Parallel Complex Network Partitioning

George M. Slota Kamesh Madduri (advisor)
Department of Computer Science and Engineering
The Pennsylvania State University
University Park, PA, USA
Email: {gms5016, madduri}@cse.psu.edu

I. Introduction

A large number parallel graph analytics follow a bulk synchronous parallel (BSP) model: periods of parallel computation followed by periods of parallel communication. To maximize parallel efficiency when a graph is distributed across a cluster, we want a partitioning of the input graph that balances both work and memory (proportional to number of vertices and edges per process) and communication (proportional to total edge cut and maximal edge cut per process). Traditional multi-level partitioners are unable to satisfy all of these requirements and are heavy-weight in terms of computational and memory requirements. This work introduces PuLP, an iterative partitioning methodology for small-world graphs that can simultaneously handle multiple constraints and multiple objectives. Partitions produced by PuLP are equal to or better than state-of-the-art partitioners in terms of edge cut. PuLP also runs very fast, being capable of partitioning multi-billion edge graphs in minutes on a single compute node.

II. PuLP Algorithm Overview

We present PuLP: Partitioning Using Label Propagation. We utilize the well-known *label propagation* technique [7], combined with iterative balancing and refinement stages to partition small-world networks. Label propagation allows us to exploit the community structure inherent in many small-world networks, while an iterative approach lets us to handle multiple constraints and objectives. Due to the near-linear work efficiency of label propagation, our partitioner runs very quickly and with low overhead.

Algorithm 1 PuLP Multi-Constraint Multi-Objective Algorithm

```
Initialize p random partitions.
Execute degree-weighted label propagation.
for k1 iterations do
    for k2 iterations do
        Balance partitions to satisfy constraint 1.
        Refine partitions to minimize objective 1.
    for k3 iterations do
        Balance partitions to satisfy constraint 2
          and minimize objective 2.
        Refine partitions to minimize objective 1.
```

An overview of the PuLP algorithm for multiple constraints and multiple objectives is given by Algorithm 1. In this algorithm, we wish satisfy user-defined vertex balance (constraint 1) and edge balance (constraint 2), while minimizing both the total edge cut (objective 1) and maximal per-part edge cut (objective 2). We term this algorithm as PuLP-MM, for PuLP multi-constraint multi-objective. For the simpler problem of only two constraints and a single objective of total edge cut minimization, we have PuLP-M, for PuLP multi-constraint. Lastly, we have just PuLP, which has a single constraint on vertex balance and single objective of total edge cut. A detailed explanation of the algorithms and parameters can be found in our recent paper [8].

III. Results

We now compare our PuLP variants to two state-of-the-art partitioners METIS [4] and KaFFPa [6]. We compare to k-way METIS with single and multiple constraints (designated as METIS-M), as well as the parallel version ParMETIS. We run KaFFPa with the `fastsocial` preconfiguration, which uses constrained label propagation for graph contraction. KaFFPa is unable to handle multiple constraints and neither code handles multiple objectives. We fix our vertex balance constraint at 10% and our edge balance constraint at 50%. The graphs used in our evaluation are given in Table I.

TABLE I
Test graphs and properties.

Network	n	m	$davg$	$dmax$	$\widetilde{D}$	Source
LiveJournal	4.8 M	43 M	18	20 K	16	[5]
R-MAT	7.7 M	133 M	35	260 K	8	[3]
uk-2005	39 M	780 M	40	1.8 M	33	[1]
Twitter	53 M	1.6 B	61	3.5 M	19	[2]

A. Performance Results

Figure 1 gives the running times of the various partitioners tested in both serial and (if possible) parallel modes, for computing 2 to 128 parts. We obtained running times on a single dual-socket Sandy Bridge compute node for all the programs except ParMETIS. For ParMETIS, we used a small cluster of 16 nodes. KaFFPa was unable to partition Twitter due to a 32-bit `int` restriction. ParMETIS failed to partition both uk-2005 and Twitter. From Figure 1, it is apparent that the PuLP variants run considerably faster than the other partitioners across most test instances.

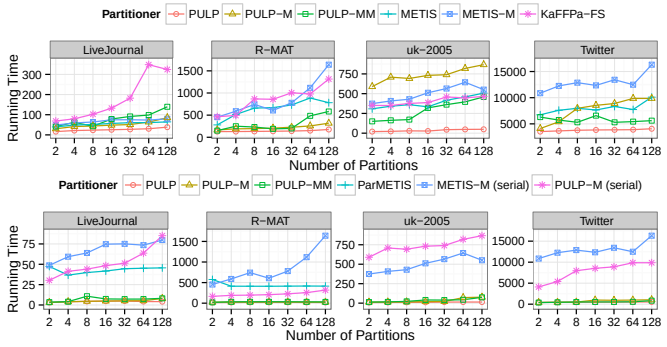

Fig. 1. Partitioning times with PuLP, METIS, and KaFFPa. Top: Comparison of **serial** running times. Bottom: Parallel and Serial running times.

TABLE II
MEMORY UTILIZATION FOR 128 PARTS.

Network	Memory Utilization for 128 Parts				**Decrease**
	METIS-M	KaFFPa	PuLP-MM	Graph Size	**vs. Best**
LiveJournal	7.2 GB	5.0 GB	0.44 GB	0.33 GB	11×
R-MAT	42 GB	64 GB	1.2 GB	1.02 GB	35×
uk-2005	41 GB	27 GB	7.9 GB	7.12 GB	3.4×
Twitter	487 GB	-	14 GB	12.2 GB	39×

Table II gives the memory utilization of PuLP multi-constraint multi-objective, METIS multi-constraint, and KaFFPa. Reported memory usage is the maximal required to partition each network into 128 parts. On these test instances, PuLP uses *at most* 3.4× less memory than the next best partitioner. PuLP shows considerable memory savings by avoiding a multi-level approach and would be capable of partitioning the Twitter graph on even a modest 16 GB machine.

B. Cut Quality

Figure 2 gives the partitioning quality resulting based on our two objectives of total edge cut (top) and max per-part cut (bottom) for PuLP-M, PuLP-MM, and METIS-M. We omit the results from METIS, KaFFPa, and PuLP because none of them produced partitions which satisfy both of our balance constraints. We report results from 2 to 128 partitions.

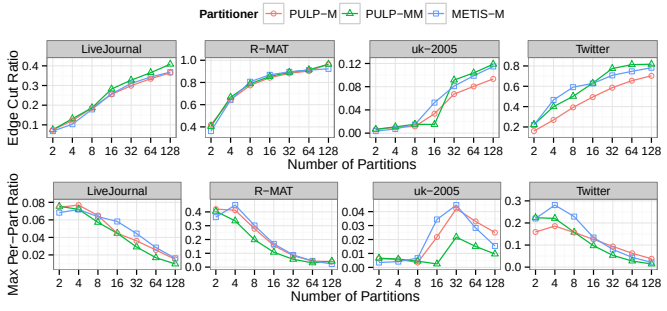

Fig. 2. Total edge cut (top) and max per-part cut (bottom) for PuLP-M, PuLP-MM and METIS-M.

From Figure 2, it can be observed that PuLP-M and PuLP-MM produce total edge cuts as good as or better than METIS-M. PuLP-M consistently produces a lower total edge cut than PuLP-MM because it optimizes directly for this metric without balancing per-part cuts as well. Looking at the bottom of Figure 2, we can see that PuLP-MM produces maximal per-part cuts consistently lower than METIS-M and PuLP-M. Taken together, the top and bottom of Figure 2 demonstrate the tradeoff from optimizing only for total edge cut and optimizing for cut balance as well.

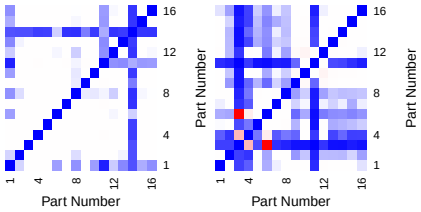

Fig. 3. Inter-part cut magnitudes produces with PuLP-MM (left) and METIS-M (right). Few cuts is blue, an average number of cuts is white, and a large number of cuts is red.

Figure 3 further demonstrates how PuLP-MM is able to balance the per-part cuts across all partitions to reduce the communication load that any single process requires. As METIS-M does not optimize for this metric, it is observed that only 1-3 of the 16 parts produced carry a large portion of the cut edges. For PuLP-MM, the cut edges are much more evenly spread out among all parts.

IV. CONCLUSIONS AND FUTURE WORK

Based on the results observed while comparing PuLP to other partitioners, utilizing an iterative label propagation-based approach for small-world graph partitioning can offer orders-of-magnitude improvement in both running time and memory utilization, while producing better cut quality under the complex objectives that modern graph analytics requires. Future work will further explore the input parameters and weighting functions governing PuLP's partitioning phases.

REFERENCES

[1] P. Boldi, B. Codenotti, M. Santini, and S. Vigna, "UbiCrawler: A scalable fully distributed web crawler," *Software: Practice & Experience*, vol. 34, no. 8, pp. 711–726, 2004.

[2] M. Cha, H. Haddadi, F. Benevenuto, and K. P. Gummadi, "Measuring user influence in Twitter: The million follower fallacy," in *Proc. Int'l. Conf. on Weblogs and Social Media (ICWSM)*, 2010.

[3] D. Chakrabarti, Y. Zhan, and C. Faloutsos, "R-MAT: A recursive model for graph mining," in *Proc. Int'l. Conf. on Data Mining (SDM)*, 2004.

[4] G. Karypis and V. Kumar, "MeTis: A software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices. version 5.1.0," http://glaros.dtc.umn.edu/gkhome/metis/metis/download, last accessed July 2014.

[5] J. Leskovec, K. Lang, A. Dasgupta, and M. Mahoney, "Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters," *Internet Mathematics*, vol. 6, no. 1, pp. 29–123, 2009.

[6] H. Meyerhenke, P. Sanders, and C. Schulz, "Partitioning complex networks via size-constrained clustering," *CoRR*, vol. abs/1402.3281, 2014.

[7] U. N. Raghavan, R. Albert, and S. Kumara, "Near linear time algorithm to detect community structures in large-scale networks," *Physical Review E*, vol. 76, no. 3, p. 036106, 2007.

[8] G. M. Slota, S. Rajamanickam, and K. Madduri, "PULP: Scalable multi-objective multi-constraint partitioning for small-world networks," in *Proc. IEEE Int'l. Conf. on Big Data (BigData)*, 2014.