Vertex Weighted Matching: Parallel Approximation Algorithms

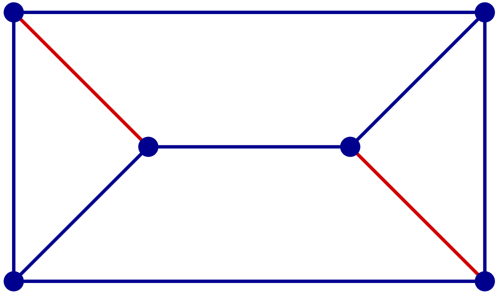
Ahmed Al-Herz and Alex Pothen
Department of Computer Science
Purdue University, USA

July 11, 2019

Outline

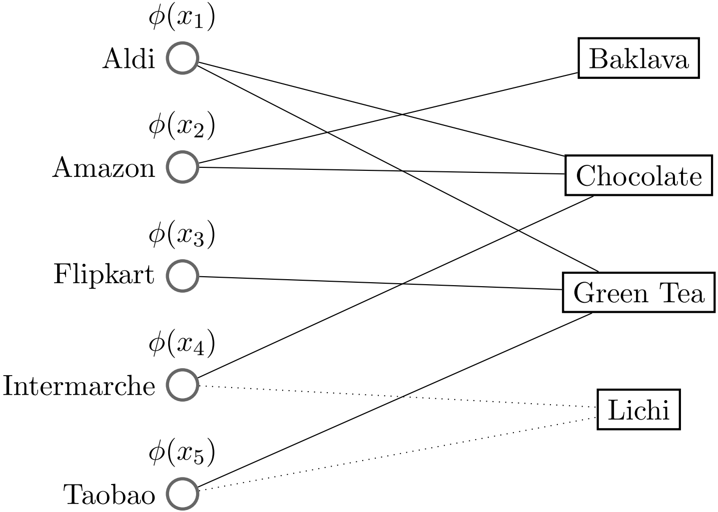
- Matching concepts
- Serial 2/3-approximation algorithms
- Parallel 2/3-approx. algorithm and Synchronization
- Experimental results
- Conclusions and References

Definitions



- Matching: a set of vertex-disjoint edges; hence at most one edge incident on each vertex.
- Maximum cardinality, edge-weighted matching
- We consider maximum vertex weighted matching. Arises in internet advertising as well as in computing sparse bases for the null space and column space of matrices, crew scheduling, etc.

Online Vertex-Weighted Matching



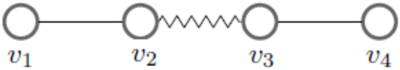
- There is an $O(1-1/e) \approx 0.6$-approx. algorithm for the online matching problem.

Online Vertex-Weighted Matching

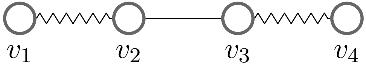
- Mehta (2012) survey: "the largest matching problem in the world, in terms of money and items matched". Asks for an approximate offline algorithm for large graphs.
- We report the first parallel 2/3-approx. algorithm (offline) for maximum vertex-weighted matching.

Matching concepts

- Alternating path: a path that has alternating matching and non-matching edges.
- Augmenting path: an odd length alternating path that begins and ends with unmatched edges.

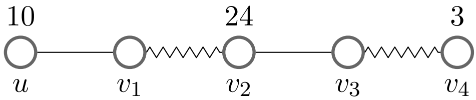


- By swapping matching and non-matching edges, we increase the weight and the cardinality of the matching.

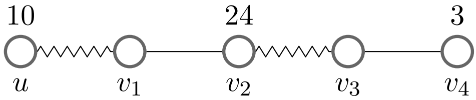


Matching concepts

- (Weight)-Increasing path: an even length alternating path whose unmatched start vertex u has higher weight than its matched end vertex v; hence $\phi(u) > \phi(v)$.



- By swapping matching and non-matching edges, we increase the weight (but not the cardinality) of the matching.



A More General Definition

- k-augmentation: an augmenting path or increasing path with *at most* k non-matching edges.
- 2-augmentation: augmenting paths of lengths one and three; increasing paths of length two and four.

Theorem

If a matching does not admit a k-augmentation, then it is a $k/(k+1)$-approximation to a maximum weight matching.

- Theorem is true for maximum cardinality and maximum edge-weighted matchings as well.

2/3-Direct Approximation Algorithm

procedure 2/3-DIRECT($G = (V, E, \phi)$)
  Initialize the matching M to be empty;
  Order the vertices in non-increasing order of weights;
  for each unmatched vertex u in order **do**
    Search for an aug. path P of length at most 3 that reaches a heaviest unmatched vertex v;
    If P is found, augment the matching M with $M \oplus P$;
  end for
end procedure

Dobrian, Halappanavar, Pothen, Al-Herz, SISC (2019)

2/3-Iterative Approximation Algorithm

```
procedure 2/3-ITER(G = (V, E, φ))
    [Initialize with a 2/3-approx. cardinality matching;]
    while a 2-augmentation exists do
        Search for a 2-augmentation P from u;
        If P is found, augment the matching with M ⊕ P;
        Update the set of unmatched vertices U;
    end while
end procedure
```

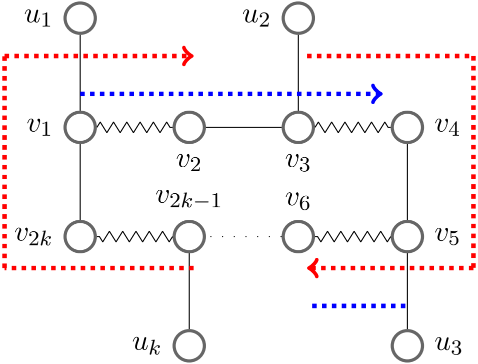
Direct vs. Iterative Algorithms for MVM

- The Direct algorithm employs augmenting paths only, and no increasing paths; Iterative algorithm employs both.
- The Direct algorithm sorts vertices and matches vertices in non-decreasing order of weights, while the Iterative algorithm processes unmatched vertices in any order, and hence can be implemented in parallel.
- The Iterative algorithm can be initialized with another matching; the Direct algorithm cannot be initialized.
- Worst-case time complexity (Δ is maximum degree, n no. of vertices, m no. of edges):
  Direct: $O(m \log \Delta + n \log n)$
  Iterative: $O(m\Delta^2)$

Parallel 2/3-approximate cardinality matching algorithm

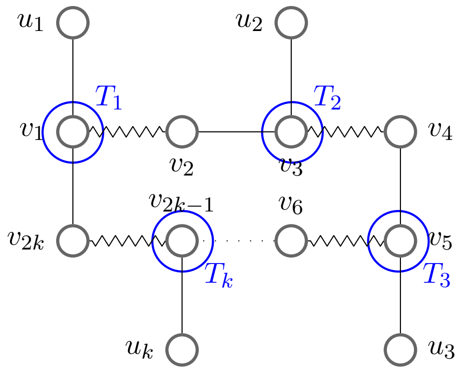
procedure PAR-2/3-ITER($G = (V, E, \phi)$)
    Initialize with 2/3-approx. cardinality matching computed in parallel;
    while a 2-augmentation exists in parallel **do**
        Search for a 2-augmentation P from an unmatched vertex u;
        If P is found, try to lock vertices on P;
        If locks obtained, augment M with $M \leftarrow M \oplus P$;
        Release all locks;
        Update set of unmatched vertices;
    end while
end procedure

Livelock



Threads $\{T_i\}$ computes an increasing path beginning at u_i that includes matched edges (v_{2i-1}, v_{2i}) and (v_{2i+1}, v_{2i+2}). Thread T_k includes matched edges (v_{2k-1}, v_k) and (v_1, v_2).

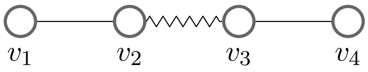
Livelock



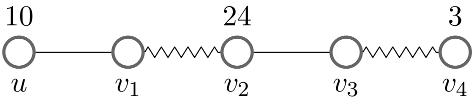
If each thread T_i locks v_{2i-1}, we have a cyclic wait!
Cyclic wait happens also for augmenting paths of length 3.

Locking procedure

- If u, v are the endpoints of an augmenting path, then we permit a thread to augment from u to v only if $u < v$.
- We lock the lower numbered endpoint of a matching edge.



Lock (in order) v_1, v_4, $\min\{v_2, v_3\}$



Let $m_1 = \min\{v_1, v_2\}$, $m_2 = \min\{v_3, v_4\}$
Lock (in order) u, $\min\{m_1, m_2\}$ and $\max\{m_1, m_2\}$

Parallel Iterative Algorithm and Locks

- If a thread fails to acquire a lock, it releases all locks and processes the next unmatched vertex or moves to the next iteration.
- Bad things that could happen:
  Deadlock: some thread waiting for ever to acquire a lock.
  Livelock: some threads cannot make any progress in matching their vertices.
  Starvation: some thread cannot match any vertices at all.

No bad news Theorem

While there is a 2-augmentation in the graph, in every iteration of the parallel iterative algorithm at least one thread will succeed in matching a vertex.

The set of test problems

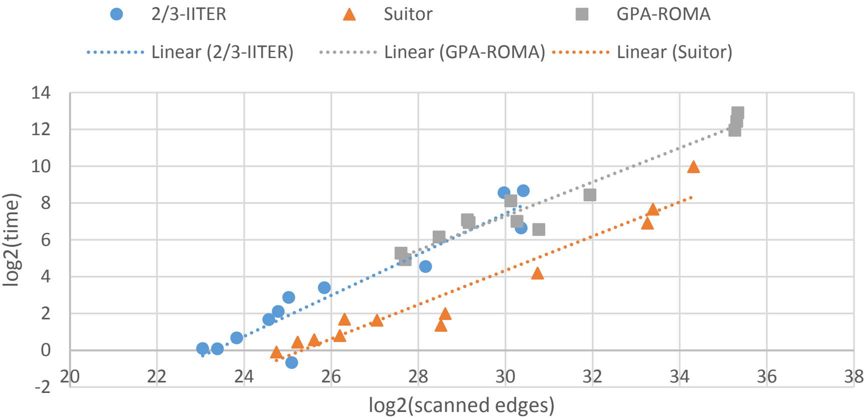
Graph	$\lvert V\rvert$	Degree **Mean**	**SD/ Mean**	$\lvert E\rvert$
kron_g500	2 097 152	117.92	7.47	91 040 932
M6	3 501 776	5.99	0.14	10 501 936
hugetric	6 592 765	2.99	0.01	9 885 854
rgg_n_2_23_s0	8 388 608	15.14	0.26	63 501 393
hugetrace	12 057 441	2.99	0.01	18 082 179
nlpkkt200	16 240 000	26.60	0.09	215 992 816
hugebubbles	19 458 087	2.99	0.01	29 179 764
road_usa	23 947 347	2.41	0.39	28 854 312
europe_osm	50 912 018	2.12	0.23	54 054 660
rmat-G500	48 877 747	85 28	15.48	2 084 251 521
rmat-SSCA	93 488 461	45.29	9.96	2 117 212 258
rmat-ER	134 217 728	32.00	0.29	2 147 483 625

Running Time of Algorithms

- Running time of a Maximum vertex weighted matching algorithm on these problems ranges from 10 to 6000 seconds.
- Relative performance of 6 approximation algorithms are computed by the ratio of the time for the Exact algorithm and the time for the approximation algorithm.

Algorithm	$1-\epsilon$-Scal $\epsilon = 1/3$	$2/3-\epsilon$ GPA-ROMA $\epsilon = 0.01$	2/3-DIR	2/3-ITER	1/2-ITER	Suitor
Geom. Mean	0.63	1.3	23	40	**110**	43

Run times vs. Edges Searched



Computed weights

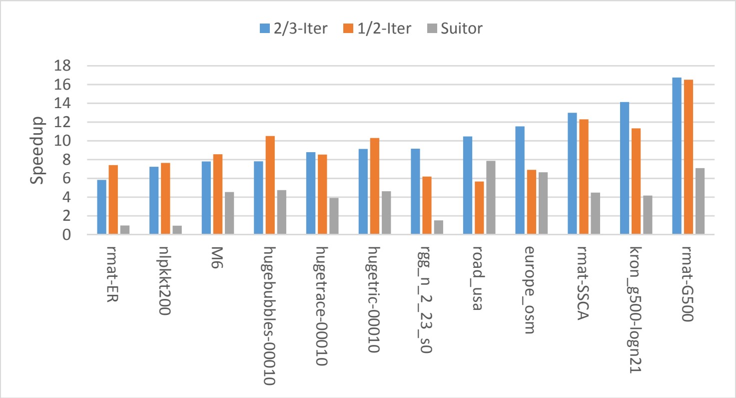
- Gap to optimality (%) is

$$(1 - \frac{\text{Weight of approx. matching}}{\text{Weight of maximum matching}}) * 100.$$

Graph	$1-\epsilon$-Scal. $\epsilon = 1/3$	$2/3-\epsilon$ GPA-ROMA $\epsilon = 0.01$	2/3-DIR	2/3-ITER	1/2-ITER	Suitor
Geom. Mean	1.25	0.23	0.32	**0.084**	0.76	3.3

The 2/3-approx. algorithms compute nearly optimal weights!

Speedup on 20 Xeon threads



Conclusions

- We have described a new serial 2/3-approximation algorithm for the maximum vertex-weighted matching problem.
- This algorithm can be adapted to provide the first parallel algorithm with approximation ratio better than 1/2 for any matching problem.
- Vertex-weighted matching problems provide hard test cases for edge-weighted matching problems.

A Puzzle
Exact Edge-weighted Matching on GL7D20

	Random Weights		
Metric	Original	edge $[0, 1]$	vertex $[0, 0.5]$ summed
Time (s)	4.61 E0	1.330 E1	1.001 E5
Cardinality	1,437,546	1,437,546	1,437,546
No. augs.	1,437,546	1,437,546	1,437,546
Aug. path length			
Maximum	9	61	2383
No. distinct	5	30	938
Mean	1.009	1.896	72.55
No. dual updates	1.35 E7	7.49 E6	3.92 E10
Time aug. paths	3.38	9.72	4.31 E4
Time dual updates	0.42	0.27	4.44 E3

References

- Ahmed Al-Herz and AP: A parallel 2/3-approx. alg. for maximum vertex-weighted matching (MVM), Preprint, 2019.
- Ahmed Al-Herz and AP: A 2/3-approx. alg. for MVM in non-bipartite graphs, Discrete Applied Mathematics, under review, 2018. (Arxiv:1902.05877)
- Florin Dobrian, Mahantesh Halappanavar, AP, Ahmed Al-Herz: A 2/3-approx. alg. for MVM in bipartite graphs, SISC 41:A566-A591, 2019.
- AP, S M Ferdous, and Fredrik Manne: Approximation algorithms in combinatorial scientific computing, Acta Numerica, 28, pp. 541-633, 2019.

Scalability on 20 Xeon threads