Enabling Scalable Social Group Analytics via Hypergraph Analysis Systems

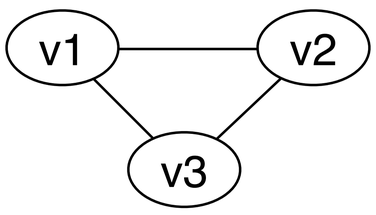
Benjamin Heintz, Abhishek Chandra
University of Minnesota
Minneapolis, MN
{heintz, chandra}@cs.umn.edu

1. Motivation

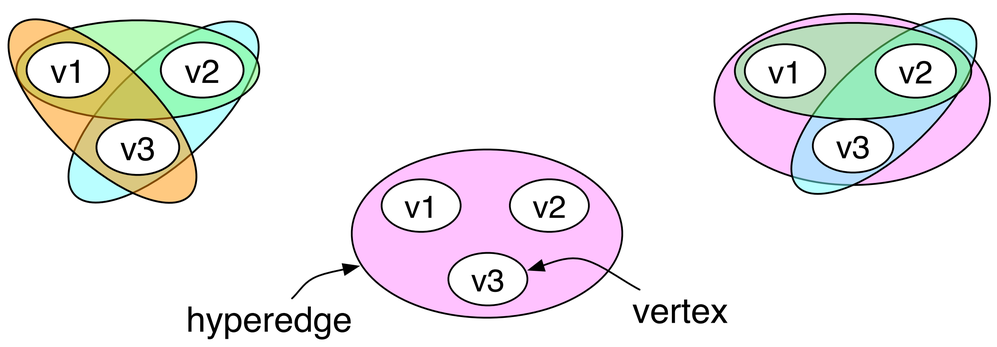
Rapid growth of **social** data
- Likes
- Tweets
- Publications

Need to transform data into **knowledge**
- Importance / centrality / influence
- Community detection
- Shortest paths
- Information flow

State of the art: systems for **graph** computation
- Pregel
- GraphLab (Dato)
- Apache Spark GraphX
- Problem: poor model for **groups**



To better model social **group** structure and behavior, we need **hypergraph** computing systems.



2. 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]
}

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

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

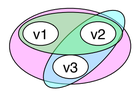
Iterative computation

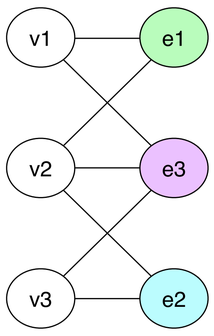
Vertex ***and hyperedge*** programs

Message flow:
- vertex → hyperedge
- hyperedge → vertex

3. Implementation Challenges

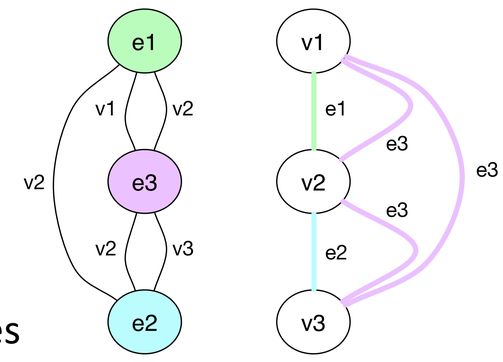
How to **represent** the hypergraph?





Bipartite graph
- ⊘ Obscures differences between hyperedges, vertices
- ✓ Portable to any graph system

Multigraph
- ⊘ Limited system support
- ⊘ Hard to implement with existing APIs
- ✓ Can exploit differences between vertices, hyperedges



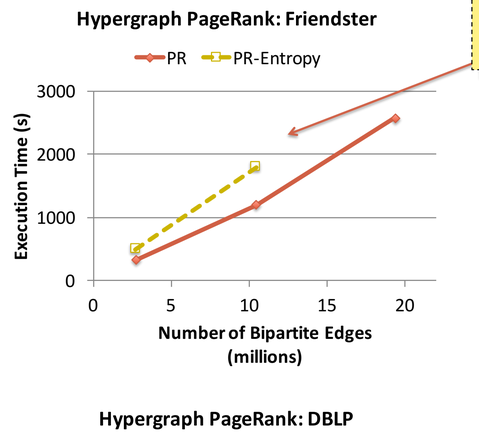
Closely related questions, constrained by underlying platform

How to **partition** the representation?
- Critical for distributed computation
- Graph: cut edges, cut vertices (PowerGraph), or both
- Hypergraph: partition underlying graph, or use a *hypergraph-aware* approach

4. Experimental Evaluation

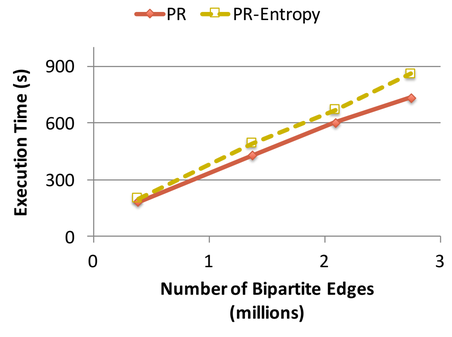
Proof-of-concept prototype
- Implemented on Apache Spark GraphX 1.2.1
- Run on shared 6-node cluster (2x6-core, 24GB RAM each)
- Using bipartite graph representation

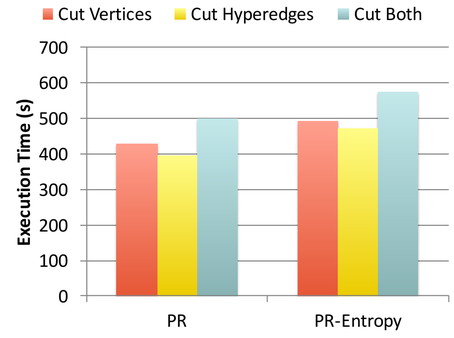
Dataset	Vertices	Hyperedges	Bipartite Edges	1-mode Projection Edges
DBLP	952,115 authors	916,947 collaborations	2,768,930	21,592,883
Friendster	7,944,949 users	1,620,991 communities	23,479,217	*> 15.1 B*



Scalability is a real challenge.

Performance heavily affected by
- Dataset + Algorithm
- Representation + Partitioning





This work was supported by NSF Grant IIS-1422802.