PULP: Fast and Simple Complex Network Partitioning

George Slota[#,*]

Kamesh Madduri[#]

Siva Rajamanickam[*]

[#]The Pennsylvania State University

*Sandia National Laboratories

Dagstuhl Seminar 14461

November 14, 2014

Most of the results in IEEE BigData'14 paper, available at graphanalysis.info

Acknowledgments

- DOE Office of Science through the FASTMath SciDAC Institute
  - Sandia National Laboratories is a multi program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.
- NSF grants ACI-1253881, OCI-0821527
- Use of NERSC systems (supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231)

PULP: Partitioning using Label Propagation

- Multi-constraint, multi-objective method for partitioning complex networks
  - But not multilevel!

- *Constraints*: for each vertex partition, ensure that
  1. $(1 - \epsilon_L)\frac{n}{p} \leq$ partition size $\leq (1 + \epsilon_U)\frac{n}{p}$
  2. intra-partition edge count $\leq (1 + \eta_U)\frac{2m}{p}$

- *Objectives*: reduce
  1. Edge cut (total number of inter-partition edges)
  2. Max inter-partition edge cut

PULP main results

- Memory-efficient: 8-40X reduction in memory utilization compared to competing methods

- Partitioning quality comparable to Metis and ParMetis for a collection of large web crawls and social networks.

- Fast: 42 s on sk-2005 (1.8 billion edges), 530 s on Twitter (1.6 billion edges) on a 16-core, 64 GB Intel system.

What's different from KAHIP-FastSocial?

- We support per-partition vertex balance
  & per partition edge balance constraints
- We reduce total edge cut
  & reduce max # edges cut per partition
- Our method is not multilevel
  - As a consequence, faster and more memory-efficient
- Different scheme to improve quality, and for the label propagation steps
- We are yet to perform partitioning quality comparisons with parallel KAHIP.
- Only shared-memory parallelism supported in our work

What are complex networks?

- We (humans, primarily Network Science community) create them and term them *complex*
  - Mostly "virtual" or physical topology + virtual interactions
  - Complex network = Graph + Vertex/edge Heterogeneity + multi* + uncertainty + incompleteness + dynamics + vertex/edge metadata + finer-grained communication + …
- What *aren't* complex networks?
  - Road networks
  - Meshes from scientific simulations
  - Meshes with underlying 2D/3D topologies

Our definition of complex networks

- Low (O(log n)) graph diameter
- Low (typically O(1)) mean shortest path length
- Irregular vertex degree distributions
- Sparse: m = O(nlog n)
- m > 10,000
- High-dimensional

Complex networks lack *good* partitions

- Our observation: reduction in edge cut over a random partitioning may be less than 5%, for graphs with million+ edges and 32-way partitioning
- Cheeger inequality gives theoretical lower bound for conductance
- Leskovec et al. [2008] studied 100 large networks, observed presence of several *tight communities* of size 100 in most networks

However …

- There is a substantial reduction in total edge cut (over random partitioning) for *some* networks
  - Good results for web crawls with high average vertex degree (~ 100)
- Partition in an exploratory manner?

Why partition complex networks?

- Reduce overhead of data replication
- Distributed-memory graph computations
  - Reduced total edge cut may lead to reduce inter-processor communication
- In addition to vertex balance, edge balance is also very important
  - Add it as a constraint
- We have a really fast and memory-efficient approach, so why not?

PULP algorithm

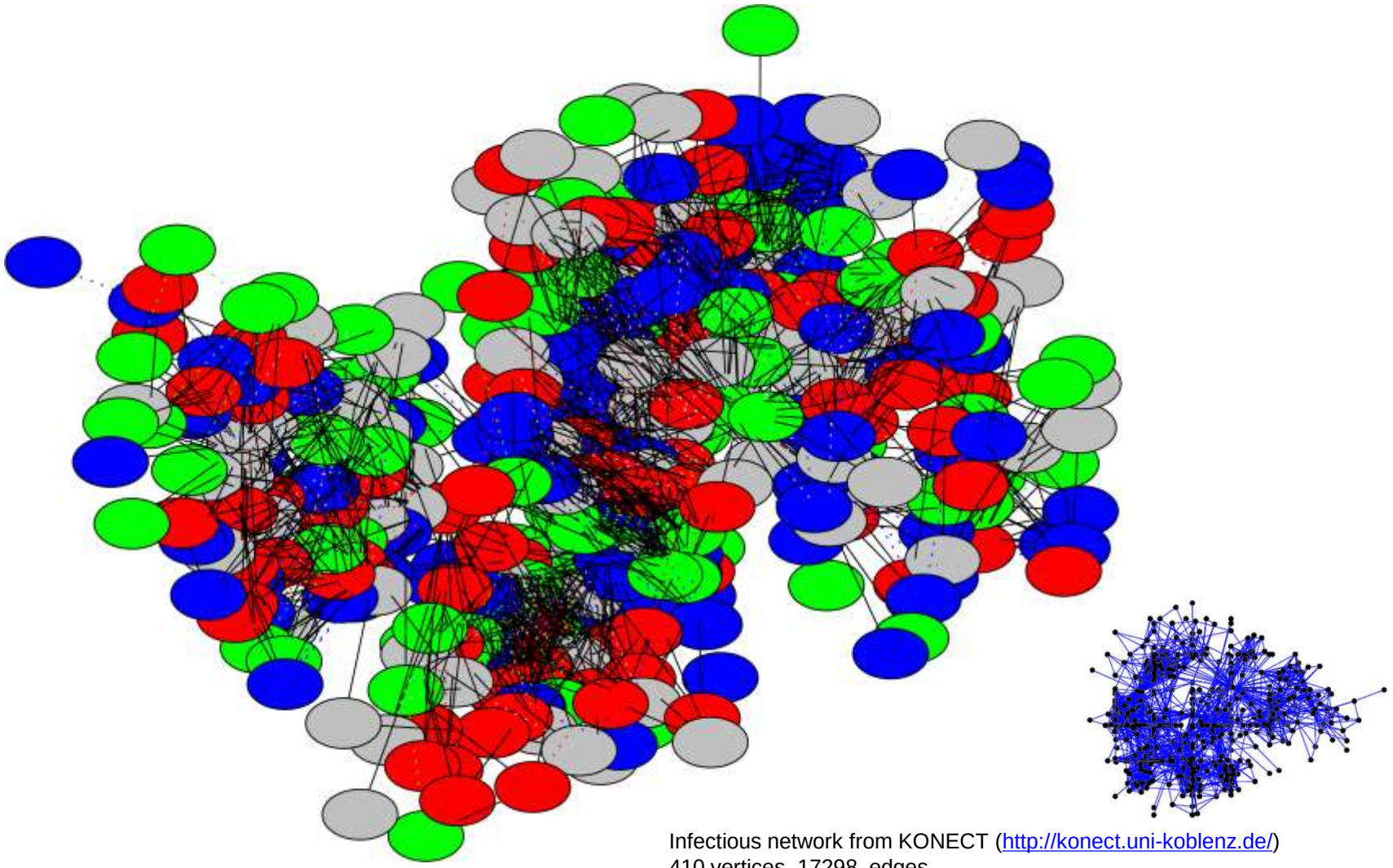
1. Assign each vertex to one of p partitions randomly
2. Degree-weighted label propagation (LP)
3. **for** k_1 iterations do
    for k_2 iterations do
        Balance partitions with LP to satisfy vertex constraint
        Heuristically improve partitions to reduce edge cut
    for k_3 iterations do
        Balance partitions with LP to satisfy edge constraint and minimize max per-part edge cut
        Heuristically improve partitions to reduce edge cut

Label propagation

- Iteratively propagate *vertex labels* along links
- Popular algorithm for community detection [Raghavan et al., 2007]: iteratively assign to each vertex the maximal per-label count over all neighbors

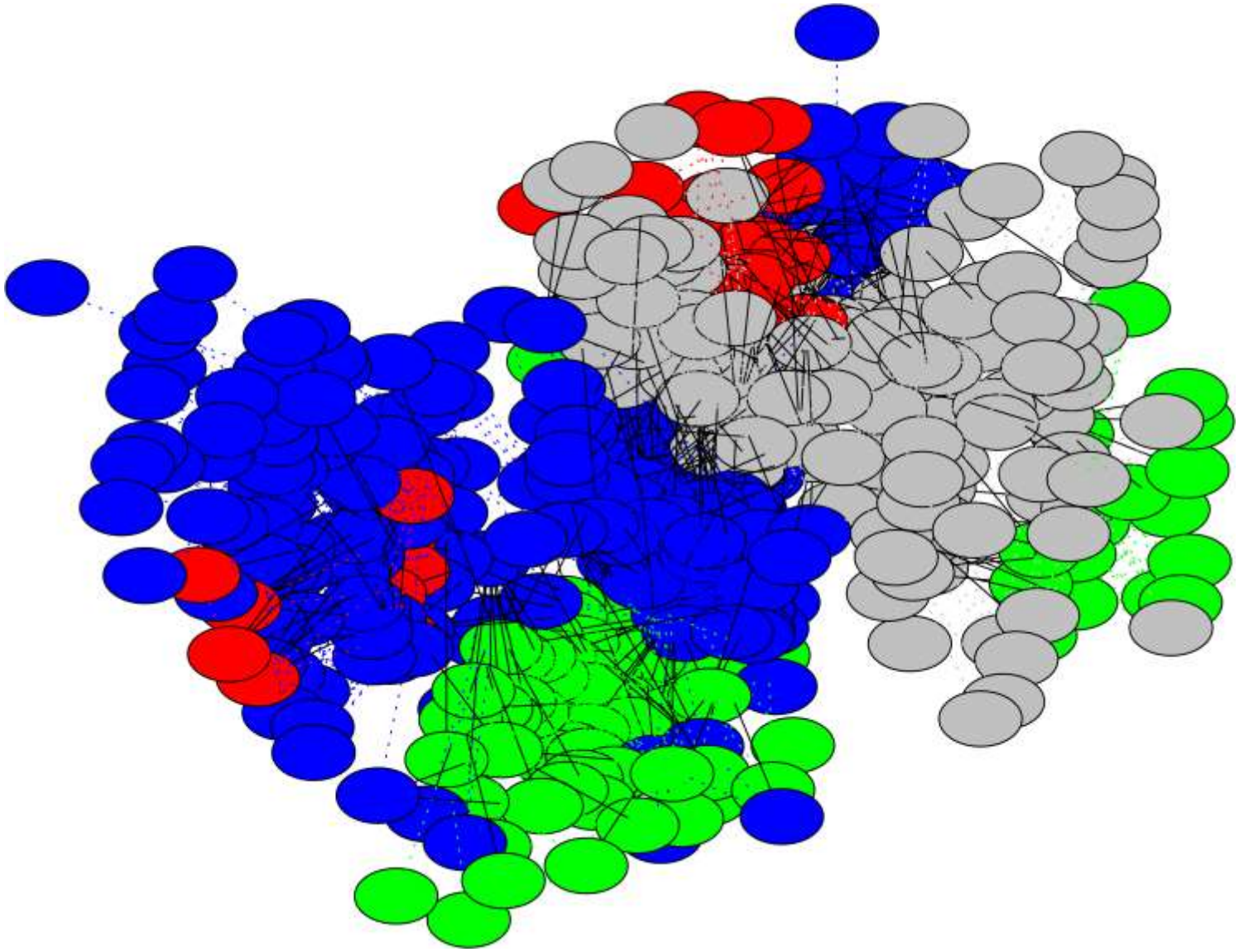
- Theoretical convergence bounds for clustered Erdos-Renyi graphs
- Fast convergence in practice
- The new betweenness centrality?

PULP with a toy example
1. Random initialization

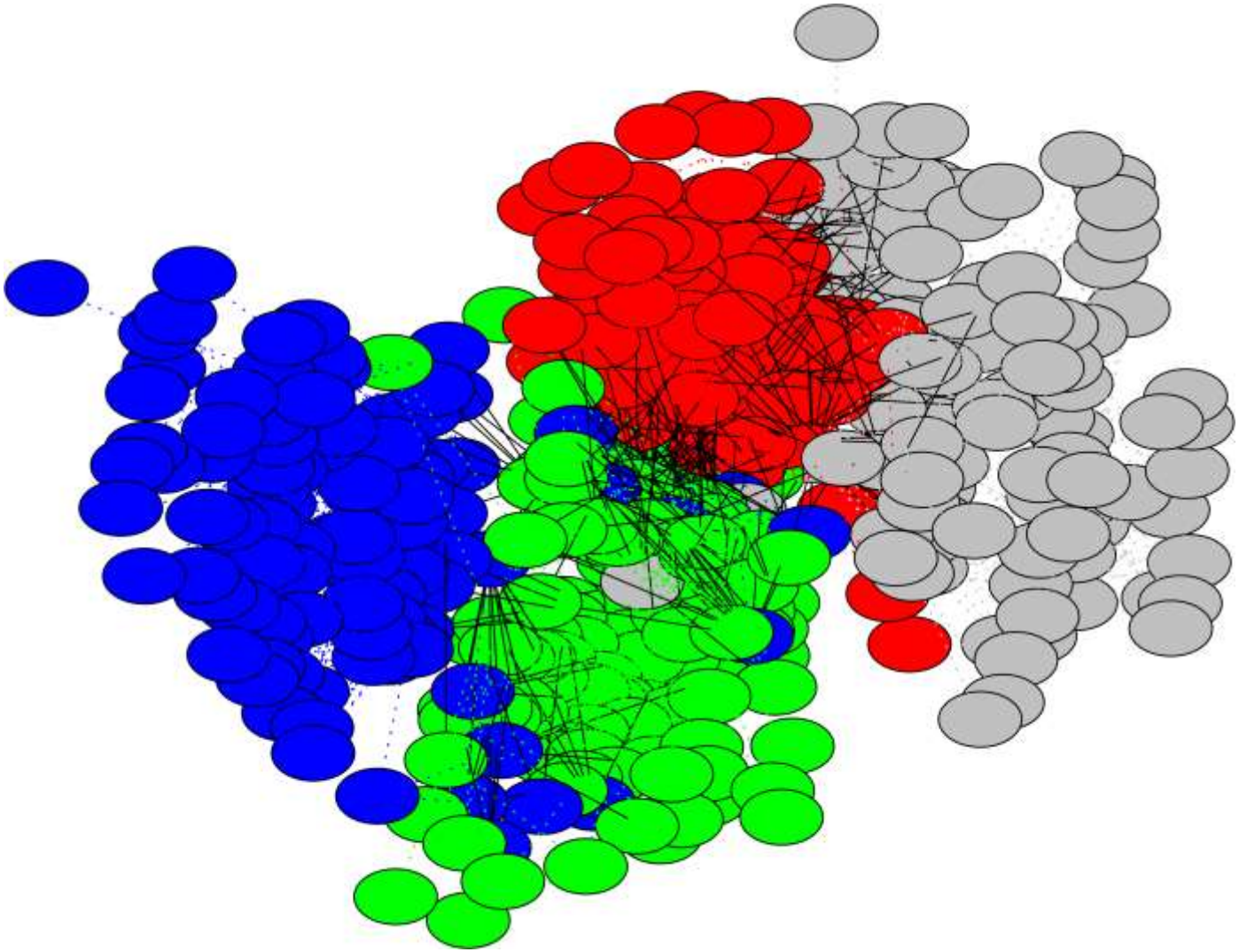


Infectious network from KONECT (http://konect.uni-koblenz.de/)
410 vertices, 17298 edges

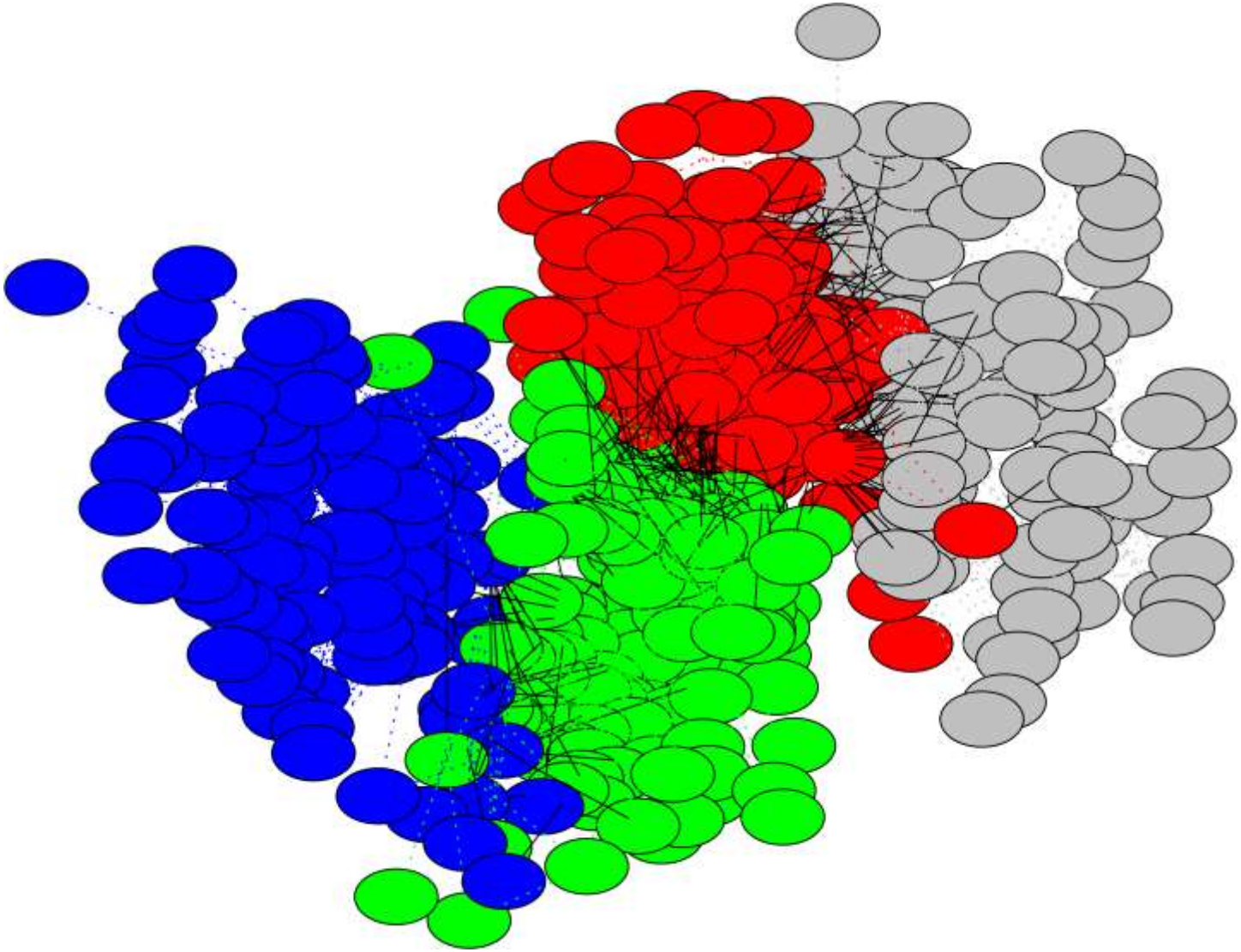
PULP Step 2. Degree-weighted label propagation



PULP Step 3. Satisfy vertex constraint, reduce total edge cut



PULP Step 4: Satisfy edge constraint, reduce max per-part edge cut



Experimental study

- Intel Xeon E5-2670 system, dual-socket, 8 cores per socket, 64 GB memory
- Test graphs
  - LAW graphs from UF Sparse matrix collection
  - Large graphs from SNAP, Koblenz, MPI repositories
  - 60K-70M vertices, 275K-2B edges
- Quality and time comparisons to Metis (v5.1.0), Metis (v5.1.0) with multiple constraints, ParMetis (v4.0.3), KaFFPa-FastSocial (v0.62, serial)
- 2-128 partitions, serial and parallel time, peak memory use

Some of the graphs used

Network	Category	n	m	$davg$	$dmax$	$\widetilde{D}$
LiveJournal	OSN	4.8 M	43 M	18	20 K	16
Orkut	OSN	3.1 M	117 M	76	33 K	9
R-MAT	SYN	7.7 M	133 M	35	260 K	8
DBpedia	RDF	62 M	190 M	6.1	7.3 M	8
WikiLinks	WWW	26 M	504 M	42	4.3 M	170
sk-2005	WWW	51 M	1.8 B	72	8.6 M	308
Twitter	OSN	53 M	1.6 B	61	3.5 M	19

Peak memory use (128-way partitioning)

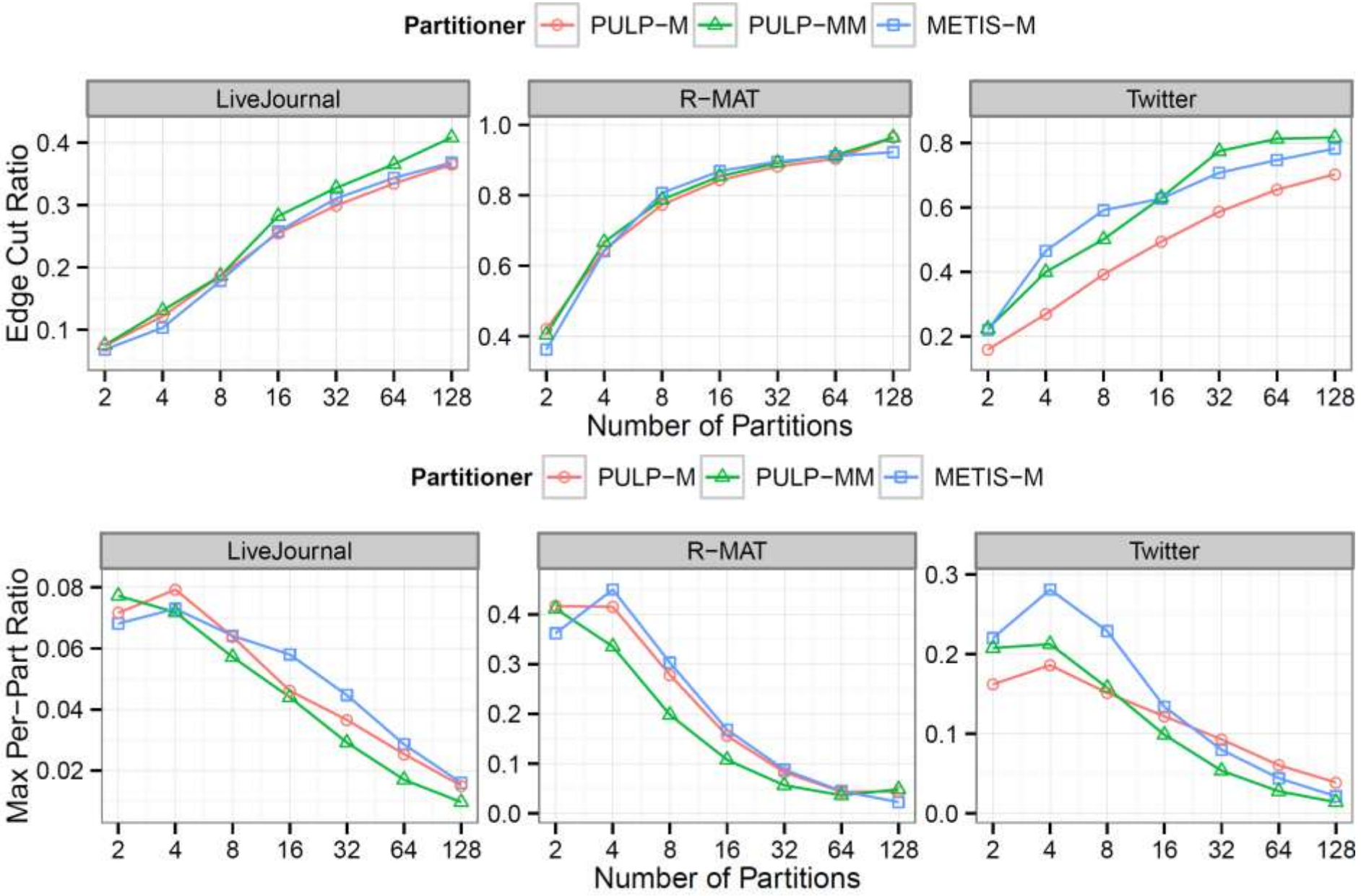
Network	Memory Utilization				Improv.
	METIS-M	KaFFPa	PuLP-MM	Graph Size	
LiveJournal	7.2 GB	5.0 GB	0.44 GB	0.33 GB	11×
Orkut	21 GB	13 GB	0.99 GB	0.88 GB	13×
R-MAT	42 GB	64 GB	1.2 GB	1.02 GB	35×
DBpedia	46 GB	-	2.8 GB	1.6 GB	16×
WikiLinks	103 GB	42 GB	5.3 GB	4.1 GB	8×
sk-2005	121 GB	-	16 GB	13.7 GB	8×
Twitter	487 GB	-	14 GB	12.2 GB	39×

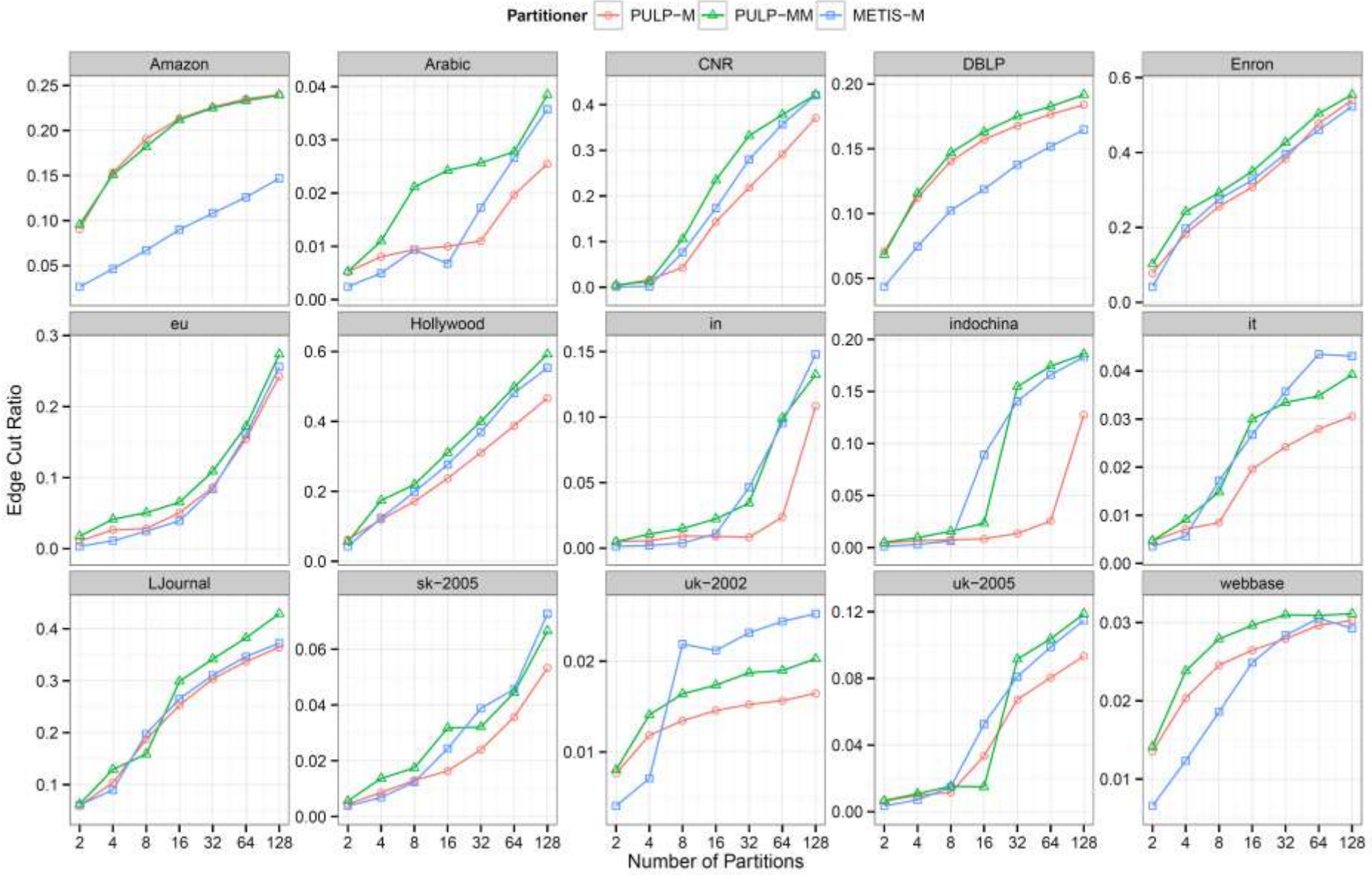
Balance constraints and other parameters

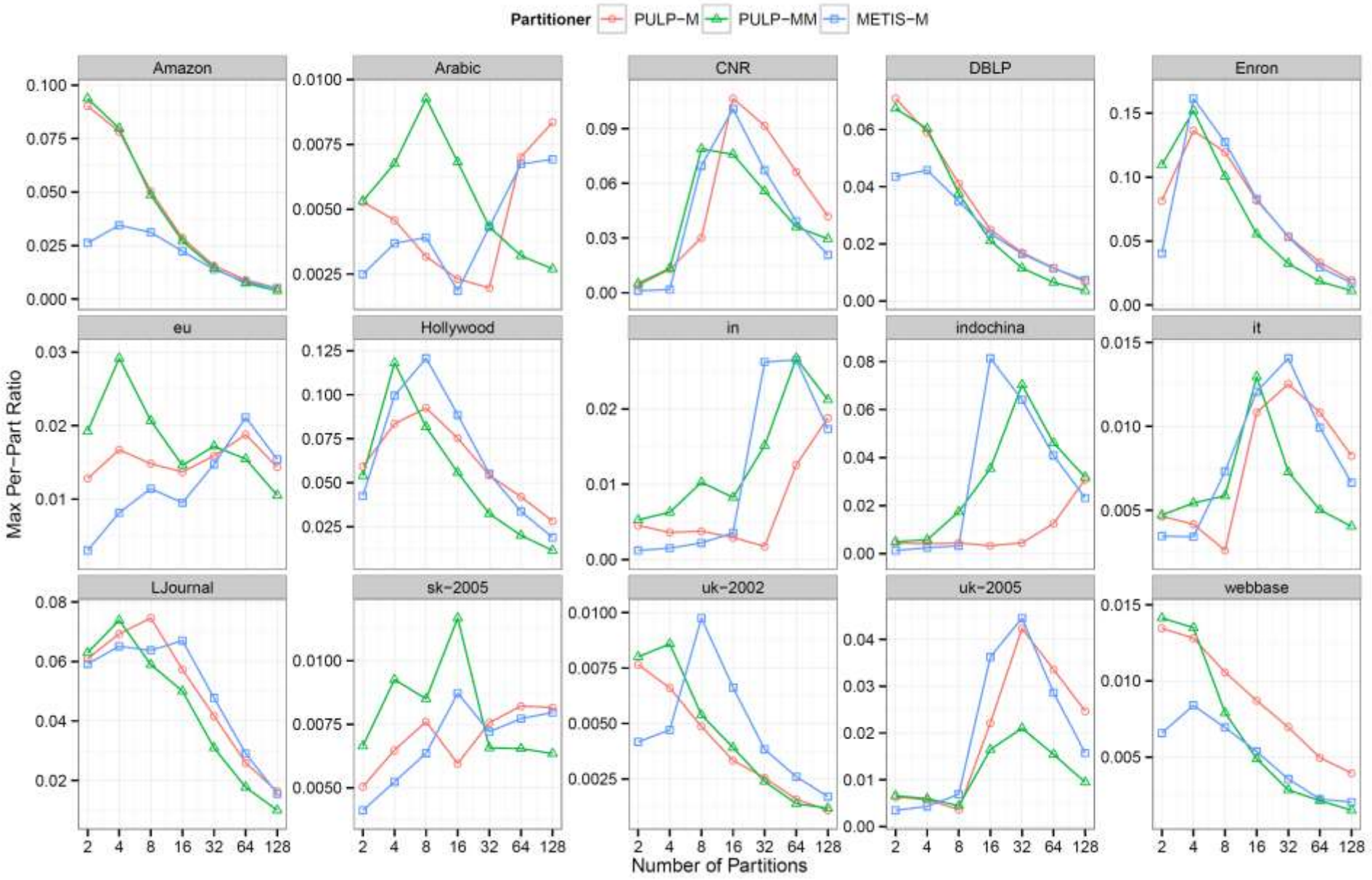
- Vertex lower balance: 0.25n/p
- Vertex upper balance: 1.1n/p
- Edge upper balance parameter: 1.5
- 3 iterations of degree-weighted label propagation
- 5 iterations of outer loop (k_1)
  - 5 iterations for objective 1 (k_2)
  - 10 iterations for objectives 1 and 2 (k_3)

Time (p = 32)

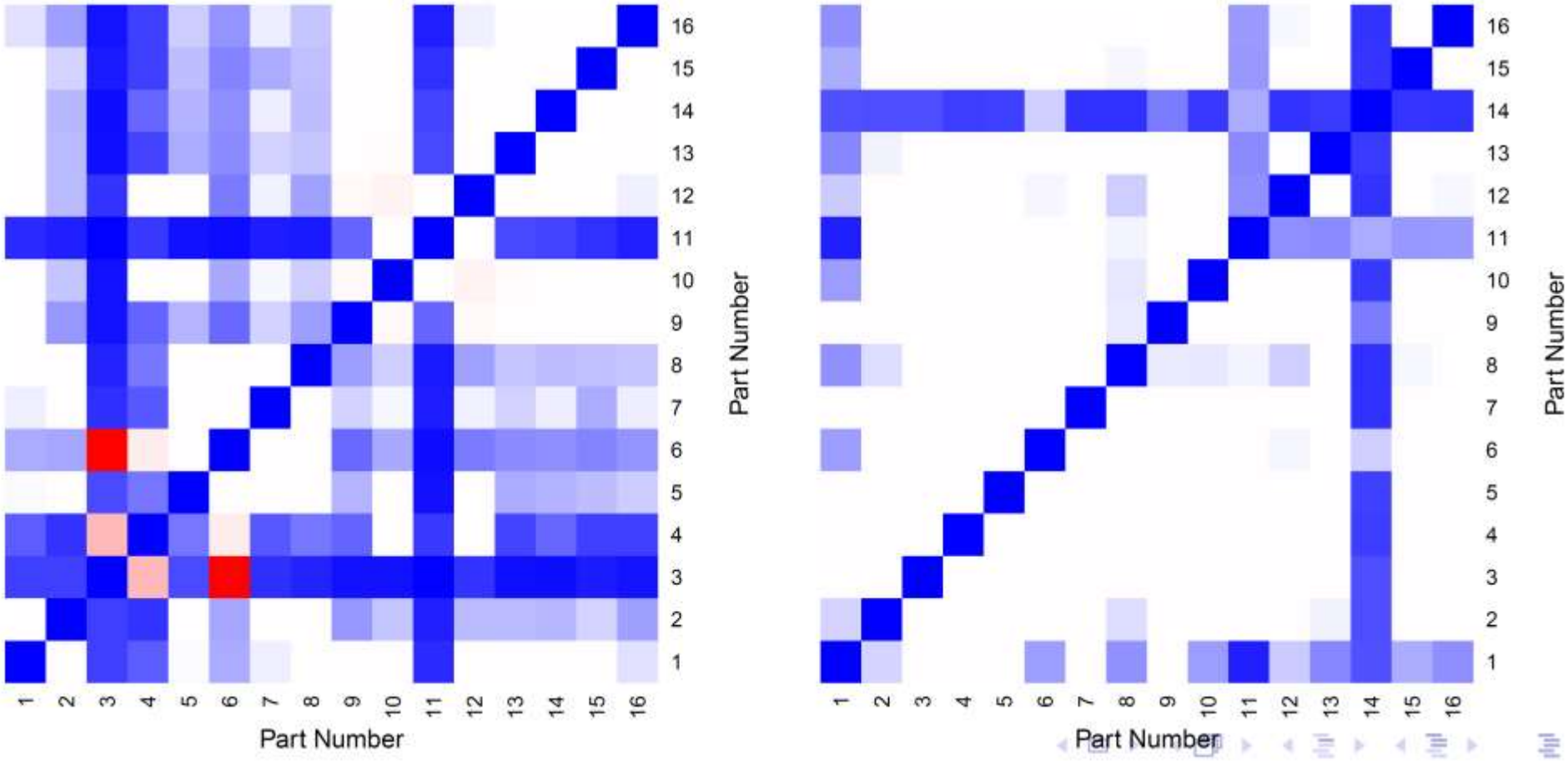
Network	Execution time (s)					PuLP-MM Speedup	
	Serial			Parallel		vs Best	All
	PuLP	METIS-M	KaFFPa	PuLP	ParMETIS	Serial	Parallel
LiveJournal	91	**75**	182	7	44	0.8×	6.1×
Orkut	**135**	170	413	12	73	1.2×	5.8×
R-MAT	**200**	778	1000	34	414	3.9×	12.2×
DBpedia	**723**	947	-	167	-	1.3×	5.6×
WikiLinks	1205	**1104**	1120	164	-	0.9×	6.7×
sk-2005	**351**	1237	-	42	-	3.5×	29.4×
Twitter	**5296**	13428	-	530	-	2.5×	25.3×


Partitioner
PULP-M
PULP-MM
METIS-M
LiveJournal
R-MAT
Twitter
Edge Cut Ratio
Number of Partitions
Max Per-Part Ratio
Number of Partitions


Partitioner
PULP-M
PULP-MM
METIS-M
Amazon
Arabic
CNR
DBLP
Enron
eu
Hollywood
in
indochina
it
LJournal
sk-2005
uk-2002
uk-2005
webbase
Edge Cut Ratio
Number of Partitions
2
4
8
16
32
64
128


Partitioner
PULP-M
PULP-MM
METIS-M
Amazon
Arabic
CNR
DBLP
Enron
eu
Hollywood
in
indochina
it
LJournal
sk-2005
uk-2002
uk-2005
webbase
Max Per-Part Ratio
Number of Partitions
2
4
8
16
32
64
128

- uk-2005 graph from LAW, METIS-M (left) vs. PuLP-MM (right)
- Blue: low comm; White: avg comm; Red: High comm
- PuLP reduces max inter-part communication requirements and balances total communication load through all tasks



Conclusions

- Our simple, non-multilevel approach seems to be performing quite well, when partitioning topologies of web crawls of high average degree
- We only use O(n)-sized data structures in the method, and do O(1) mallocs

We're just getting started … Future work

- Graphs to networks: vertex weights?
- Make it single-objective again: why do total edge cut?
- Swap order of edge and vertex balance constraints
- Parameter sensitivity
- Partitioning with vertex delegates
- Distributed-memory, scaling to larger graphs
- Flow-based partitioning improvement
- Performance of graph analytics before/after partitioning

Thank you!

- Questions? Feedback?

- madduri@cse.psu.edu
- http://www.cse.psu.edu/~madduri