How to Match in Parallel: The Power of Approximation Algorithms

Alex Pothen

Computer Science Department
Purdue University
www.cs.purdue.edu/homes/apothen/

HPC Days in Lyon, April 2016

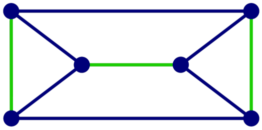
Structure of Talk

- Two stories in parallel matching
  - Approximate b-Matching (1/2)
  - Approximate b-Edge cover (3/2)
- Adaptive Anonymity

Matching Problems

A Matching M

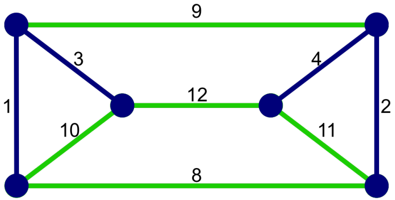
A set of independent edges in a graph G, i.e., at most one edge in M is incident on each vertex.



Maximum: Cardinality, Edge-weight, Vertex-weight, etc.

b-Matching Problems

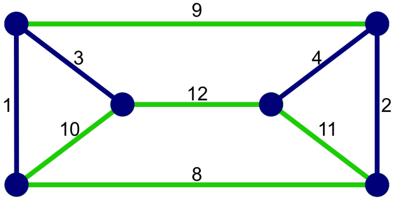
Given a graph $G = (V, E)$, and a function $b(v)$ for each vertex v, a b-Matching is a subset of edges M such that at most $b(v)$ edges in M are incident on a vertex v.



Here $b(v) = 2$ for all vertices. They do not all need to be equal.

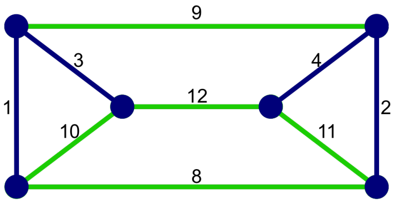
Weighted b-Matching

Given a graph $G = (V, E, W)$, and a function $b(v)$ for each vertex v, a maximum edge weighted b-Matching is a subset of edges M such that at most $b(v)$ edges in M are incident on a vertex v, and the weight of the edges in M is maximized.



Weighted b-Matching

Given a graph $G = (V, E, W)$, and a function $b(v)$ for each vertex v, a maximum edge weighted b-Matching is a subset of edges M such that at most $b(v)$ edges in M are incident on a vertex v, and the weight of the edges in M is maximized.



Message

- Exact algorithms have $O(B|V||E|)$ complexity, so approximate!
- Greedy algorithm, $1/2$-approximation, sort $+$ linear time.
- But little concurrency.
- Approximate more!

Message

- Exact algorithms have $O(B|V||E|)$ complexity, so approximate!
- Greedy algorithm, $1/2$-approximation, sort $+$ linear time.
- But little concurrency.
- Approximate more!

Message

- Locally Dominant edge algorithm.
- 1/2-approximation algorithm, near linear-time complexity (sort adjacency lists, not edge list)
- These compute with a local ordering that limits concurrency.
- Approximate even more!

Message

- Locally Dominant edge algorithm.
- $1/2$-approximation algorithm, near linear-time complexity (sort adjacency lists, not edge list)
- These compute with a local ordering that limits concurrency.
- Approximate even more!

Message

- Abandon orderings (global or local).
- Suitor algorithm for matching, and *b*-matching (does not work for edge cover).
- $1/2$-approximation, more concurrency, need to limit additional work in the algorithm.
- Greedy to Locally Dominant to Suitor algorithms, the solutions do not change!

Message

- Abandon orderings (global or local).
- Suitor algorithm for matching, and *b*-matching (does not work for edge cover).
- $1/2$-approximation, more concurrency, need to limit additional work in the algorithm.
- Greedy to Locally Dominant to Suitor algorithms, the solutions do not change!

Message

- Abandon orderings (global or local).
- Suitor algorithm for matching, and *b*-matching (does not work for edge cover).
- $1/2$-approximation, more concurrency, need to limit additional work in the algorithm.
- Greedy to Locally Dominant to Suitor algorithms, the solutions do not change!

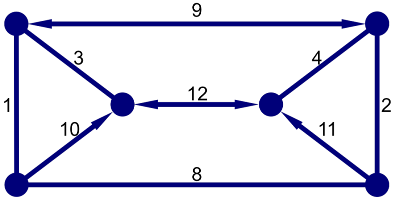
Serial Algorithms for Weighted *b*-Matching

We can modify half-approximation algorithms for 1-Matching to compute *b*-Matchings.

- Greedy: Match edges in decreasing order of weights. When an edge is matched, delete all edges incident on its endpoints.
- Path Growing + Dynamic Programming (PGA, PGA')
- Global Paths
- Locally Dominant edge (Preis)
- Suitor (Manne and Halappanavar)

Locally Dominant Edge Algorithm

A locally dominant edge is at least as heavy as any other edge incident on its end points.
Each vertex can set a pointer to its heaviest neighbor. If two vertices point to each other, then the edge is locally dominant.



Suitor Algorithm

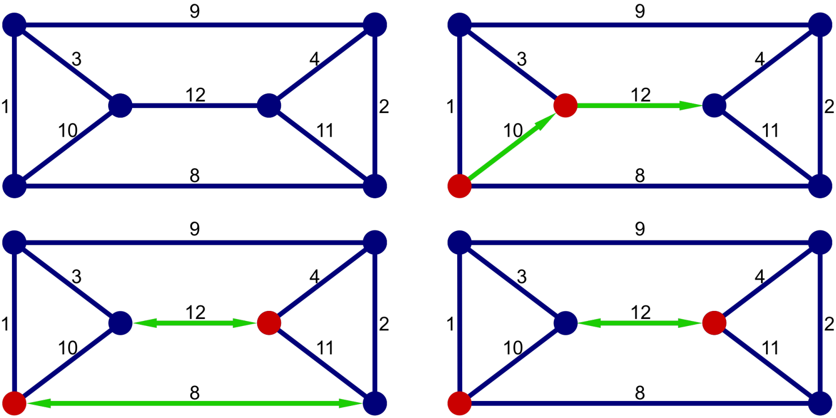
Vertices make proposals to their neighbors (in arbitrary order).
Vertices propose to neighbors in decreasing order of weights.
Each vertex records the best offer it has so far.
A vertex u proposes to its current heaviest neighbor v,
if it does not already have a better offer.
When two vertices propose to each other, they are matched.

Suitor Algorithm

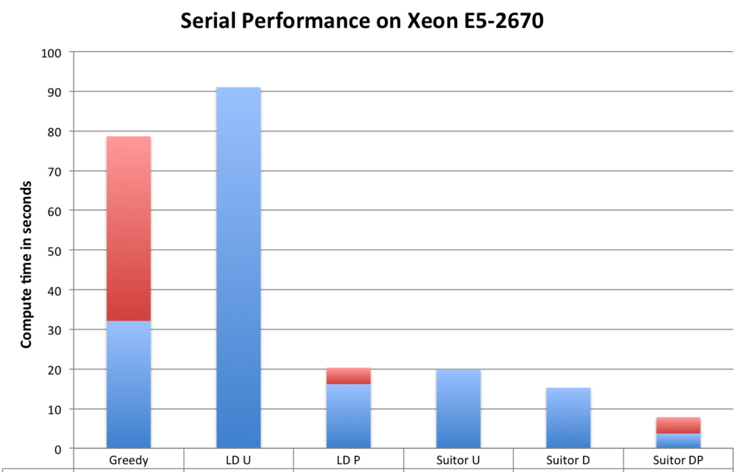
Consider u proposing to v, its current heaviest neighbor.

If v has a better offer, then consider next heaviest neighbor.
If v has a lower offer (from w, say), propose to v, annul the proposal of w to v.
Now w needs to make a new proposal.

Suitor Algorithm

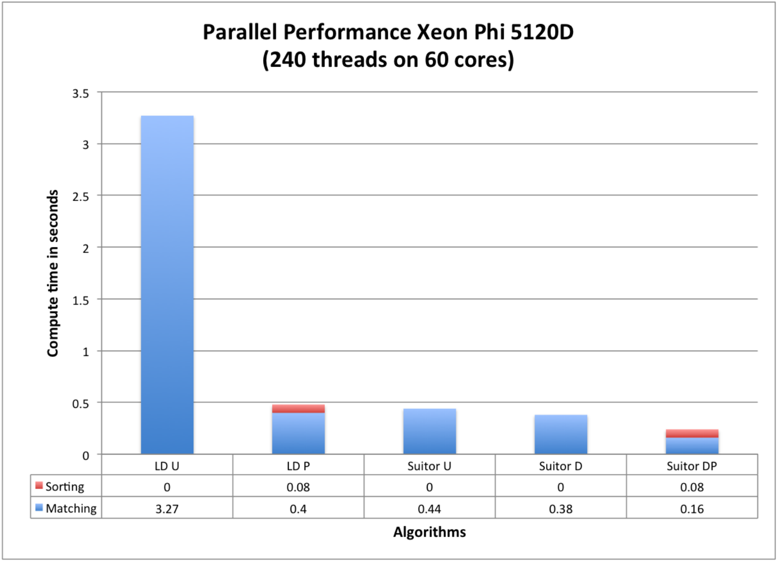


b-Matching, RMAT, 1M nodes, 67M edges



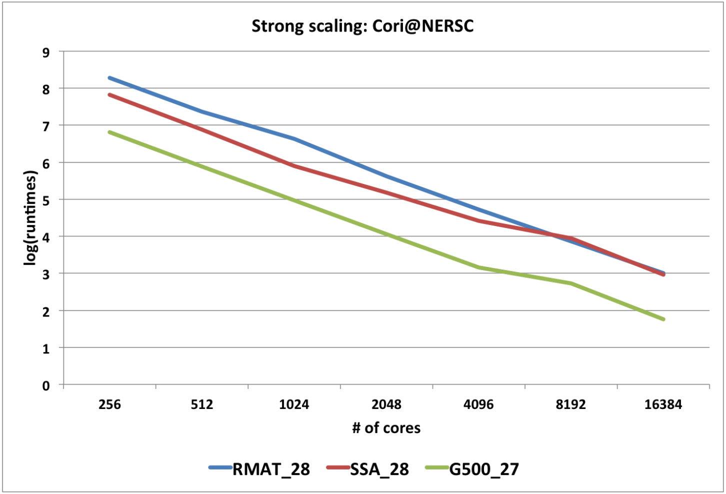
	Greedy	LD U	LD P	Suitor U	Suitor D	Suitor DP
Sorting	46.52	0	4.13	0	0	4.11
Matching	32.12	91	16.18	19.93	15.26	3.73

b-Matching, RMAT, 1M nodes, 67M edges



	LD U	LD P	Suitor U	Suitor D	Suitor DP
Sorting	0	0.08	0	0	0.08
Matching	3.27	0.4	0.44	0.38	0.16

b-Matching, 134/268M nodes, 2B edges



Distributed Memory Parallel Suitor Algorithm

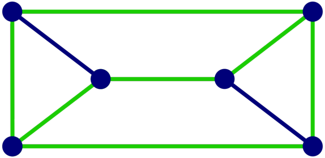
- Organize computations in aynchronous supersteps of computation and communication.
- Three classes of messages (proposal, annulment, rejection).
- Processors make proposals from vertices in some order (e.g., decreasing heaviest weight edge) to reduce the number of proposals.
- Processors process a subset of their vertices, and then send messages.
- Balance communication costs with the need for updated information about the state of current matching.
- Processors receive messages in any order, and can compute once they receive a message from any processor.

Structure of Talk

- Two stories in parallel matching
  - Approximate b-Matching ($1/2$)
  - Approximate b-Edge cover ($3/2$)
- Adaptive Anonymity

b-Edge Cover

Given a function $b(v)$, a set of edges with at least $b(v)$ edges incident on each vertex v.



Here $b(v) = 2$.
Minimum cardinality, edge weight, etc.
Solvable in polynomial time, but expensive.

Approximation Algorithms

- Greedy algorithm:
- effective weight of an edge is its weight divided by the number of its endpoints not included in the cover yet.
- add edge of least effective weight to cover
- update $b(v)$ value of neighboring edges
- update their effective weights

Approximation Algorithms

- This is a $3/2$-approximation algorithm.
- Note the weights change in the algorithm (unlike matching).
- Other approximation algorithms (e.g., Hall and Hochbaum, $O(\Delta)$ approximation ratio, max degree).
- Not much concurrency.

Locally Subdominant Edge Approximation

- Locally subdominant edge (LS): an edge whose effective weight is minimum among all edges incident on its endpoints.
  Add LS edges to the Edge cover, update effective weights, and repeat.
- $3/2$-approximation algorithm, with more concurrency than the Greedy algorithm.
- It computes exactly the same edge cover as Greedy!

Adaptive Anonymity

Users	*Features*					
	F_1	F_2	F_3	F_4	F_5	F_6
$U1$	1	0	1	0	1	0
$U2$	1	1	1	1	1	0
$U3$	0	1	0	1	0	1
$U4$	0	0	0	0	0	1
$U5$	1	1	0	0	0	0
$U6$	1	1	0	0	0	1

Adaptive Anonymity

Users	*Features*					
	F_1	F_2	F_3	F_4	F_5	F_6
$U1$	1	0	1	0	1	0
$U2$	1	1	1	1	1	0
$U3$	0	1	0	1	0	1
$U4$	0	0	0	0	0	1
$U5$	1	1	0	0	0	0
$U6$	1	1	0	0	0	1
$U1$	1	$*$	1	$*$	1	0
$U2$	1	$*$	1	$*$	1	0
$U3$	0	$*$	0	$*$	0	1
$U4$	0	$*$	0	$*$	0	1
$U5$	1	1	0	0	0	$*$
$U6$	1	1	0	0	0	$*$

Adaptive Anonymity algorithm

$X \in \mathbb{R}^{n \times f}$ is the instance-feature matrix;
Initialize weight matrix $W \in \mathbb{R}^{n \times f}$ with all one's;

for $i = 1$ to niter
    Calculate a weighted graph G from W and X
    Compute a min weight b-edge cover C in G
    Recalculate weight matrix W using C
    if convergence criterion is met **break**; **endif**
endfor

The edge cover groups each instance v with $\geq b(v)$ others

k-Anonymity with Approximation Algorithms

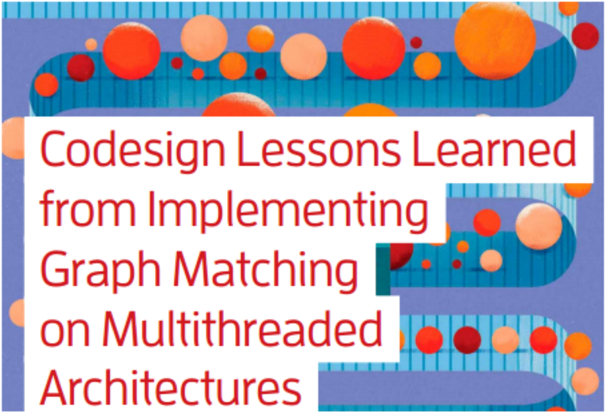
Prob.	*Inst.*	*Feat.*	b−*Matching*		b−*EdgeCover*	
			Time	*Util.*	*Time*	*Util.*
Caltech	768	101	0.9	94.5	6.6	93.7
Reed	7962	139	1.5	95.6	229	95.0
Haverford	1,446	145	3.0	97.2	34	96.6
*Simmons*81	1,518	140	3.2	96.7	62	96.0
UCIAdult	32.6*K*	101	26	97		
*Census*1990	158*K*	68	9*m*	90		
PokerHands	500*K*	95	2*h*17*m*	84		
CMS	1*M*	512	10*h*33*m*	81		

Two orders faster than exact *b*-Matching.

Approx. Algs. for Adaptive Anonymity

- b-Edge cover algorithm: $3/2$-approximation of the minimum weight edge cover. Satisfies the privacy requirements. The algorithm takes more time since effective weights of edges updated in the algorithm. Currently we store the entire dissimilarity matrix, but this could be avoided.
- b-Matching algorithm: $1/2$-approximation of the maximum weight matching. May not satisfy the privacy requirements exactly, but is faster since edge weights are static. The violations are few, and can be fixed easily. Stores only a subset of the dissimilarity matrix at a time, and has linear space complexity.

COVER FEATURE **IRREGULAR APPLICATIONS**



Codesign Lessons Learned from Implementing Graph Matching on Multithreaded Architectures

Mahantesh Halappanavar, Pacific Northwest National Laboratory
Alex Pothen, Purdue University
Ariful Azad, Lawrence Berkeley National Laboratory
Fredrik Manne, University of Bergen
Johannes Langguth, Simula Research Laboratory
Arif Khan, Purdue University

Executing irregular, data-intensive workloads on multithreaded

Recent Work: Matching

- Halappanavar, P, Azad, Langguth, Manne, Khan: Codesign of matching algorithms and multicore machines, IEEE Computer, August 2015.
- Khan, P, Patwary, Manne, MH et al: Approximation algorithms for weighted b-Matching, SISC, 2016, to appear.
- Azad, Buluc, P: Parallel Tree-Grafting Algorithm for Maximum Cardinality Matching, IPDPS 2015, TPDS 2016.
- Khan, Gleich, P: Network Alignment via Approximate Matching, Supercomputing 2012.
- Azad, Rajwa, P: Classifying Immunophenotypes with Templates from Flow Cytometry, WABI 2010; BMC Bioinformatics 2012; ACM Bioinformatics 2013.

Collaborators

b-matching, *b*-edge cover	Arif Khan
Suitor Algorithm	Fredrik Manne
	Mahantesh Halappanavar
Adaptive Anonymity	Krzysztof Choromanski (Google)
b-matching	Intel PCL
	(Mostofa Patwary, Nadathur Satish,
	Narayanan Sunderam, Pradeep Dubey)
Multicore, XMT, GPU	MH, John Feo, Antonino Tumeo,
	Oreste Villa
max cardinality	Ariful Azad, Aydin Buluc
Push Relabel	Johannes Langguth
Network Alignment	David Gleich
Comp. Immunology	Ariful Azad, Bartek Rajwa
Preconditioners	Bora Ucar, Michele Benzi

Seventh SIAM Workshop on Combinatorial Scientific Computing

October 10-12, 2016
Albuquerque, New Mexico USA

Webpage at www.siam.org/meetings/csc16
Organizers: Erik Boman (Sandia) and Assefaw Gebremedhin (Washington State Univ.)
Please submit papers by May 2, 2016.
Proceedings published by SIAM (electronic).