Scalable Positional Analysis for Studying Evolution of Nodes in Networks

Pratik V. Gupte, B. Ravindran
Reconfigurable and Intelligent Systems Engineering Lab,
Department of Computer Science and Engineering,
IIT Madras

October 3, 2014

Position and Role

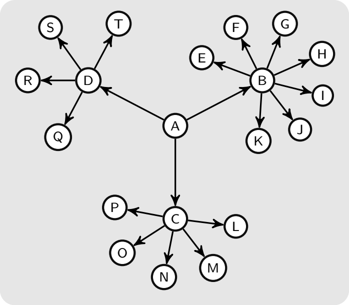
- Position and Roles
  - **Position** refers to set of individuals (actors or nodes) who are embedded similarly in the network
  - **Role** is the pattern of relation that exists between actors or different positions
- Positional Analysis
  - Partition actors (nodes) in the network into discrete subgroups based on some kind of structural similarity
  - Structural equivalence, Automorphic equivalence, Regular Equivalence are few such equivalence relations from social sciences; RolX from Henderson et al.
  - Equitable partitions is a similar notion from graph isomorphism literature

Why Positional Analysis?

- A very intuitive way of understanding interactions in networks
- Nodes having similar structural signature tend to have similar behaviour
- Important tool in the analysis of social networks [1]
- Blockmodels - To facilitate further processing on social networks by obtaining a reduced representation
- Position plays important role in evolution of networks
- Mining “roles” from network data easy for humans, difficult to automate!
- Need new methods to do this in a scalable fashion

[1] S. Wasserman and K. Faust, Social Network Analysis: methods and applications. Cambridge University Press, 1994

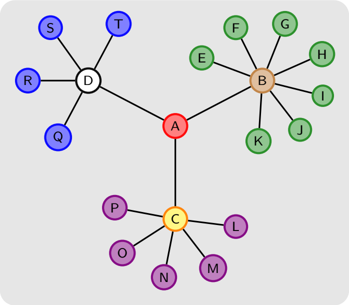
Teacher - Teaching Assistant (TA) - Student Network



Example: Teacher - TA - Student Network

- Possible **positions/blocks** of actors (nodes) in the current Example:
  - Position/Block 1 (Teacher): {A}
  - Position/Block 2 (TAs): {B,C,D}
  - Position/Block 3 (Students): {E,F,G,...,R,S,T}

Equitable Partition [1] on Teacher - TA - Student N/w

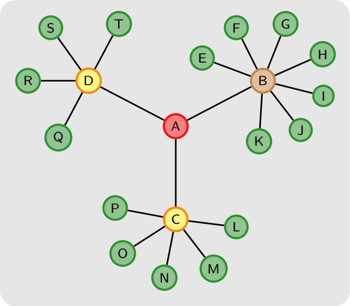


Equitable Partition of Teacher - TA - Student N/w

- For 2 actors (nodes) to be at **same position/block** under Equitable partition equivalence relation
  - They need to be connected to other positions with **exactly the same degree!**
- Equitable Partitioning leads to **7 positions/blocks** of the Teacher - TA - Student N/w:
  - {{A},{B},{C},{D},{E,F,G,H,I,J,K},{L,M,N,O,P},{Q,R,S,T}}

[1] B. D. McKay, "Practical graph isomorphism," Congressus Numerantium, vol. 30, pp. 45-87, 1981

ϵ - Equitable Partition [2] on Teacher - TA - Student N/w



1 - Equitable Partition of Teacher - TA - Student N/w ($\epsilon = 1$)

- For 2 actors (nodes) to be at **same position/block** under the ϵ - Equitable partition equivalence relation
    - They get a **freedom of ϵ degrees** in number of connections to every other position/block
- ϵ - Equitable Partitioning with $\epsilon = 1$ leads to **4 positions/blocks** of the Teacher - TA - Student N/w:
    - {{A},{B},{C,D},{E,F,G,I,J,K,L,M,N,O,P,Q,R,S,T}}

[2] Kate and Ravindran, 2009

Advantages of ϵ-Equitable Partition

- Two nodes are equivalent if the strengths of connections to corresponding blocks is within ϵ of each other
  - Can consider a zero strength connection equivalent to an ϵ strength connection
- Advantages:
  - More inclusive
  - ϵ of zero yields an equitable partition
  - Corresponds to intuitive notions
- Problem:
  - Uniqueness?

Fast ϵ-Equitable Partition Algorithm

- New algorithm with better heuristics to find ϵEP
- Based on modification of McKay's Equitable Partition [3] Algorithm
- Running time complexity of proposed algorithm is
  - $O(n^2 \log n)$ for *Sparse Graphs*
  - $O(n^3)$ for *Dense Graphs*
  - Efficient in Practice
  - Complexity of ϵEP Algorithm from [2] is $O(n^3)$ for all class of graphs

Fast ϵ-Equitable Partition Algorithm

Algorithm 1 Fast ϵ-Equitable Partition

Input: graph G, *ordered unit* partition π, epsilon ϵ
Output: ϵ-equitable partition π

active = indices(π)
while (active $\neq \phi$) **and** (π is **not** $discrete$) **do**
  $idx = min$(active)
  active = active $\setminus \{idx\}$
  $f(u) = deg_{\mathrm{G}}(u, \pi[idx])\ \forall u \in V$
  $\pi' = \mathrm{SPLIT}(\pi, f, \epsilon)$
  active = active $\cup$ [*ordered* indices of newly *split* cells from π', while replacing (*in place*) the indices from π which were *split*]
  $\pi = \pi'$
end while

Scalable ϵ-Equitable Partition Algorithm

- Each iteration of the Fast ϵEP Algorithm translates nicely to MapReduce [5] paradigm
  - Implemented the most computationally intensive step of the ϵEP algorithm, computation of function f , as a $\textsc{Map}$ operation. f maps every vertex $u \in V$ to its degree to current active cell C_a
  - Algorithm is *iterative* in nature! Frameworks introduce high job setup overhead times across the iterative MapReduce steps
  - Implemented a bare minimum MapReduce framework using open source tools
  - Execution time empirical studies on random power-law graphs for the proposed algorithm show encouraging results

[5] J. Dean and S. Ghemawat, MapReduce: Simplified Data Processing on Large Clusters, Communications of the ACM, 51 (2008), pp. 107113.

Scalable ϵ-Equitable Partition Algorithm

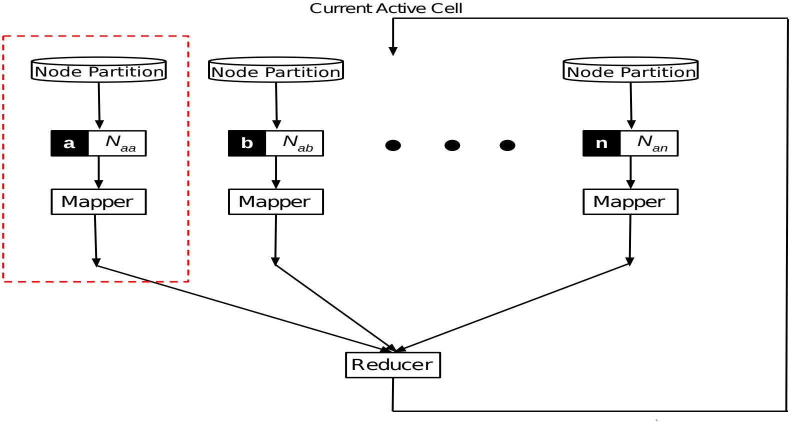
Algorithm 2 MapReduce step of the Parallel ϵ-Equitable Partition

class MAPPER
    method INITIALIZE()
        $C_a \leftarrow$ *Current Active Cell*
    method MAP(id n, vertex N)
        $d \leftarrow |N.AdjacencyList \cap C_a|$
        EMIT(id n, value d)

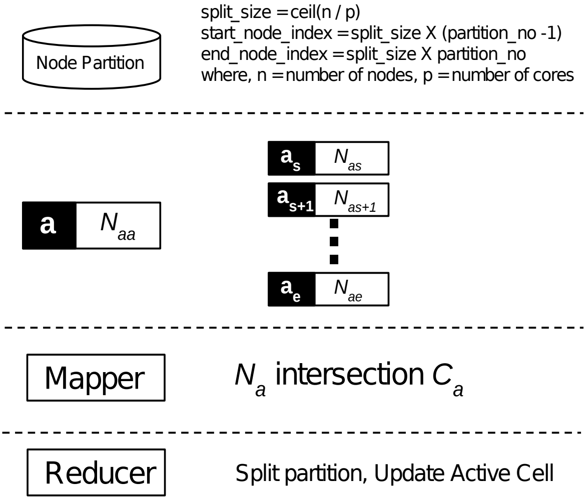
---

class REDUCER
    method REDUCE()
        SPLIT(π, f, ϵ)
        UPDATE(*active*)
        UPDATE(π)

Scalable ϵ-Equitable Partition MapReduce



Scalable ϵ-Equitable Partition MapReduce Step



Scalable ϵEP Algorithm: Computational Aspects

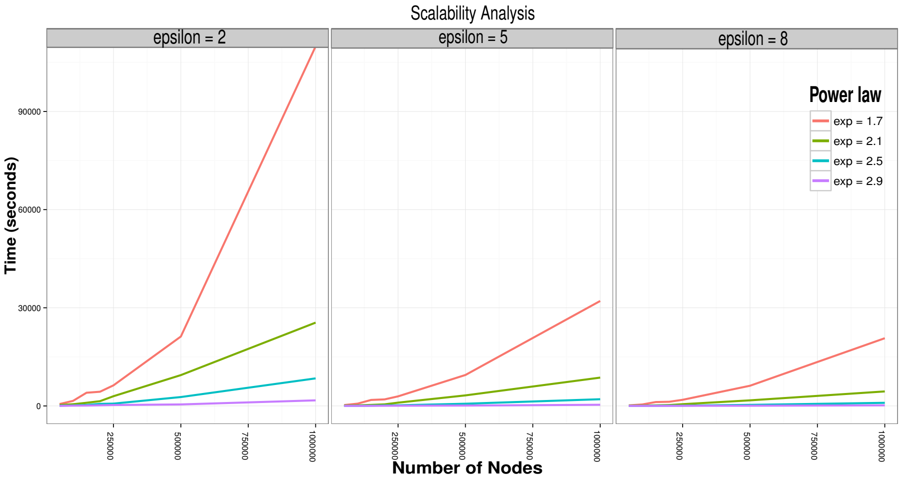
- Complexity bound for sparse graphs is $O(\frac{n^2}{p} \log n)$, where p is the number of cores
- Total additional overhead $i \times c$, data copy cost c for i iterations
- Data copy cost c is small and constant in average, since we copy only the active cell data C_a. Node data is cached on each compute node, hence doesn't require copy overhead
- i is proportional to size of the graph G, inversely proportional to ϵ

γ	2.9	2.5	2.1
Complexity	$O(n)$	$O(n \log n)$	$O(n^2)$

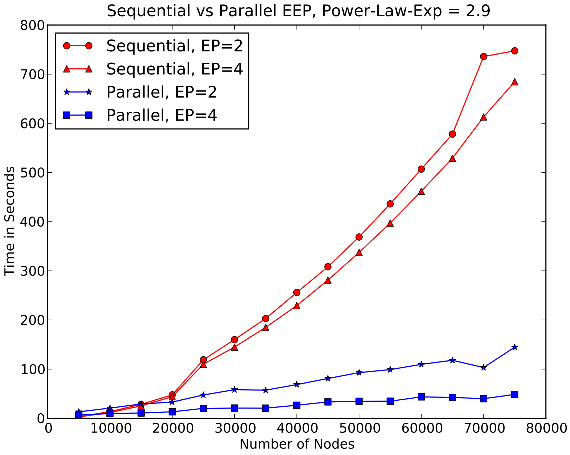
Table: Curve Fitting Results, $\epsilon = 5$

- For $\gamma = 2.1$, difference between sum squared residual for the curves $n \log n$ and n^2 was negligible

Scalability Analysis



EEP: Sequential vs Parallel on Sparse Graphs, $\gamma = 2.9$



Node Evolution Studies

- Datasets Used - Facebook (New Orleans Regional Network) [6] and Flickr [7]

Table: Facebook Dataset Details

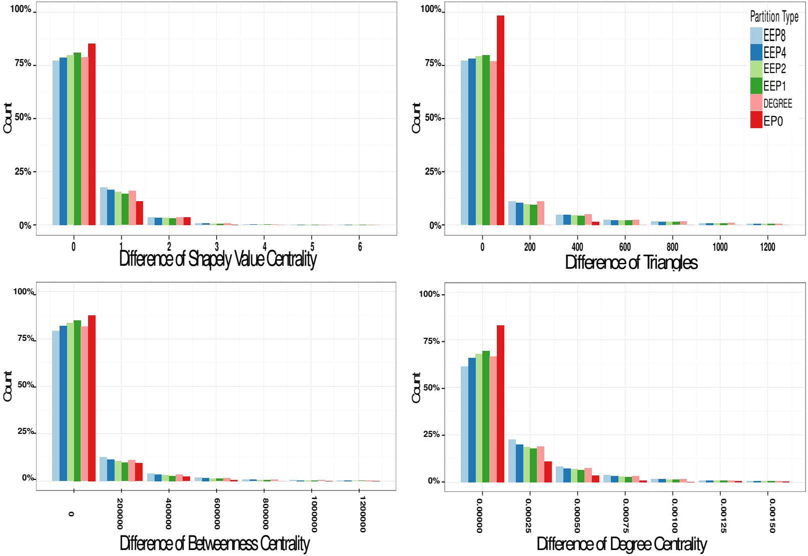
Graph	Vertices	Edges	Upto Date
1	15273	80005	2007-06-22
2	31432	218292	2008-04-07
3	61096	614796	2009-01-22

Table: Flickr Dataset Details

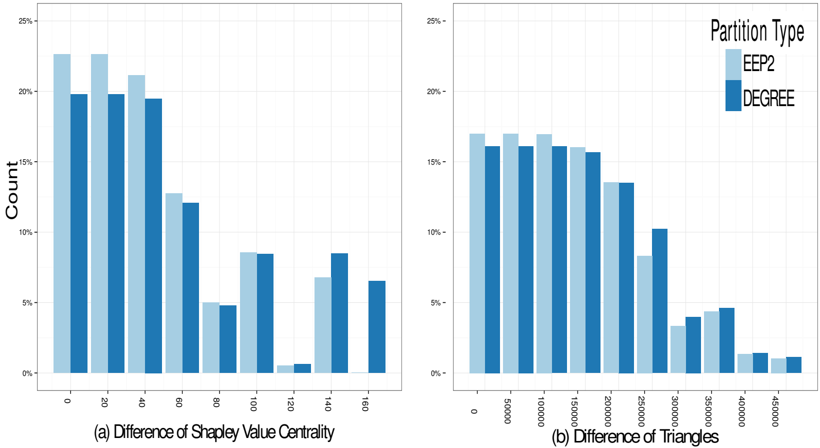
Graph	Vertices	Edges	Upto Date
1	1277145	6042808	2006-12-03
2	1856431	10301742	2007-05-19

[6] B. Viswanath, A. Mislove, M. Cha, and K. P. Gummadi, On the Evolution of User Interaction in Facebook, in Proceedings of the 2nd ACM workshop on Online Social Networks, ACM, 2009.
[7] A. Mislove, H. S. Koppula, K. P. Gummadi, P. Druschel, and B. Bhattacharjee, Growth of the Flickr Social Network, in Proceedings of the first workshop on Online Social Networks, ACM, 2008, pp. 2530.

Results: Co-Evolving node pairs for Facebook Graph $G_1 \to G_3$



Results: Co-Evolving node pairs for Flickr Graph



Results: Evolution of Positions

Epsilon	0	1	2	3	4	5	6	7	8	d*
$G1 \rightarrow G2$	59.59	66.19	76.60	83.00	86.57	89.43	91.29	92.88	94.18	86.93
$G1 \rightarrow G3$	54.11	57.17	69.33	76.61	80.85	84.37	86.60	88.95	90.72	79.42
$G2 \rightarrow G3$	56.88	67.18	76.80	82.12	85.55	87.99	89.87	91.48	92.93	78.11

Table: Percentage of ϵEP overlap using the Partition Similarity Score for time evolving graphs of the Facebook Network. ϵ varied from 0 to 8, $\epsilon = 0$ corresponds to an equitable partition. d* denotes the partition based on degree.

$$sim(\pi_t, \pi_{t+\delta t}) = \frac{1}{2}\left[\left(\frac{N - |\pi_t \cap \pi_{t+\delta t}|}{N - |\pi_t|}\right) + \left(\frac{N - |\pi_t \cap \pi_{t+\delta t}|}{N - |\pi_{t+\delta t}|}\right)\right] \quad (1)$$

Conclusion and Future Work

- Conclusions
  - Proposed a new algorithm with better heuristics to find ϵ-equitable partition of a graph
  - Proposed a scalable version using MapReduce paradigm
- Future Scope of Work
  - Explore the implied advantage of our Parallel ϵEP Algorithm to find the *coarsest equitable partition* of very large graphs for an $\epsilon = 0$.
  - Explore algorithms for positional analysis of very large graphs using vertex-centric computation paradigms

Thank you!

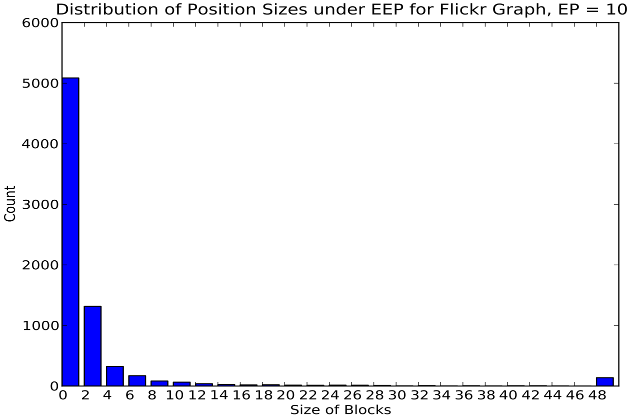
Thank you!

Split function for Fast ϵEP Algorithm

Algorithm 3 Split Function

```
split(π, f, ϵ)
idx = 0
for each currentCell in π do
    sortedCell = SORT(currentCell) using f as the comparison key
    currentDegree = f(sortedCell[0])
    for each vertex in sortedCell do
        if (f(vertex) − currentDegree) ≤ ϵ then
            Add vertex to cell π_s[idx]
        else
            currentDegree = f(vertex)
            idx = idx + 1
            Add vertex to cell π_s[idx]
        end if
    end for
    idx = idx + 1
end for
return(π_s)
```

Results: Distribution of Block Size, Flickr Graph EP $= 10$



Results: Distribution of Block Size, Flickr EP = 2