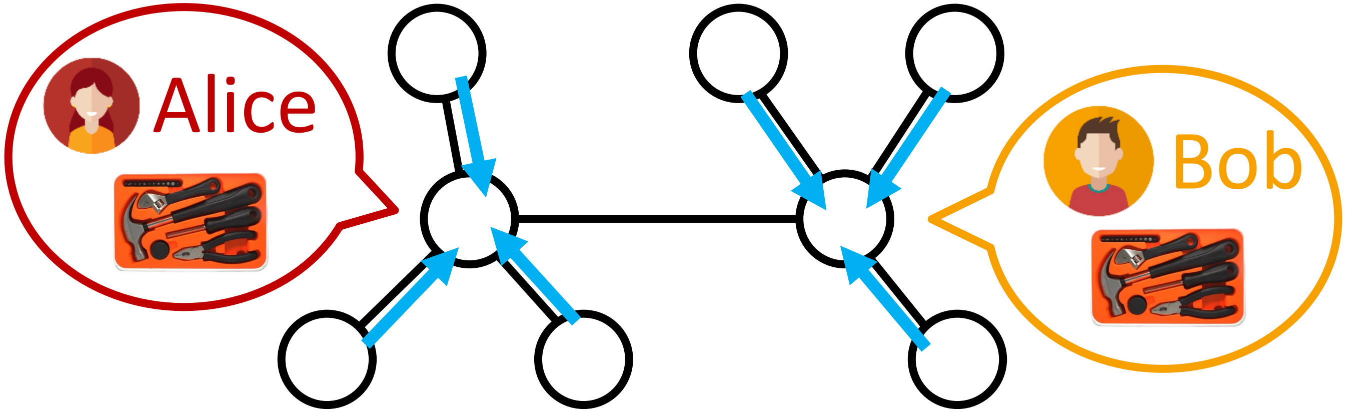
- Consider a **social network**, which is a graph
  - **Nodes**: people
  - **Edges**: friendship



"How many people should buy an IKEA toolkit for everyone to use it?"

Socially Optimal Decision

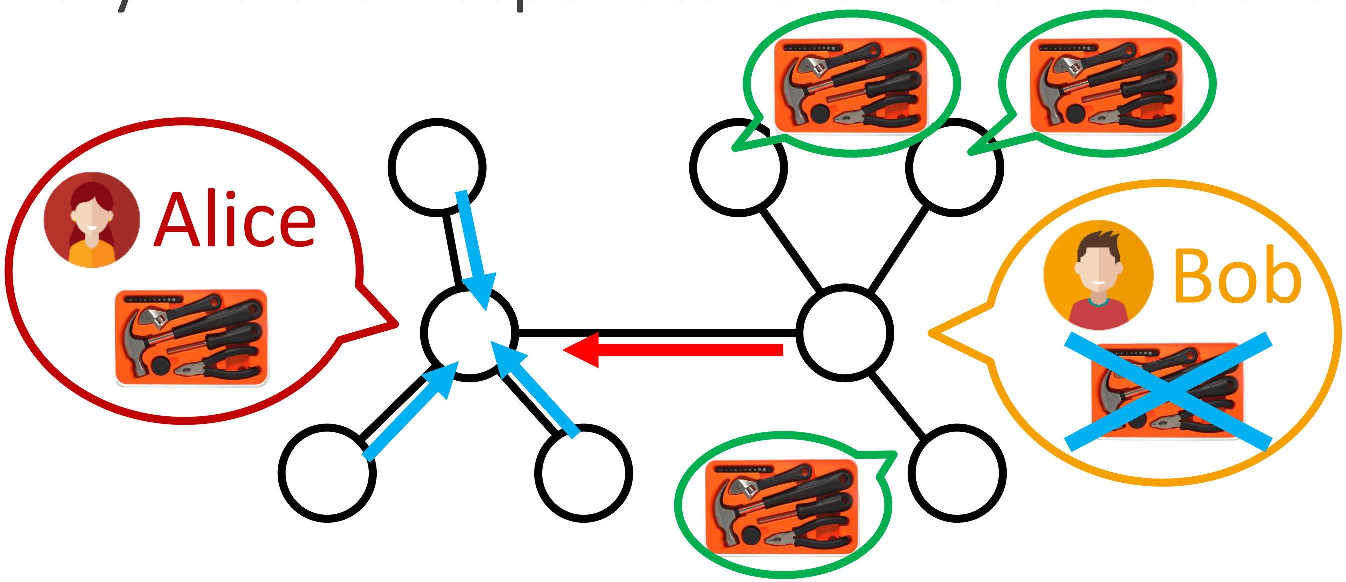
- The answer is **at least 2**
- **Socially optimal**:
  - everyone uses a toolkit
  - with minimum purchases (or with minimum cost)



“Does everyone want to stick to their current decisions?”

Individually Optimal Decision

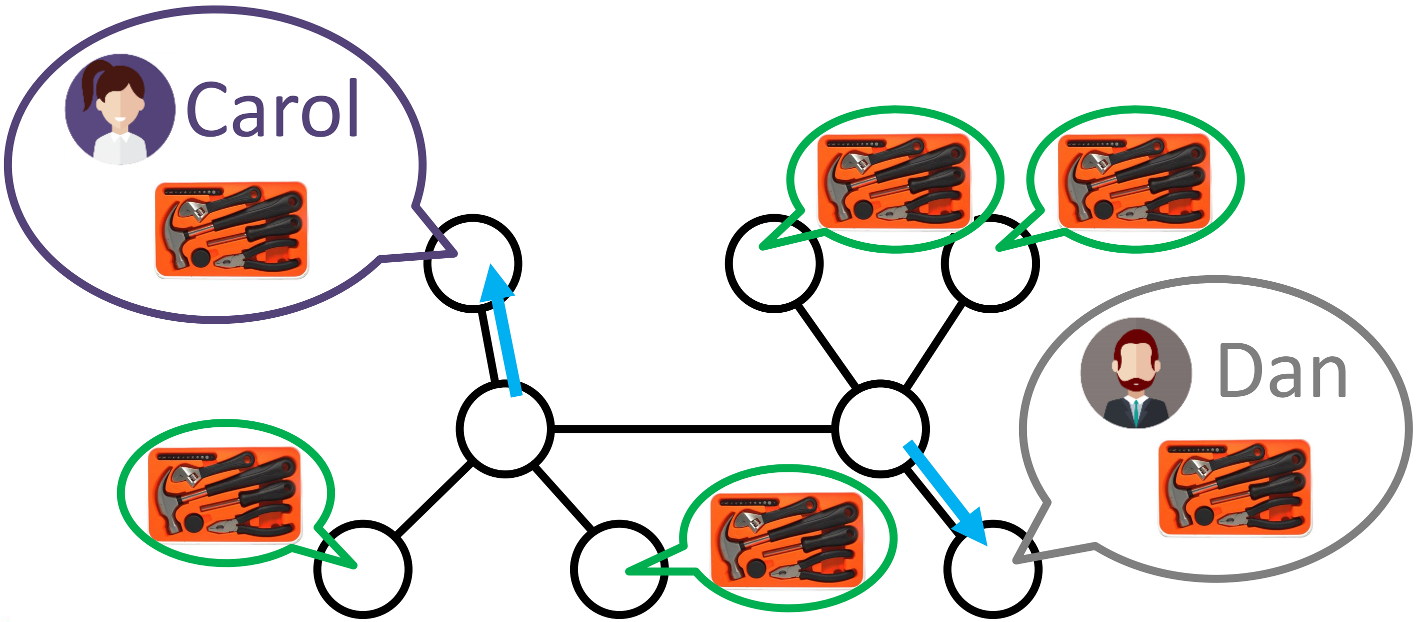
- The answer is **No**
- **Individually optimal**:
  - everyone best responses to others' decisions



- **Socially inefficient** (suboptimal):
  - 4 purchases happen when 2 are optimal

Social Inefficiency

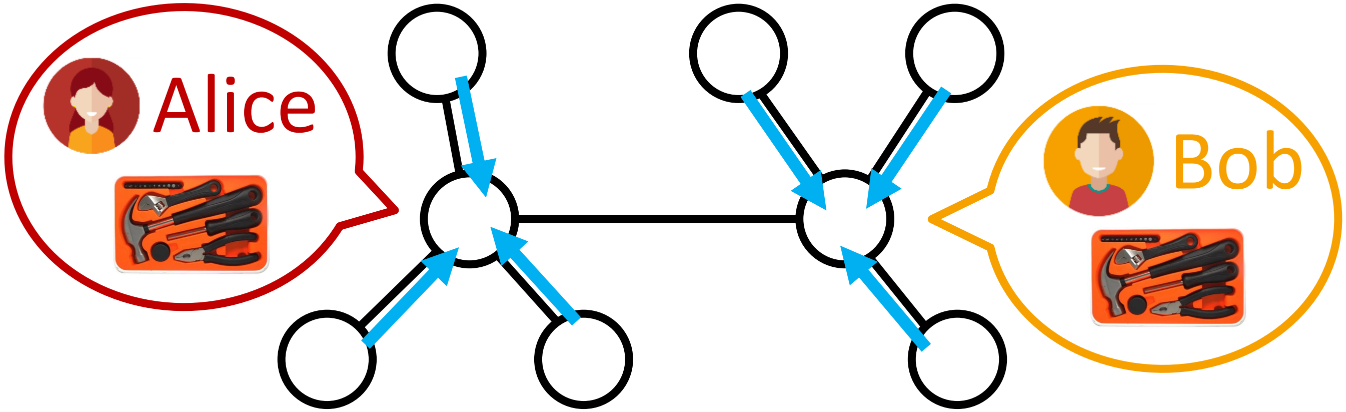
- Individually optimal outcome with 6 purchases



"How can we prevent this social inefficiency?"

Moving toward Social Optimum

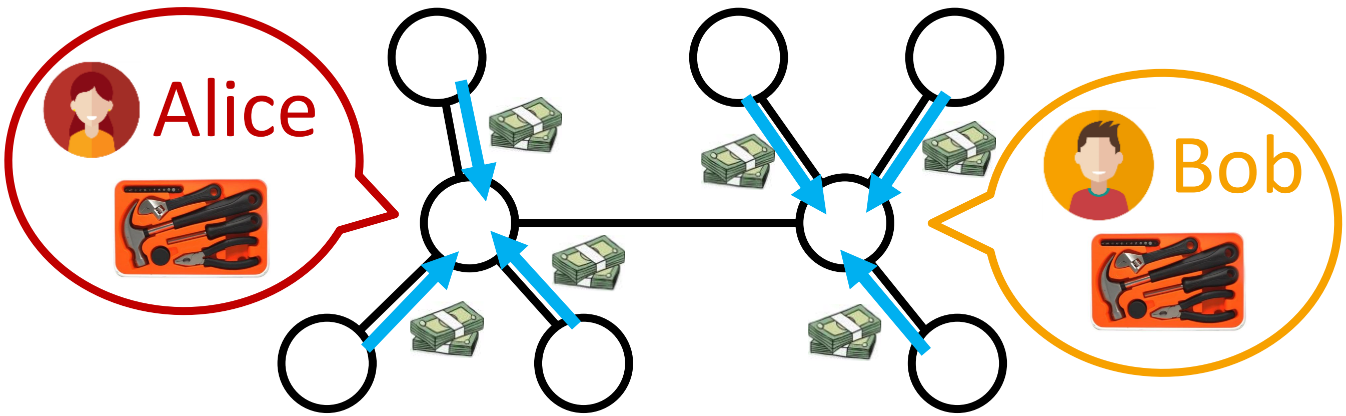
- Recall the **socially optimal outcome**



"How can we make people stick with this socially optimal outcome?"

Imposing Rental Fee

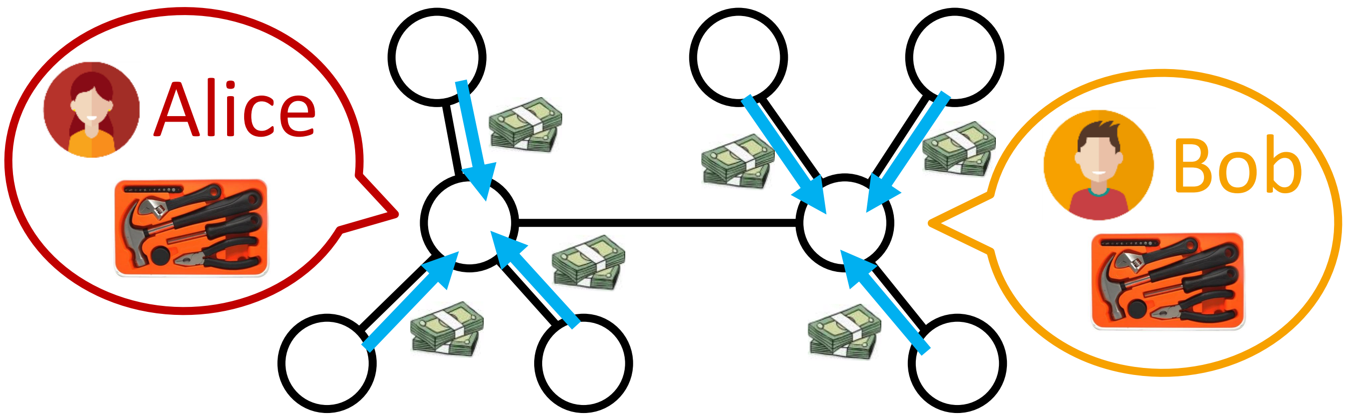
- Renters pay rental fee for getting **permanent access**
- Rental fee is **half** the price of a toolkit



"Does everyone want to stick to their current decisions?"

Socially & Individually Optimal

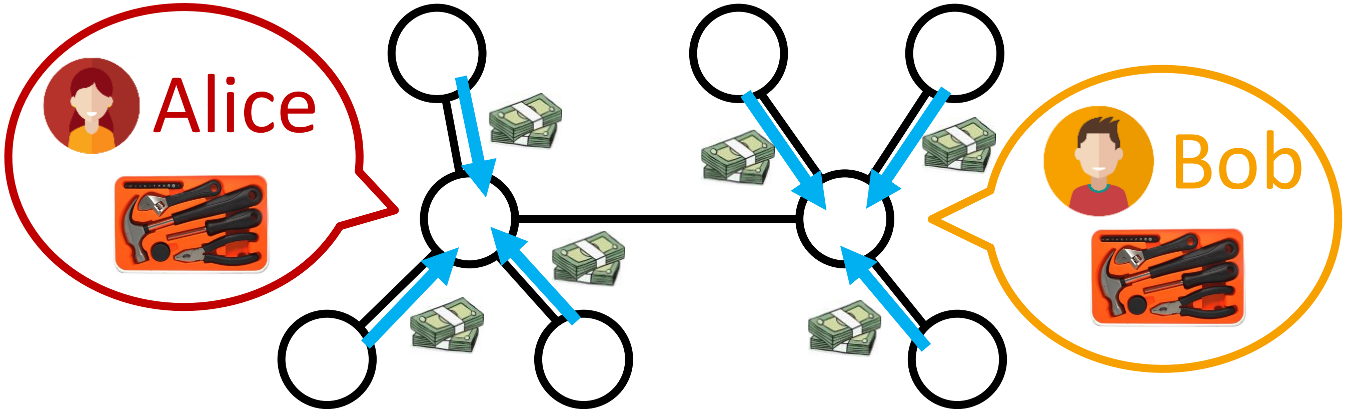
- The answer is **Yes**
  - Alice & Bob: are paid more than the price
  - The others: renting is cheaper than buying



- **Individually optimal**
- **Socially optimal** with minimum (2) purchases

Socially & Individually Optimal

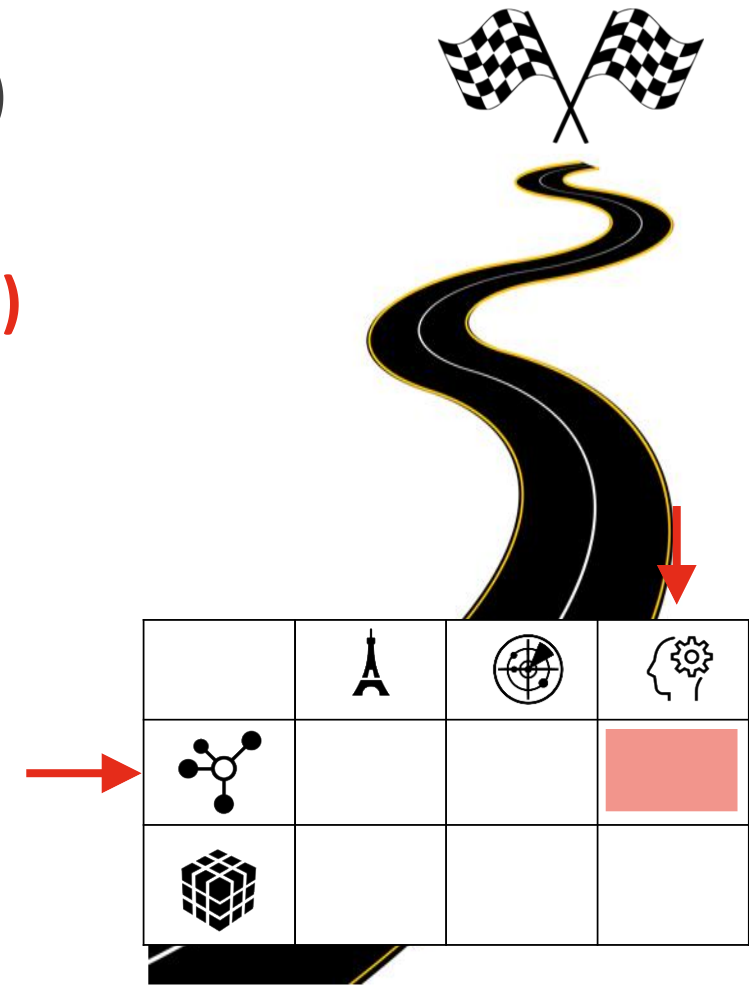
- The answer is **Yes**
  - Alice & Bob: are paid more than the price
  - The others: renting is cheaper than buying



Q1 "How do rental fees affect social inefficiency?"
Q2 "What are the 'socially optimal' rental fees?"

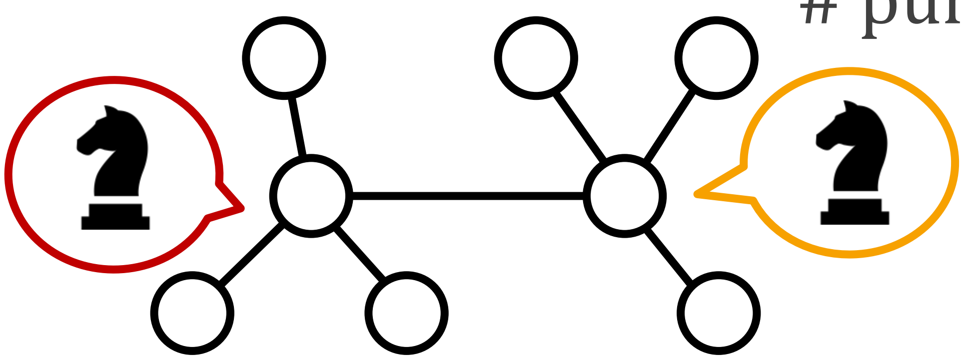
Roadmap

- T1. Structure Analysis (Part 1)
- T2. Anomaly Detection (Part 2)
- T3. Behavior Modeling (Part 3)
  - **T3-1. Modeling Purchases (§14)**
    - Toy Example
    - **Game-theoretic Model <<**
    - Best Rental-fee Algorithm
  - …
- Future Directions
- Conclusions



Formal Game-theoretic Model

- **Players:** nodes in a social network
- **Strategies:**
  - **buy** a good / **rent** a good from a friend with a good
- **Nash Equilibrium (NE):** individually optimal outcome
- **Social Optimum:** socially optimal outcome
- **Inefficiency** of an NE: $\dfrac{\text{\# purchases in the NE}}{\text{\# purchases in a social optimum}}$



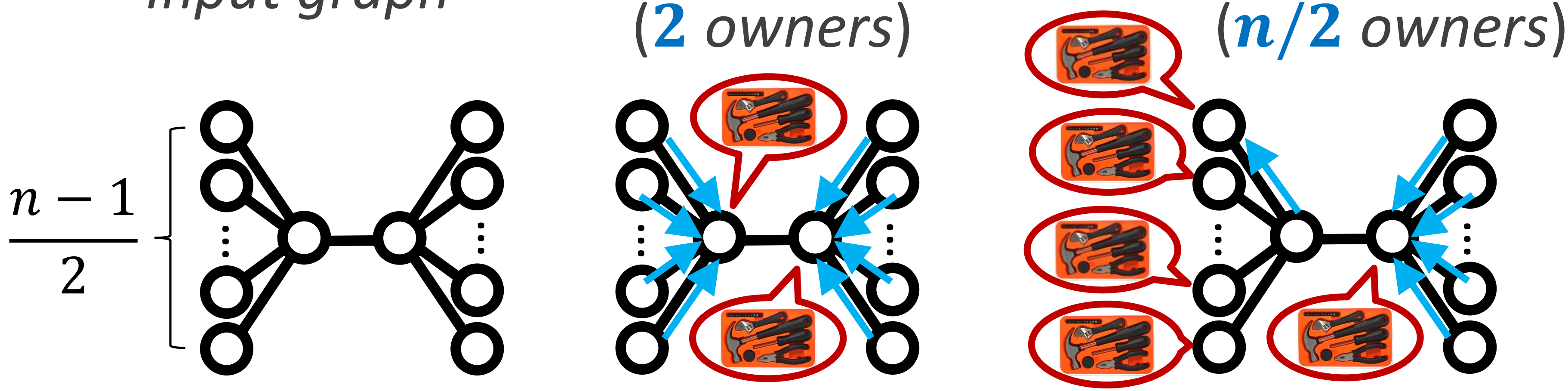
Detail

Inefficiency without Rental Fee

- [THM 1] **Existence of NEs**
  - In ***every*** social network, there exists an NE.

- [THM 2] **Inefficiency without Rental Fee**
  - There exists a social network with $\boldsymbol{n}$ nodes
  - where ***all*** NEs have $\boldsymbol{\Theta}(\boldsymbol{n})$ inefficiency.

Input graph | *Social optimum* (**2** *owners*) | *Best NE* ($\boldsymbol{n/2}$ *owners*)



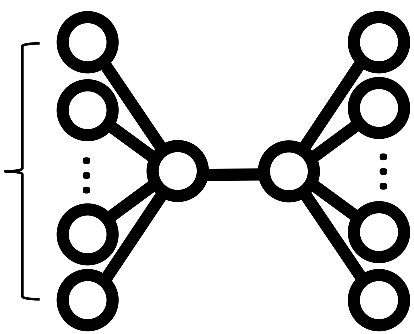
Detail

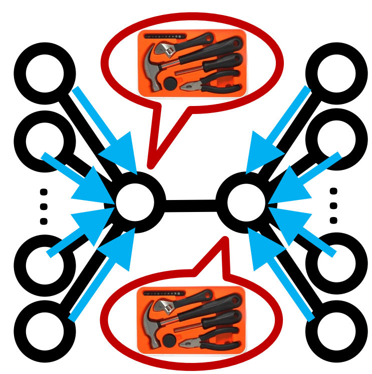
Inefficiency with Rental Fee

- [THM 3] **Inefficiency with Rental Fee**
  - In ***every*** social network,
  - if $\frac{price}{3} < rental\ fee < price$,
  - then, there exists a **socially optimal NE,**
  - otherwise **…**

Input graph

$\frac{n-1}{2}$

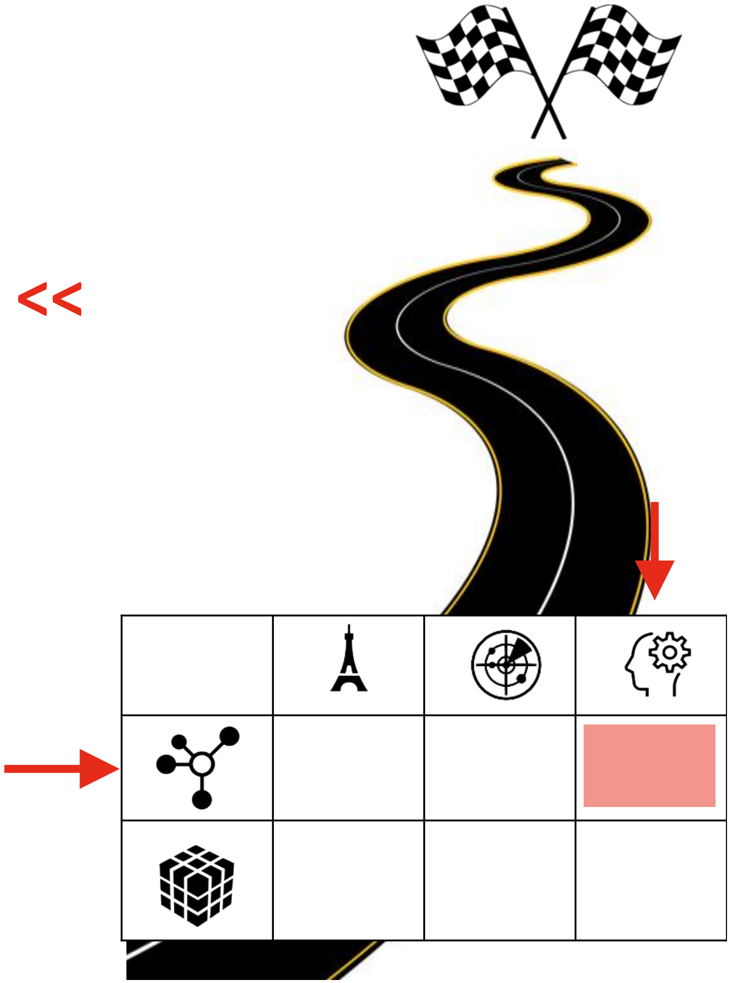




Social optimum
*(**2** owners)*
→ ***NE** with proper rental fee*

Roadmap

- T1. Structure Analysis (Part 1)
- T2. Anomaly Detection (Part 2)
- T3. Behavior Modeling (Part 3)
  - **T3-1. Modeling Purchases (§14) <<**
    - Toy Example
    - Game-theoretic Model
    - **Best Rental-fee Search <<**
  - …
- Future Directions
- Conclusions



Finding Best Rental Fee

- **Given:**
  - a social network
  - a sharable good with price p
- **Find:** a range of rental fees
- **To Minimize:** inefficiencies of NEs

Detail

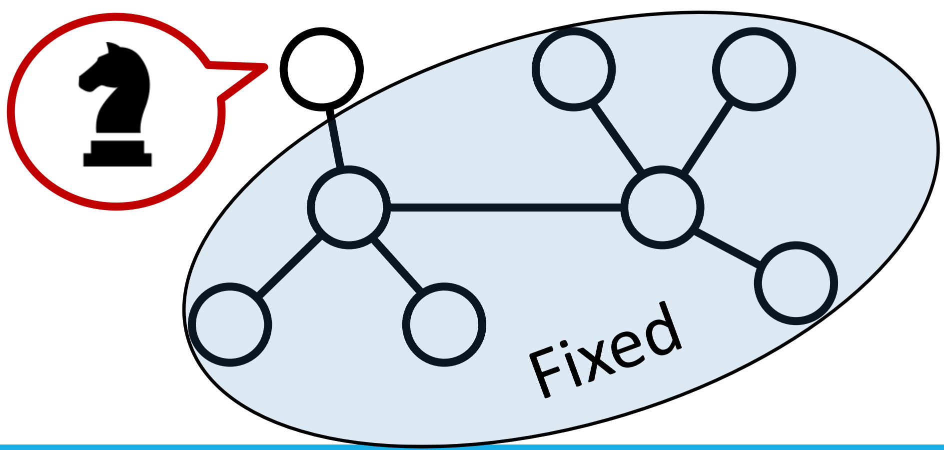
Searching NEs (SGG-Nash)

- **Linear-time** algorithm for searching NEs

randomly initialize strategies

repeat until an NE is reached
- *for each* node
  - optimize its strategy while fixing the others'

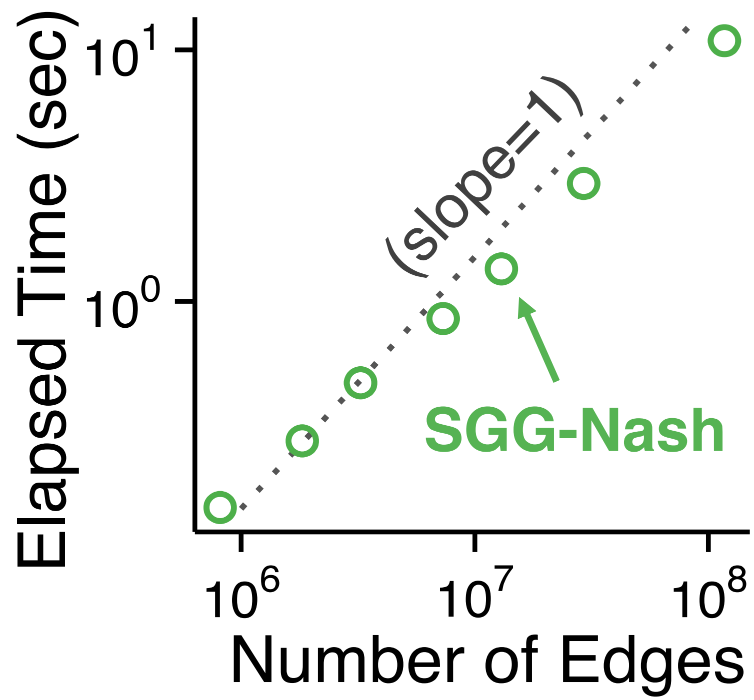
- Gives different NEs depending on initial strategies



[THM 4] Convergence
In ***every*** social network, an NE is reached within **3 *iterations.***

Scalability of SGG-Nash

- SGG-Nash is linear in the number of edges

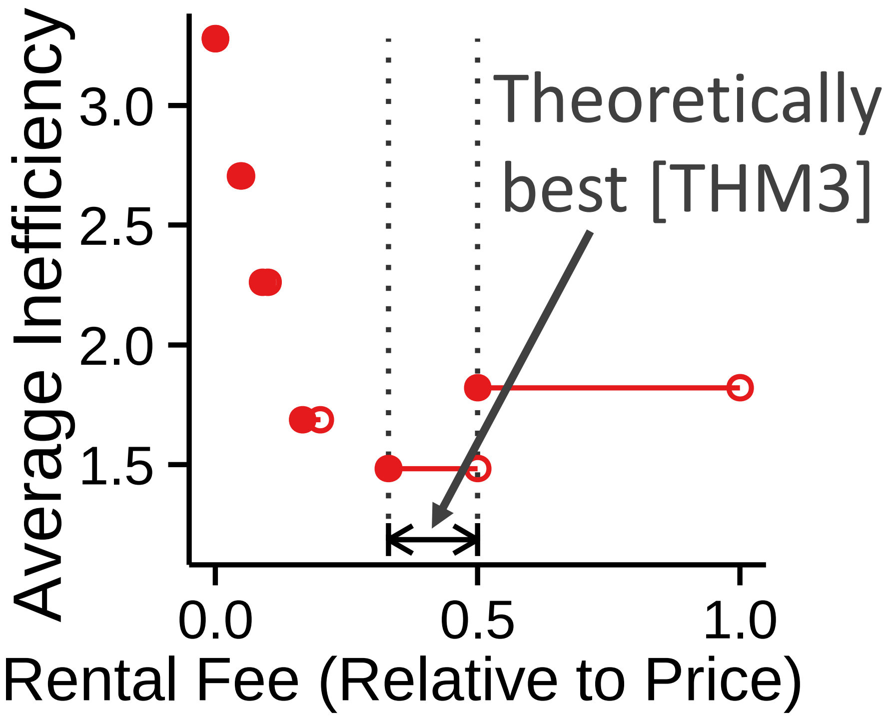


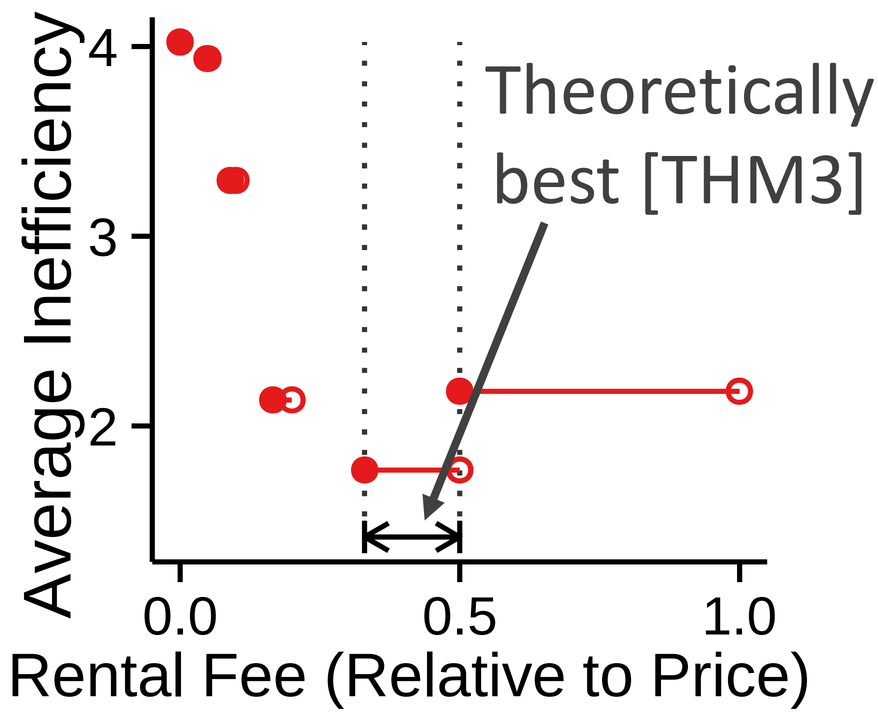
- Dataset: orkut

Best Rental Fee in Real Graphs

- Datasets:
- Inefficiency is minimized consistently when

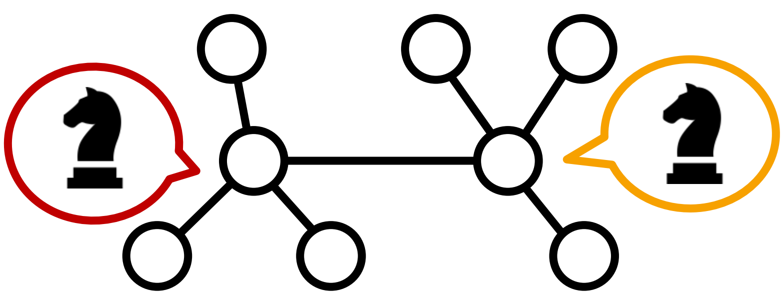
$$price/3 < rental\ fee < price/2$$

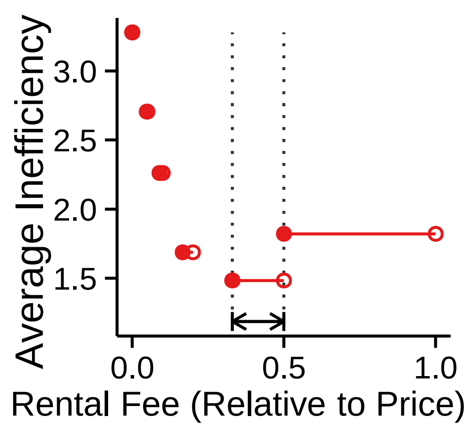


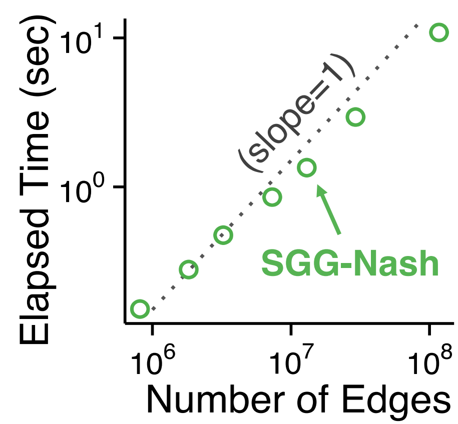


Summary of §14

- **Game-theoretic Model:** Sharable good game
- **Theoretical Analysis:**
  - Existence of NEs
  - Inefficiency of NEs
- **Algorithm:** for linear-time NE search
- **Suggestion**: "socially optimal" rental fees







Contributions and Impact (Part 3)

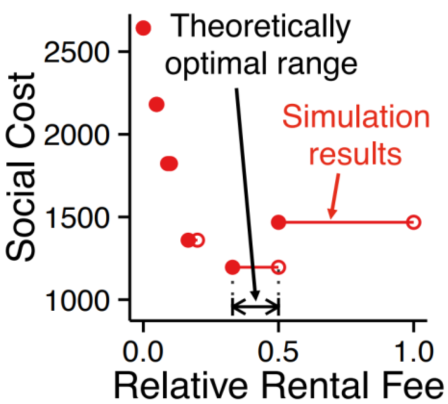
Tools for **purchase modeling** [IJCAI17]

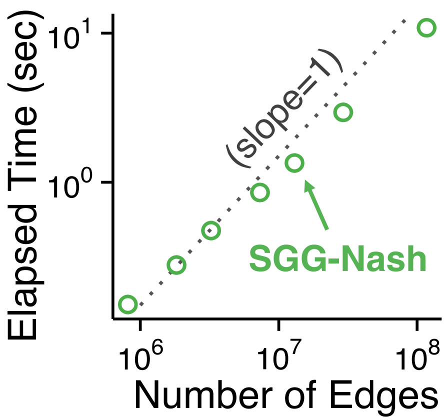
- Suggest *'socially optimal'* rental fees
- Media: NewScientist

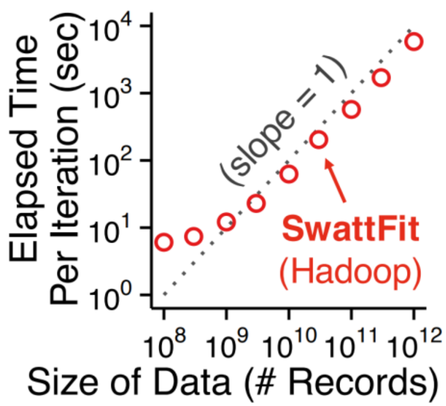

NewScientist
DAILY NEWS 26 May 2017
Game theory says you should charge your friends to borrow things
By Timothy Revell

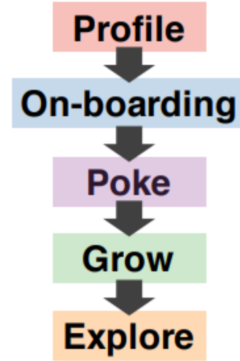
Tools for **progression modeling** [WWW18]

- Scale to datasets with a *trillion records*
- Real-world usage: LinkedIn









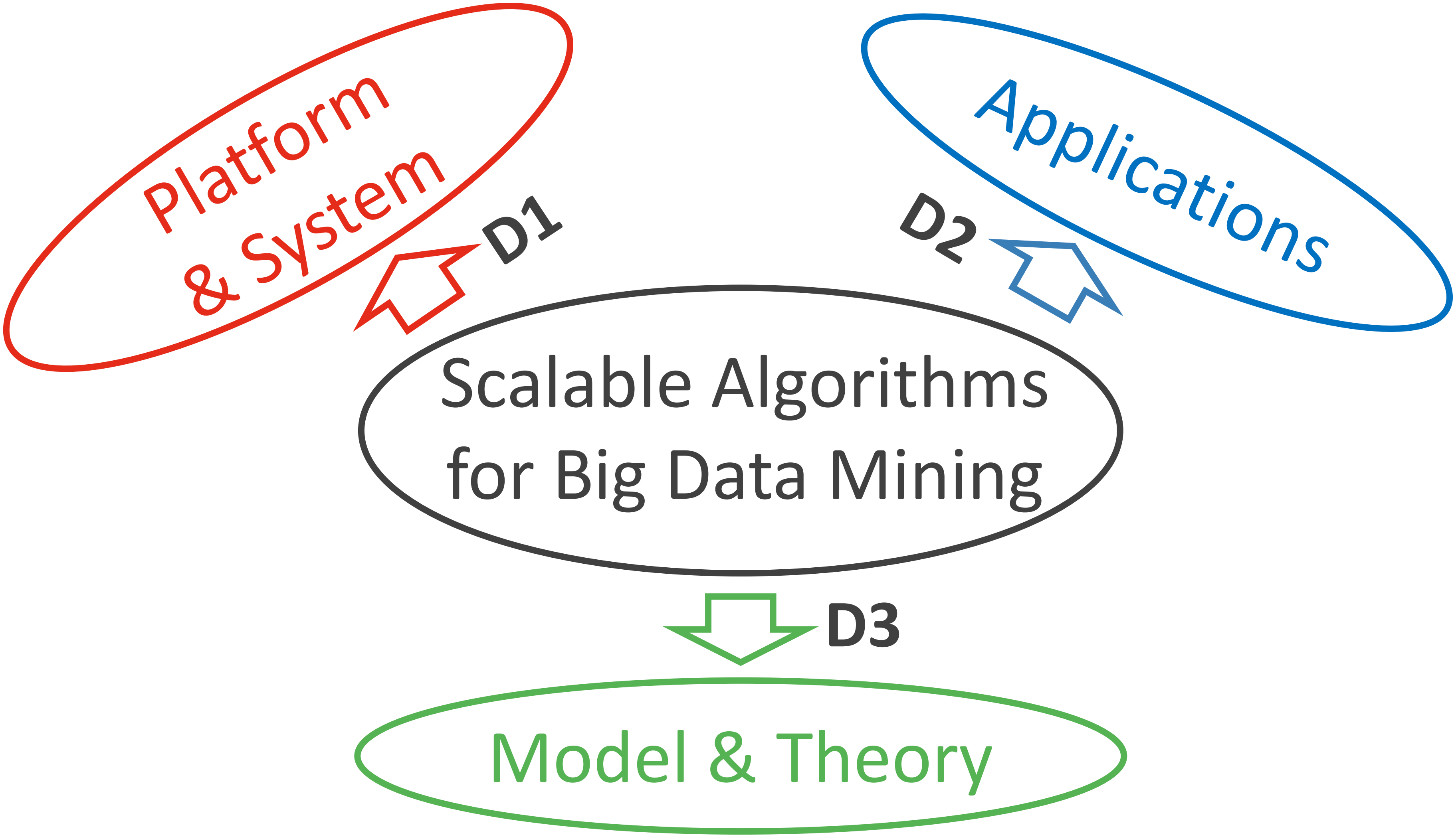
Organization of the Thesis (Recall)

	Part1. Structure Analysis	**Part2.** Anomaly Detection	**Part3.** Behavior Modeling
Graphs	Triangle Count (§§ 3-6)	Anomalous Subgraph (§ 9)	Purchase Behavior (§ 14)
	Summarization (§ 7)		
Tensors	Summarization (§ 8)	Dense Subtensors (§§ 10-13)	Progression (§ 15)

Roadmap

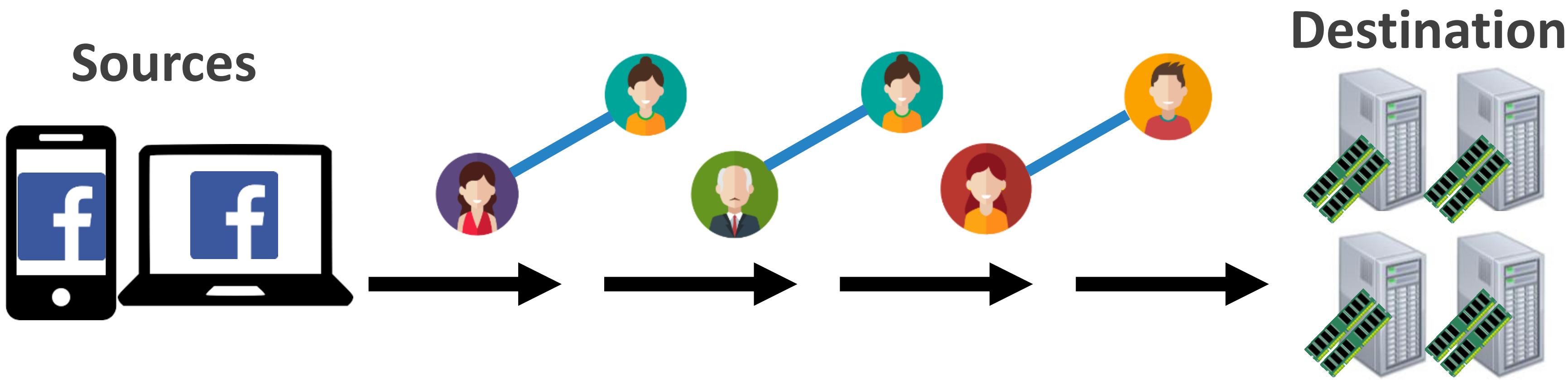
- T1. Structure Analysis (Part 1)
- T2. Anomaly Detection (Part 2)
- T3. Behavior Modeling (Part 3)
- **Future Directions <<**
- Conclusions

Vision: Algorithms for “Big Data”



D1: Distributed Graph Stream Processing

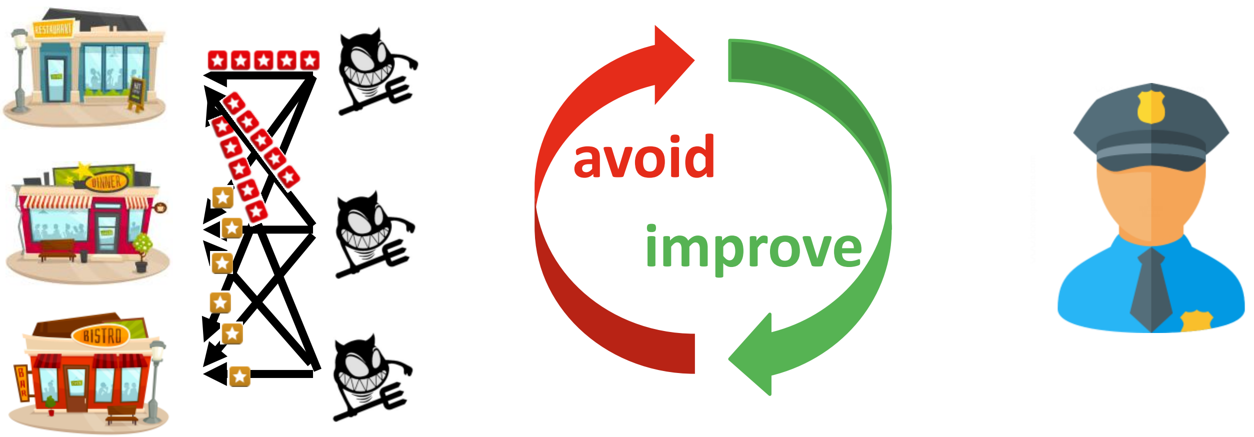
"How to analyze ***large dynamic graphs*** on a ***cluster*** of machines?"



Current		Short Term		Mid Term		Long Term
Triangle Counting (§5)	➡	More Graph Mining Tasks	➡	Programming Model (Generalization)	➡	System / Platform

D2: Detecting Adversarial Anomalies

Anomalies / Fraudsters **Detection System**

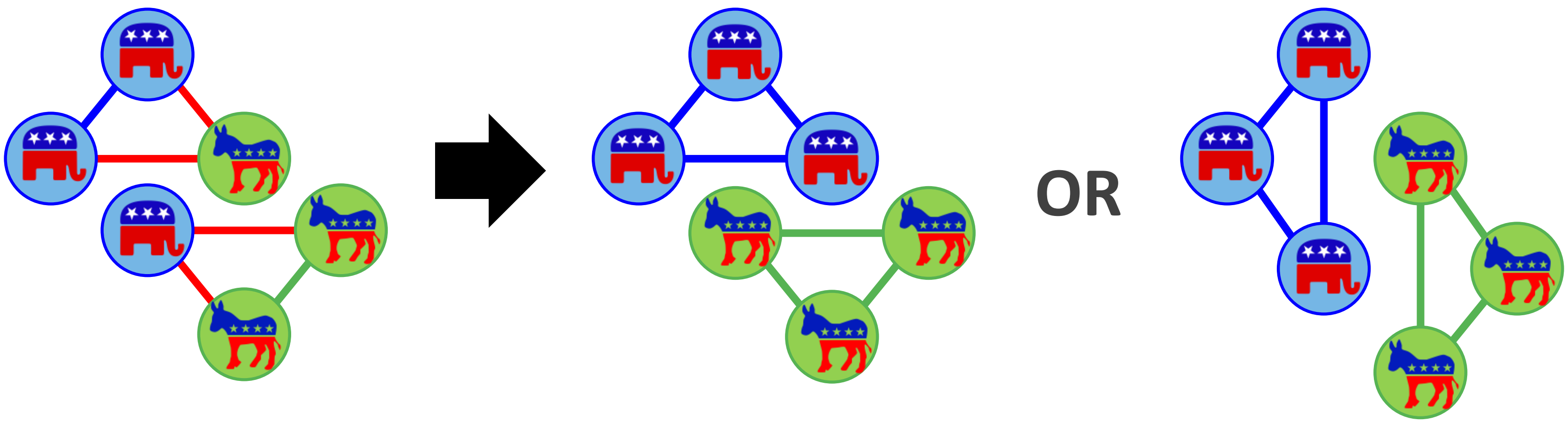


Current	Short Term	Mid Term	Long Term
Algorithms for Static Anomalies	Profits of Anomalies	Cost to Avoid Algorithms	Algorithms Costly to Avoid

D3: Co-Evolution of Beliefs and Graphs

*"How to model the **co-evolution** of **nodes' beliefs** and **edges**?"*

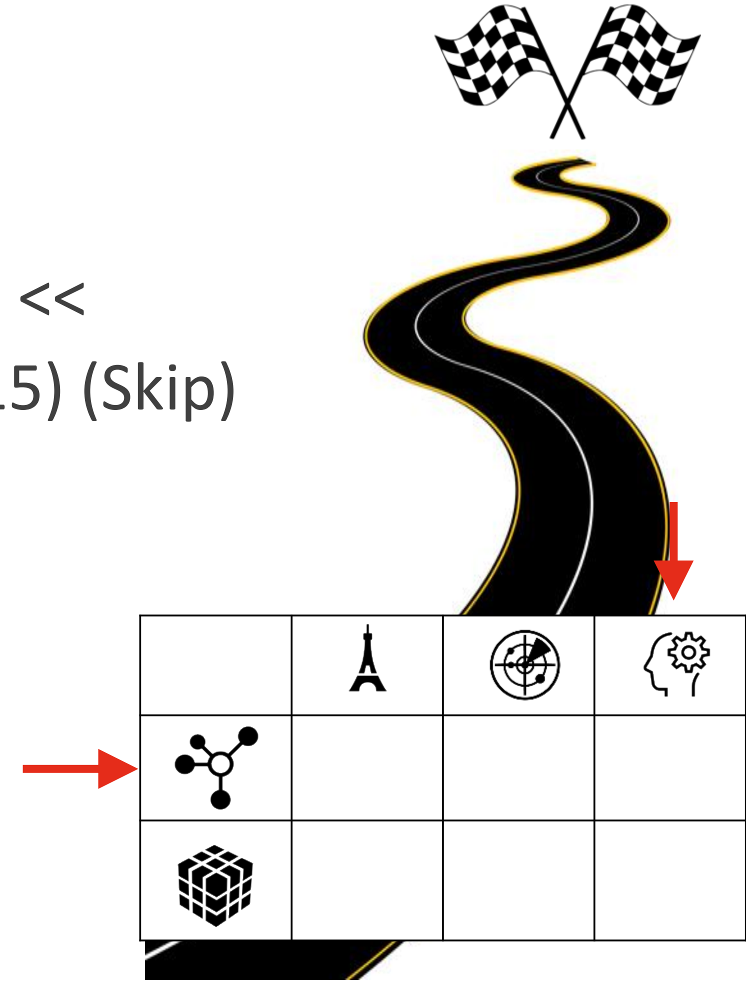
Change of ***Beliefs*** Change of ***Edges***



Current	Short Term	Mid Term	Long Term
Regression Model [PKDD18]	Game Theory / Nash Equilibrium	Prediction Algorithms	Reducing Polarization

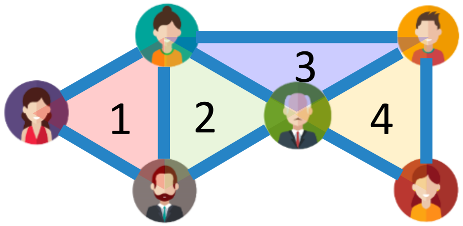
Roadmap

- T1. Structure Analysis (Part 1)
- T2. Anomaly Detection (Part 2)
- T3. Behavior Modeling (Part 3)
  - T3-1. Modeling Purchases (§14) <<
  - T3-2. Modeling Progression (§ 15) (Skip)
- Future Directions
- **Conclusions <<**



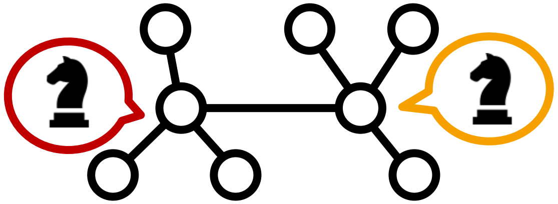
Conclusion

- **Goal**: “To fully understand and utilize *large dynamic graphs* and *tensors*”
- **Contributions**: developing *scalable algorithms* for



T1. Structure Analysis (Part 1)

T2. Anomaly Detection (Part 2)



T3. Behavior Modeling (Part 3)

- **Impact**: Massachusetts Institute of Technology, NAVER, NewScientist, LinkedIn

github.com/kijungs

References

[1] **K Shin**, B Hooi, and C Faloutsos, “M-Zoom: Fast Dense-Block Detection in Tensors with Quality Guarantees”, **ECML/PKDD 2016** (§11)

[2] **K Shin**, T Eliassi-Rad, and C Faloutsos, “CoreScope: Graph Mining Using k-Core Analysis - Patterns, Anomalies and Algorithms”, **ICDM 2016** (§9)

[3] **K Shin**, “Mining Large Dynamic Graphs and Tensors for Accurate Triangle Counting in Real Graph Streams”, **ICDM 2017** (§4)

[4] **K Shin**, B Hooi, J Kim, and C Faloutsos, “D-Cube: Dense-Block Detection in Terabyte-Scale Tensors”, **WSDM 2017** (§12)

[5] **K Shin**, E Lee, D Eswaran, and AD. Procaccia, “Why You Should Charge Your Friends for Borrowing Your Stuff”, **IJCAI 2017** (§14)

[6] **K Shin**, B Hooi, J Kim, and C Faloutsos, “DenseAlert: Incremental Dense-Subtensor Detection in Tensor Streams”, **KDD 2017** (§13)

[7] J Oh, **K Shin**, EE Papalexakis, C Faloutsos, and Hwanjo Yu, “S-HOT: Scalable High-Order Tucker Decomposition”, **WSDM 2017** (§8)

References (cont.)

[8] **K Shin**, B Hooi, and C Faloutsos, “Fast, Accurate and Flexible Algorithms for Dense Subtensor Mining”, **TKDD 2018** (§11)

[9] **K Shin**, M Shafiei, M Kim, A Jain, and H Raghavan, “Discovering Progression Stages in Trillion-Scale Behavior Logs**”** **WWW 2018** (§14)

[10] **K Shin**, T Eliassi-Rad, and C Faloutsos, “Patterns and Anomalies in k-Cores of Real-world Graphs with Applications”, **KAIS 2018** (§9)

[11] **K Shin**, M Hammoud, E Lee, J Oh, and C Faloutsos. “Tri-fly: Distributed estimation of global and local triangle counts in graph streams**”** **PAKDD 2018** (§5)

[12] **K Shin**, J Kim, B Hooi, and C Faloutsos, “Think before You Discard: Accurate Triangle Counting in Graph Streams with Deletions”, **ECML/PKDD 2018** (§6)

[13] **K Shin**, A Ghoting, M Kim and H Raghavan, “SWeG: Lossless and Lossy Summarization of Web-Scale Graphs”, **WWW 2019** (§7)

Thank You!

- Sponsors:
- Admins:

- Collaborators:

Thank You!

- Homepage (Software & Datasets):
  http://www.cs.cmu.edu/~kijungs/defense/