Scalable Parallel Algorithms for Clustering Large-Scale Biological Graphs

Inna Rytsareva, Ananth Kalyanaraman (Advisor)

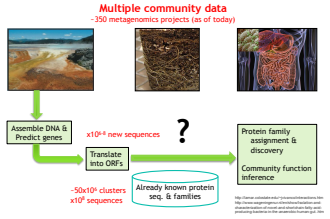
School of Electrical Engineering and Computer Science, Washington State University Pullman, WA, USA

ABSTRACT

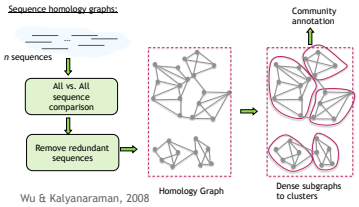
Clustering is one of the advanced analytical functions that has an immense potential to transform the knowledgespace when applied particularly to a data-rich domain such as computational biology. The computational hardness of the underlying theoretical problem necessitate use of heuristics in practice. In this study, we evaluate the application of a randomized sampling-based heuristic called shingling on unweighted biological graphs and propose a new variant of this heuristic that extends its application to weighted graph inputs and is better positioned to achieve qualitative gains on unweighted inputs as well. We also present parallel algorithms for this heuristic using the MapReduce paradigm. Experimental results on subsets of a medium-scale real world input (containing up to 10.3M vertices and 640M edges) demonstrate significant qualitative improvements in the reported clustering, both with and without using edge weights. Furthermore, performance studies indicate near-linear scaling on up to 4K cores of a distributed memory supercomputer.

OBJECTIVES

Characterizing protein families in environmental microbial communities



Problem formulation



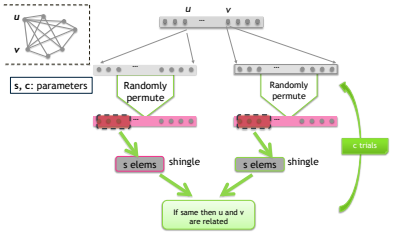
Wu & Kalyanaraman, 2008

Challenges

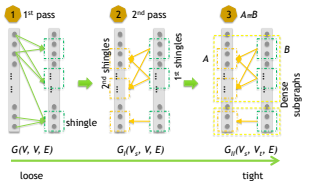
- Theoretical formulations are NP-Hard (Feige et al., 2001)
  - Need for efficient heuristics
- Lack of parallel tools
  - Developments recent/ongoing
  - (e.g., 10th DIMACS challenge)
- To allow for overlaps or not
- Metrics for quality evaluation

METHODS

The Shingling heuristic (Broder et al., 2000)



Dense subgraph detection using the Shingling heuristic (Gibson et al., 2005)



Limitations of the standard Shingling heuristic

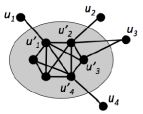
- Large number of singletons
- Inability to handle edge weights
  - e.g., degree of sequence similarity
- Scaling to very large data sets
  - Parallelization required for reducing time to solution

Improving sensitivity by singleton reduction

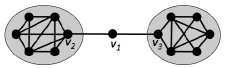
Cause: Low degree vertices tend to be left out by Shingling sampling approach

Two contrasting scenarios for low degree vertices:

(+)ve case: Periphery

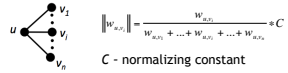


(-)ve case: Chimeric



Normalization procedure for handling edge weights

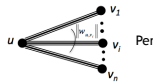
Step 1) Normalize edge weight for every edge (if not already normalized)



$$\|w_{u,v_i}\| = \frac{w_{u,v_i}}{w_{u,v_1} + \ldots + w_{u,v_i} + \ldots + w_{u,v_n}} * C$$

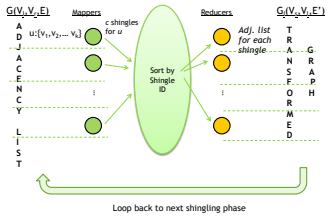
C - normalizing constant

Step 2) Generate $w_{u,vi}$ copies for edge (u, v_i)



Perform shingling on the multigraph

Parallel MapReduce Algorithm



EXPERIMENTAL RESULTS

Experimental Setup

environmental microbial communities

Input label	Number of edges	Number of vertices
UBC-25M	25 x 10^6	3,965 x 10^3
UBC-50M	50 x 10^6	4,525 x 10^3
UBC-100M	100 x 10^6	6,795 x 10^3
UBC-200M	200 x 10^6	7,336 x 10^3
UBC-400M	400 x 10^6	8,958 x 10^3
UBC-640M	640 x 10^6	10,346 x 10^3

Experimental platform:

- Hopper (NERSC-6): Cray XE6
  - 6,392 compute nodes, 153,216 cores
  - 32 GB RAM per node
  - Custom mpich-2 version 5.5.5 for Cray XE
  - MapReduce-MPI library

Qualitative Assessment

Input	Data set, Algorithm	Quality metrics		Clustering statistics		
		Modularity	Density	# non-singleton clusters	% of singletons	Largest cluster size
UBC-25M	Unweighted, Standard	0.8454	0.3126±0.2506	27,479	35.15%	21,973
	Unweighted, Normalized	0.9928	0.5603±0.3866	95,505	2.39%	25,827
	Weighted, Normalized	0.9937	0.5603±0.3866	95,505	2.12%	25,864
	Weighted, Louvain	0.9695	0.5309±0.3864	116,558	0%	13,297
	Weighted, MCL	0.9383	0.5259±0.3837	118,518	0%	5,507

Performance Study

Input graph	Run-time (in seconds) using p cores						
	p=64	p=128	p=256	p=512	p=1024	p=2048	p=4096
UBC-25M	104.4	59.81	37.21	26.66	17.5	14.8	
UBC-50M	159.87	100.43	50.89	32.66	25.96	17.84	20.69
UBC-100M		158.22	90.74	53.51	36.48	27.88	24.86
UBC-200M			110.57	60.23	43.66	30.53	31.14
UBC-400M				121.81	73.91	36.71	25.53
UBC-640M					102.49	70.78	36.08

Input: Unweighted UBC subgraphs. Run-times are for our MPI-MapReduce implementation.

Performance observations

- MapReduce implementation matters
  - MPI-MapReduce implementation is at least two orders of magnitude *faster* than Hadoop's
- Unweighted vs. weighted analysis takes about the same time
- Adjacency list implementation is about *3x faster* than edge list implementation

CONTRIBUTIONS

- ***New variant*** of the Shingling heuristic based on a *normalization technique*
  - *Reduction of # singletons*
  - Ability to handle unweighted and ***weighted*** graphs
- ***MapReduce*** algorithm for parallelizing the new variant
- Extensive experimental results
  - qualitative improvement over the standard heuristic
  - performance improvement of the MPI implementation over the Hadoop implementation

REFERENCES

1. Broder, A.Z. et al., (2000), 'Min-wise independent permutations', Journal of Computer and System Sciences, Vol. 60, pp.630-659.
2. Gibson, D., Kumar, R. and Tomkins, A. (2005), 'Discovering large dense subgraphs in massive graphs', Proceedings of the International Conference on Very Large Data Bases, pp.721-732.
3. Rytsareva, I., Kalyanaraman, A. (2013), 'Scalable heuristics for clustering biological graphs', 3rd IEEE International Conference on Computational Advances in Bio and Medical Sciences (ICCABS), pp.1-6.
4. Wu, C. and Kalyanaraman, A. (2008), 'An efficient parallel approach for identifying protein families in large-scale metagenomic data sets', Proceedings ACM/IEEE conference on Supercomputing, pp.1-10.

ACKNOWLEDGMENT and CONTACTS

US Department of Energy DE-SC-0006516
National Science Foundation IIS 0916463
SC'13 Student Volunteer Scholarship
ACM Microsoft Research Travel Award
This research used resources of the National Energy Research Scientific Computing Center, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231

Inna Rytsareva: inna.rytsareva@email.wsu.edu
Ananth Kalyanaraman: ananth@eecs.wsu.edu