Enabling Scalable Social Group Analytics via Hypergraph Analysis Systems

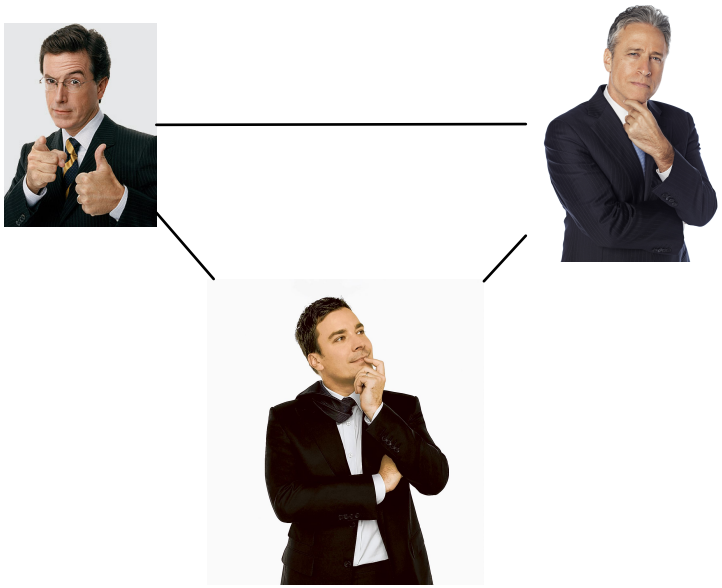
Benjamin Heintz, Abhishek Chandra
University of Minnesota

Big Social Data

- Rapid growth in *social* data
  - Likes
  - Tweets
  - Publications
- Transform into *knowledge*
  - Importance / centrality / influence
  - Community detection
  - Information flow

State of the Art: *Graph* Systems

- *Pairwise* interactions
  - Writing a paper
  - Attending an event
  - Appearing on a TV show
- Wave of systems work
  - Pregel
  - GraphLab
  - GraphX



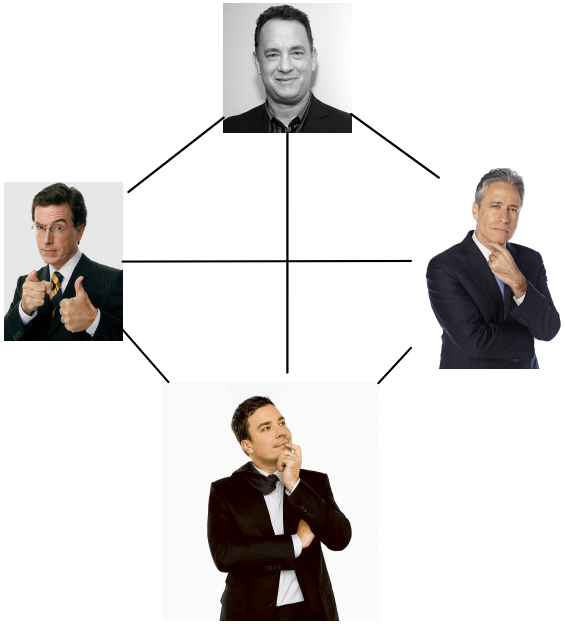
Vertex- or edge-centric programming models

What about *Groups*?

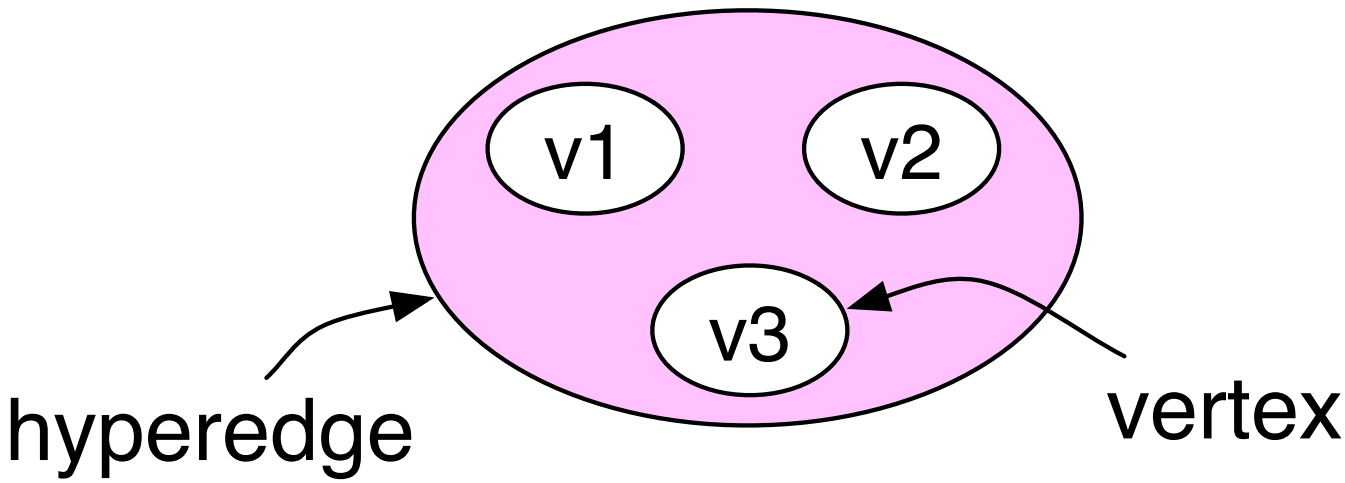
- Interactions: more than just ~~pairs~~
  - Papers
  - Photos
  - Events
- How to model groups?

Groups are the basis for many social interactions.

Graphs poorly model groups.

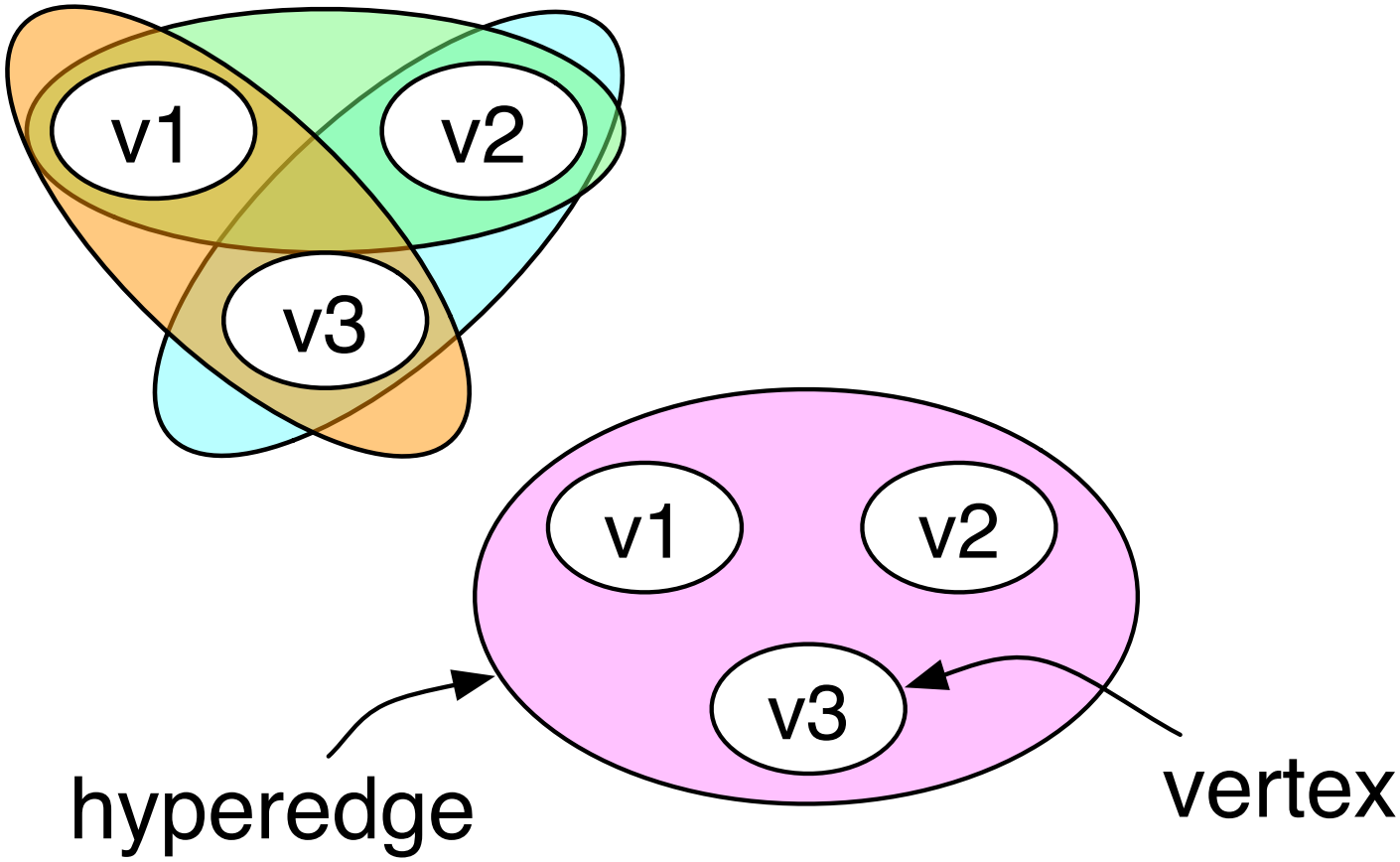


Hypergraphs Model Groups



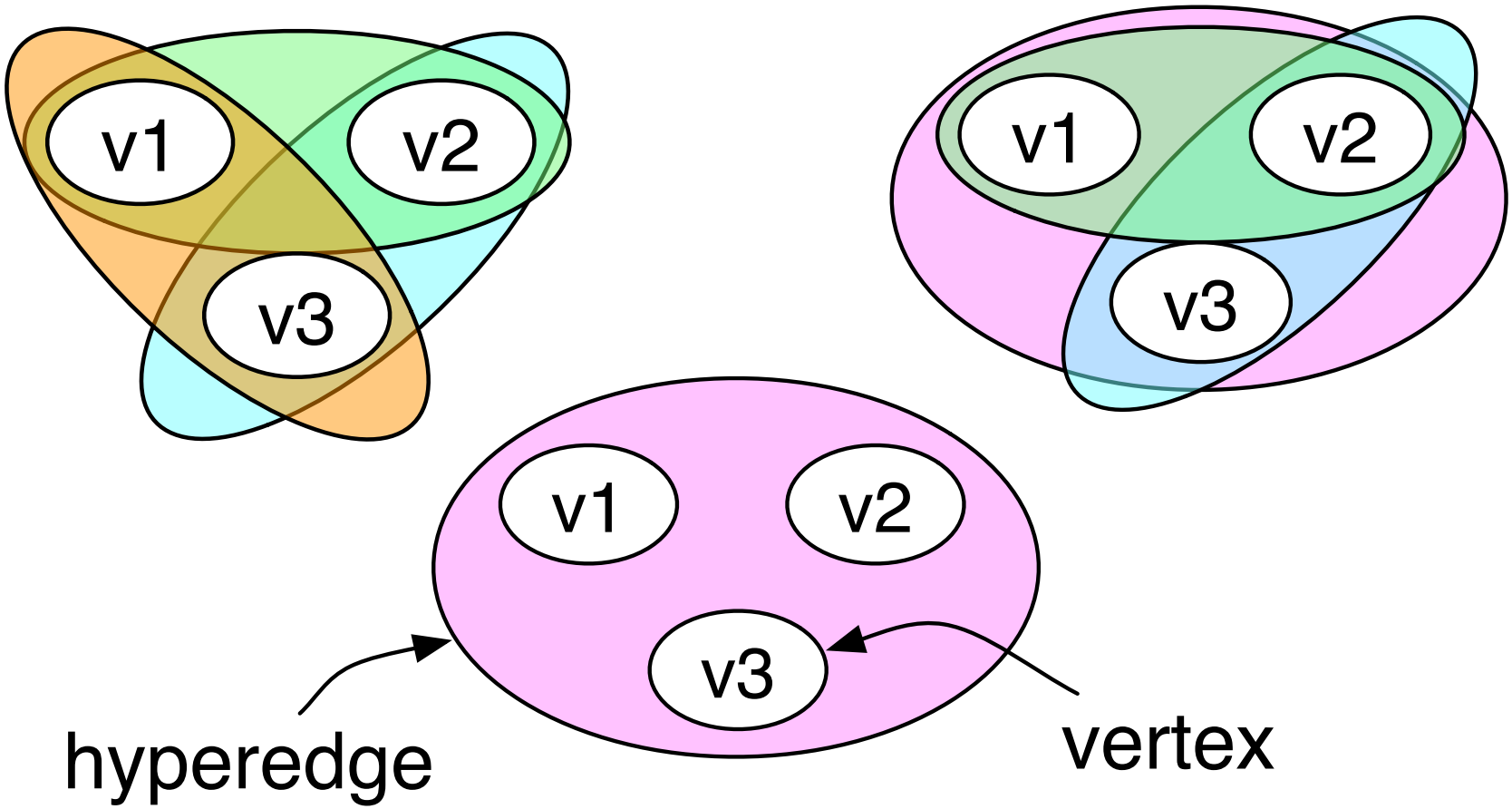
$$H = (V, E)$$

Hypergraphs Model Groups



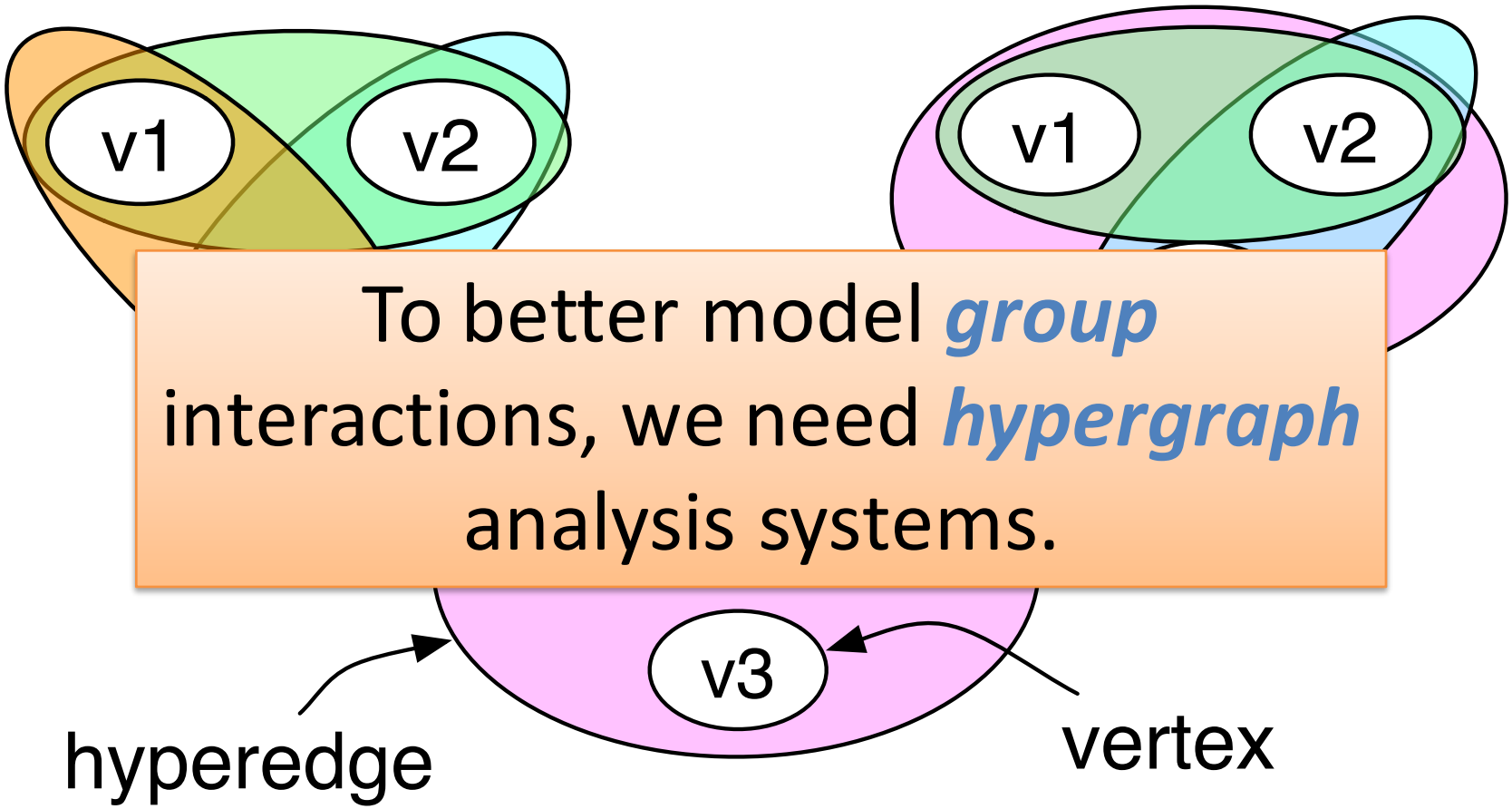
$$H = (V, E)$$

Hypergraphs Model Groups



$H = (V, E)$

Hypergraphs Model Groups



To better model ***group*** interactions, we need ***hypergraph*** analysis systems.

$H = (V, E)$

Alternative Approaches

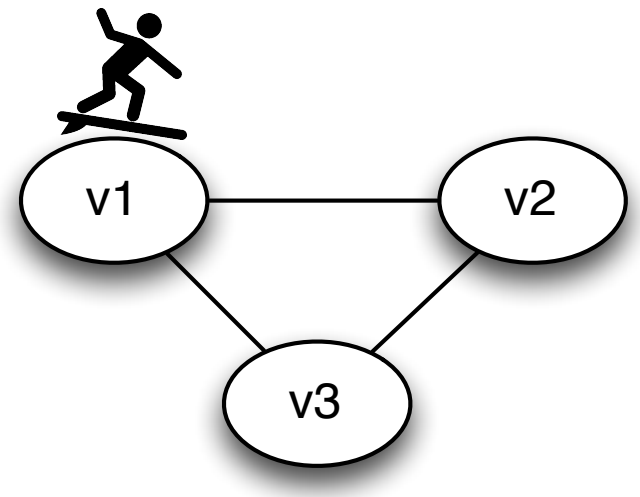
Approach	Pros	Cons
Affiliation network	Models groups explicitly	Tedious to implement; wrong level of abstraction
1-mode graph projection	No need for hypergraph systems	Tedious; large size; not always applicable
Hypergraphs	Level of abstraction matches the domain	*Requires hypergraph systems!*

Roadmap

- Motivation
- ***Interface***
- Implementation
- Evaluation
- Discussion

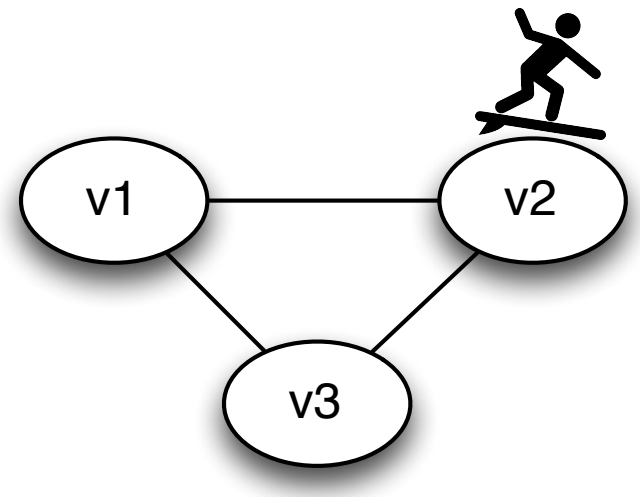
Example: PageRank

- Rank vertices by importance



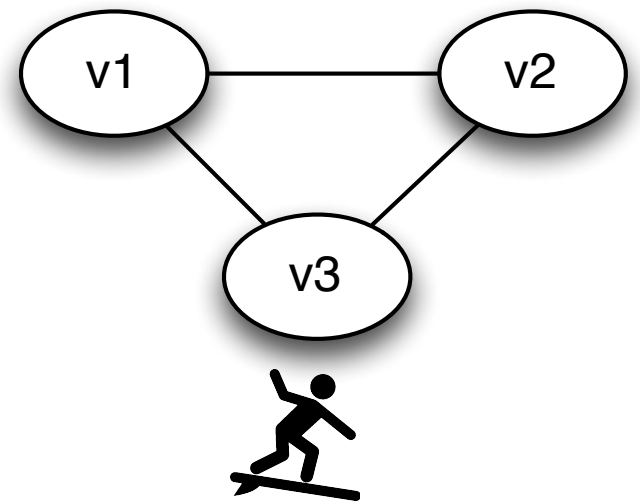
Example: PageRank

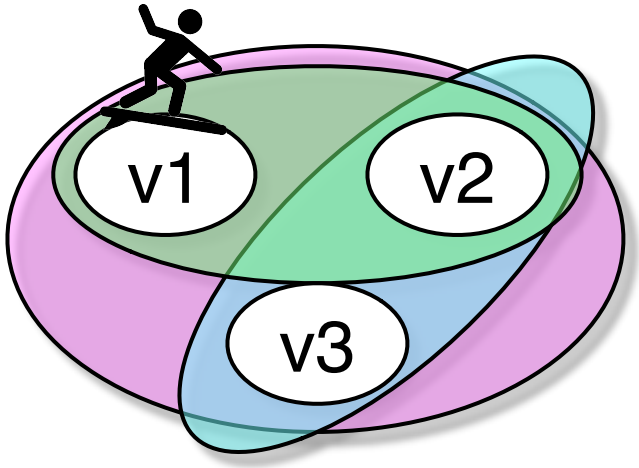
- Rank vertices by importance



Example: PageRank

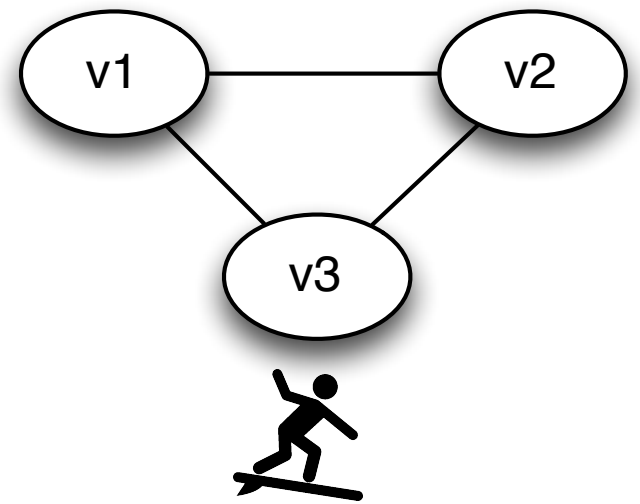
- Rank vertices by importance
- Extension: rank *hyperedges* too

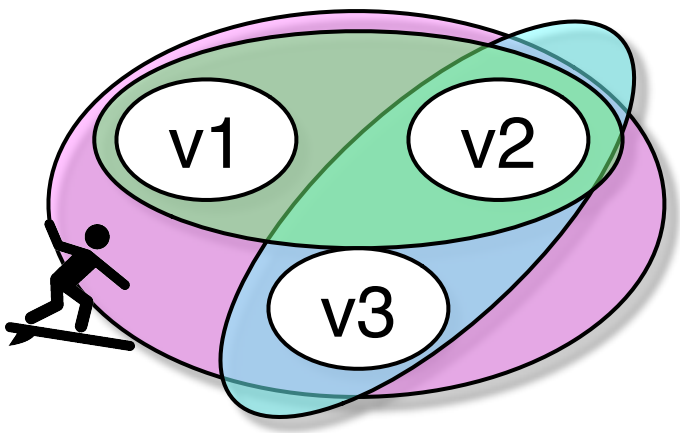




Example: PageRank

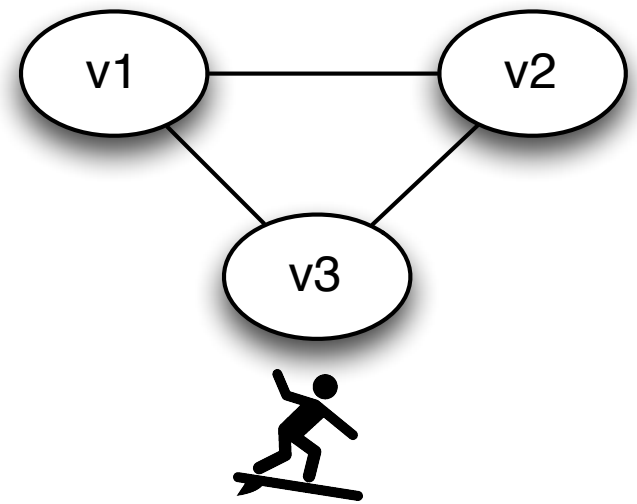
- Rank vertices by importance
- Extension: rank *hyperedges* too

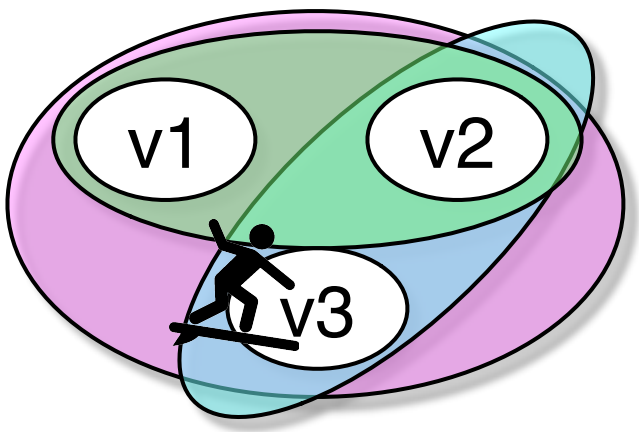




Example: PageRank

- Rank vertices by importance
- Extension: rank *hyperedges* too
- Extension: arbitrary hyperedge behavior



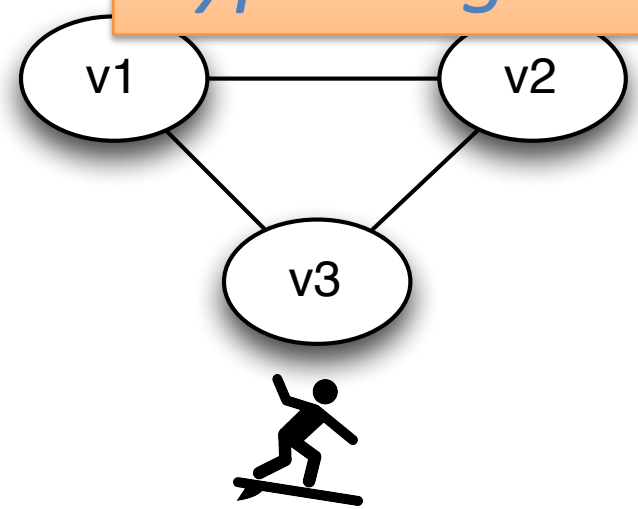


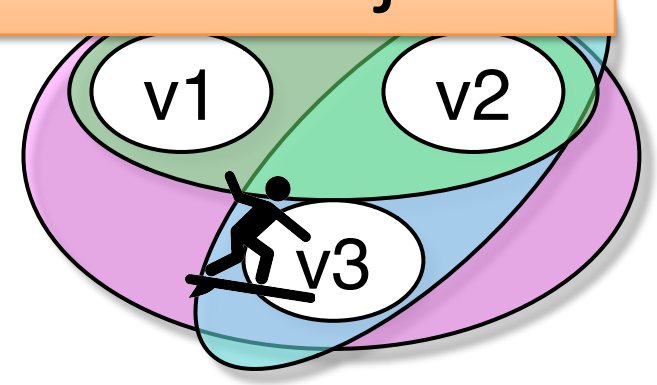
Example: PageRank

- Rank vertices by importance
- Extension: rank *hyperedges* too
- Ext

or

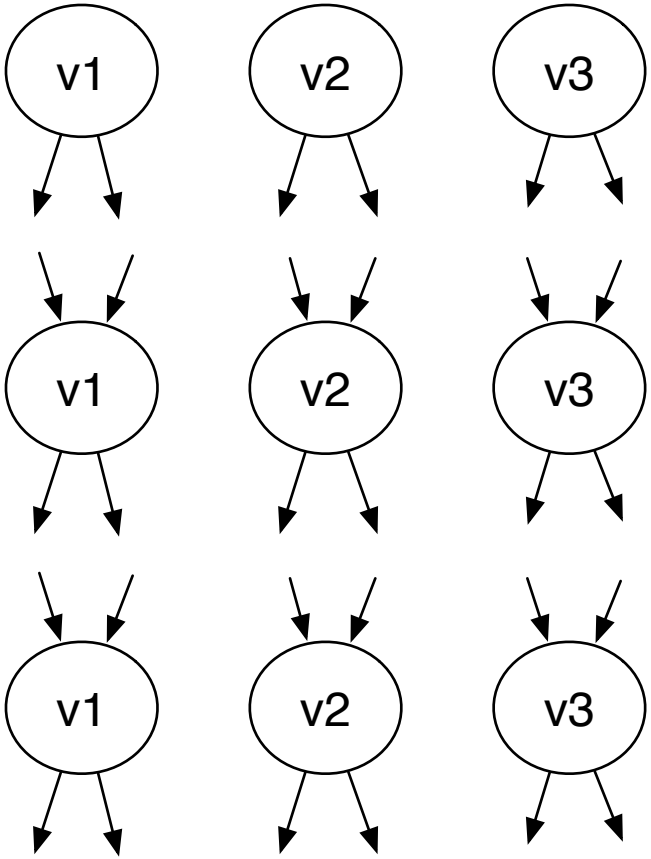
In a hypergraph system,
hyperedges are first-class objects.





Iterative Computation

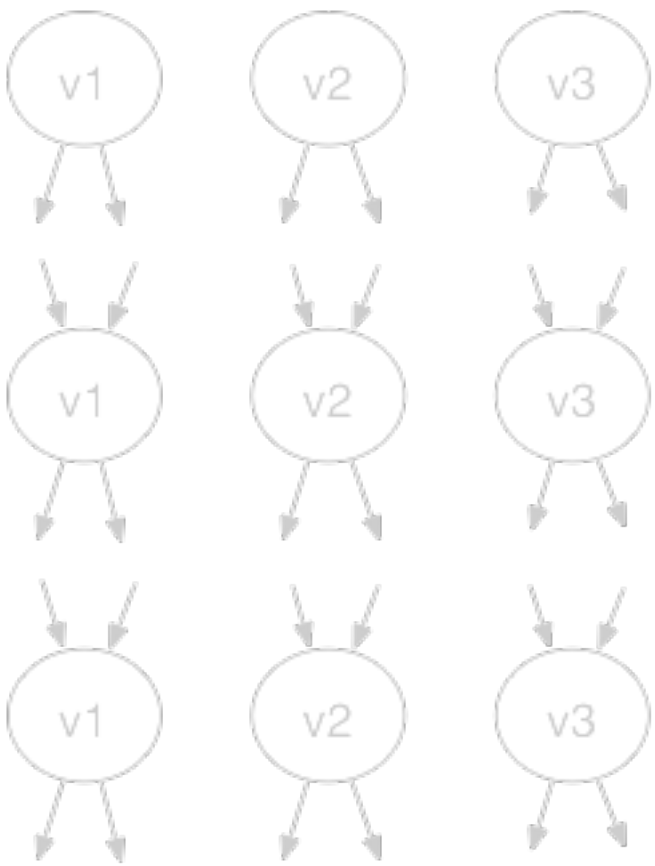
Graph



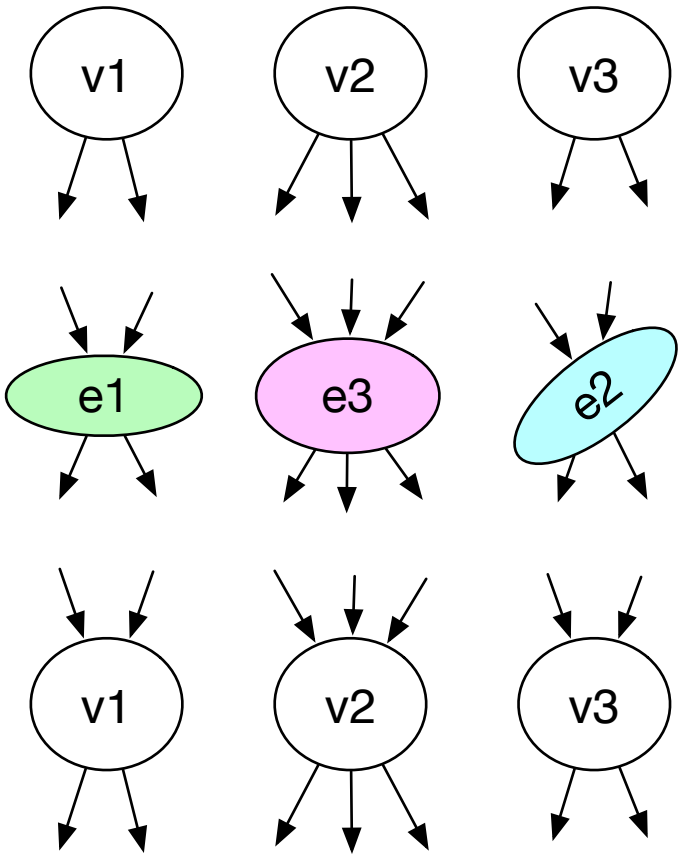
Hypergraph

Iterative Computation

Graph



Hypergraph



A Hypergraph API

```
trait HyperGraph[HVD, HED] {
  def compute[ToE, ToV](
    maxIters: Int,
    initialMsg: ToV,
    hvProgram: Program[HVD, ToV, ToE],
    heProgram: Program[HED, ToE, ToV])
    : HyperGraph[HVD, HED]
}
```

Iterative computation

```
object HyperGraph {
  trait Program[A, InMsg, OutMsg] {
    def messageCombiner: MessageCombiner[OutMsg]
    def procedure: Procedure[A, InMsg, OutMsg]
  }

  type MessageCombiner[Msg] = (Msg, Msg) => Msg

  type Procedure[A, InMsg, OutMsg] =
    (Int, NodeId, A, InMsg, Context[A, OutMsg]) => Unit
```

Vertex ***and hyperedge*** programs

```
  trait Context[A, OutMsg] {
    def become(attr: A): Unit
    def send(msgF: NodeId => OutMsg,
             to: Recipients): Unit
  }
}
```

Messages to neighbors

Roadmap

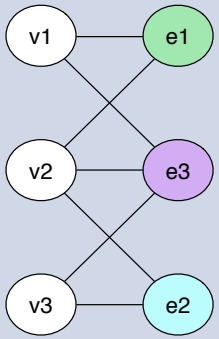
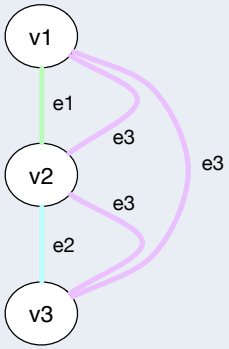
- Motivation
- Interface
- ***Implementation***
- Evaluation
- Discussion

Implementation

- Challenges
  - Representation
  - Partitioning

- Initial approach
  - Build upon existing graph systems

Hypergraph Representations

Representation	Pros	Cons
Bipartite Graph 	Portable to any graph system	Obscures hyperedge / vertex differences
Multigraph 	Exploits hyperedge / vertex differences	Size overhead

Hypergraph Partitioning

- Partition the underlying *graph*
  - Cut vertices, cut edges, both

- Or use *hypergraph-aware* partitioning
  - Differentiate hyperedges and vertices

Roadmap

- Motivation
- Interface
- Implementation
- ***Evaluation***
- Discussion

Datasets

Dataset	Vertices	Hyperedges	Bipartite Edges	1-mode Projection Edges
DBLP	952,115 authors	916,947 collaborations	2,768,930	21,592,883
Friendster	7,944,949 users	1,620,991 communities	23,479,217	***> 15.1 B***

Datasets

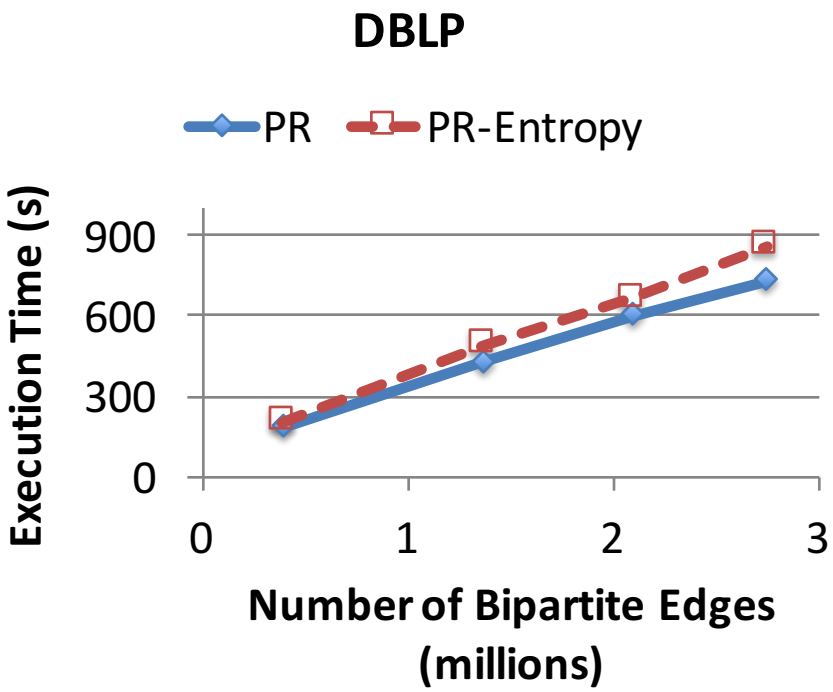
Dataset	Vertices	Hyperedges	Bipartite Edges	1-mode Projection Edges
DBLP	952,115	916,947	2,768,930	21,592,883
Friendster	7,944,949 users	1,620,991 communities	23,479,217	***> 15.1 B***

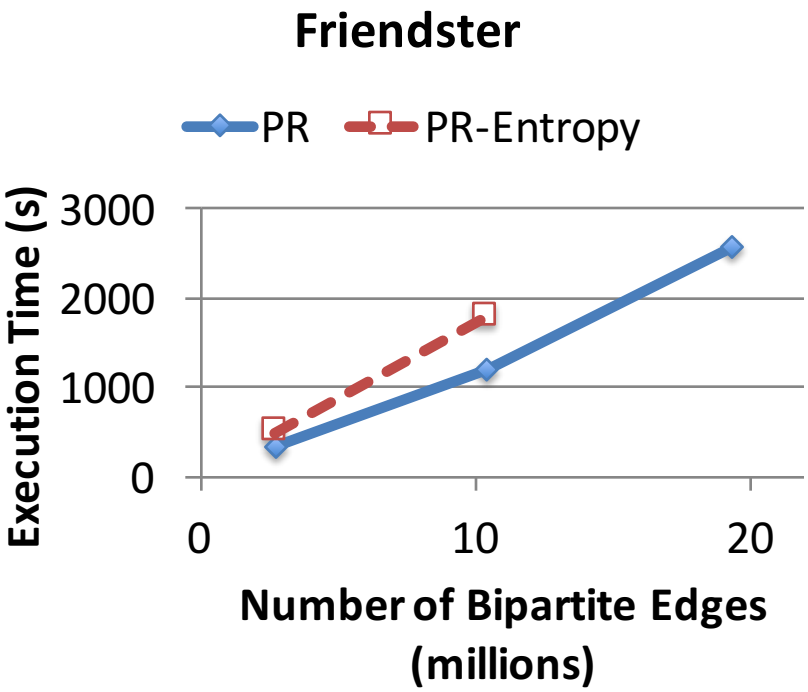
Real-world hypergraphs exhibit *diverse* characteristics.

Prototype

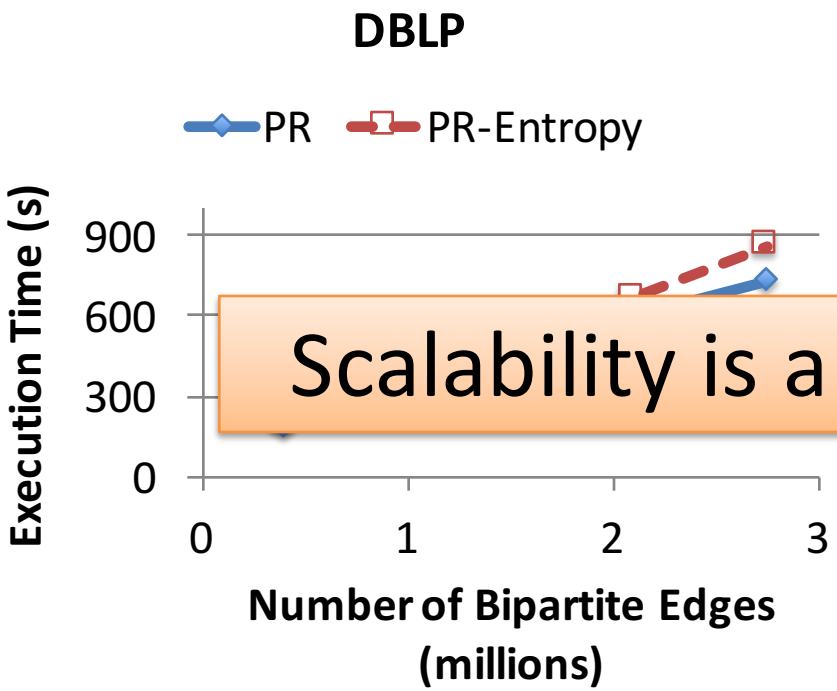
- *Proof-of-concept* prototype
- Implemented on Apache Spark GraphX 1.2.1
- Run on shared 6-node cluster (2x6-core, 24GB RAM each)
- Using bipartite graph representation

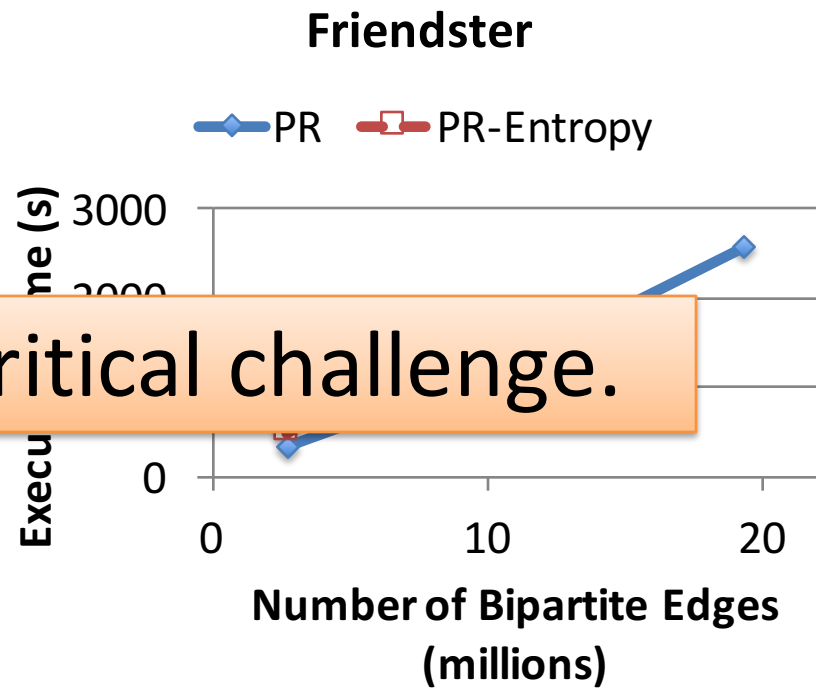
Experimental Results (1/2)





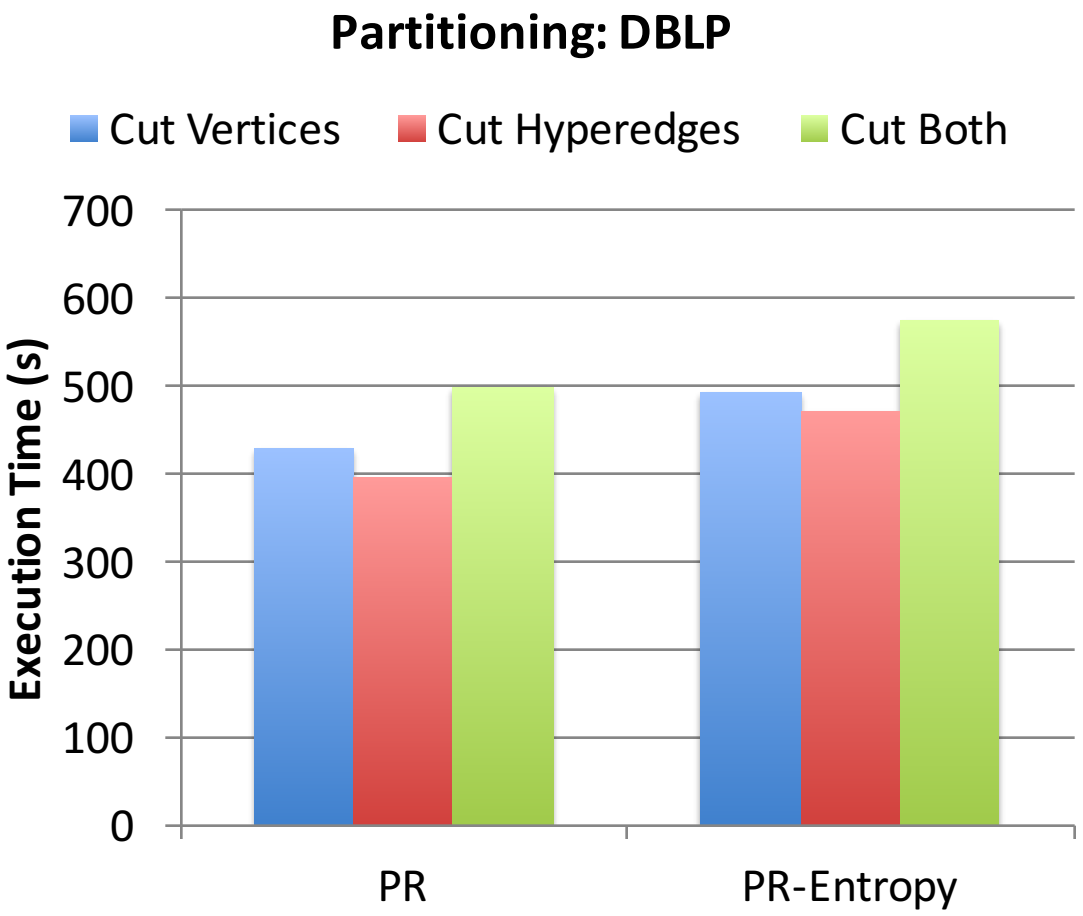
Experimental Results (1/2)



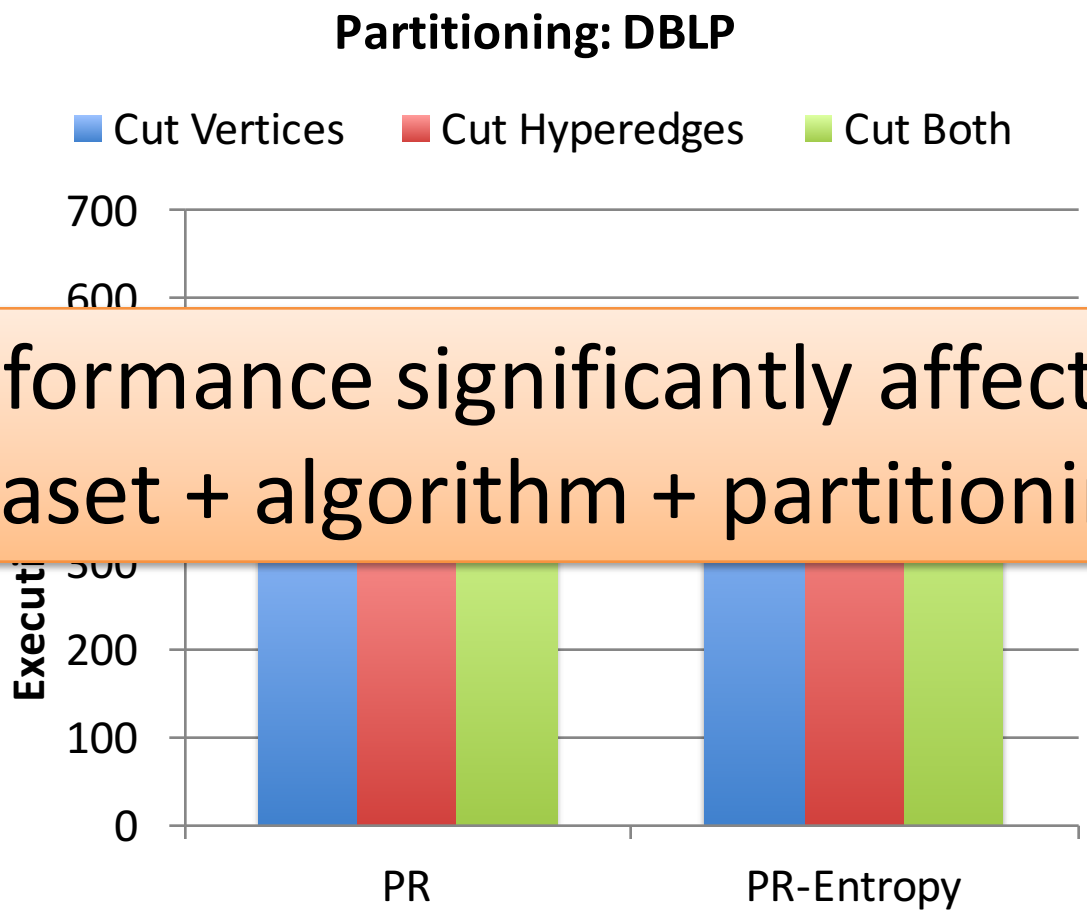


Scalability is a critical challenge.

Experimental Results (2/2)



Experimental Results (2/2)



Performance significantly affected by dataset + algorithm + partitioning

Conclusion

- Social data → knowledge
- State of the art: graphs
- Need to model groups
- Hypergraphs are the right abstraction
- System challenges

Roadmap

- Motivation
- Interface
- Implementation
- Evaluation
- ***Discussion***

Discussion

- *Programming Model & API*
  - Synchronous vs. asynchronous
  - Directed, temporal, …
- *Implementation*
  - Build on graph systems or from scratch?
  - Representations, partitioning techniques

Thank you!

dcsg.cs.umn.edu