GraSP: Distributed Streaming Graph Partitioning

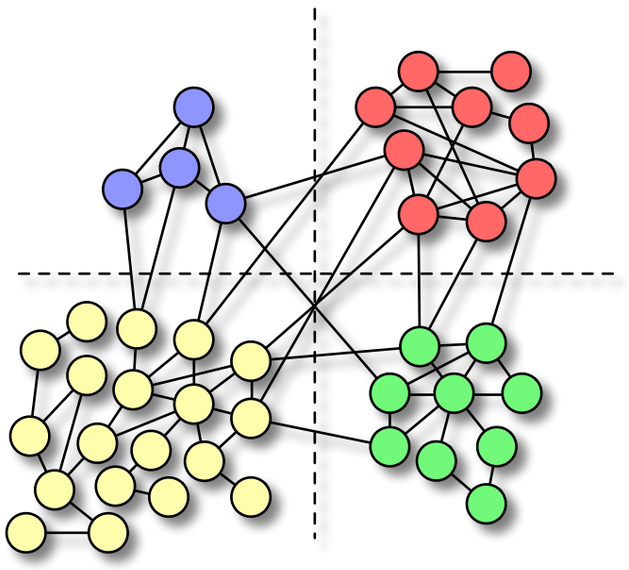
Casey Battaglino, Robert Pienta, Richard Vuduc
Georgia Institute of Technology

KDD HPGM ‘15

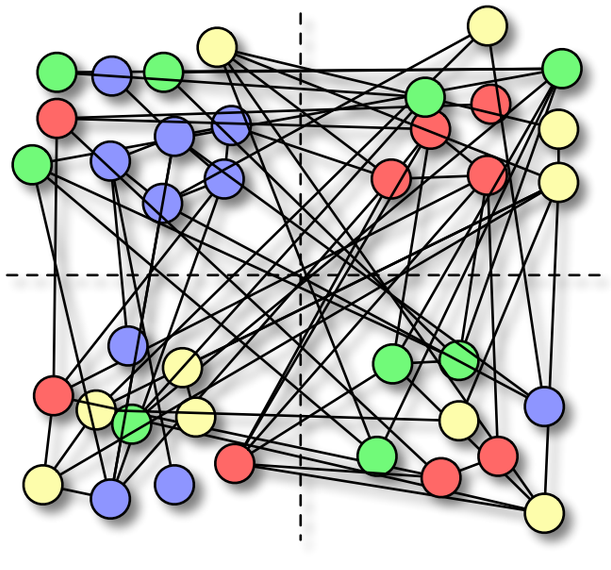
Contributions

- Parallel implementation of re-streaming partitioning algorithm
- GraSP interface is easily substituted for existing HPC partitioners such as ParMETIS
- Performance evaluation on large graphs on a state-of-the-art machine, with very favorable comparison to existing methods

Motivation

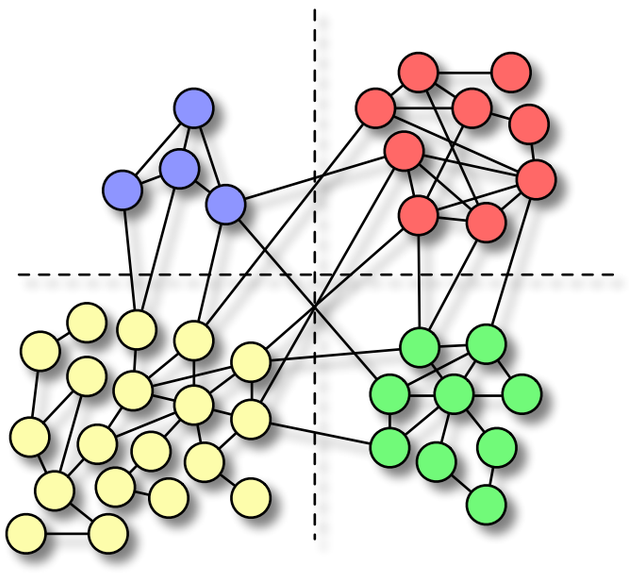


load imbalance

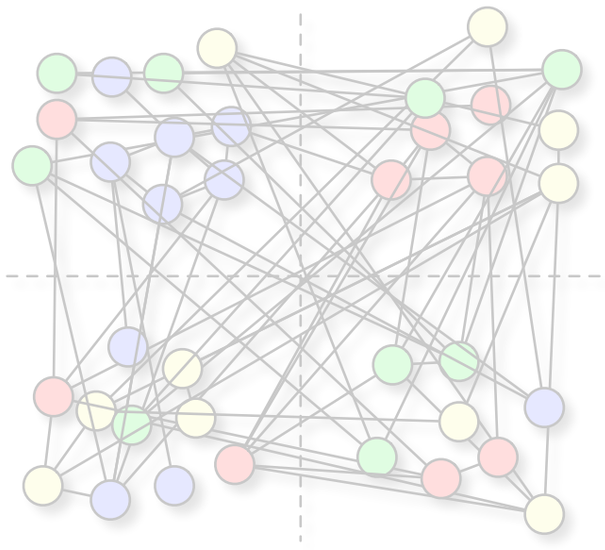


high edgecut

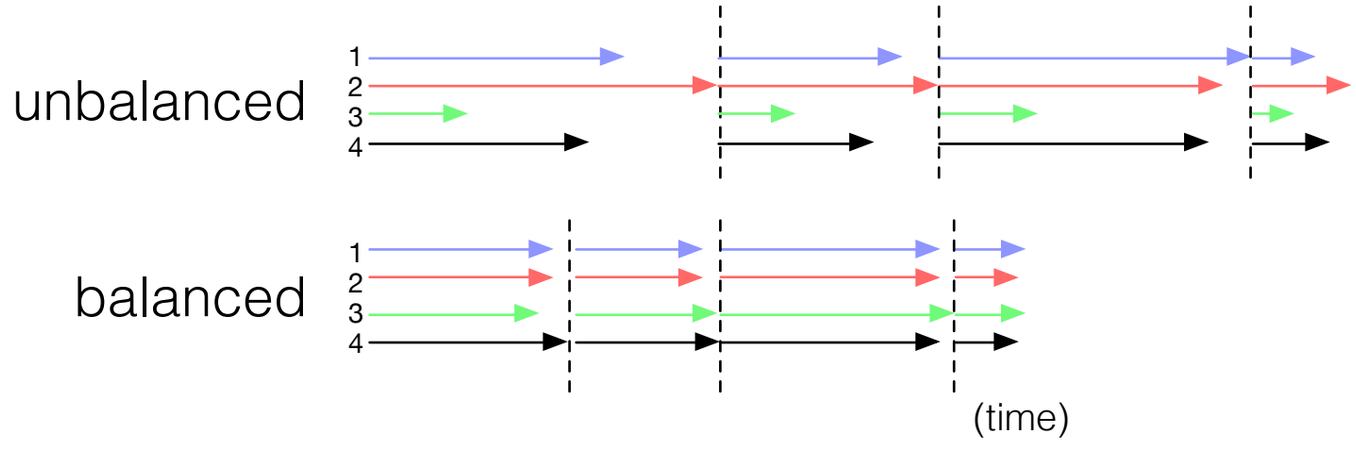
Motivation



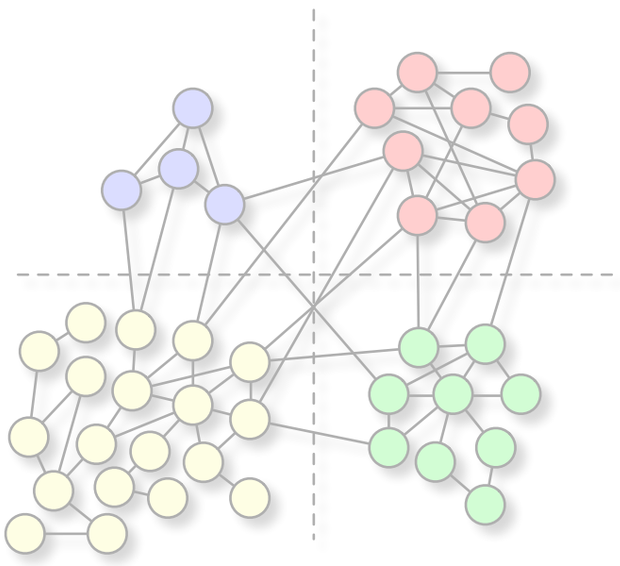
load imbalance



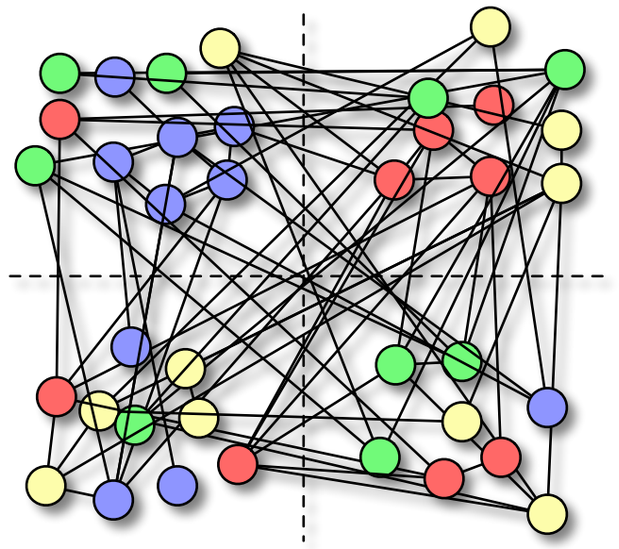
high edgecut



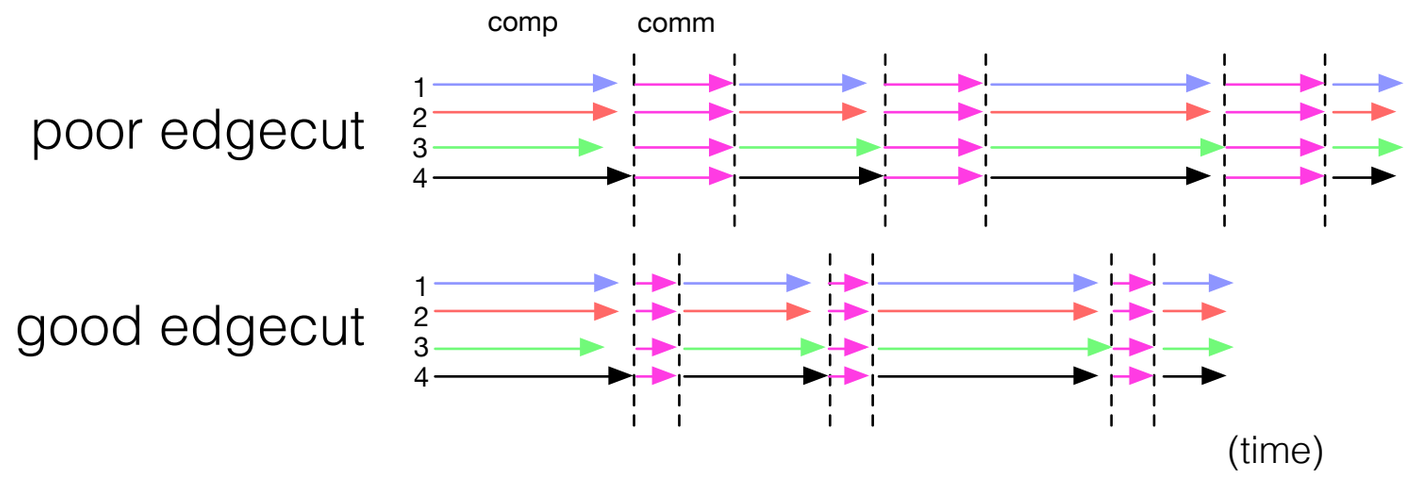
Motivation



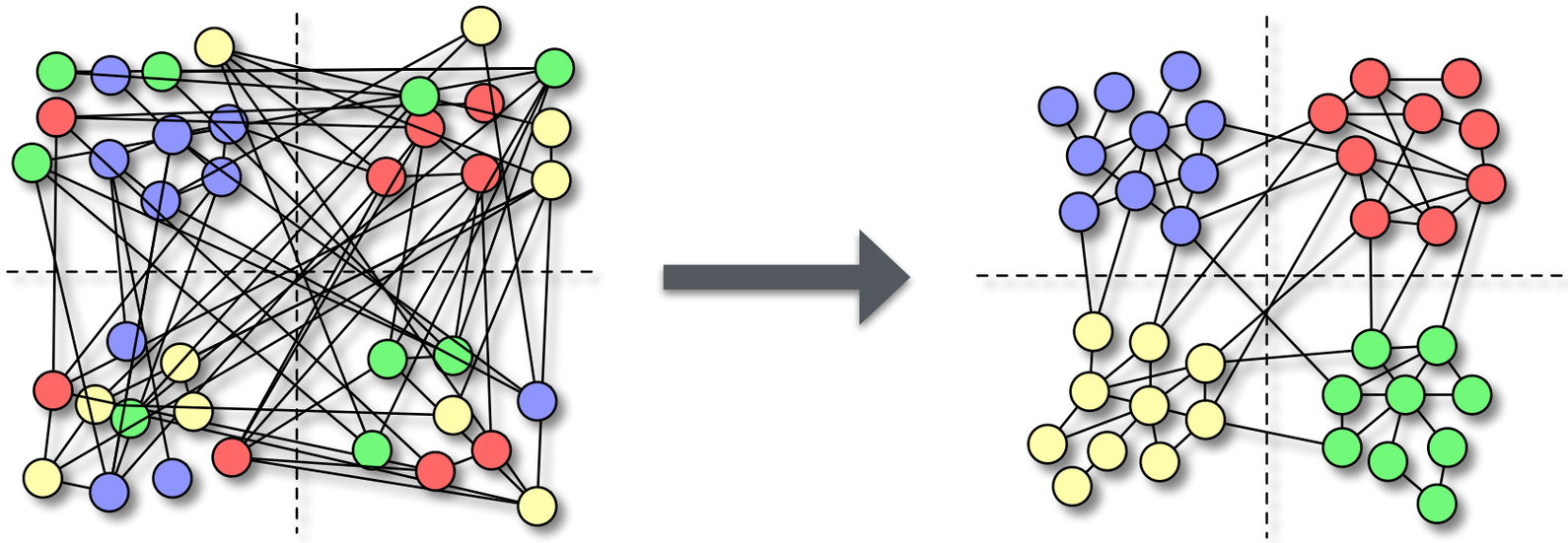
load imbalance



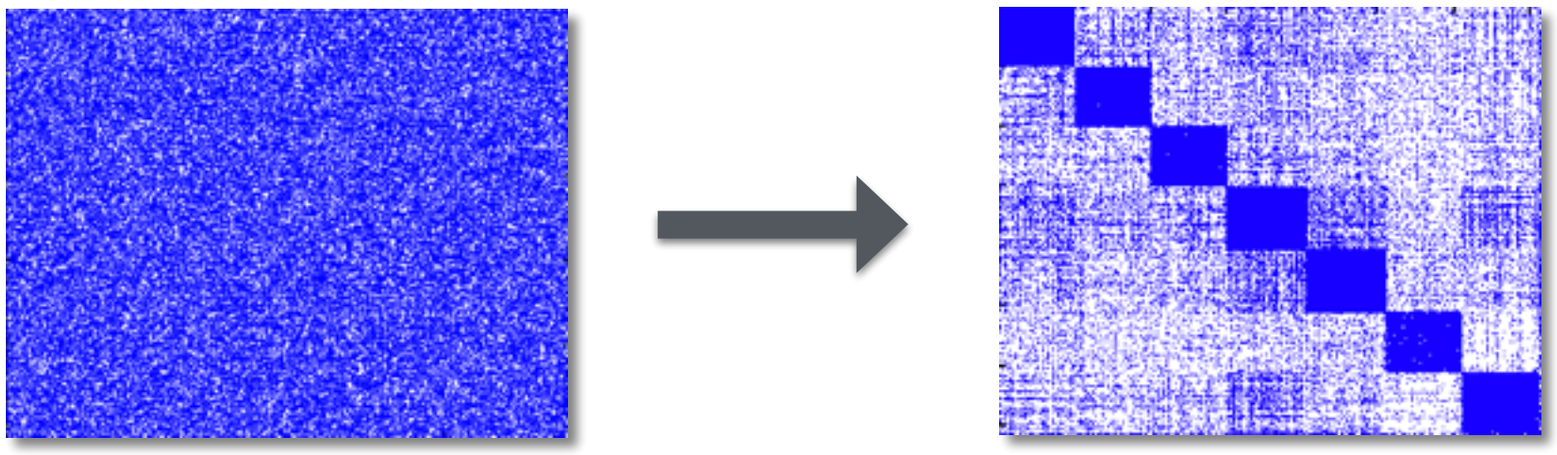
high edgecut



Motivation

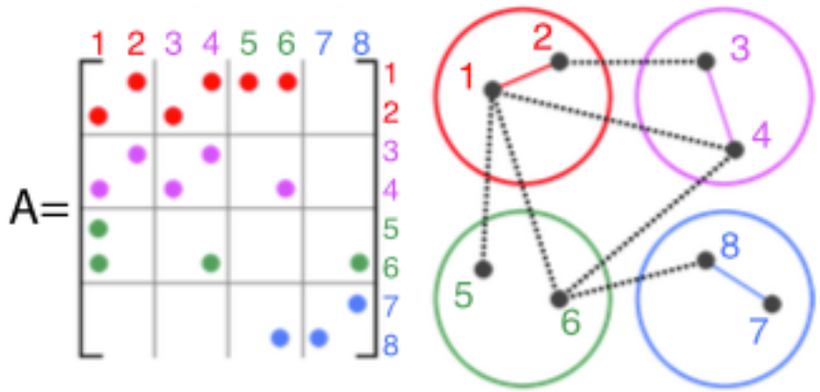


(spy plot)



Graph Partitioning

Motivation

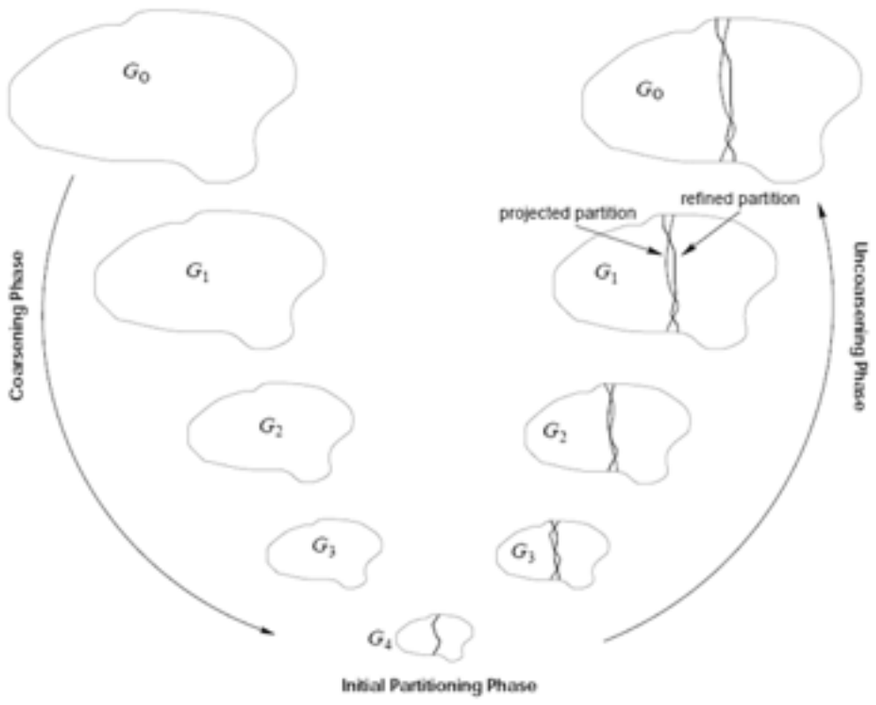


BFS/Shortest Path
Connectivity
PageRank
Betweenness Centrality
etc. etc.

HPC: Solvers, SpMV

Motivation

HPC Graph Partitioners



METIS/ParMETIS

Scotch/PT-Scotch

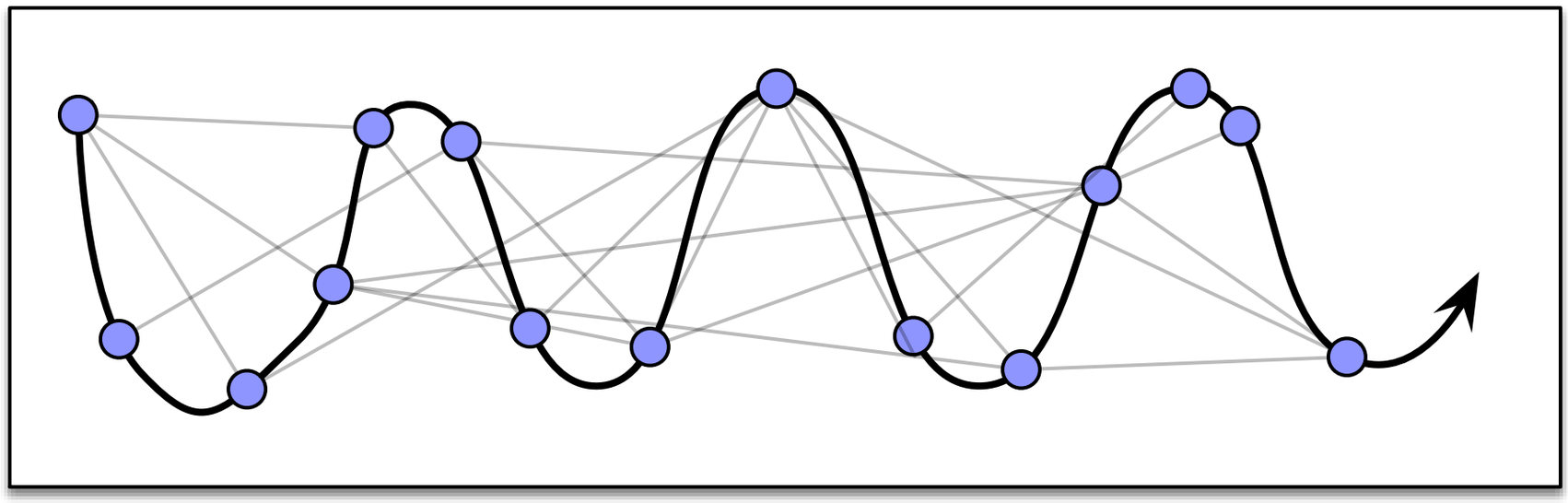
PaToH

- Offline, generally divide-and-conquer (‘multilevel’)
- Achieve excellent partitions for wide variety of graphs
- Suitable for moderate-sized graphs, but heavy parallel overhead
- Re-partitioning is slow

algorithmic alternatives: spectral, geometric, graph growing, random walk

Motivation

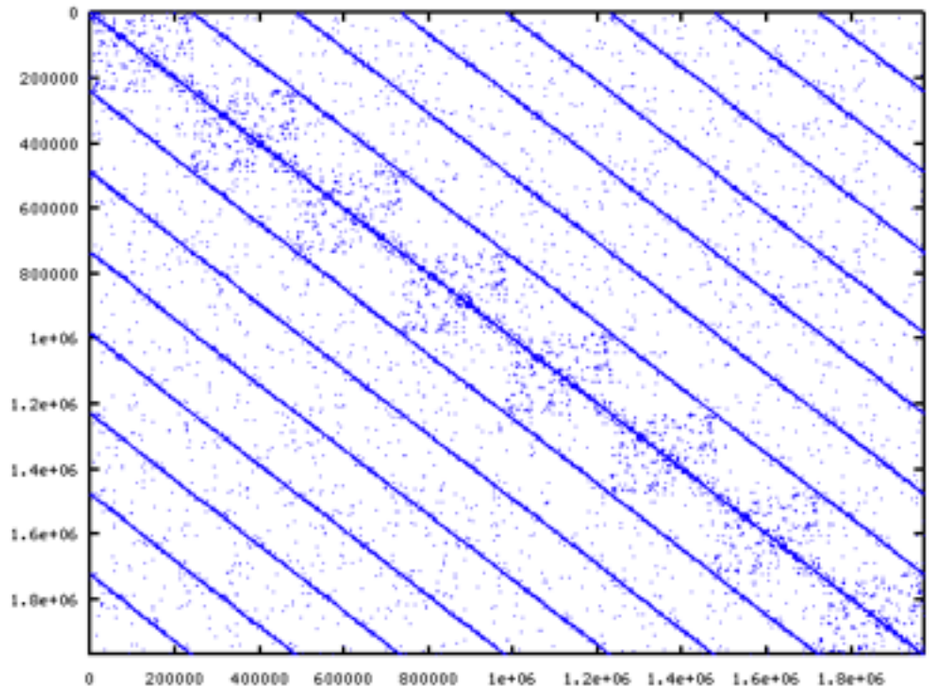
Streaming Graph Partitioners



- Stream over vertices in any order
- Touch each vertex once
- Simple to implement and very fast
- Quality partitions for low-diameter/scale-free graphs
- Suitable for re-partitioning / dynamic partitioning

Motivation

Streaming Graph Partitioners



roadNet-CA8.mat

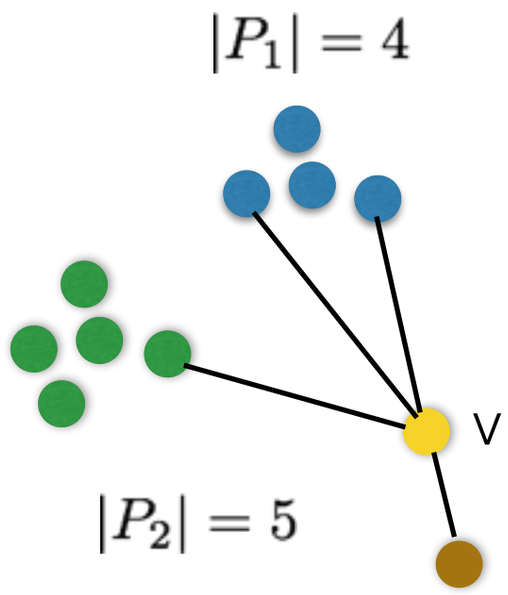
Note: Struggle on higher diameter graphs

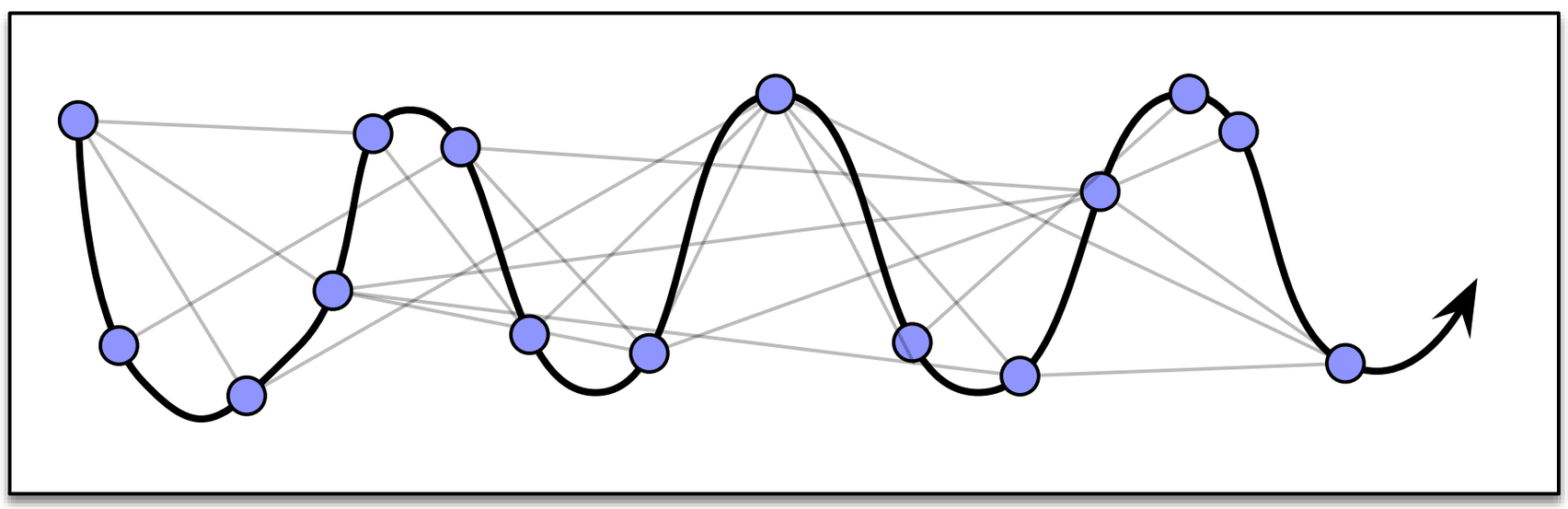
Methodology

Streaming Partitioning [SK2012, TKRV2012]

Set all P_i to $\emptyset$;
foreach $v \in V(G)$ *as it arrives at time* t **do**
    $j \leftarrow \underset{i \in \{1,\ldots,p\}}{\operatorname{argmax}} |P_i^t \cap N(v)| - \alpha \frac{\gamma}{2} |P_i^t|^{\gamma-1}$;
    Add v to set P_j^{t+1};
end

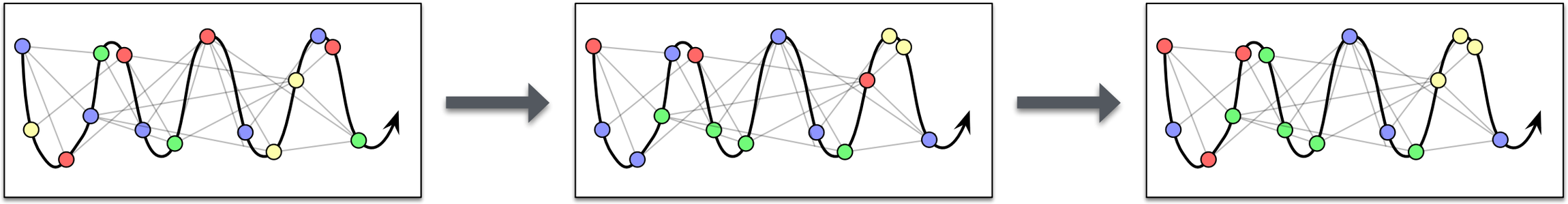
Algorithm 1: Serial streaming FENNEL partitioner





Methodology

Restreaming Partitioning [NU2013]



init random partition

quality converges

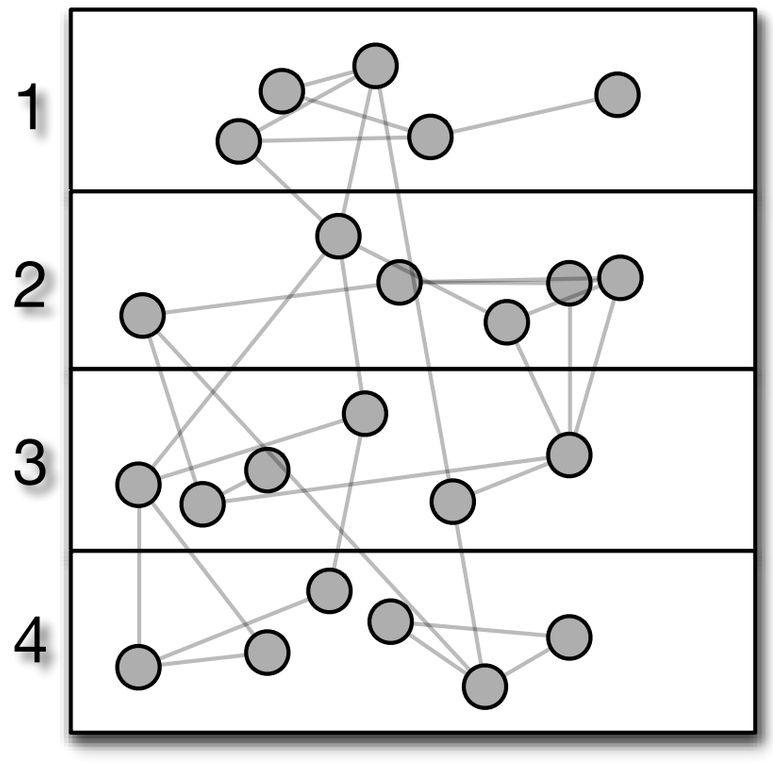
temper parameters over time to favor balance

Methodology

GraSP: Parallel Restreaming

for *each process* p **do in parallel**
  $vorder \leftarrow rand\_perm(\{0, \ldots, |V(G_{local})|\})$;
  Randomly assign local vertices to partitions $P^0_{i,p}$;
end

for $run \leftarrow \{1 \ldots n_s\}$ **do**
  for *each process* p **do in parallel**
    foreach $v \in vorder$ **do**
      $j \leftarrow \underset{i \in \{1,\ldots,p\}}{\text{argmax}} |P^t_i \cap N(v)| - \alpha \frac{\gamma}{2} |P^t_i|^{\gamma-1}$;
      Add v to set $P^{t+1}_{j,p}$;
    end
  end
  MPI_AllGather global partition assignments;
  $\alpha \leftarrow t_\alpha \alpha$
end

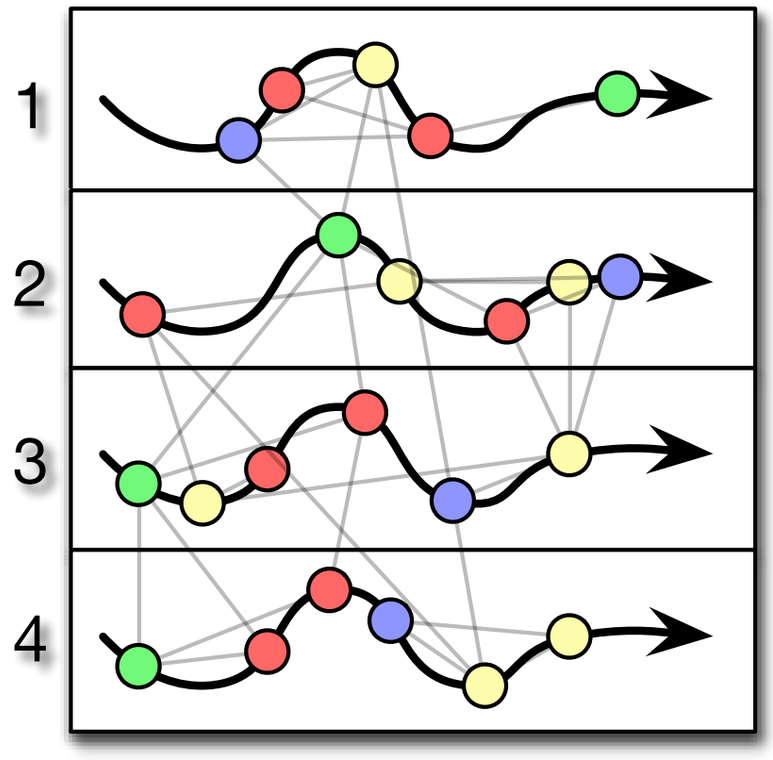


Methodology

Initialization

for *each process* p **do in parallel**
  $vorder \leftarrow rand\_perm(\{0, \ldots, |V(G_{local})|\})$;
  Randomly assign local vertices to partitions $P^0_{i,p}$;
end

for $run \leftarrow \{1 \ldots n_s\}$ **do**
  for *each process* p **do in parallel**
    foreach $v \in vorder$ **do**
      $j \leftarrow \underset{i \in \{1,\ldots,p\}}{\text{argmax}} |P^t_i \cap N(v)| - \alpha \frac{\gamma}{2} |P^t_i|^{\gamma-1}$;
      Add v to set $P^{t+1}_{j,p}$;
    end
  end
  MPI_AllGather global partition assignments;
  $\alpha \leftarrow t_\alpha \alpha$
end



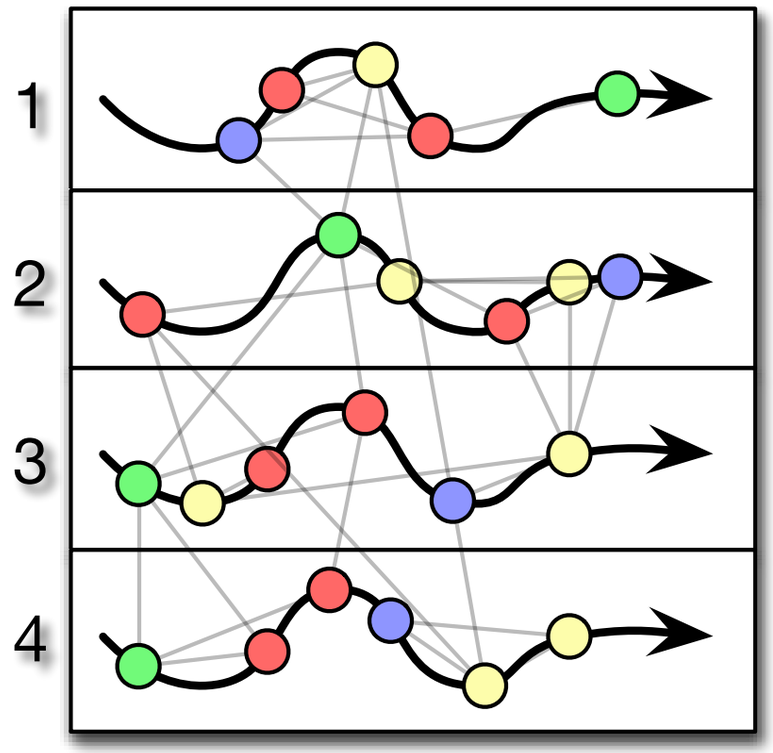
Methodology

Multiple Runs

for *each process* p **do in parallel**
  $vorder \leftarrow rand\_perm(\{0, \ldots, |V(G_{local})|\})$;
  Randomly assign local vertices to partitions $P^0_{i,p}$;
end

for $run \leftarrow \{1 \ldots n_s\}$ **do**
  for *each process* p **do in parallel**
    foreach $v \in vorder$ **do**
      $j \leftarrow \underset{i \in \{1,\ldots,p\}}{\text{argmax}} |P^t_i \cap N(v)| - \alpha \frac{\gamma}{2} |P^t_i|^{\gamma-1}$;
      Add v to set $P^{t+1}_{j,p}$;
    end
  end
  MPI_AllGather global partition assignments;
  $\alpha \leftarrow t_\alpha \alpha$
end

Algorithm 2: Parallel Restreaming performed by GraSP.



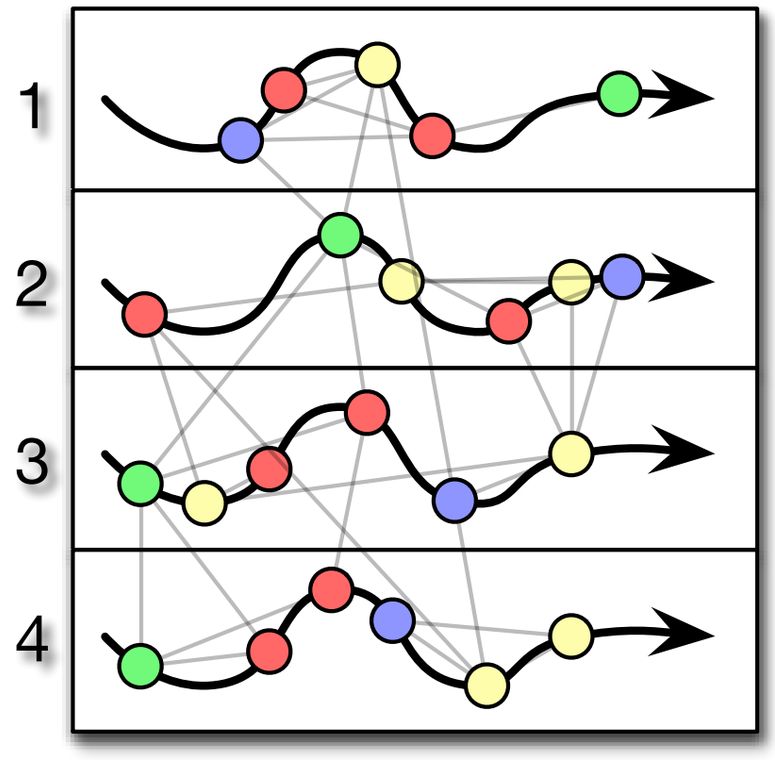
Methodology

Stream

for *each process* p **do in parallel**
  $vorder \leftarrow rand\_perm(\{0, \ldots, |V(G_{local})|\})$;
  Randomly assign local vertices to partitions $P^0_{i,p}$;
end

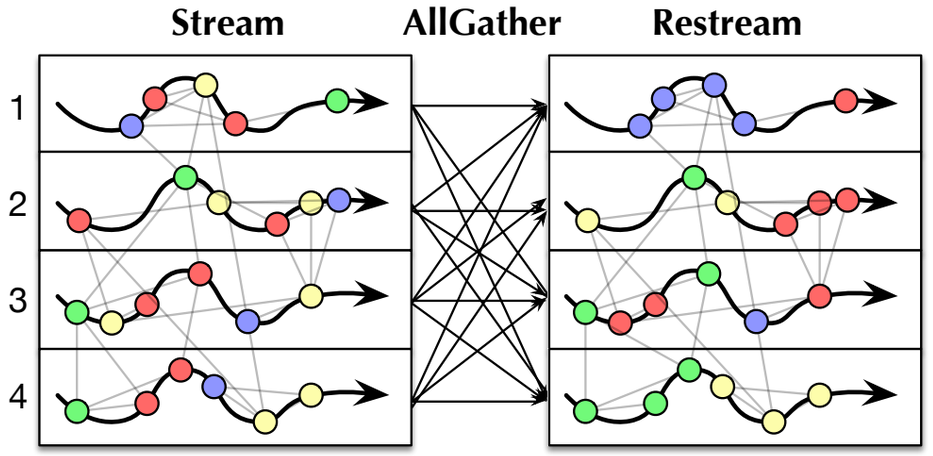
for $run \leftarrow \{1 \ldots n_s\}$ **do**
  for *each process* p **do in parallel**
    foreach $v \in vorder$ **do**
      $j \leftarrow \underset{i \in \{1,\ldots,p\}}{\text{argmax}} |P^t_i \cap N(v)| - \alpha \frac{\gamma}{2} |P^t_i|^{\gamma-1}$;
      Add v to set $P^{t+1}_{j,p}$;
    end
  end
  MPI_AllGather global partition assignments;
  $\alpha \leftarrow t_\alpha \alpha$
end

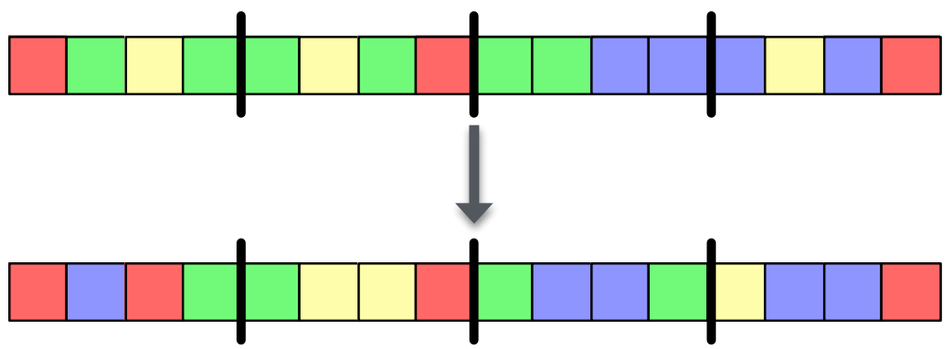
Algorithm 2: Parallel Restreaming performed by GraSP.



Methodology

Communicate





for *each process* p **do in parallel**
  $vorder \leftarrow rand\_perm(\{0, \ldots, |V(G_{local})|\})$;
  Randomly assign local vertices to partitions $P^0_{i,p}$;
end

for $run \leftarrow \{1 \ldots n_s\}$ **do**
  for *each process* p **do in parallel**
    foreach $v \in vorder$ **do**
      $j \leftarrow \underset{i \in \{1,\ldots,p\}}{\text{argmax}} |P^t_i \cap N(v)| - \alpha \frac{\gamma}{2} |P^t_i|^{\gamma-1}$;
      Add v to set $P^{t+1}_{j,p}$;
    end
  end
  MPI_AllGather global partition assignments;
  $\alpha \leftarrow t_\alpha \alpha$
end

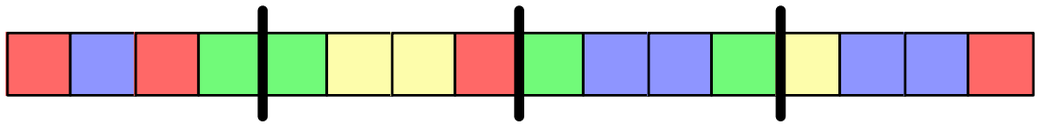
Algorithm 2: Parallel Restreaming performed by GraSP.

Methodology

for *each process* p **do in parallel**
  $vorder \leftarrow rand\_perm(\{0, \ldots, |V(G_{local})|\})$;
  Randomly assign local vertices to partitions $P^0_{i,p}$;
end
for $run \leftarrow \{1 \ldots n_s\}$ **do**
  for *each process* p **do in parallel**
    foreach $v \in vorder$ **do**
      $j \leftarrow \underset{i \in \{1,\ldots,p\}}{\text{argmax}} |P^t_i \cap N(v)| - \alpha \frac{\gamma}{2} |P^t_i|^{\gamma-1}$;
      Add v to set $P^{t+1}_{j,p}$;
    end
  end
  MPI_AllGather global partition assignments;
  $\alpha \leftarrow t_\alpha \alpha$
end

Algorithm 2: Parallel Restreaming performed by GraSP.

post: final partition computed



Evaluation

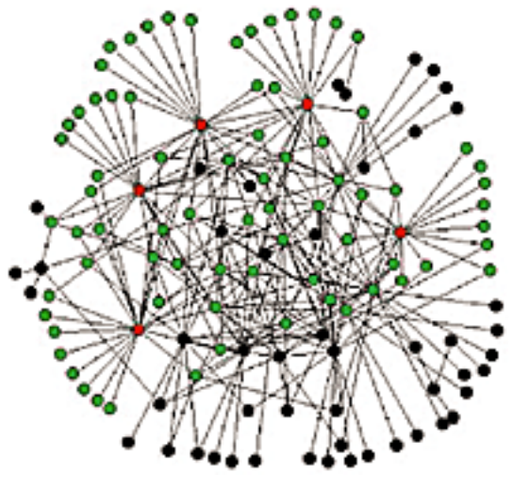
System

- Edison @ NERSC
- 5576 Compute Nodes
- Two 12-core “Ivy Bridge” processors per node
- Cray Aries interconnect
- MPI v3.0

Evaluation

Real-World Data: SNAP Data Sets†

Synthetic Data: R-MAT



†: https://snap.stanford.edu/data/

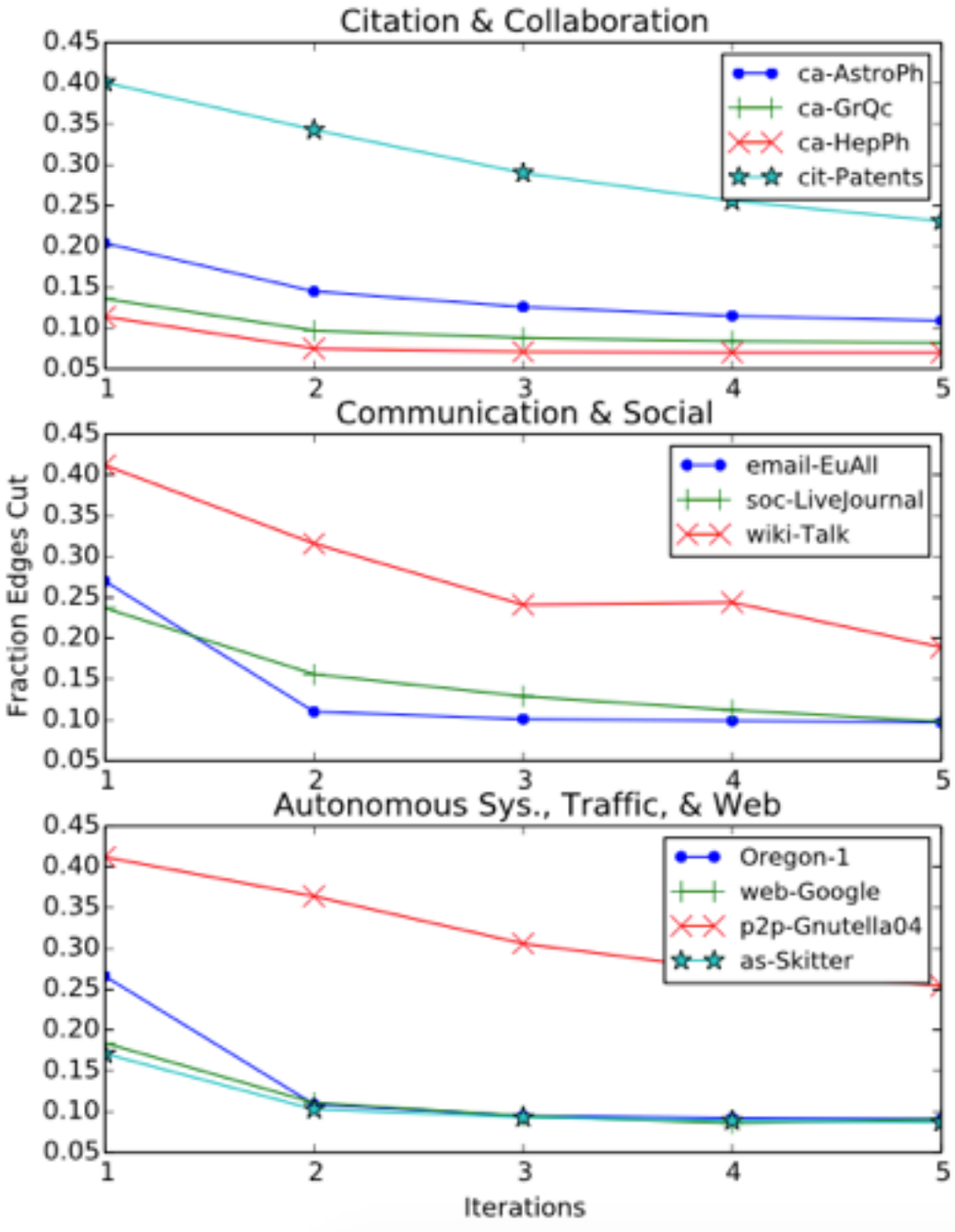
Evaluation

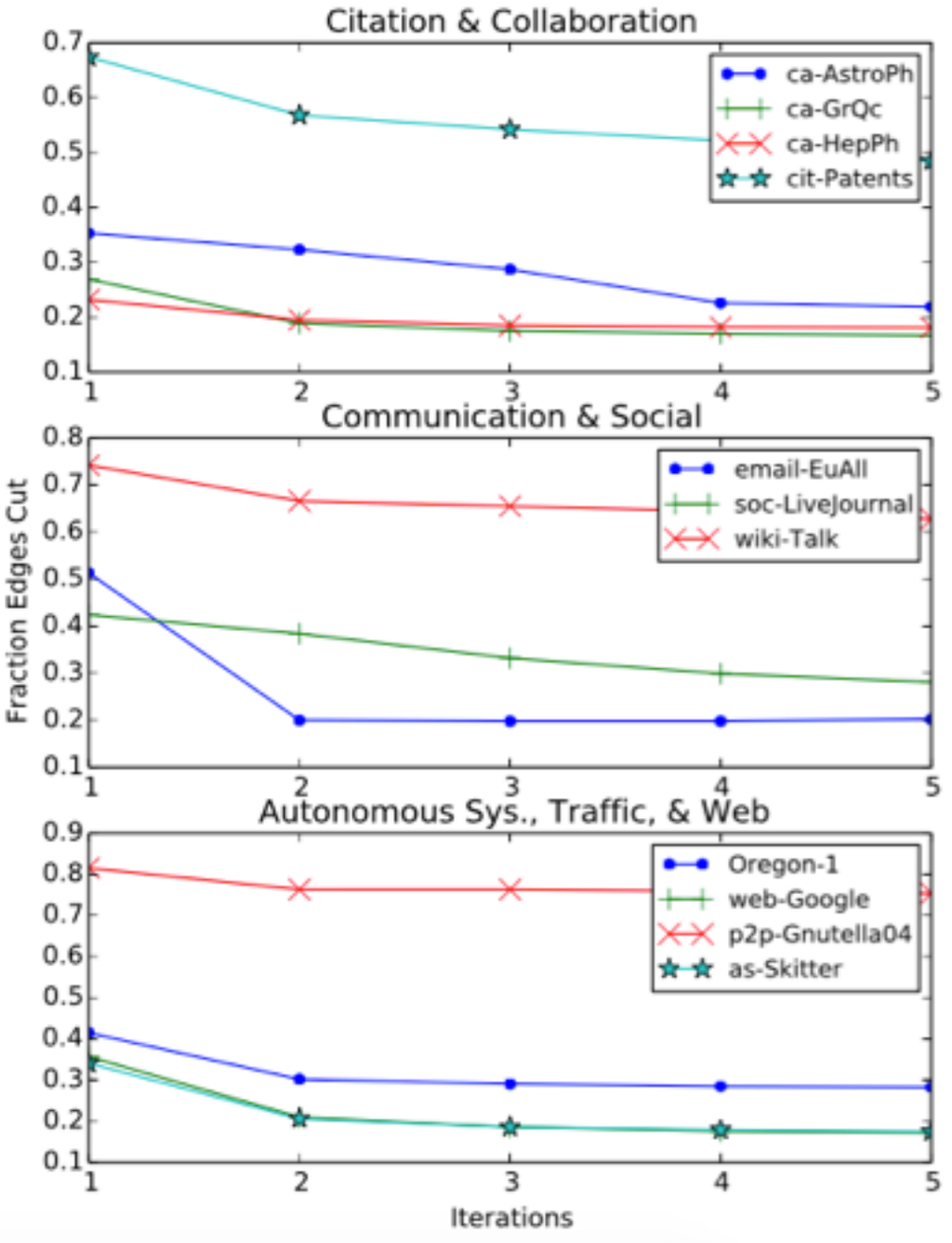
Verification on real-world data sets

Data Set	N	nnz	$\lambda_{p=2}$	$\lambda_{p=8}$
soc-LiveJournal	4,847,571	68,993,773	0.234	0.463
as-Skitter	1,696,415	22,190,596	0.166	0.324
cit-Patents	3,774,768	16,518,948	0.402	0.726
roadNet-CA	1,971,281	5,533,214	0.186	0.360
web-Google	916,428	5,105,039	0.189	0.336
wiki-Talk	2,394,385	5,021,410	0.411	0.752
amazon0302	262,111	1,234,877	0.202	0.370
soc-Slashdot0902	82,168	948,464	0.236	0.382
ca-AstroPh	18,772	396,160	0.232	0.413
cit-HepPh	34,546	421,578	0.343	0.646
email-EuAll	265,214	420,045	0.280	0.538
Oregon-1	11,492	46,818	0.224	0.406
p2p-Gnutella04	10,879	39,994	0.415	0.747

Evaluation

Verification on real-world data sets





Evaluation

Synthetic graphs: R-MAT

Scale	26	27	28	29	30	31
|V(G)|	67M	134M	268M	537M	1.07B	2.15B
|E(G)|	1.07B	2.14B	4.29B	8.58B	17.1B	34.3B

Evaluation

GraSP vs ParMETIS: Scale-22 R-MAT†

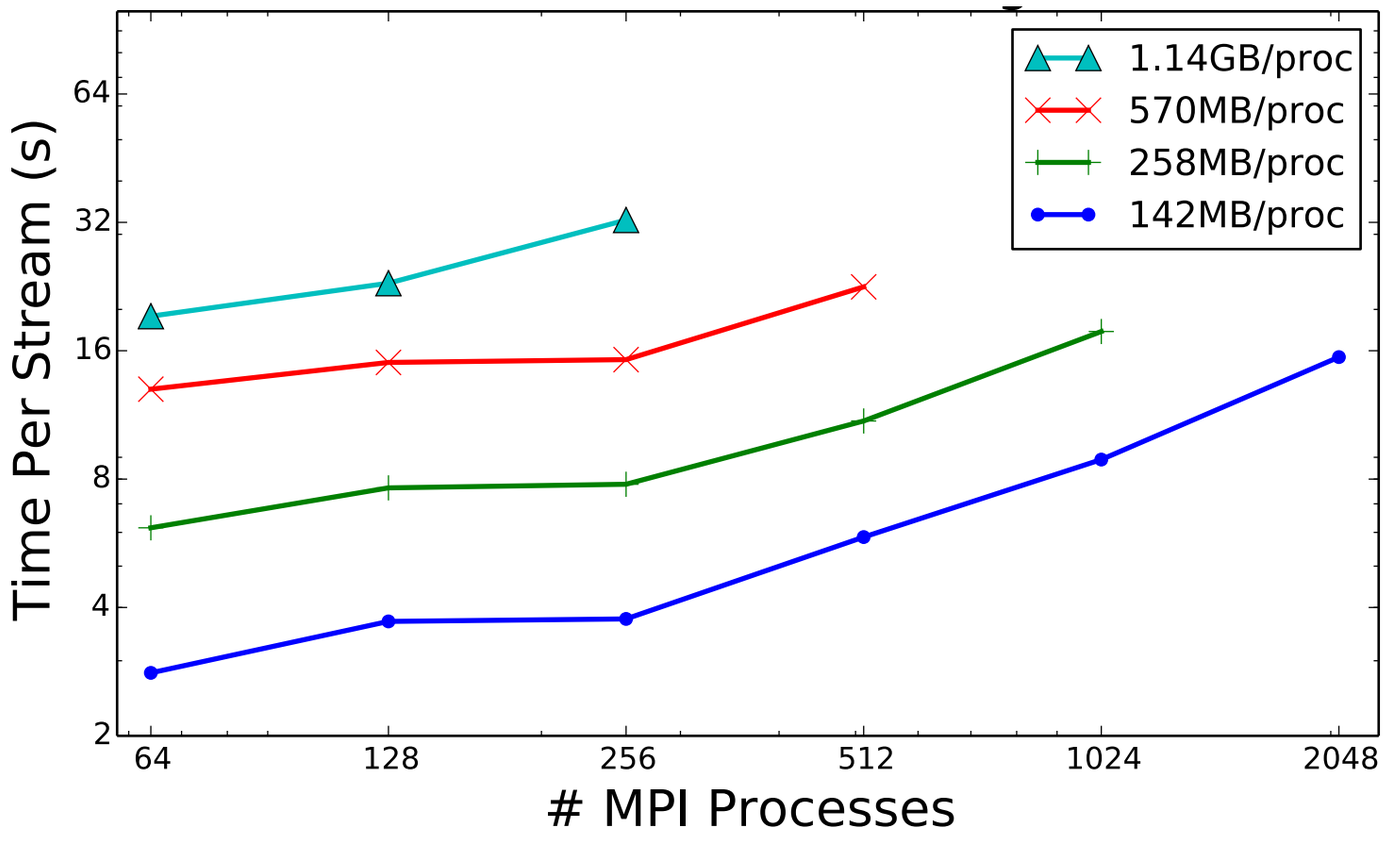
#procs	λ_{metis}	λ_{grasp}	t_{metis}(s)	$t_{grasp}(s)$
8	0.36	0.29	307.8	0.72
16	0.38	0.41	221.9	0.45
32	0.40	0.54	194.9	0.31

We generally encountered performance ~3 order of magnitude better.

To save core-hours we didn't run ParMETIS above Scale-22

†: 4194304 nodes, 67108864 edges

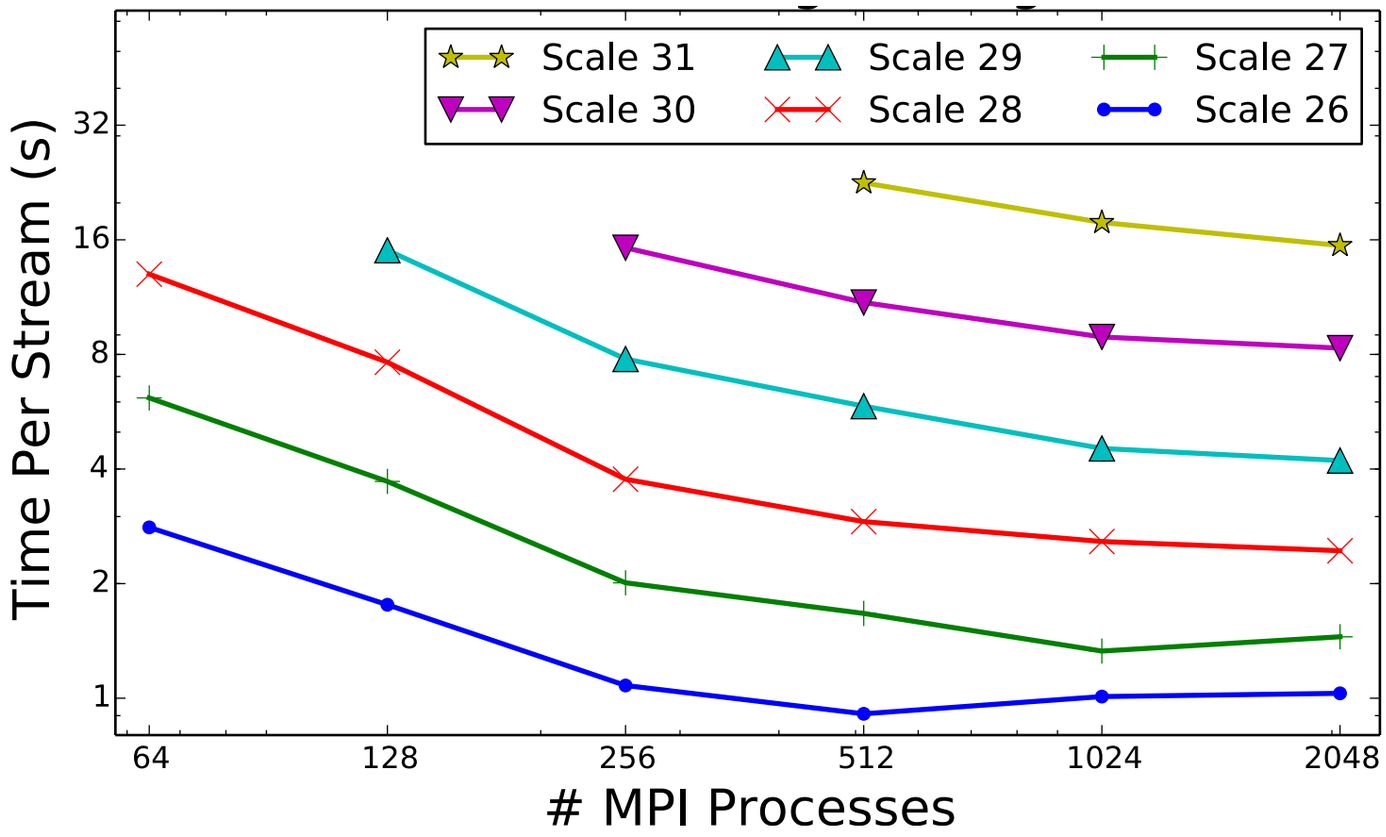
Evaluation



Weak Scaling

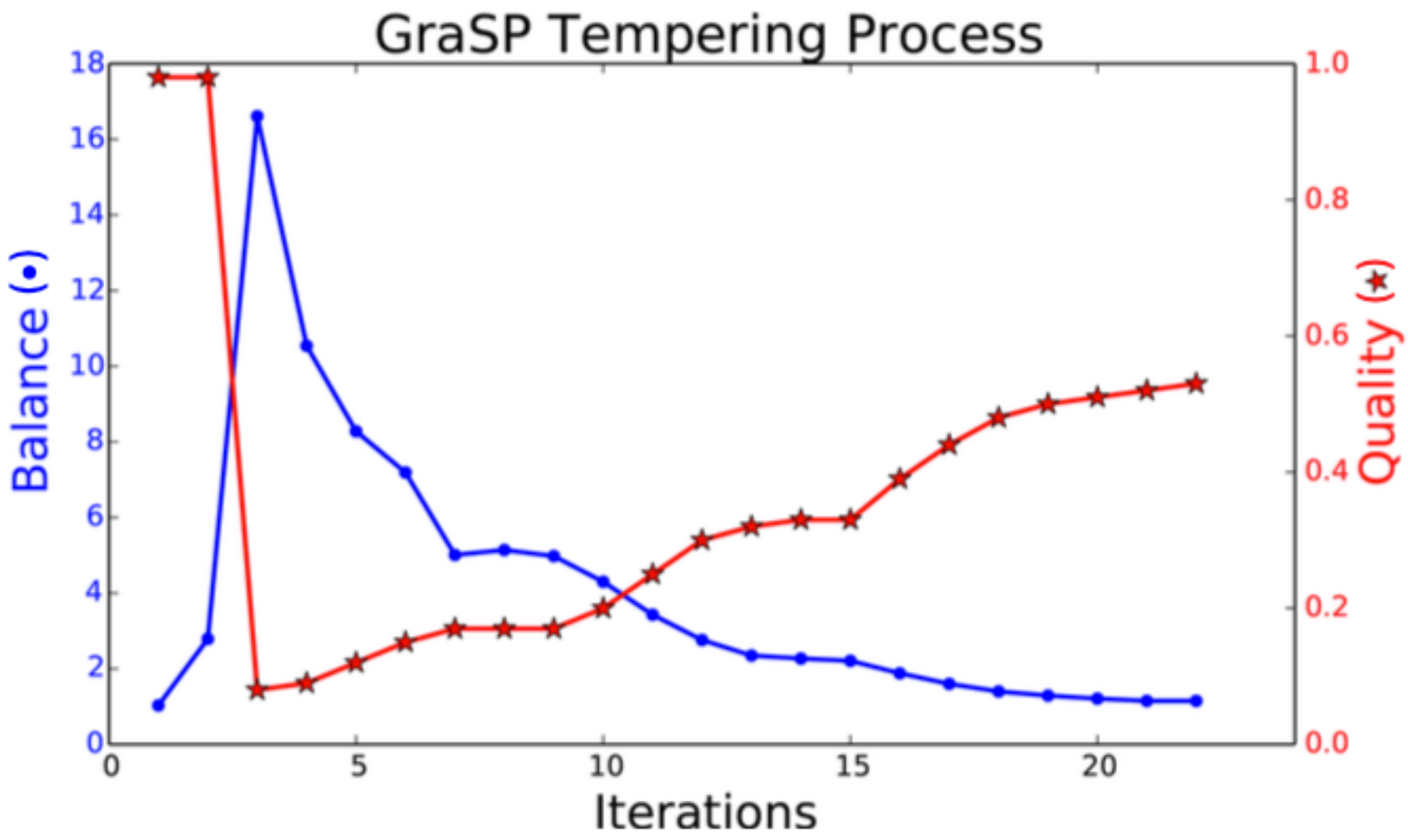
Evaluation

Scale	26	27	28	29	30	31
|V(G)|	67M	134M	268M	537M	1.07B	2.15B
|E(G)|	1.07B	2.14B	4.29B	8.58B	17.1B	34.3B



Strong Scaling

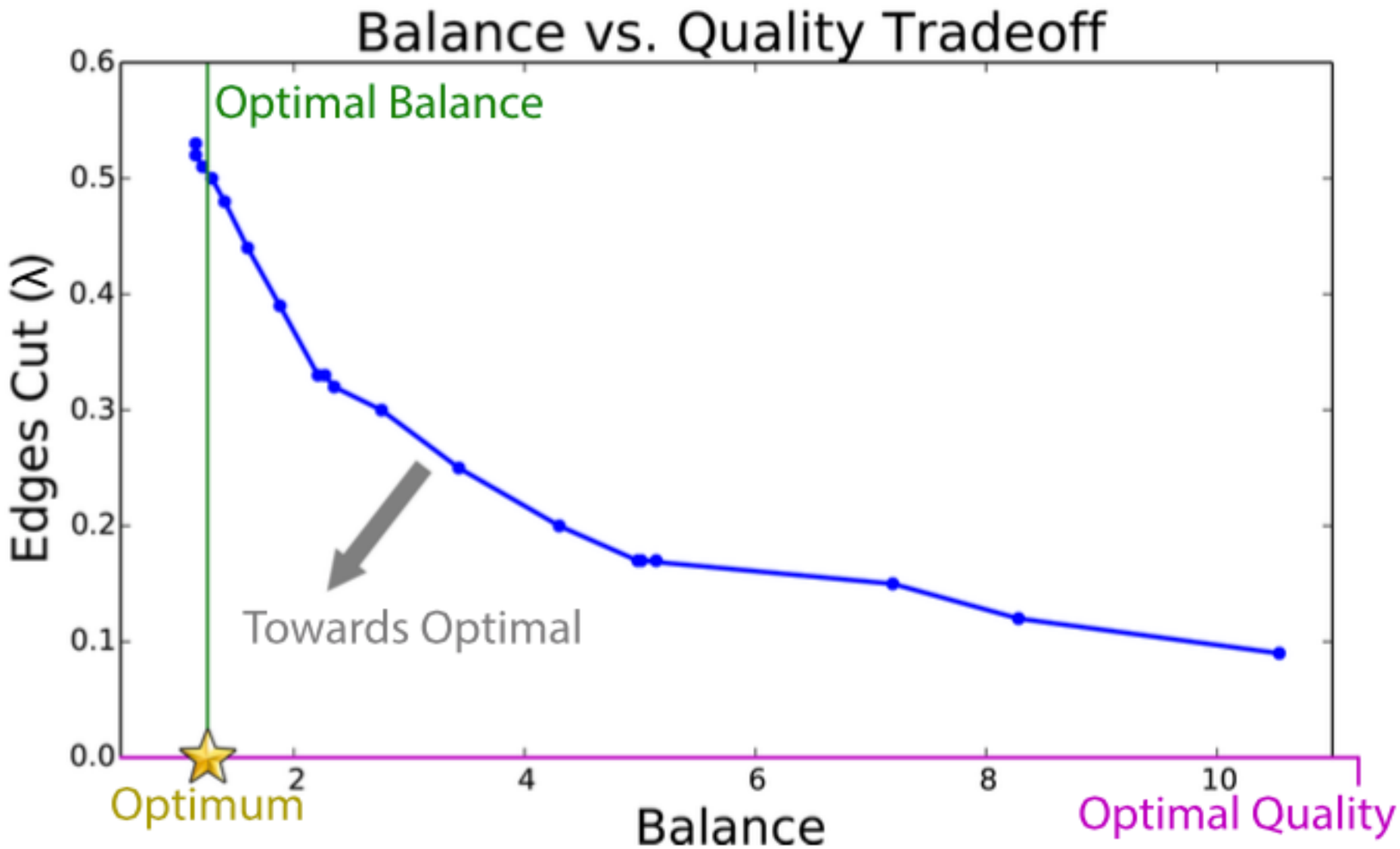
Evaluation



Scale-28, 64 MPI Processes

“Tempering”

Evaluation



Scale-28, 64 MPI Processes

“Tempering”

Conclusion

- Streaming partitioning is simple, scalable and effective (for the right kinds of graphs)

- Streaming partitioning can operate orders of magnitude faster than sophisticated parallel partitioners with similar quality

- Streaming partitioning deserves more attention from the HPC community