PuLP

Complex Objective Partitioning of Small-World Networks Using Label Propagation

George M. Slota[1,2] Kamesh Madduri[2]
Sivasankaran Rajamanickam[1]

[1]Sandia National Laboratories, [2]The Pennsylvania State University
gslota@psu.edu, madduri@cse.psu.edu, srajama@sandia.gov

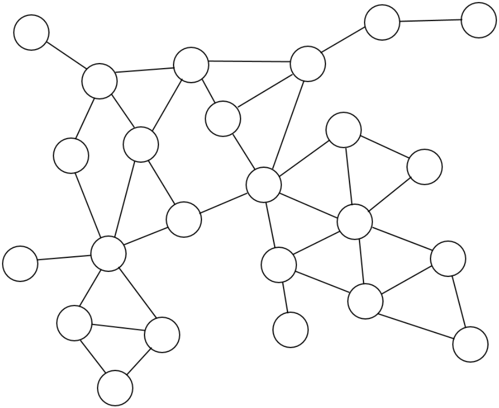
SIAM CSE15 17 March 2015

Highlights

- We present PuLP, a multi-constraint multi-objective partitioner
- Shared-memory parallelism (OpenMP)
- PuLP demonstrates an average speedup of 14.5× relative to state-of-the-art partitioners on small-world graph test suite
- PuLP requires 8-39× less memory than state-of-the-art partitioners
- PuLP produces partitions with comparable or better quality than state-of-the-art partitioners for small-world graphs
- Exploits the label propagation clustering algorithm

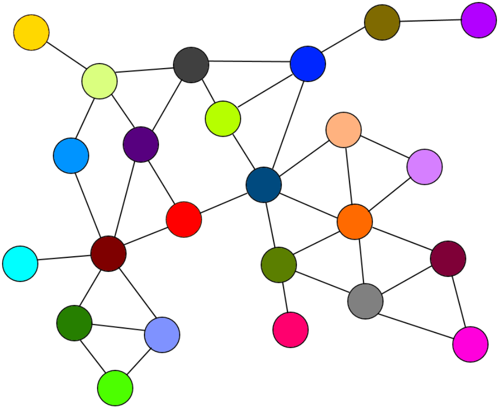
Label Propagation
Algorithm progression

- Randomly label with n labels



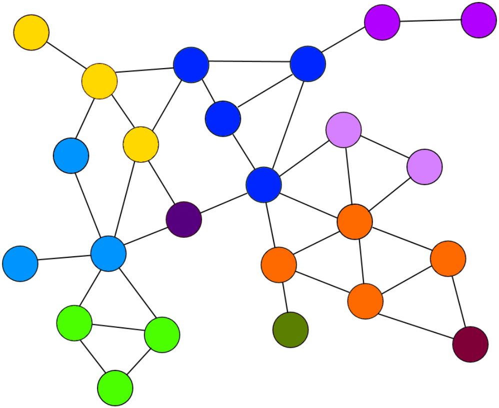
Label Propagation
Algorithm progression

- Randomly label with n labels



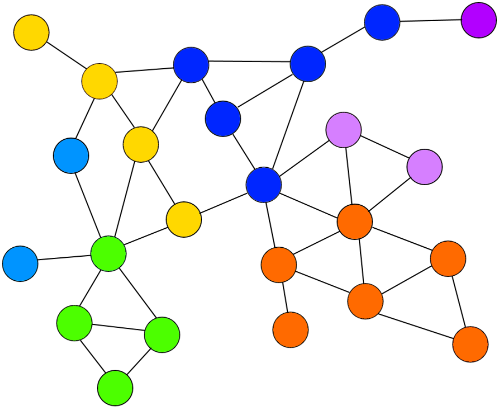
Label Propagation
Algorithm progression

- Randomly label with n labels
- Iteratively update each v with max per-label count over neighbors, ties broken randomly



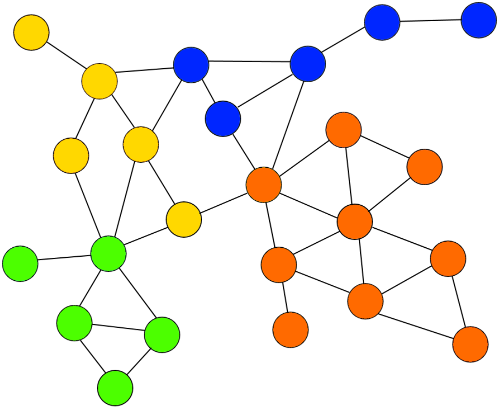
Label Propagation
Algorithm progression

- Randomly label with n labels
- Iteratively update each v with max per-label count over neighbors, ties broken randomly



Label Propagation
Algorithm progression

- Randomly label with n labels
- Iteratively update each v with max per-label count over neighbors, ties broken randomly
- Algorithm completes when no new updates possible



Label Propagation

Overview and observations

- **Label propagation**: initialize a graph with n labels, iteratively assign to each vertex the maximal per-label count over all neighbors to generate clusters, ties broken randomly (Raghavan et al. 2007)
  - Clustering algorithm - dense clusters hold same label
  - Fast - each iteration in $O(n + m)$
  - Naïvely parallel - only per-vertex label updates
  - *Observation*: Possible applications for large-scale small-world graph partitioning

Partitioning

Problem description, objectives and constraints

- **Goal**: minimize execution time for small-world graph analytics
- **Constraints**:
  - *Vertex* and *edge balance*: per-task memory and computation requirements
- **Objectives**:
  - *Edge cut* and *max per-part cut*: total and per-task communication requirements
- Single level vs. multi level partitioners
  - Multi level produces high quality at cost of computing
  - Single level produces sub-optimal quality

Partitioning

Problem description, objectives and constraints

- **Goal**: minimize execution time for small-world graph analytics
- **Constraints**:
  - *Vertex* and *edge balance*: per-task memory and computation requirements
- **Objectives**:
  - *Edge cut* and *max per-part cut*: total and per-task communication requirements
- Single level vs. multi level partitioners
  - Multi level produces high quality at cost of computing
  - Single level produces sub-optimal quality
    - Until now...

PuLP : **P**artitioning **u**sing **L**abel **P**ropagation

- Utilize label propagation for:
  - Vertex balanced partitions, minimize edge cut (PuLP)
  - Vertex and edge balanced partitions, minimize edge cut (PuLP-M)
  - Vertex and edge balanced partitions, minimize edge cut and maximal per-part edge cut (PuLP-MM)
  - Any combination of the above - multi objective, multi constraint

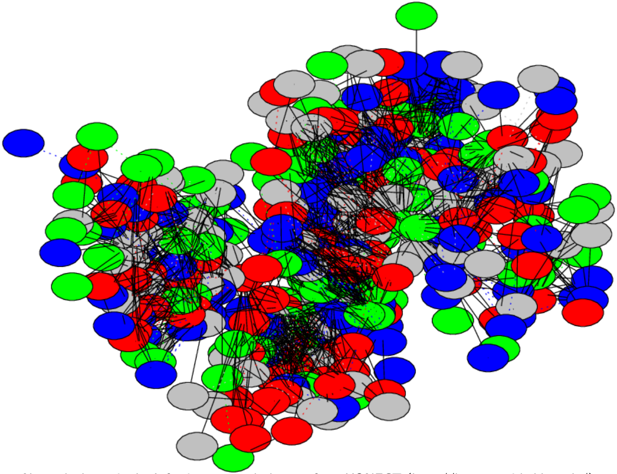
PULP-MM

Algorithm overview

Randomly initialize p partitions
Create partitions with degree-weighted label propagation
for Some number of iterations **do**
    for Some number of iterations **do**
        Balance partitions to satisfy vertex constraint
        Refine partitions to minimize edge cut
    for Some number of iterations **do**
        Balance partitions to satisfy edge constraint and minimize per-part edge cut
        Refine partitions to minimize edge cut

PuLP-MM
PuLP-MM algorithm

- Randomly initialize partition labels for p parts

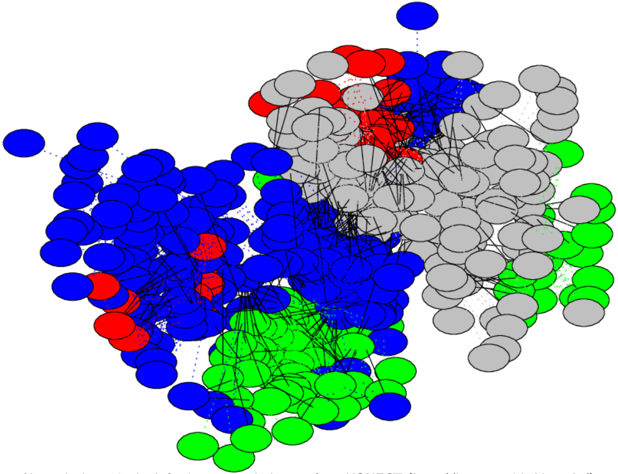


Network shown is the Infectious network dataset from KONECT (http://konect.uni-koblenz.de/)

PuLP-MM

PuLP-MM algorithm

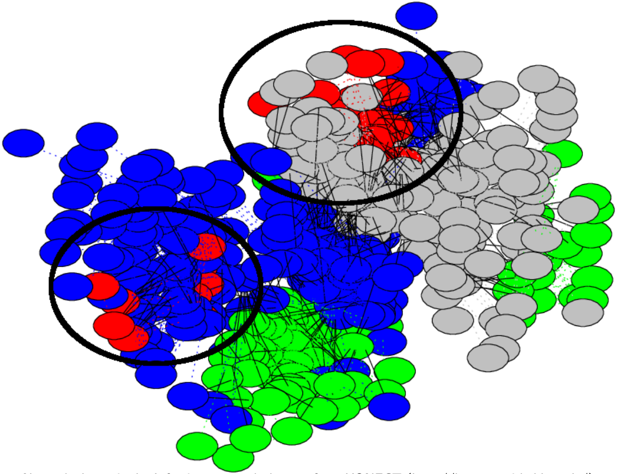
- Randomly initialize partition labels for p parts
- Run *degree-weighted* label prop to create initial parts



Network shown is the Infectious network dataset from KONECT (http://konect.uni-koblenz.de/)

PuLP-MM
PuLP-MM algorithm

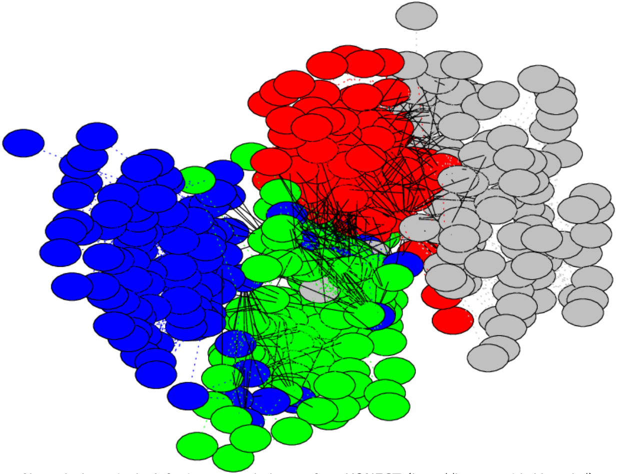
- Randomly initialize partition labels for p parts
- Run *degree-weighted* label prop to create initial parts



Network shown is the Infectious network dataset from KONECT (http://konect.uni-koblenz.de/)

PuLP-MM
PuLP-MM algorithm

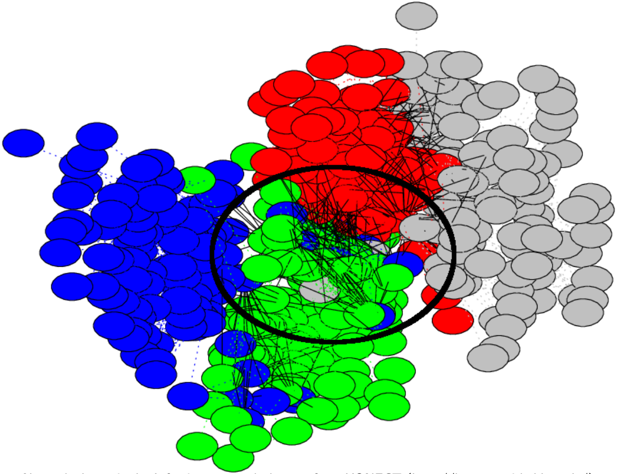
- Randomly initialize partition labels for p parts
- Run *degree-weighted* label prop to create initial parts
- Iteratively balance for vertices, minimize edge cut



Network shown is the Infectious network dataset from KONECT (http://konect.uni-koblenz.de/)

PuLP-MM
PuLP-MM algorithm

- Randomly initialize partition labels for p parts
- Run *degree-weighted* label prop to create initial parts
- Iteratively balance for vertices, minimize edge cut

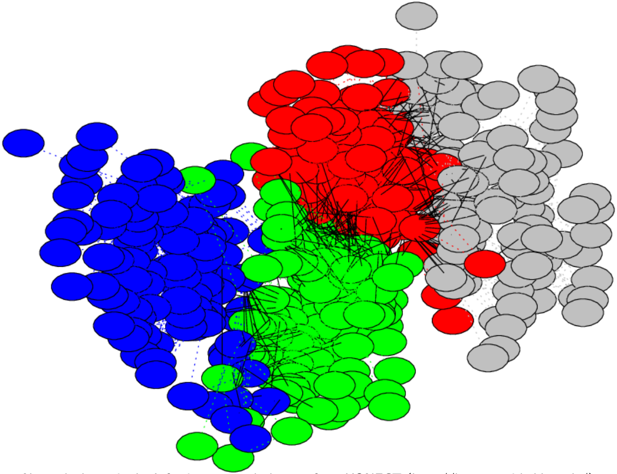


Network shown is the Infectious network dataset from KONECT (http://konect.uni-koblenz.de/)

PuLP-MM

PuLP-MM algorithm

- Randomly initialize partition labels for p parts
- Run *degree-weighted* label prop to create initial parts
- Iteratively balance for vertices, minimize edge cut
- Balance for edges, minimize per-part edge cut



Network shown is the Infectious network dataset from KONECT (http://konect.uni-koblenz.de/)

PuLP-MM

Label propagation step

$P \leftarrow \text{PULP-lp}(G(V,E), p, P, N, I_p)$
$Min_v \leftarrow (n/p) \times (1 - \epsilon_l)$
$i \leftarrow 0$, $r \leftarrow 1$
while $i < I_p$ **and** $r \neq 0$ **do**
    $r \leftarrow 0$
    for all $v \in V$ **do**
        $C(1 \cdots p) \leftarrow 0$
        for $u \in E(v)$ **do**
            $C(P(u)) \leftarrow C(P(u)) + |E(u)|$
        $x \leftarrow \text{Max}(C(1 \cdots p))$
        if $x \neq P(v)$ **and** $N(P(v)) - 1 > Min_v$ **then**
            $P(v) \leftarrow x$
            $r \leftarrow r + 1$
    $i \leftarrow i + 1$
return P

PuLP-MM

Vertex Balancing Step

$P \leftarrow$ PULP-vp$(G(V,E), P, p, N, I_b)$
$i \leftarrow 0$, $r \leftarrow 1$
$Max_v \leftarrow (n/p) \times (1 + \epsilon_u)$
$W_v(1 \cdots p) \leftarrow$ **Max(**$Max_v / N(1 \cdots p) - 1$**,**$0$**)**
while $i < I_b$ **and** $r \neq 0$ **do**
    $r \leftarrow 0$
    for all $v \in V$ **do**
        $C(1 \cdots p) \leftarrow 0$
        for all $u \in E(v)$ **do**
            $C(P(u)) \leftarrow C(P(u)) + |E(u)|$
        for $j = 1 \cdots p$ **do**
            if Moving v **to** P_j **violates** Max_v **then**
                $C(j) \leftarrow 0$
            else
                $C(j) \leftarrow C(j) \times W_v(j)$
        $x \leftarrow$ Max$(C(1 \cdots p))$
        if $x \neq P(v)$ **then**
            Update($N(P(v)), N(x)$**)**
            Update($W_v(P(v)), W_v(x)$**)**
            $P(v) \leftarrow x$
            $r \leftarrow r + 1$
    $i \leftarrow i + 1$

PuLP-MM

Edge balancing and max per-part cut minimization step

$P \leftarrow$ PULP-cp$(G(V, E), P, p, N, M, T, U, I_b)$
$i \leftarrow 0$, $r \leftarrow 1$
$Max_v \leftarrow (n/p) \times (1 + \epsilon_u)$, $Max_e \leftarrow (m/p) \times (1 + \eta_u)$
$Cur_{Max_e} \leftarrow$ **Max**$(M(1 \cdots p))$, $Cur_{Max_c} \leftarrow$ **Max**$(T(1 \cdots p))$
$W_e(1 \cdots p) \leftarrow Cur_{Max_e}/M(1 \cdots p) - 1$, $W_c(1 \cdots p) \leftarrow Cur_{Max_c}/T(1 \cdots p) - 1$
$R_e \leftarrow 1$, $R_c \leftarrow 1$
while $i < I_p$ **and** $r \neq 0$ **do**
  $r \leftarrow 0$
  for all $v \in V$ **do**
    $C(1 \cdots p) \leftarrow 0$
    for all $u \in E(v)$ **do**
      $C(P(u)) \leftarrow C(P(u)) + 1$
    for $j = 1 \cdots p$ **do**
      if Moving v to P_j violates Max_v, Cur_{Max_e}, Cur_{Max_c} **then**
        $C(j) \leftarrow 0$
      else
        $C(j) \leftarrow C(j) \times (W_e(j) \times R_e + W_v(j) \times R_c)$
    $x \leftarrow$ Max$(C(1 \cdots p))$
    if $x \neq P(v)$ **then**
      $P(v) \leftarrow x$
      $r \leftarrow r + 1$
      Update all variables for x and $P(v)$
  if $Cur_{Max_e} < Max_e$ **then**
    $Cur_{Max_e} \leftarrow Max_e$
    $R_c \leftarrow R_c \times Cur_{Max_c}$
    $R_e \leftarrow 1$
  else
    $R_e \leftarrow R_e \times (Cur_{Max_e}/Max_e)$
    $R_c \leftarrow 1$
  $i \leftarrow i + 1$

Results

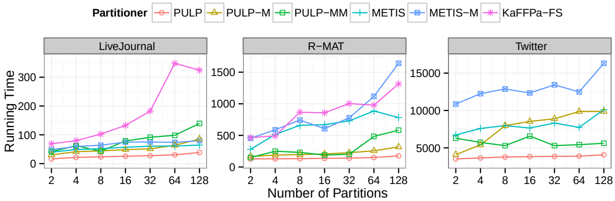
Test Environment and Graphs

- Test system: *Compton*
  - Intel Xeon E5-2670 (Sandy Bridge), dual-socket, 16 cores, 64 GB memory.
- Test graphs:
  - LAW graphs from UF Sparse Matrix, SNAP, MPI, Koblenz
  - Real (one R-MAT), small-world, 60 K–70 M vertices, 275 K–2 B edges
- Test Algorithms:
  - **METIS** - single constraint single objective
  - **METIS-M** - multi constraint single objective
  - **ParMETIS** - METIS-M running in parallel
  - **KaFFPa** - single constraint single objective
  - **PuLP** - single constraint single objective
  - **PuLP-M** - multi constraint single objective
  - **PuLP-MM** - multi constraint multi objective
- Metrics: 2–128 partitions, serial and parallel running times, memory utilization, edge cut, max per-partition edge cut

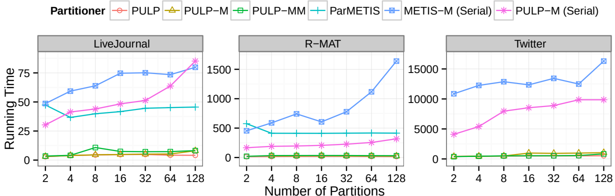
Results

PuLP Running Times - Serial (top), Parallel (bottom)

- In serial, PuLP-MM runs 1.7× faster (geometric mean) than next fastest of METIS and KaFFPa



- In parallel, PuLP-MM runs 14.5× faster (geometric mean) than next fastest (ParMETIS times are fastest of 1 to 256 cores)



Results

PuLP memory utilization for 128 partitions

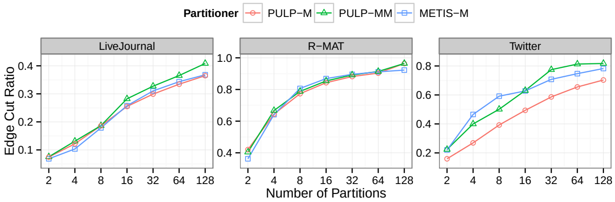
- PuLP utilizes minimal memory, $O(n)$, 8-39$\times$ less than KaFFPa and METIS
- Savings are mostly from avoiding a multilevel approach

Network	Memory Utilization				**Improv.**
	METIS-M	KaFFPa	PuLP-MM	Graph Size	
LiveJournal	7.2 GB	5.0 GB	0.44 GB	0.33 GB	21$\times$
Orkut	21 GB	13 GB	0.99 GB	0.88 GB	23$\times$
R-MAT	42 GB	-	1.2 GB	1.02 GB	35$\times$
DBpedia	46 GB	-	2.8 GB	1.6 GB	28$\times$
WikiLinks	103 GB	42 GB	5.3 GB	4.1 GB	25$\times$
sk-2005	121 GB	-	16 GB	13.7 GB	8$\times$
Twitter	487 GB	-	14 GB	12.2 GB	39$\times$

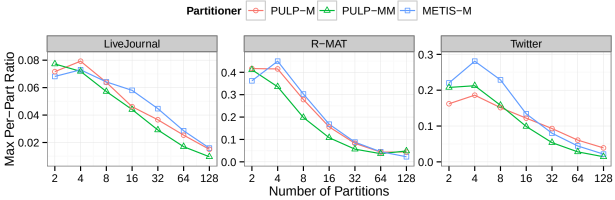
Results

PuLP quality - edge cut and max per-part cut

- PuLP-M produces better edge cut than METIS-M over most graphs



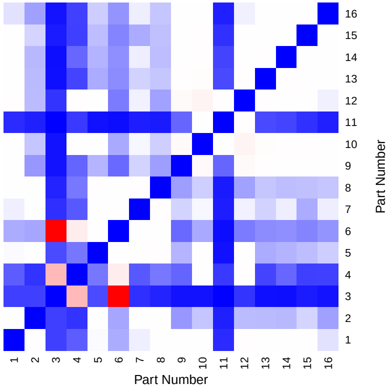
- PuLP-MM produces better max edge cut than METIS-M over most graphs

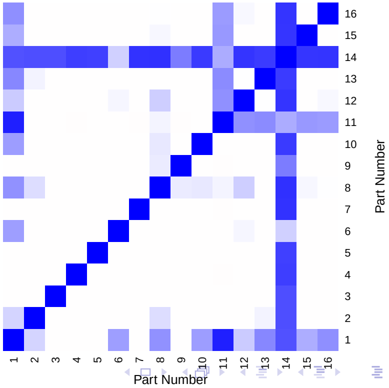


Results

PuLP– balanced communication

- 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

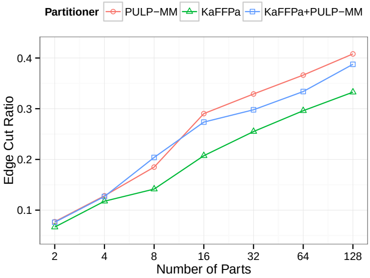


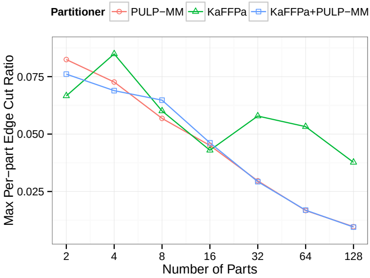


Results

Re-balancing for multiple constraints and objectives

- PuLP can balance a quality but unbalanced (single objective and constraint) partition
- We observe up to 11% further improvement in edge cut produced by PuLP-MM when starting with a single constraint partition from KaFFPa
- No cost to second objective (max per-part cut), our approaches show up to 400% improvement relative to KaFFPa





Conclusions
and future work

- Average of **14.5**× faster, **23**× less memory, **better edge cut** or **better per-part edge cut** than next best tested partitioner on small-world graph test suite
- **Future work:**
  - Implementation in Zoltan2
  - Explore techniques for avoiding local minima, such as simulated annealing, etc.
  - Further parallelization in distributed environment for massive-scale graphs
  - Explore tradeoff and interactions in various parameters and iteration counts
  - New initialization procedures for various graph types (meshes, road nets, etc.)