Scalable and Memory-Efficient Clustering of Large-Scale Social Networks

Joyce Jiyoung Whang, Xin Sui, Inderjit S. Dhillon
Department of Computer Science
The University of Texas at Austin

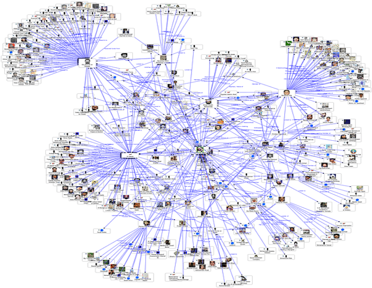
IEEE International Conference on Data Mining (ICDM)
December 10 - 13, 2012

Contents

Introduction

Problem Statement

- Graph Clustering
  - Graph $G = (\mathcal{V}, \mathcal{E})$
  - k disjoint clusters $\mathcal{V}_1, \cdots, \mathcal{V}_k$ such that $\mathcal{V} = \mathcal{V}_1 \cup \cdots \cup \mathcal{V}_k$.
- Social Networks
  - Vertices: actors, edges: social interactions
  - Distinguishing properties
    - Power law degree distribution
    - Hierarchical structure



Contributions

- Multilevel Graph Clustering Algorithms
  - PMetis, KMetis, Graclus
  - ParMetis (Parallel implementation of KMetis)
  - Performance degradation
    - KMetis – 19 hours, more than 180 Gigabytes memory to cluster a Twitter graph (50 million vertices, one billion edges).

- **GEM** (**G**raph **E**xtraction + weighted kernel *k*-**M**eans)
  - Scalable & memory-efficient clustering algorithm
    - Comparable or better quality
    - Much faster and consumes much less memory
  - PGEM (Parallel implementation of GEM)
    - Higher quality of clusters
    - Much better scalability
  - GEM takes less than three hours on Twitter (40 Gigabytes memory).
  - PGEM takes less than three minutes on Twitter on 128 processes.

Preliminaries: Graph Clustering Objectives

- Kernighan-Lin objective
  - PMetis, KMetis
  - k equal-sized clusters

$$\min_{\mathcal{V}_1,\ldots,\mathcal{V}_k} \sum_{i=1}^{k} \frac{links(\mathcal{V}_i, \mathcal{V} \setminus \mathcal{V}_i)}{|\mathcal{V}_i|} \text{ such that } |\mathcal{V}_i| = \frac{|\mathcal{V}|}{k}.$$

- Normalized cut objective
  - Graclus
  - Minimize cut relative to the degree of a cluster

$$\min_{\mathcal{V}_1,\ldots,\mathcal{V}_k} \sum_{i=1}^{k} \frac{links(\mathcal{V}_i, \mathcal{V} \setminus \mathcal{V}_i)}{degree(\mathcal{V}_i)}.$$

Preliminaries: Weighted Kernel k-Means

- **A general weighted kernel k-means objective** is equivalent to **a weighted graph clustering objective**.
- Weighted kernel k-means
  - Objective

$$J = \sum_{c=1}^{k} \sum_{\mathbf{x}_i \in \pi_c} w_i ||\varphi(\mathbf{x}_i) - \mathbf{m}_c||^2, \text{ where } \mathbf{m}_c = \frac{\sum_{\mathbf{x}_i \in \pi_c} w_i \varphi(\mathbf{x}_i)}{\sum_{\mathbf{x}_i \in \pi_c} w_i}.$$

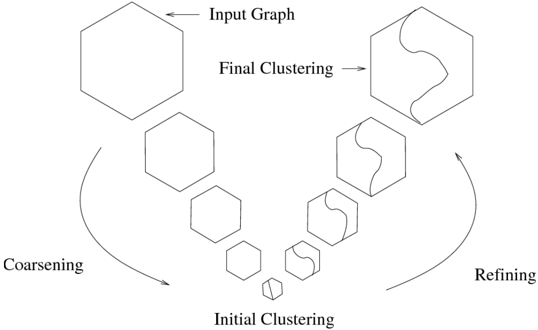
  - Algorithm
    - Assigns each node to the closest cluster.
    - After all the nodes are considered, the centroids are updated.
    - Given the Kernel matrix K, where $K_{ij} = \varphi(\mathbf{x}_i) \cdot \varphi(\mathbf{x}_j)$,

$$||\varphi(\mathbf{x}_i) - \mathbf{m}_c||^2 = K_{ii} - \frac{2 \sum_{\mathbf{x}_j \in \pi_c} w_j K_{ij}}{\sum_{\mathbf{x}_j \in \pi_c} w_j} + \frac{\sum_{\mathbf{x}_j, \mathbf{x}_l \in \pi_c} w_j w_l K_{jl}}{(\sum_{\mathbf{x}_j \in \pi_c} w_j)^2}.$$

Multilevel Graph Clustering Algorithms

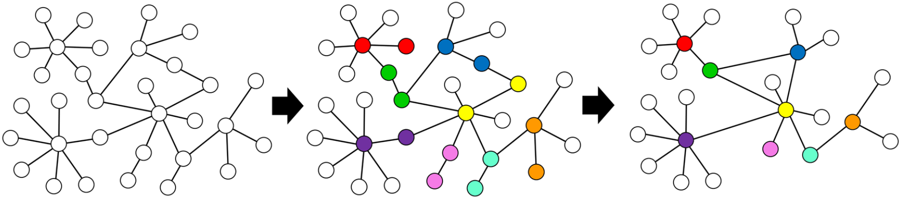
Multilevel Framework for Graph Clustering

- PMetis, KMetis, Graclus
- Multilevel Framework
  - Coarsening phase
  - Initial clustering phase
  - Refinement phase



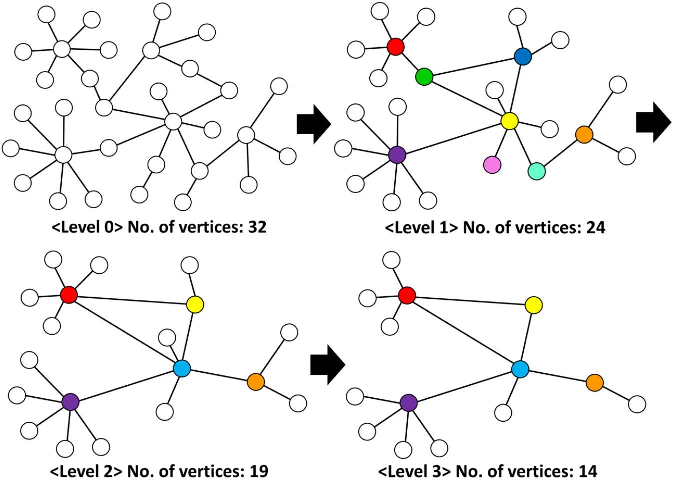
Problems with Coarsening Phase

- Coarsening of a scale-free network can lead to serious problems.
  - Most low degree vertices tend to be attached to high degree vertices.
  - Low degree vertices have little chance to be merged.
  - Example:
    - No. of vertices in the original graph: 32
    - No. of vertices in the coarsened graph: 24



Problems with Coarsening Phase

- Coarsening of a scale-free network
  - Transform the original graph into smaller graphs level by level

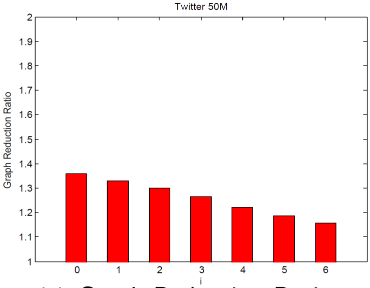


Limitations of Multilevel Framework

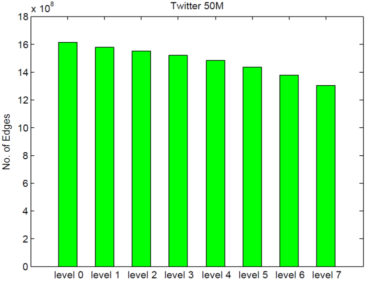
- Difficulties of Coarsening in Large Social Networks
  - In the coarsening from G_i to G_{i+1},

$$\text{Graph Reduction Ratio} = \frac{|\mathcal{V}_i|}{|\mathcal{V}_{i+1}|}.$$

  - Ideally, graph reduction ratio would equal 2.
  - The success of multilevel algorithms – high graph reduction ratio
  - Power law degree distribution – graph reduction ratio becomes small.



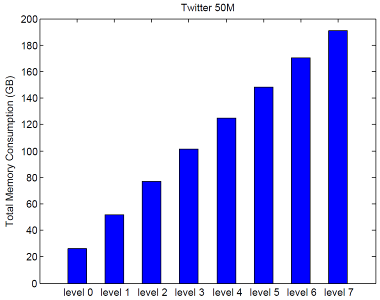
(a) Graph Reduction Ratio



(b) No. of Edges at Each Level

Limitations of Multilevel Framework

- Memory Consumption
  - Multilevel algorithms generate a series of graphs.
  - Total memory consumption increases rapidly during coarsening phase.

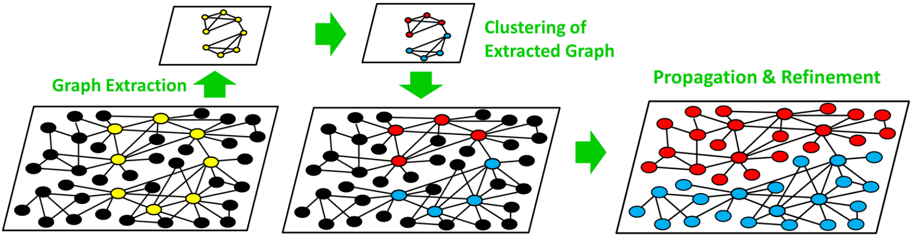


- Difficulties in Parallelization
  - Coarsening requires intensive communication between processes.

Proposed Algorithm: GEM

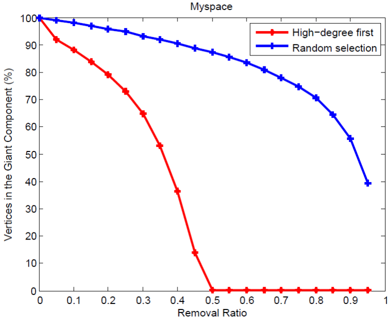
Overview of GEM

- Graph Extraction
- Clustering of Extracted Graph
- Propagation and Refinement



Graph Extraction

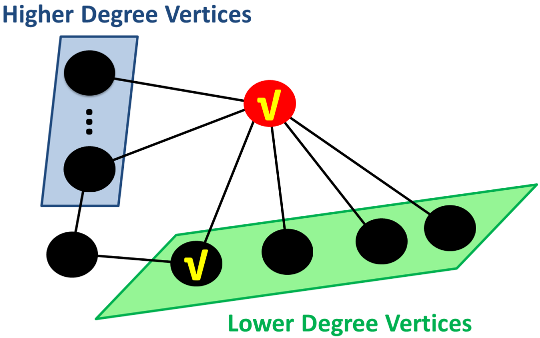
- Extract a skeleton of the original graph using high degree vertices.



- High degree vertices
  - Tend to preserve the structure of a network
  - Popular and influential people

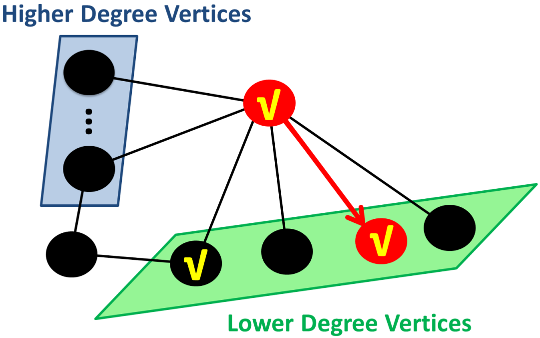
Clustering of Extracted Graph

- **Down-Path Walk Algorithm**
  - Refer to a path $v_i \to v_j$ as a *down-path* if $d_i \geq d_j$.
  - Follow a certain number of down-paths.
  - Generate a seed by selecting the final vertex in the path.
  - Mark the seed, and its neighbors.
  - Repeat this procedure until we get k seeds in the graph.



Clustering of Extracted Graph

- **Down-Path Walk Algorithm**
  - Refer to a path $v_i \rightarrow v_j$ as a *down-path* if $d_i \geq d_j$.
  - Follow a certain number of down-paths.
  - Generate a seed by selecting the final vertex in the path.
  - Mark the seed, and its neighbors.
  - Repeat this procedure until we get k seeds in the graph.



Clustering of Extracted Graph

- **Online Weighted Kernel *k*-means**
  - Initialization of clusters

Input: $\mathcal{V}$: the vertex set, s_i $(i = 1, ..., k)$: seeds.
Output: $\mathcal{V}_i$ $(i = 1, ..., k)$: initial clusters.

for each cluster $\mathcal{V}_i$ **do**
    $\mathcal{V}_i = s_i$.
end for
for each vertex $\hat{v} \in \mathcal{V}$ **do**
    for each cluster $\mathcal{V}_i$ **do**
        $$\hat{\alpha} = \frac{\sigma}{\hat{d}} - \frac{2links(\hat{v}, \mathcal{V}_i)}{\hat{d} \cdot degree(\mathcal{V}_i)} + \frac{links(\mathcal{V}_i, \mathcal{V}_i)}{degree(\mathcal{V}_i)^2}.$$
        $$\delta_i = \frac{\hat{d} \cdot degree(\mathcal{V}_i)}{degree(\mathcal{V}_i) + \hat{d}} \left\{ \hat{\alpha} + \frac{\sigma}{degree(\mathcal{V}_i)} \right\}.$$
    end for
    Find $\mathcal{V}_p$ s.t. $\delta_p \leq \delta_j$ for all j $(j = 1, ..., k)$.
    $\mathcal{V}_p = \{\hat{v}\} \cup \mathcal{V}_p$.
end for

Initialize clusters: each cluster only contains the seed.

Assign each vertex to the cluster which allows the least increase of the total objective.

Clustering of Extracted Graph

- **Online Weighted Kernel *k*-means**
  - Graph clustering using online weighted kernel k-means

Input: $\mathcal{V}$: the vertex set, $\mathcal{V}_i$ $(i = 1, ..., k)$: initial clusters, τ_{max}: maximum number of iterations.

Output: $\mathcal{V}_i^*$ $(i = 1, ..., k)$: final clusters.

Initialize $\tau = 0$.
repeat
  for each vertex $\hat{v} \in \mathcal{V}$ **do**
    $\mathcal{V}_p \longleftarrow$ the current cluster of $\hat{v}$.
    for each cluster $\mathcal{V}_i$ **do**
      $\hat{\alpha} = \frac{\sigma}{\hat{d}} - \frac{2links(\hat{v}, \mathcal{V}_i)}{\hat{d} \cdot degree(\mathcal{V}_i)} + \frac{links(\mathcal{V}_i, \mathcal{V}_i)}{degree(\mathcal{V}_i)^2}.$
      if $\mathcal{V}_i = \mathcal{V}_p$ **then**
        $\delta_i = \frac{\hat{d} \cdot degree(\mathcal{V}_i)}{degree(\mathcal{V}_i) - \hat{d}} \left\{ \hat{\alpha} - \frac{\sigma}{degree(\mathcal{V}_i)} \right\}.$
      else
        $\delta_i = \frac{\hat{d} \cdot degree(\mathcal{V}_i)}{degree(\mathcal{V}_i) + \hat{d}} \left\{ \hat{\alpha} + \frac{\sigma}{degree(\mathcal{V}_i)} \right\}.$
      end if
    end for
    Find $\mathcal{V}_q$ s.t. $\delta_q \leq \delta_j$ for all j $(j = 1, ..., k)$.
    if $\mathcal{V}_p \neq \mathcal{V}_q$ **then**
      $\mathcal{V}_p = \mathcal{V}_p \setminus \{\hat{v}\}$, $\mathcal{V}_q = \mathcal{V}_q \cup \{\hat{v}\}$.
      Update $links(\mathcal{V}_p, \mathcal{V}_p), degree(\mathcal{V}_p),$ $links(\mathcal{V}_q, \mathcal{V}_q), degree(\mathcal{V}_q).$
    end if
  end for
  $\tau = \tau + 1$.
until not converged and $\tau < \tau_{max}$
$\mathcal{V}_i^* = \mathcal{V}_i$ $(i = 1, ..., k)$.

Compute the actual change in objective function for each cluster.

Assign the vertex to the cluster which allows the greatest decrease of the total objective change.

Propagation and Refinement

- Propagation
  - Propagate clustering of extracted graph to the entire original graph.
  - Visit vertices not in extracted graph in a breadth-first order (starting from vertices of extracted graph).
  - Arrive at initial clustering of the original graph (by online weighted kernel k-means).
- Refinement
  - Refine the clustering of the original graph.
  - Good initial clusters are achieved by the propagation step.
  - Refinement step efficiently improves the clustering result (by online weighted kernel k-means).

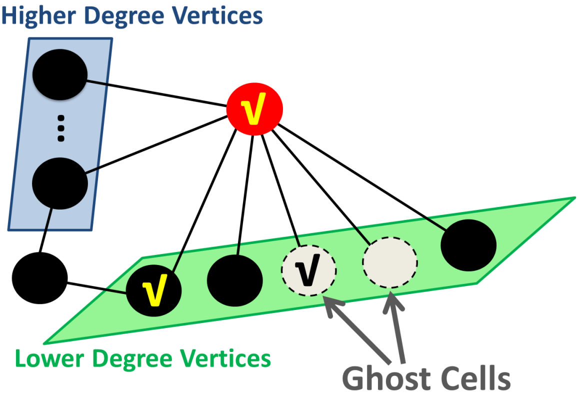
Parallel Algorithm: PGEM

PGEM

- GEM is easy to parallelize for large number of processes.
- Graph distribution across different processes
  - Each process: an equal-sized subset of vertices & their adjacency lists.
- Graph extraction
  - Each process scans its local vertices, and picks up high degree vertices.
  - The extracted graph is randomly distributed over all the processes.
- Seed selection phase
  - Seeds are generated in rounds.
  - Leader process decides the number of seeds each process will generate.
    - Proportional to the number of currently unmarked vertices in that process

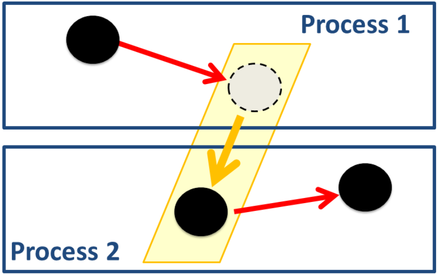
PGEM

- Parallel Down-Path Walk Algorithm
  - Neighbor vertices might be located in a different process.
  - Ghost Cells
    - For each remote neighbor vertex, a mirror vertex is maintained.
    - Buffer the information of its remote counterpart



PGEM

- Parallel Down-Path Walk Algorithm (continued)
  - Passing a walk to another process
    - Process 1 finds that the next vertex it will visit belongs to process 2.
    - Then process 1 stops this walk and notifies process 2 to continue it.



PGEM

- Parallel Online Weighted Kernel k-means
  - Initialization of clusters
    - Ghost cells: accessing cluster information of remote vertices
    - Each process maintains a local copy of cluster centroids
  - Refinement
    - Visit local vertices in random order
    - Updating cluster centroids and ghost cells – relax the synchronization
- Propagation Phase
  - Similar to initialization of clusters in the extracted graph
- Refinement of the entire graph
  - Same strategy as clustering of the extracted graph

Experimental Results

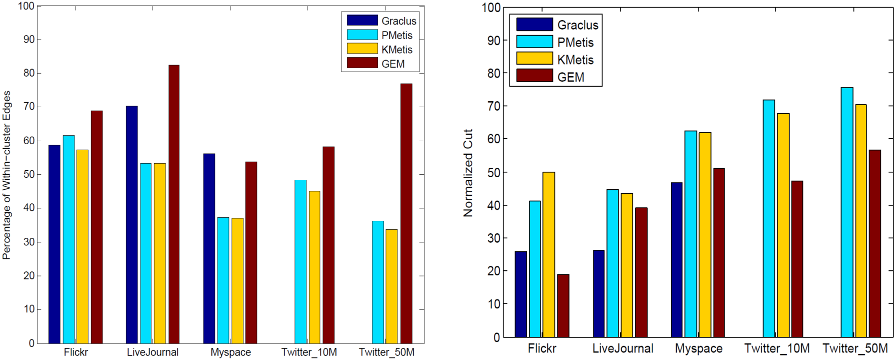
Experimental Setting & Dataset

- Sequential experiments
  - **GEM vs. PMetis, KMetis (Metis), Graclus**
  - Shared memory machine (AMD Opteron 2.6GHz CPU, 256GB memory)
- Parallel experiments
  - **PGEM vs. ParMetis**
  - Ranger at Texas Advanced Computing Center (TACC)
    - 3,936 machine nodes (4×4-core AMD Opteron CPU, 32GB memory)
- Dataset

Graph	No. of vertices	No. of edges
Flickr	1,994,422	21,445,057
LiveJournal	1,757,326	42,183,338
Myspace	2,086,141	45,459,079
Twitter (10M)	11,316,799	63,555,738
Twitter (50M)	51,161,011	1,613,892,592

Evaluation of GEM

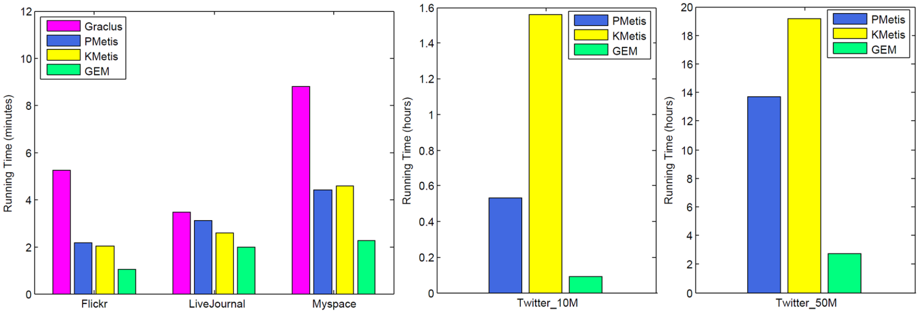
- Quality of clusters
  - Higher percentage of within-cluster edges / lower normalized cut indicates better quality of clusters.



(a) Percentage of Within-cluster Edges (b) Normalized Cut

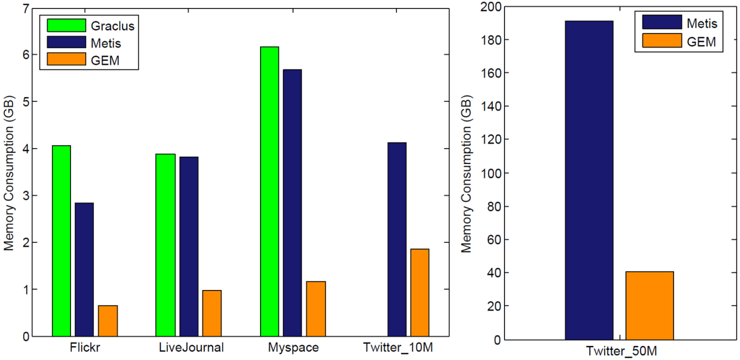
Evaluation of GEM

- Running Time
  - GEM is the fastest algorithm across all the datasets.
  - Twitter 10M: GEM (6 min.) vs. PMetis (30 min.) vs. KMetis (90 min.)
  - Twitter 50M: GEM (3 hrs.) vs. PMetis (14 hrs.) vs. KMetis (19 hrs.)



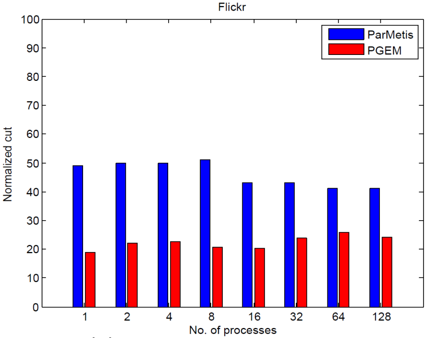
Evaluation of GEM

- Memory Consumption
  - PMetis and KMetis use the same coarsening strategy (as in Metis).
  - GEM directly extracts a subgraph from the original graph.
  - Multilevel algorithms gradually reduce the graph size by multilevel coarsening.

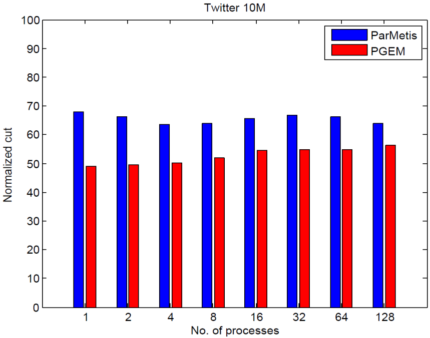


Evaluation of PGEM

- PGEM performs consistently better than ParMetis in terms of normalized cut, running time and speedup across all the datasets.
  - Quality of clusters



(a) Flickr normalized cut

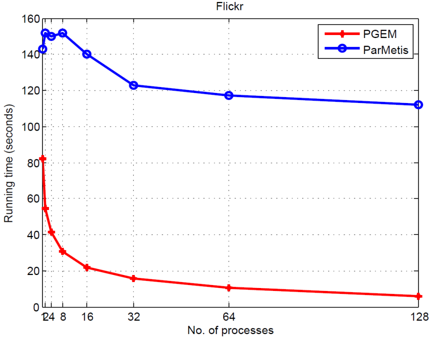


(b) Twitter 10M normalized cut

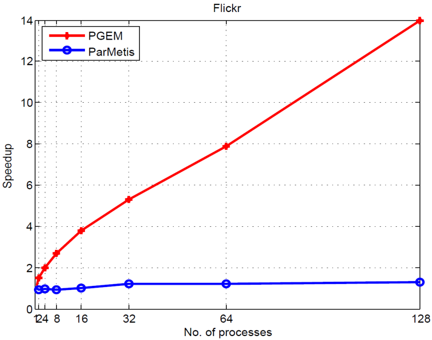
Evaluation of PGEM

- Running time & speedup on Flickr

$$\text{Speedup} = \frac{\text{Runtime of the program with one process}}{\text{Runtime with } p \text{ processes}}$$



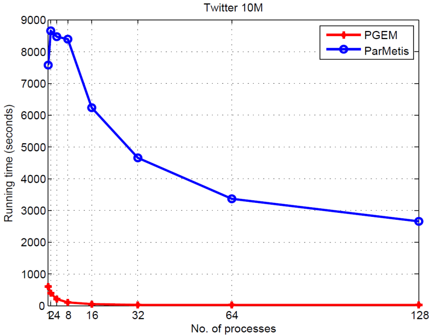
(a) Flickr running time



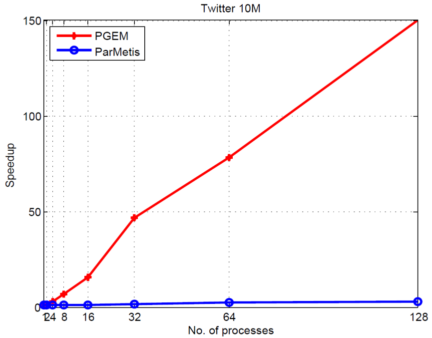
(b) Flickr speedup

Evaluation of PGEM

- Running time & speedup on Twitter 10M
  - PGEM achieves a super-linear speedup.
  - In all cases, the speedup of ParMetis is less than 10.
    - The multilevel scheme is hard to be scaled to large number of processes.



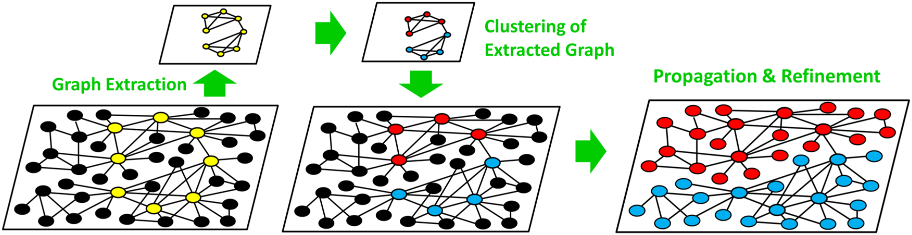
(a) Twitter 10M running time



(b) Twitter 10M speedup

Conclusions

GEM & PGEM



- GEM produces clusters of quality better than state-of-the-art clustering algorithms while it saves much time and memory.
- PGEM achieves significant scalability while producing high quality clusters.
- Future Research
  - Theoretical justification (can we theoretically show that extracted graph preserves structure of original graph).
  - Automatical detection of number of clusters.

References

- I. S. Dhillon, Y. Guan, and B. Kulis. Weighted graph cuts without eigenvectors: A multilevel approach. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 29, no. 11, pp. 1944-1957, 2007.
- G. Karypis and V. Kumar. Multilevel *k*-way partitioning scheme for irregular graphs. *Journal of Parallel and Distributed Computing*, vol. 48, pp. 96129, 1998.