Seminar: Massive-Scale Graph Analysis
Summer Semester 2015

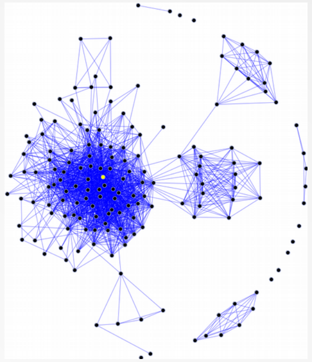
Graph Analysis Using Map/Reduce

Laurent Linden
s9lalind@stud.uni-saarland.de

May 14, 2015

Introduction

Peta-Scale Graph



\+ Graph mining

\+ MapReduce

= PEGASUS

Related Work

- Page Rank (Google)
- Random Walk w/ Restart (Pan et al.)
- Diameter Estimation (Kang et al.)
- Connected Components (Kang et al.)

Problems

- No unified solution
- Individual optimizations
- Hard to understand

Ideas

- Find common primitive
- Translate mining tasks
- Solve generalized problem
- Optimize solution

PeGaSus

- Implements common primitive
- Users specify:
  - stopping criterion (e.g. $y^{t+1} = y^t$)
  - 3 operations (combine2, combineAll, assign)
- Provides MR Jobs for
  - PR, RWR, DE, CC

Generalized Iterative Matrix-Vector mult.

- Graph = Adjacency matrix

$$E = \begin{bmatrix} e_{11} & e_{12} & \cdots & e_{1n} \\ e_{21} & & & e_{2n} \\ \vdots & & & \vdots \\ e_{n1} & e_{n2} & \cdots & e_{nn} \end{bmatrix}$$

- Graph Mining
  - Generalized Matrix-Vector multiplication
  - Iterative solution finding

$$y_i = \sum_{j=1}^{n} e_{i,j} \cdot y_j$$

$$y^{t+1} = E \times_G y^t$$

GIM-V

- Combine2($e_{i,j}$, y_j^t)
- CombineAll$_i$({...})
- Assign(y_i^t, y_i^{t+1})

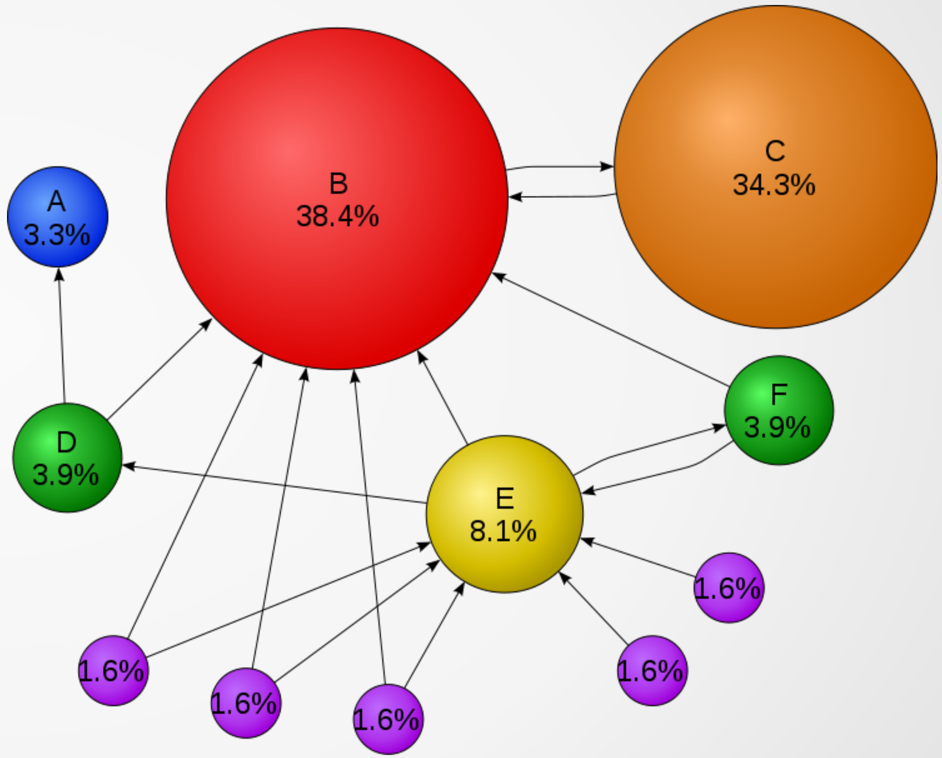
$$y_i^{t+1} = \boxed{\sum_{j=1}^{n} \boxed{e_{i,j} \cdot y_j^t}}$$

- In general:

$$y_i^{t+1} = assign(y_i^t, combineAll_i(\{z_j \mid j = 1..n, \text{ and } z_j = combine2(e_{i,j}, y_j^t)\}))$$

Example: Page Rank

- Rank pages by relevance
- Start out with normal distribution
- Compute next step via Power Method



Source: wikipedia.org/wiki/PageRank

Example: Page Rank

- E column-normalized adjacency matrix $E = \left(e_{i,j}\right)$
- U uniform matrix $U = \left(\frac{1}{n}\right)_{i,j}$
- c damping factor $0 \leq c \leq 1$
- p page rank vector
  - Satisfies the equation $p = \left(c \cdot E + \left(1 - c\right) \cdot U\right) \cdot p$
- Iterative scheme

$$p_i^0 = \frac{1}{n}$$

$$p^{t+1} = M \times_G p^t$$

Example: Page Rank

$$p^{t+1} = \overbrace{\left(c \cdot E^T + (1-c) \cdot U\right)}^{\text{M}} \times_G p^t$$

$$combine2(e_{i,j}, p_j^t) = c \times e_{i,j} \times p_j^t$$

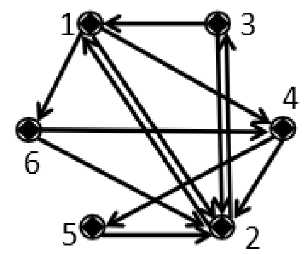
$$assign(p_i^t, p_i^{t+1}) = p_i^{t+1}$$

$$combineAll_i(\{x_1, \ldots, x_n\}) = \frac{(1-c)}{n} + \sum_{j=1}^{n} x_j$$

$$p_i^{t+1} = assign(p_i^t, combineAll_i(\{x_j \mid j=1..n, \text{ and } x_j = combine2(m_{i,j}, p_j^t)\}))$$

Example: Random Walks w/ Restart

- Measures proximity of nodes
- Start with arbitrary distribution
- Refine distribution until results converge



P	p_1	p_2	p_3	p_4	p_5	p_6
	0.32	0.24	0.24	0.19	0.20	0.18
	0.28	0.39	0.29	0.31	0.33	0.30
	0.12	0.17	0.27	0.13	0.14	0.13
	0.13	0.10	0.10	0.23	0.08	0.14
	0.06	0.04	0.04	0.10	0.18	0.06
	0.09	0.07	0.07	0.05	0.06	0.20

Source: Pan et al, "Fast Random Walk with Restart and its Application"

Example: Random Walks w/ Restart

- E column-normalized adjacency matrix

$$E=(e_{i,j})$$

- e_k vector with k-th component = 1

$$e_k=(0,\dots,0,1,0,\dots,0)$$

- c restart probability

$$0\leq c\leq 1$$

- r_k proximity vector
  - Satisfies the equation

$$r_k=c\cdot E\cdot r_k+(1-c)\cdot e_k$$

- Iterative Scheme

$$r_k^0=initial\,distribution$$

$$r_k^{t+1}=M\times_G r_k^t$$

Example: Random Walks w/ Restart

$$r_k^{t+1} = \overbrace{\left(c \cdot E \cdot r_k + (1-c) \cdot e_k\right)}^{M} \times_G r_k^t$$

$$combine2(e_{i,j}, r_j^t) = c \times e_{i,j} \times r_j^t$$

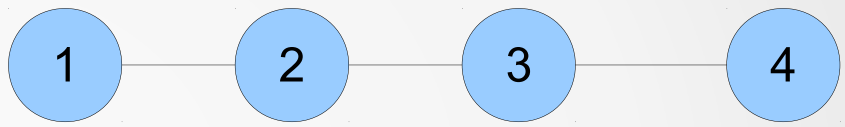
$$assign(r_i^t, r_i^{t+1}) = r_i^{t+1}$$

$$combineAll_i(\{x_1, \ldots, x_n\}) = (1-c)\, I(i=k) + \sum_{j=1}^{n} x_j$$

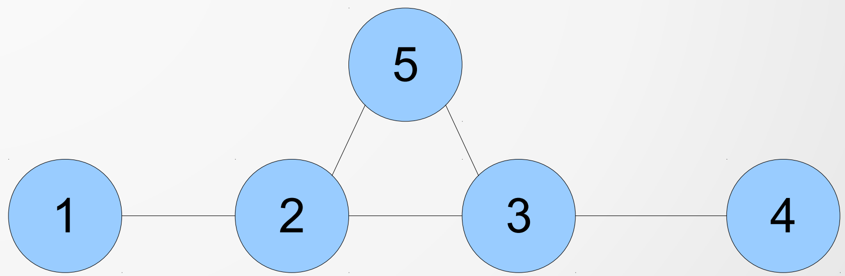
$$r_k^{t+1} = assign(r_k^t, combineAll_i(\{x_j \mid j=1..n, \text{ and } x_j = combine2(m_{i,j}, r_k^t)\}))$$

Example: Diameter Estimation

- Estimate radius and diameter of subgraphs
- Radius of a node
  - #Hops to farthest-away node
- Diameter of a graph
  - Maximal length of shortest path between every pair of nodes



Radius(3) = 2



- Diameter = 3

Example: Diameter Estimation

- M adjacency matrix

$$M = (m_{i,j})$$

- b_i probabilistic bitstring
  - Solution satisfies equation

$$b_i^{h+1} = b_i^h \, (for\ all\ i)$$

- Iterative Scheme

$$b_i^0 = 0 \ldots 010 \ldots 0$$

$$b_i^{h+1} = b_i^h \text{ BITWISE-OR } \{ b_k^h \mid (i,k) \in E \}$$

Example: Diameter Estimation

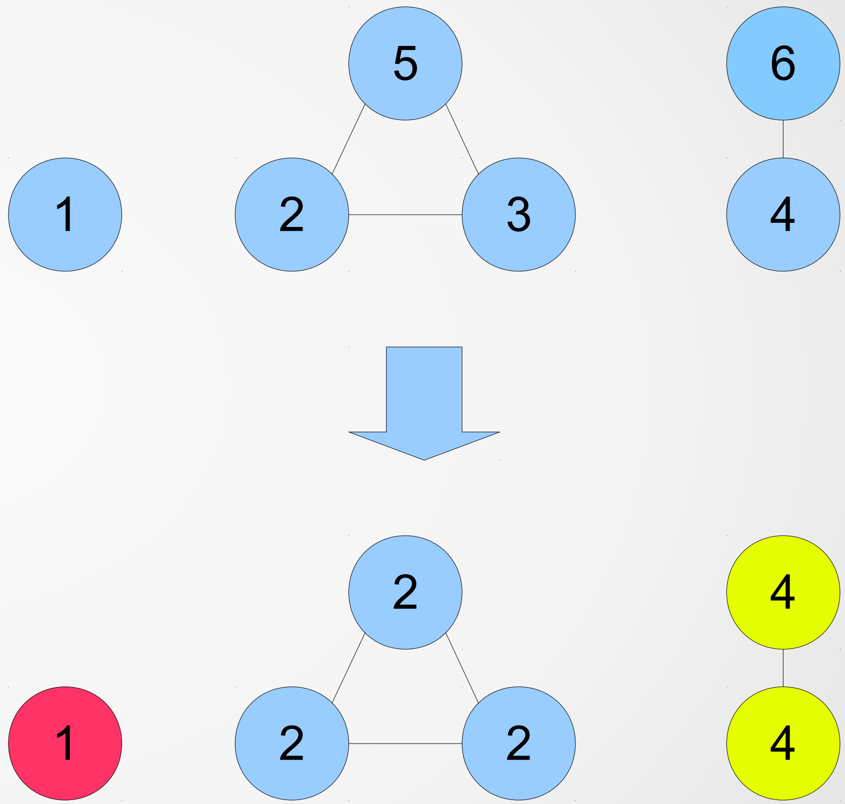
$$b_i^{h+1} = M \times_G b_i^h$$

$$combine2(m_{i,j}, b_j^h) = m_{i,j} \times b_j^h$$
$$combineAll_i(\{x_1, \ldots, x_n\}) = \text{BITWISE-OR } \{x_j \mid j = 1..n\}$$
$$assign(b_i^h, b_i^{h+1}) = \text{BITWISE-OR}(b_i^h, b_i^{h+1})$$

$$b_i^{h+1} = assign(b_i^h, combineAll_i(\{x_j \mid j = 1..n, \text{ and } x_j = combine2(m_{i,j}, b_j^h)\}))$$

Example: Connected Components

- Find connected components
- Assign weight to nodes
- Smallest weight gets propagated to neighbors
- Terminate when result does not change anymore



Example: Connected Components

- M adjacency matrix

$$M = (m_{i,j})$$

- c minimum node vector
  - Solution satisfies equation

$$c_i^{h+1} = c_i^h (for\, all\, i)$$

- Iterative scheme

$$c_i^0 = i$$

$$c_i^{h+1} = \mathrm{MIN}\{\ c_j^h \mid \mathrm{j} \in \mathrm{E}\}$$

Example: Connected Components

$$c_i^{h+1} = M \times_G c_i^h$$

$$combine2(m_{i,j}, c_j^h) = m_{i,j} \times c_j^h$$

$$combineAll_i(\{x_1, \ldots, x_n\}) = \mathrm{MIN}\ \{x_j \mid j = 1..n\}$$

$$assign(c_i^h, c_i^{h+1}) = \mathrm{MIN}(c_i^h, c_i^{h+1})$$

$$c_i^{h+1} = assign(c_i^h, combineAll_i(\{x_j \mid j = 1..n, \text{ and } x_j = combine2(m_{i,j}, c_j^h)\}))$$

GIM-V BASE

- Input
  - Edge file $E(id_{src},id_{dst},mval)$
  - Vector file V(id,vval)
- Output
  - Vector file V(id,vval)

- 2 Stages (MR Jobs)
  - Stage1: Applies combine2
  - Stage2: Applies combineAll, assign

GIM-V BASE (Stage 1)

Input: $M = \{(id_{src}, (id_{dst}, mval))\}$
$V = \{(id, vval)\}$

Output: $V' = \{(id_{src}, combine2(mval,vval))\}$

Mapper(k, v):

(k,v) of type V:

(id,vval) → (id,vval)

(k,v) of type M:

$(id_{src},(id_{dst},mval))$ → $(id_{dst},(id_{src},mval))$

...

Reducer(k, v[1..m]):

Extract vval from v:

→ (k,(“self”,vval))

Extract $(id_{src}$,mval) pairs from v:

store pairs in list A

foreach $(id_{src}$, mval) in A:

x = combine2(mval, vval)

→ $(id_{src},(\text{“others”},x))$

GIM-V BASE (Stage 2)

Input: $V' = \{(id_{src}, vval)\}$

Output: $V = \{(id_{src}, vval)\}$

Mapper(k, v):

(k, v) → (k, v)

Reducer(k, v[1..m]):

Extract vval from v

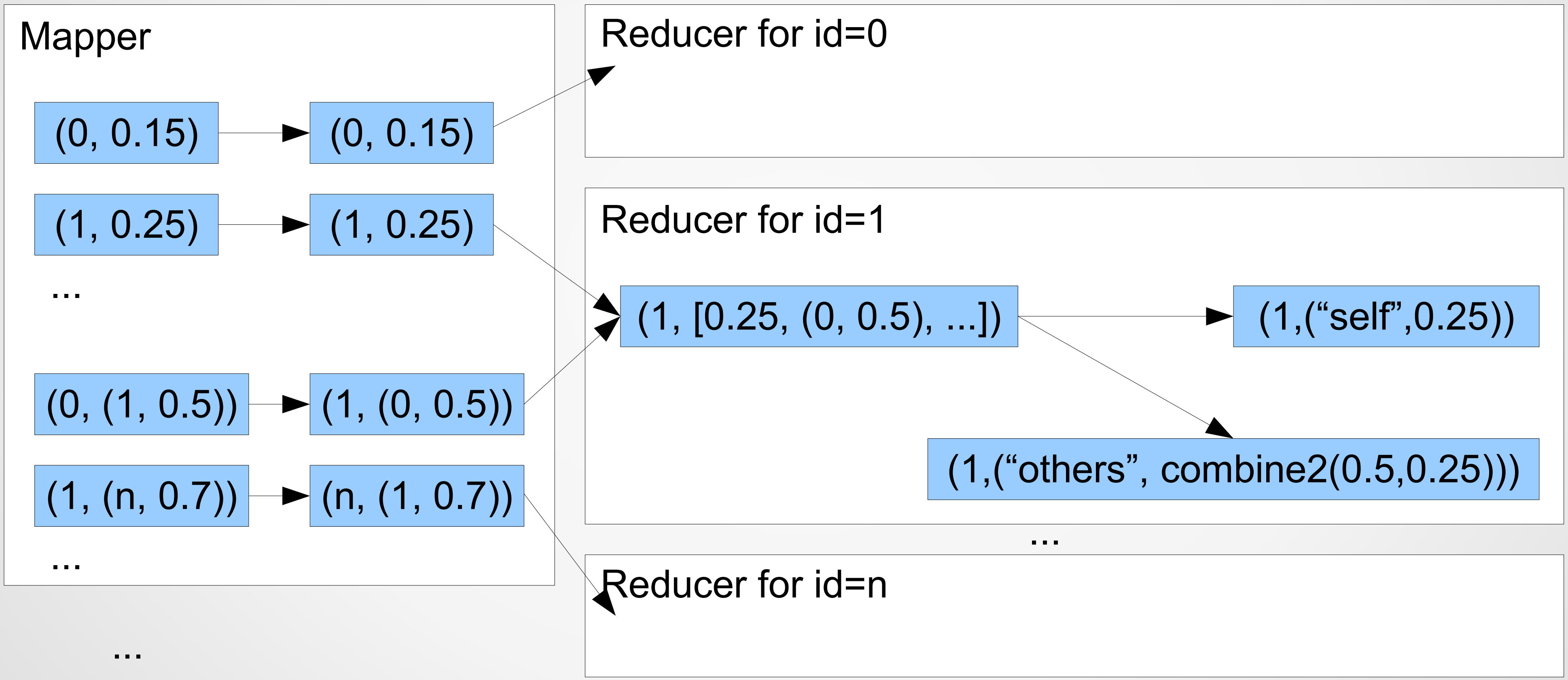
Extract Stage1 output from v:
Store values in list A

$c = \text{combineAll}_k(A)$

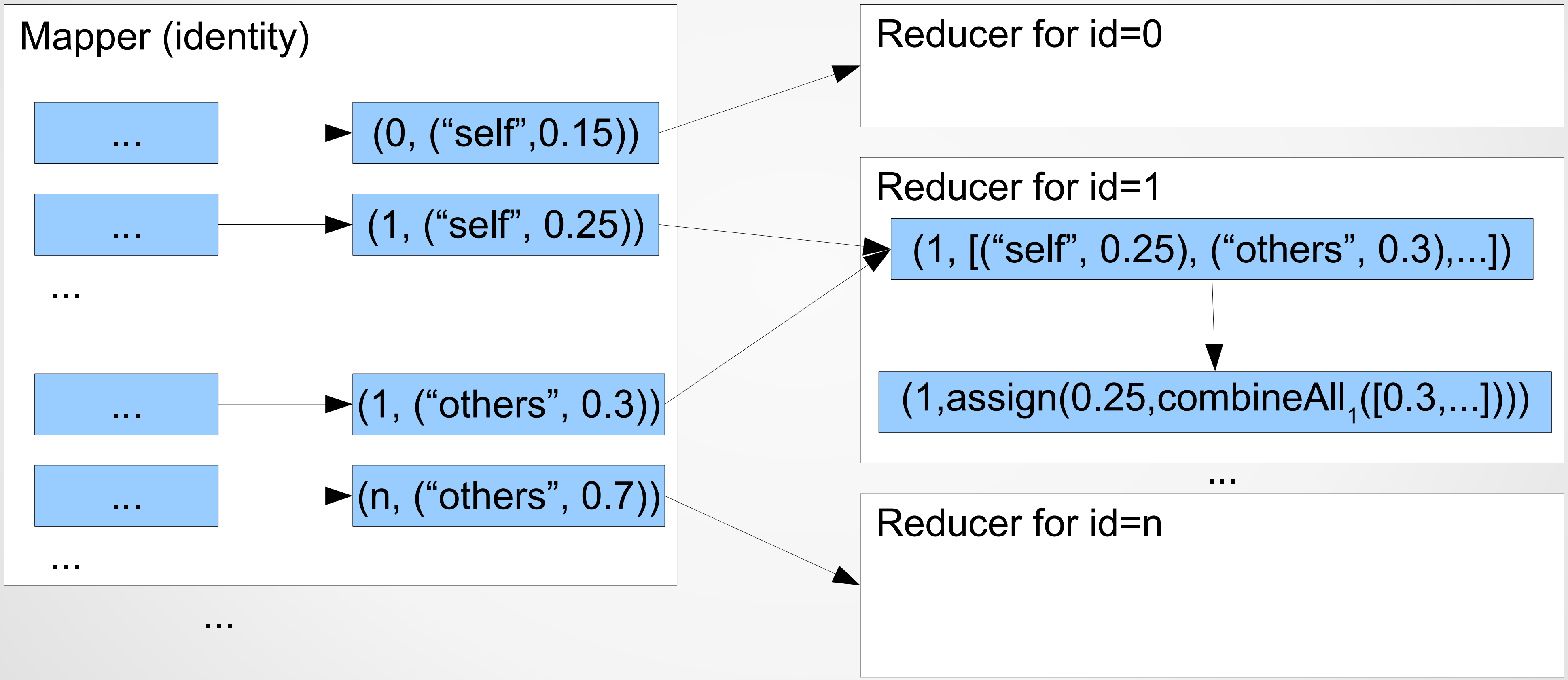
$a = \text{assign}(vval, c)$

→ (k, a)

GIM-V BASE (Stage 1)



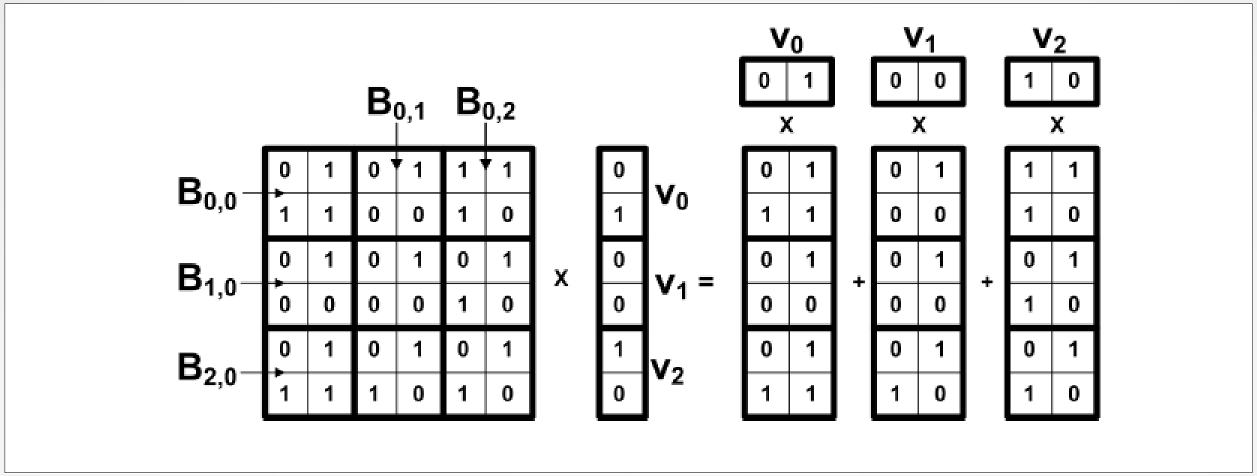
GIM-V BASE (Stage 2)



GIM-V BASE Bottlenecks

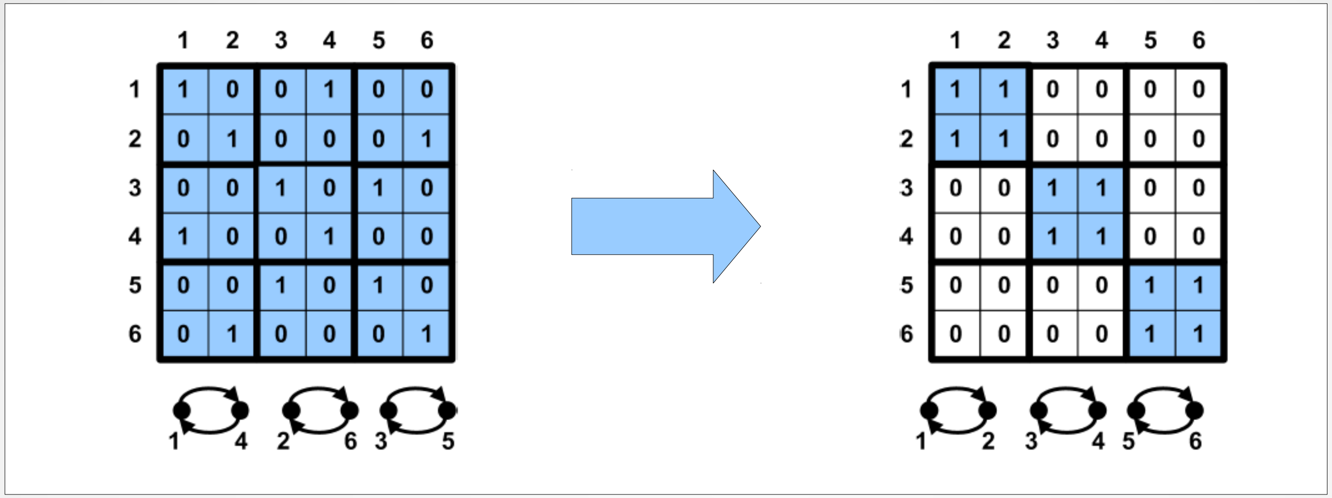
- Network traffic
- Disk IO
- Number of iterations

GIM-V BL (Block Multiplication)



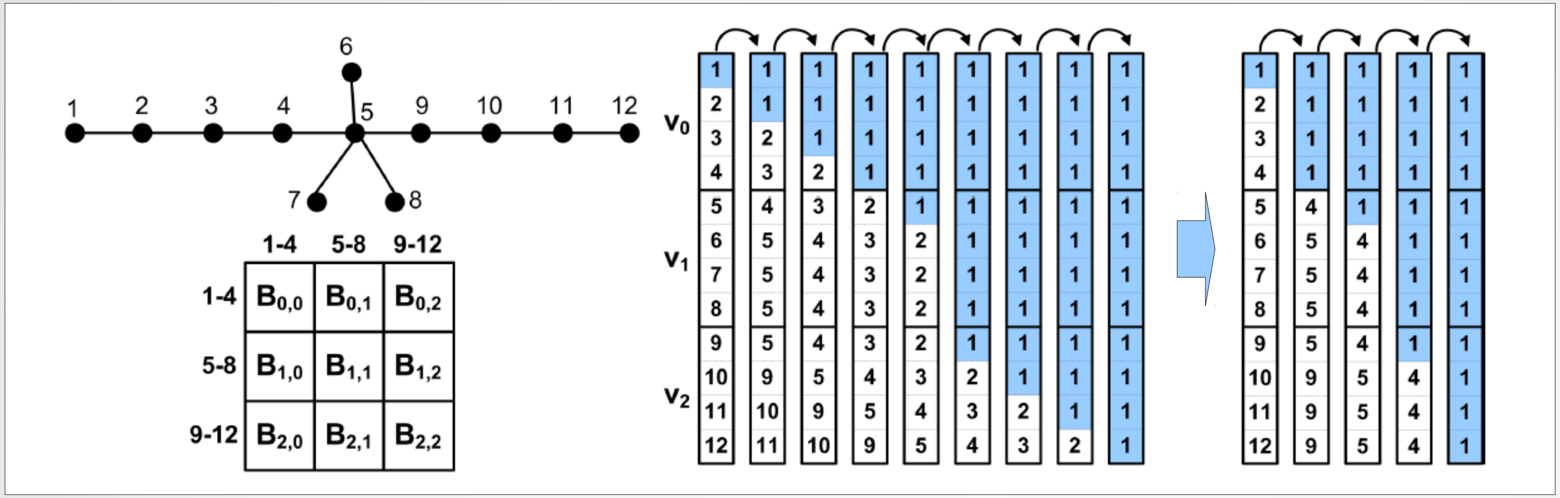
- Groups elements into blocks
  - M, V can be joined block-wise not element-wise
  - Guarantees co-location of nearby edges
- Groups of 2 x 2 blocks saved in a single line
  - $(row_{block}, col_{block}, row_{e1}, col_{e1}, val_{e1}, \ldots)$
  - $(id_{block}, id_{e1}, vval_{e1}, \ldots)$

GIM-V CL (Clustered Edges)



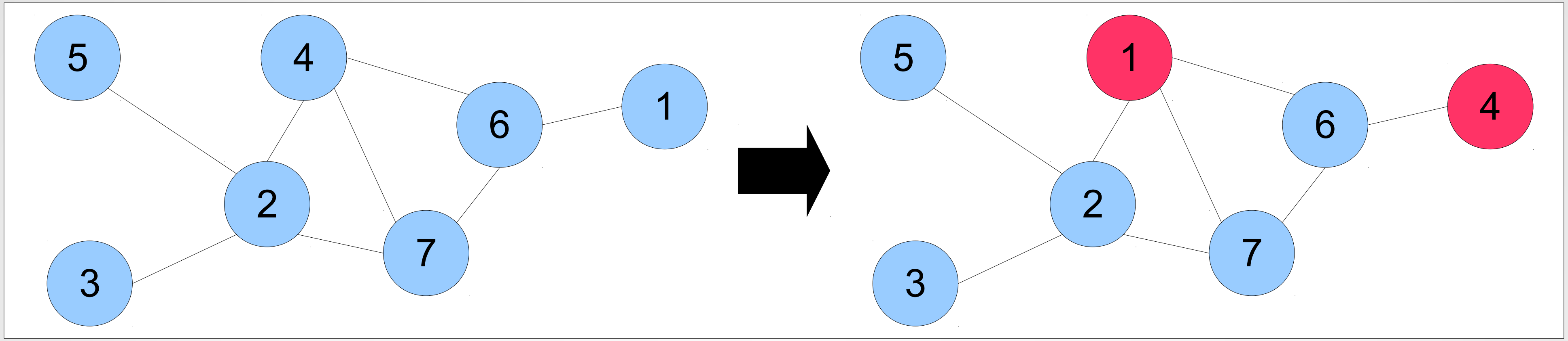
- Clusters connected vertices into the same blocks
  - Move connected vertices close to each other
  - No need to serialize 0 blocks anymore
- Preprocessing step (one time cost)
- Only useful in conjunction with GIM-V BL

GIM-V DI (Diagonal Block Iteration)



- Iterate GIM-V on block level
  - Repeated block multiplication
  - Repeated GIM-V application
- Block multiplication cheaper then GIM-V
- Only useful in conjunction with GIM-V BL

GIM-V NR (Node Renumbering)



- Renumber nodes for better convergence
  - Find center node via heuristics (highest degree)
  - Renumber center node to value of minimum node
- Preprocessing step (one time cost)

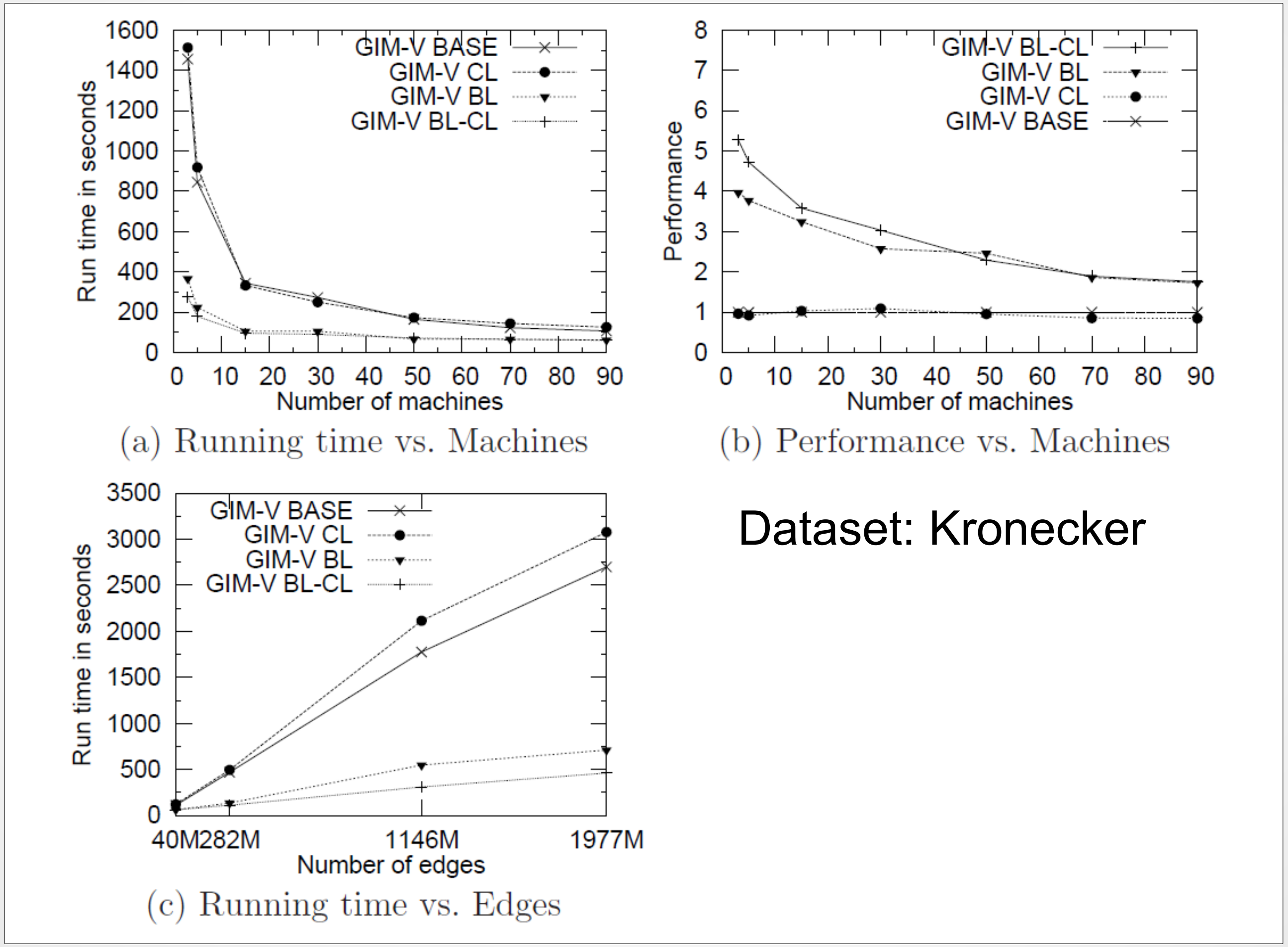
Evaluation

- Evaluation of GIM-V
  - Performance & Scalability
- Evaluation of optimizations
  - Relative to GIM-V BASE

Experimental Setup

- Yahoo M45
  - 4,000 processors
  - 3 TB of memory
  - 1.5 PB of disk space
  - 27 teraflops (peak performance)
- Datasets
  - Kronecker (177 K nodes, 1977 M edges)
  - LinkedIn (7.5 M nodes, 58 M edges)
  - Wikipedia (4.4 M nodes, 27 M edges)
  - And many more (not shown here)

PR & GIM-V Evaluation



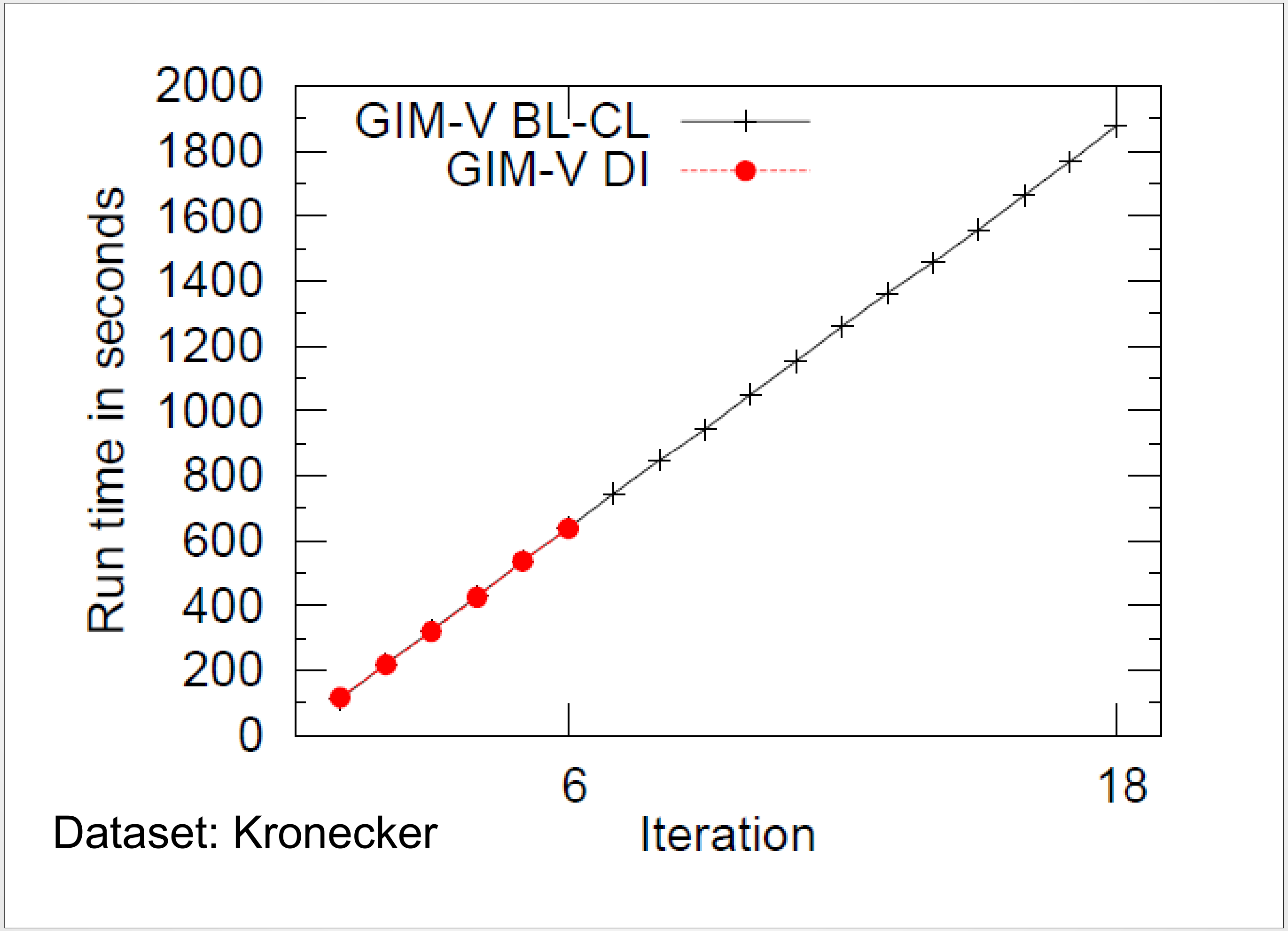
(a) Running time vs. Machines

(b) Performance vs. Machines

(c) Running time vs. Edges

Notice the impact block multiplication has on performance

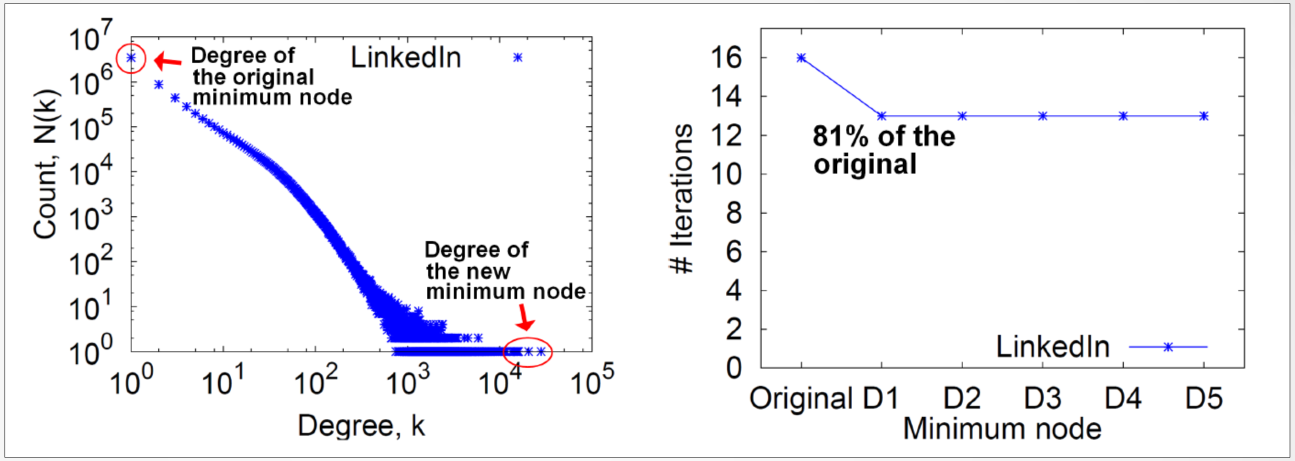
CC & GIM-V Evaluation



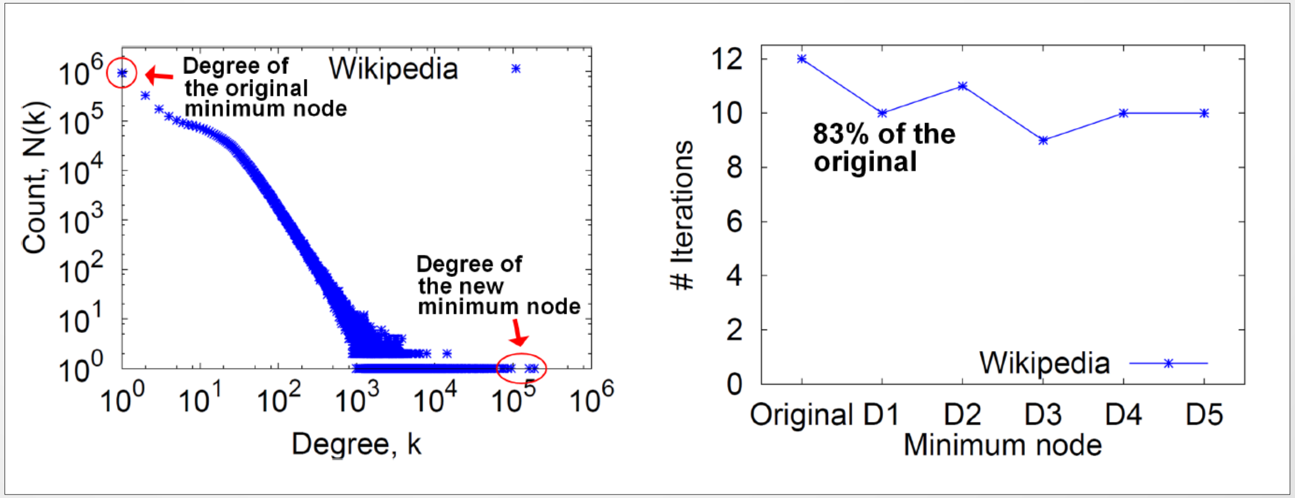
Notice the big reduction in #iterations

DE & GIM-V Evaluation

- LinkedIn



- Wikipedia



Notice the reduction in #iterations

Conclusion

- Feasability of Peta-Scale Graph analysis
- Identification of common primitive
- Unification of GM algorithms

Q&A