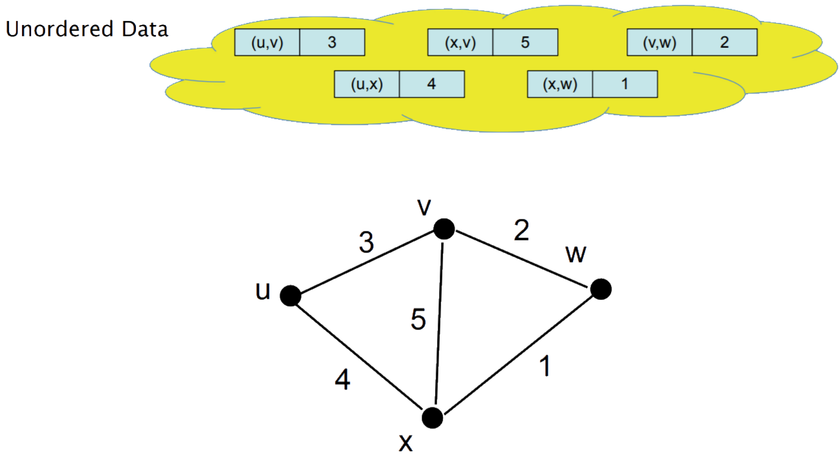
Greedy and Local Ratio Algorithms in the MapReduce Model

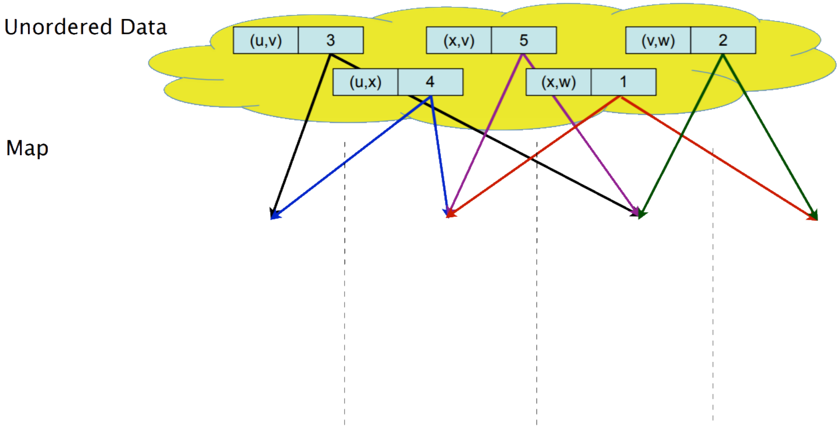
Nick Harvey [1] Chris Liaw [1] **Paul Liu** [2]

[1]University of British Columbia

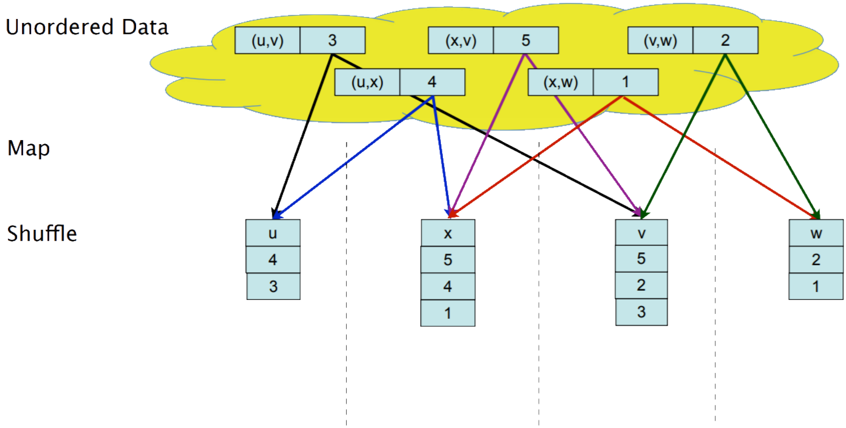
[2]Stanford University

MapReduce at a glance [Vassilvitskii, 2012]

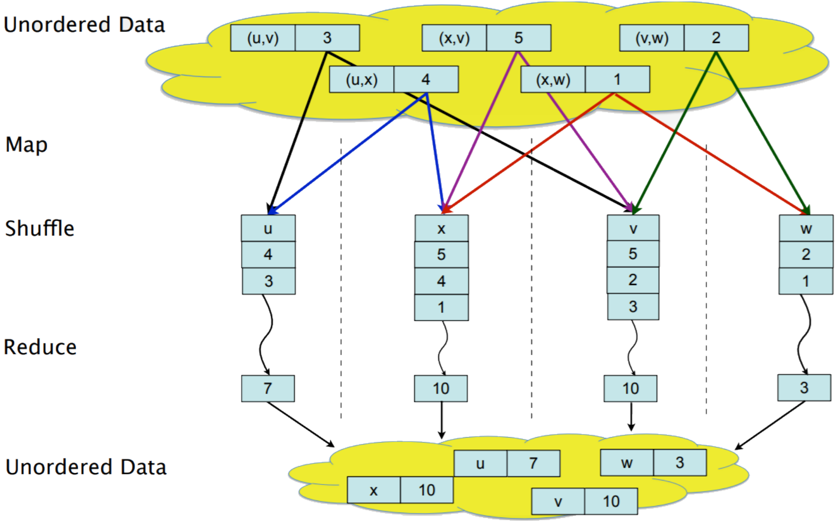



Unordered Data
Map
(u,v) 3
(x,v) 5
(v,w) 2
(u,x) 4
(x,w) 1

MapReduce at a glance [Vassilvitskii, 2012]



MapReduce at a glance [Vassilvitskii, 2012]



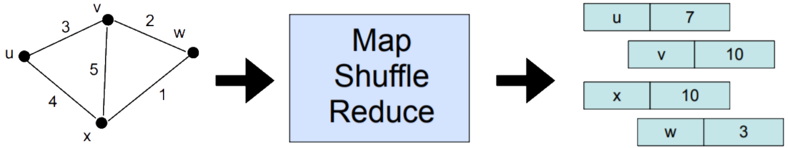
MapReduce at a glance [Vassilvitskii, 2012]

Data:

- Represented as <Key, Value> pairs

Operations:

- Map: <Key, Value> → List(<Key, Value>)
- Shuffle: Aggregate all pairs with the same key
- Reduce: <Key, List(Value)> → <Key, List(Value)>



MRC model for graphs [Karloff et al., 2010]

- n number of vertices
- $m = n^{1+c}$ input size (number of edges)
- $n^{1+\mu}$ memory on each machine, $0 < \mu \leq c$
- $n^{c-\mu}$ machines

Efficiency: algorithms that run in $O(1)$ rounds (often $O(c/\mu)$).

Real world graphs [Leskovec et al., 2005]

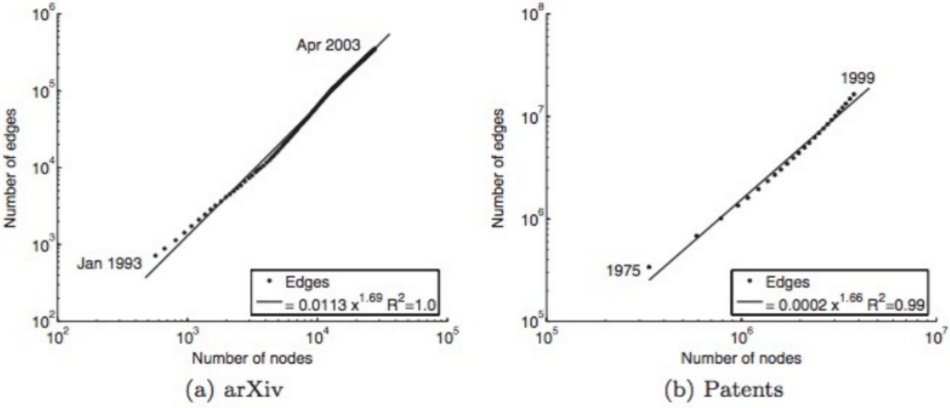


Figure: $|E| \approx n^{1.69}$ (left) and $n^{1.66}$ (right)

Graph problems in MapReduce

- Filtering: unweighted vertex cover, matching, edge cover [Lattanzi et al., 2011]
- Rounding and linear programming for weighted matching [Lattanzi et al., 2011, Crouch and Stubbs, 2014, Grigorescu et al., 2016, Ahn and Guha, 2015]

Graph problems in MapReduce

- Filtering: unweighted vertex cover, matching, edge cover [Lattanzi et al., 2011]
- Rounding and linear programming for weighted matching [Lattanzi et al., 2011, Crouch and Stubbs, 2014, Grigorescu et al., 2016, Ahn and Guha, 2015]

Our paper:

- Randomized local ratio (weighted matching, vertex cover, etc.)
- Hungry-greedy (maximal indep. set, maximal clique, etc.)
- Additional results on colouring problems

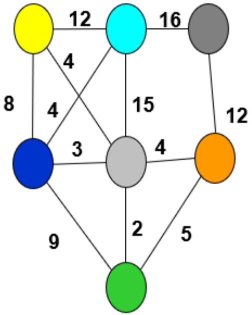
Randomized Local Ratio

Sequential Local Ratio

- Approximation technique for NP-hard optimization problems
- Initially used in the 80s for finding low cost vertex covers [Bar-Yehuda and Even, 1985]
- Recently used by [Paz and Schwartzman, 2017] to find weighted matchings in the streaming model

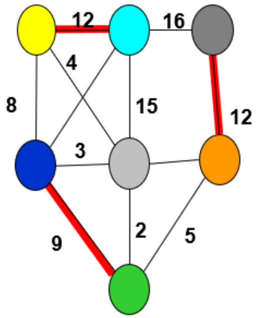
Local Ratio - Weighted Matching

A maximum weight matching is a set $M \subseteq E$ s.t. no two $e \in M$ shares an endpoint and the sum of weights in M is maximized.

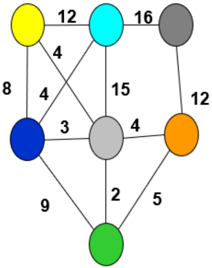


Local Ratio - Weighted Matching

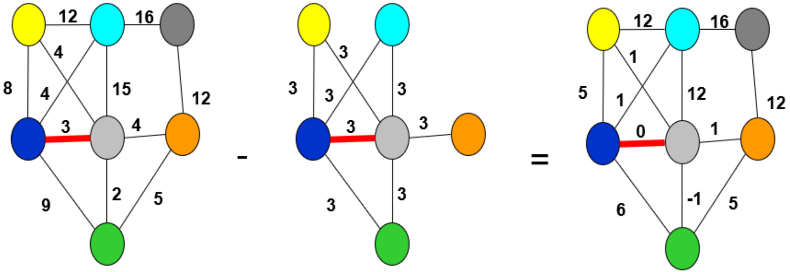
A maximum weight matching is a set $M \subseteq E$ s.t. no two $e \in M$ shares an endpoint and the sum of weights in M is maximized.



Select **any** edge e with positive weight. Reduce its weight from itself and its neighboring edges. Push e onto a stack. Repeat this procedure until no positive weight edges remain. Pop edges off the stack, adding greedily to the matching.



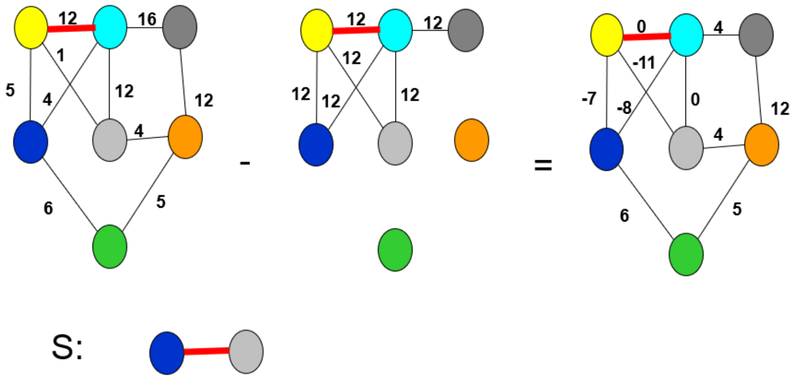
Select **any** edge e with positive weight. Reduce its weight from itself and its neighboring edges. Push e onto a stack. Repeat this procedure until no positive weight edges remain. Pop edges off the stack, adding greedily to the matching.



S:

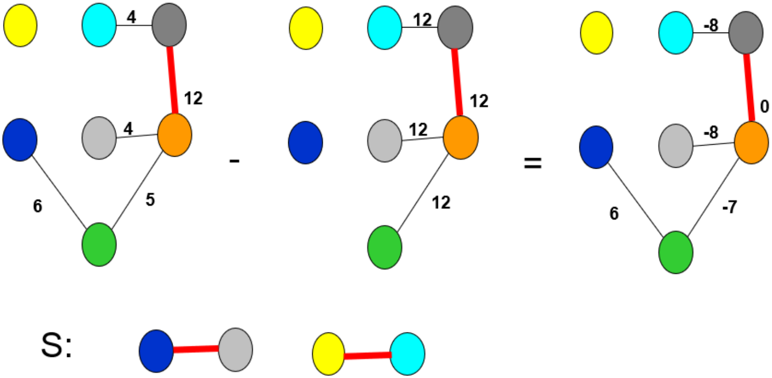
Local Ratio - Weighted Matching

Select **any** edge e with positive weight. Reduce its weight from itself and its neighboring edges. Push e onto a stack. Repeat this procedure until no positive weight edges remain. Pop edges off the stack, adding greedily to the matching.



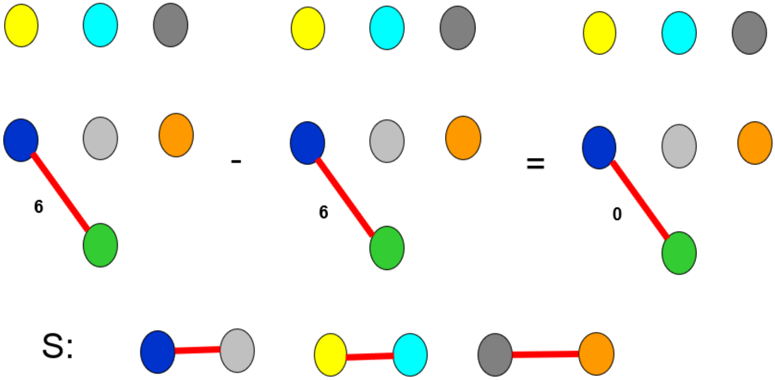
Local Ratio - Weighted Matching

Select **any** edge e with positive weight. Reduce its weight from itself and its neighboring edges. Push e onto a stack. Repeat this procedure until no positive weight edges remain. Pop edges off the stack, adding greedily to the matching.



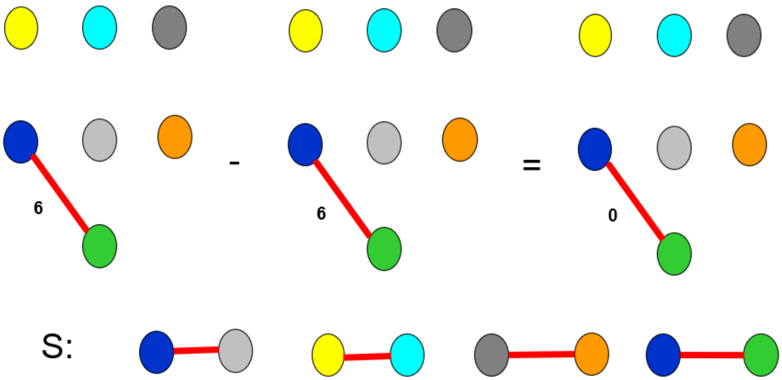
Local Ratio - Weighted Matching

Select **any** edge e with positive weight. Reduce its weight from itself and its neighboring edges. Push e onto a stack. Repeat this procedure until no positive weight edges remain. Pop edges off the stack, adding greedily to the matching.



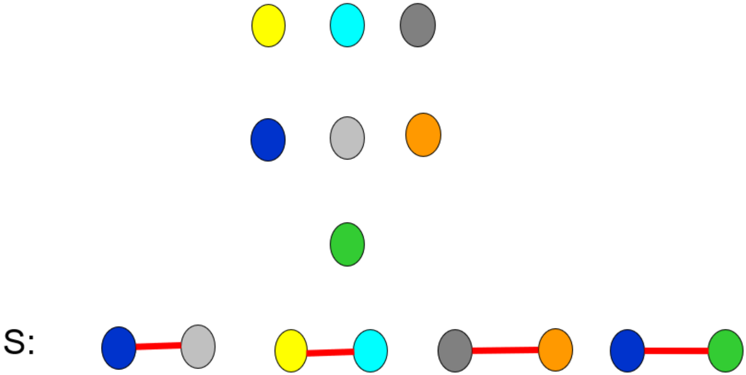
Local Ratio - Weighted Matching

Select **any** edge *e* with positive weight. Reduce its weight from itself and its neighboring edges. Push *e* onto a stack. Repeat this procedure until no positive weight edges remain. Pop edges off the stack, adding greedily to the matching.



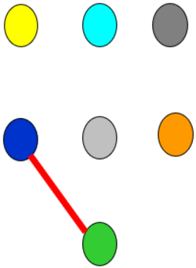
Local Ratio - Weighted Matching

Select **any** edge e with positive weight. Reduce its weight from itself and its neighboring edges. Push e onto a stack. Repeat this procedure until no positive weight edges remain. Pop edges off the stack, adding greedily to the matching.

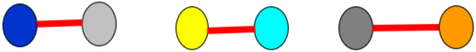


Local Ratio - Weighted Matching

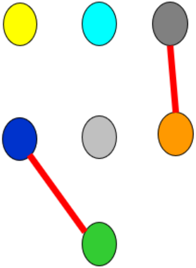
Select **any** edge e with positive weight. Reduce its weight from itself and its neighboring edges. Push e onto a stack. Repeat this procedure until no positive weight edges remain. Pop edges off the stack, adding greedily to the matching.

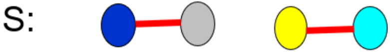


S:

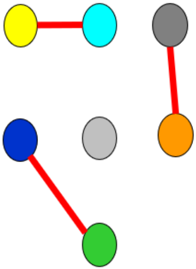


Select **any** edge e with positive weight. Reduce its weight from itself and its neighboring edges. Push e onto a stack. Repeat this procedure until no positive weight edges remain. Pop edges off the stack, adding greedily to the matching.

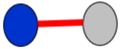




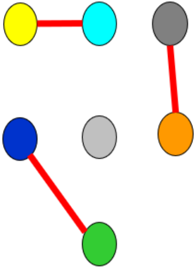
Select **any** edge e with positive weight. Reduce its weight from itself and its neighboring edges. Push e onto a stack. Repeat this procedure until no positive weight edges remain. Pop edges off the stack, adding greedily to the matching.



S:

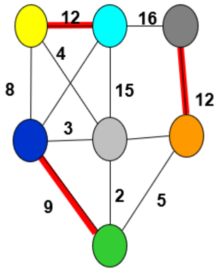


Select **any** edge e with positive weight. Reduce its weight from itself and its neighboring edges. Push e onto a stack. Repeat this procedure until no positive weight edges remain. Pop edges off the stack, adding greedily to the matching.



S:

Select **any** edge e with positive weight. Reduce its weight from itself and its neighboring edges. Push e onto a stack. Repeat this procedure until no positive weight edges remain. Pop edges off the stack, adding greedily to the matching.



Local Ratio - Weighted Matching

Select **any** edge e with positive weight. Reduce its weight from itself and its neighboring edges. Push e onto a stack. Repeat this procedure until no positive weight edges remain. Pop edges off the stack, adding greedily to the matching.

Theorem ([Paz and Schwartzman, 2017])

The sequential local ratio algorithm gives a 2-approximation for weighted matching.

Randomized Local Ratio

- Can cover a large fraction of the remaining sets with just a random sample of elements
- Random sample will be comparable to optimal solution since the processing order of local ratio is arbitrary

MapReduce - Weighted Matching

Initialize an empty stack S and repeat while E is not empty:

- Sample $8n^{1+\mu}$ edges uniformly from E
- Run local ratio on sampled edges, processing edges in order of **decreasing** weight, adding onto S
- Remove non-positive edges from E

Pop edges off S, adding greedily to the matching.

MapReduce - Weighted Matching

Initialize an empty stack S and repeat while E is not empty:

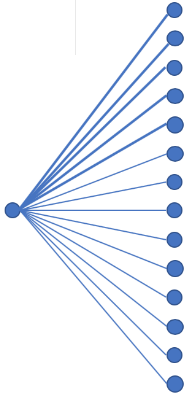
- Sample $8n^{1+\mu}$ edges uniformly from E
- Run local ratio on sampled edges, processing edges in order of **decreasing** weight, adding onto S
- Remove non-positive edges from E

Pop edges off S, adding greedily to the matching.

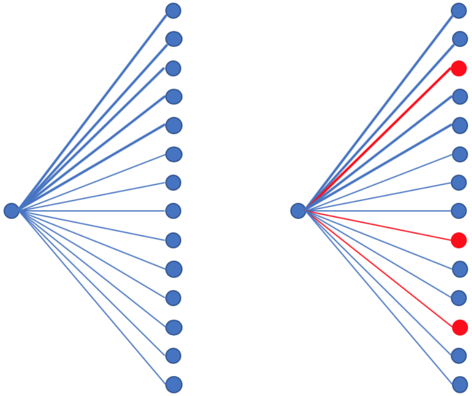
Theorem (Our paper)

Each iteration, the number of edges decrease by a factor of n^{μ}. With $|E| = n^{1+c}$ edges, the entire algorithm terminates in $O(c/\mu)$ iterations.

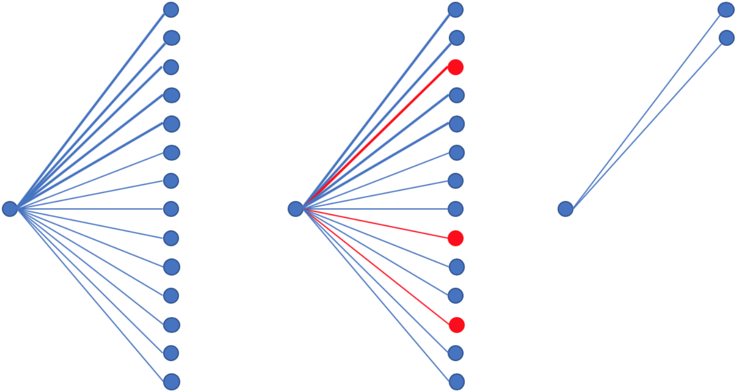
MapReduce - Weighted Matching



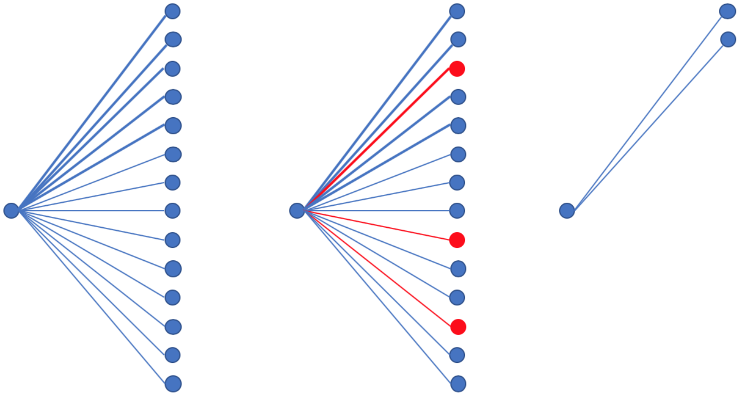
MapReduce - Weighted Matching



MapReduce - Weighted Matching

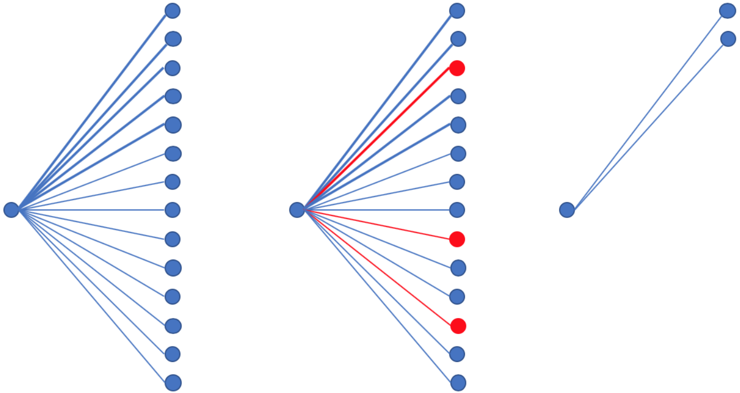


MapReduce - Weighted Matching



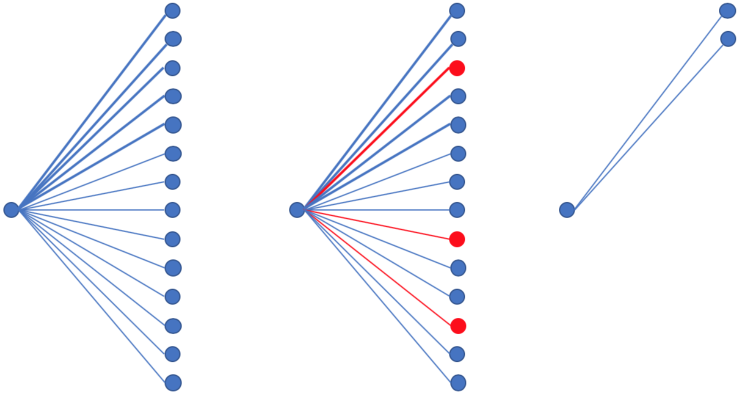
- If the k-th heaviest edge adjacent to a node v is added to S, all but the top k edges are removed.

MapReduce - Weighted Matching



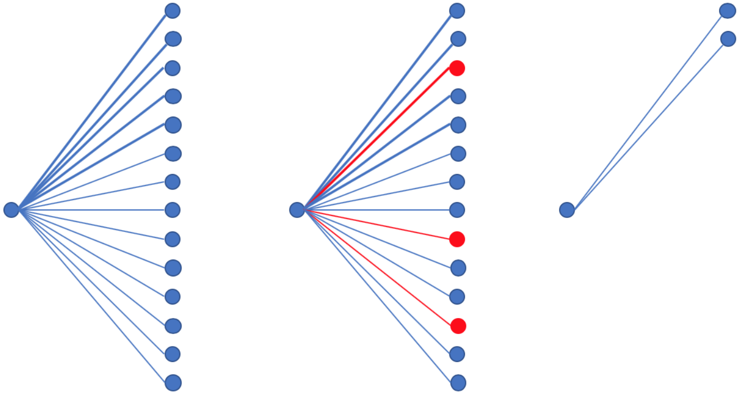
- Each node v has $O(n^{\mu})$ edges sampled, so we get one of the top $O(\deg(v)/n^{\mu})$ heaviest edges

MapReduce - Weighted Matching



- Degree of each node decreases by a factor of n^{μ} each iteration

MapReduce - Weighted Matching



- Further analysis gives $O(c/\mu)$ rounds instead of $O(1/\mu)$

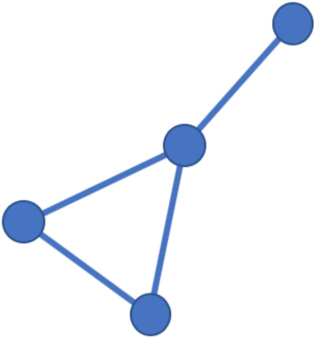
Hungry-greedy

Hungry-greedy

- Unlike local ratio, sampling is non-uniform
- **Heavy** elements are sampled first, to quickly disqualify a large fraction of elements from the solution

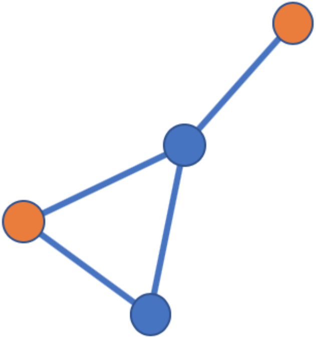
Hungry-greedy - Maximal Independent Set

Given a graph $G = (V, E)$, an independent set (IS) is a set $S \subseteq V$ s.t. no two $v \in S$ are adjacent. A **maximal** independent set (MIS) is not a subset of any other IS.



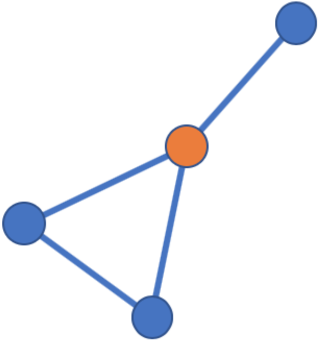
Hungry-greedy - Maximal Independent Set

Given a graph $G = (V, E)$, an independent set (IS) is a set $S \subseteq V$ s.t. no two $v \in S$ are adjacent. A **maximal** independent set (MIS) is not a subset of any other IS.



Hungry-greedy - Maximal Independent Set

Given a graph $G = (V, E)$, an independent set (IS) is a set $S \subseteq V$ s.t. no two $v \in S$ are adjacent. A **maximal** independent set (MIS) is not a subset of any other IS.

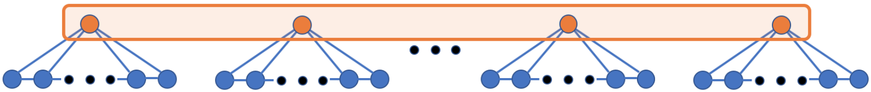


Hungry-greedy - Maximal Independent Set

Hungry-greedy - Maximal Independent Set

Hungry-greedy - Maximal Independent Set

Hungry-greedy - Maximal Independent Set



Hungry-greedy - Maximal Independent Set

For $i = 1, \ldots, 2/\mu$

- $V_H \leftarrow \{v \in V \mid \deg_I(v) > n^{1-i\mu/2}\}$
- While $|V_H| > n^{i\mu/2}$:
  - Draw $n^{i\mu/2}$ groups of $n^{\mu/2}$*vertices*
  - For each group, add any vertices v to I if $\deg_I(v) > n^{i\mu/2}$
  - Remove any vertices from V_H with small degree
- Find an MIS in V_H and add to I

Theorem (Our paper)

Each iteration, the maximum degree of the graph decreases by a factor of $n^{\mu/4}$.

Additional Results

Problem	Approximation	MapReduce Rounds	Space per machine
Weighted b-matching	$3 - 2/b$	$O(c/\mu)$	$O(n^{1+\mu})$
Weighted vertex cover	2	$O(c/\mu)$	$O(n^{1+\mu})$
Weighted set cover	f	$O((c/\mu)^2)$	$O(f \cdot n^{1+\mu})$
	$(1+\varepsilon)\ln\Delta$	$O\left(\frac{\log\left(\frac{w_{\max}}{w_{\min}}\Delta\right)}{\mu^2\varepsilon}\right)$ if $n = \text{poly}(m)$	$O(m^{1+\mu})$
Maximal Clique		$O(1/\mu)$	$O(n^{1+\mu})$
Vertex Colouring	$(1+o(1))\Delta$ colours	$O(1)$	$O(n^{1+\mu})$
Edge Colouring	$(1+o(1))\Delta$ colours	$O(1)$	$O(n^{1+\mu})$

See full paper for definition of various parameters.

Techniques:

- Randomized local ratio
- Hungry-greedy

Thanks for listening! Questions?

The $\mu = 0$ case

Problem	Weighted?	Approximation	MapReduce Rounds	Space per machine
Vertex Cover	Y	2	$O(\log n)$	$O(n)$
Matching	Y	2	$O(\log n)$	$O(n)$
b-matching	Y	$3 - 2/b$	$O(\log n)$	$O(n)$
Maximal Indep. Set			$O(\log n)$	$O(n)$
Maximal Clique			$O(\log n)$	$O(n)$
Set Cover	Y	f	$O(\log^2 n)$	$O(f \cdot n)$
Vertex Colouring		$(1 + o(1))\Delta$ colours	$O(1)$	$\tilde{O}(n)$
Edge Colouring		$(1 + o(1))\Delta$ colours	$O(1)$	$\tilde{O}(n)$

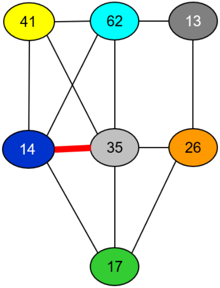
Techniques:

- Randomized local ratio
- Hungry-greedy

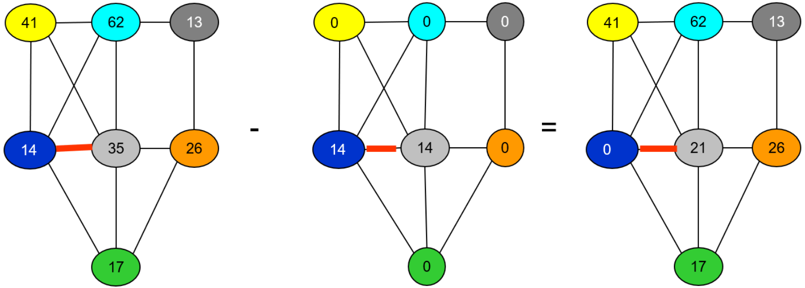
Sequential Local Ratio - Weighted Vertex Cover

Suppose $G = (V, E)$ with weights w_v on the vertices $v \in V$.
A minimum vertex cover is a set $S \subseteq V$ s.t. any $e \in E$ has at least one endpoint in S and the sum of weights in S is minimized.

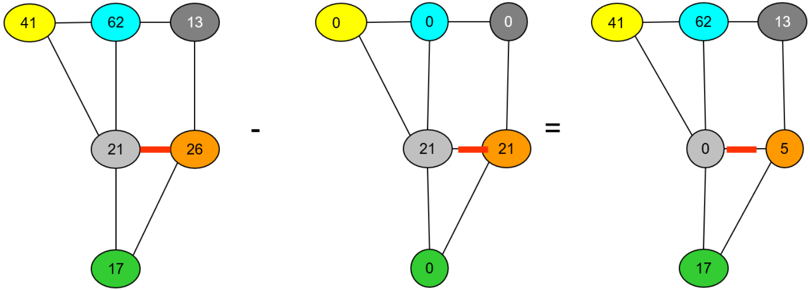
Select **any** uncovered edge e whose vertices have positive weight. Reduce the minimum of the two vertex weights from the two vertices. Remove all vertices with weight 0 and add them to the cover. Repeat while edges remain.



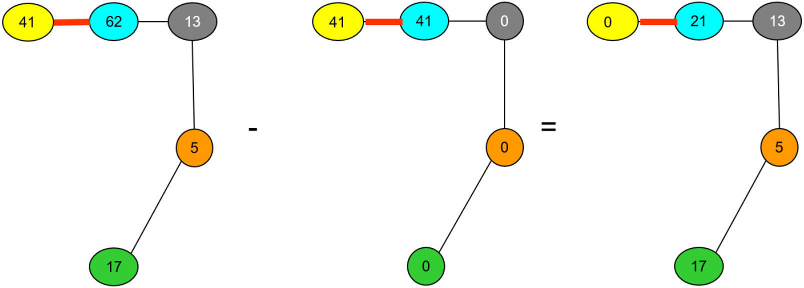
Select **any** uncovered edge e whose vertices have positive weight. Reduce the minimum of the two vertex weights from the two vertices. Remove all vertices with weight 0 and add them to the cover. Repeat while edges remain.



Select **any** uncovered edge e whose vertices have positive weight. Reduce the minimum of the two vertex weights from the two vertices. Remove all vertices with weight 0 and add them to the cover. Repeat while edges remain.

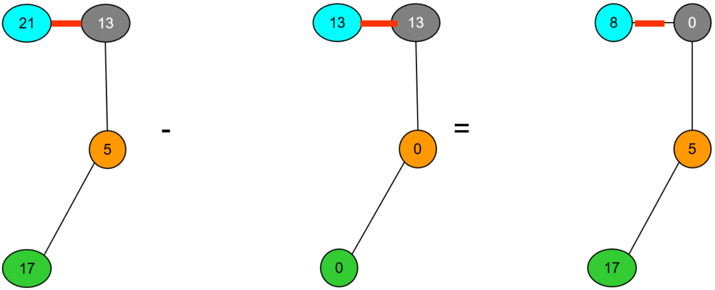


Select **any** uncovered edge e whose vertices have positive weight. Reduce the minimum of the two vertex weights from the two vertices. Remove all vertices with weight 0 and add them to the cover. Repeat while edges remain.

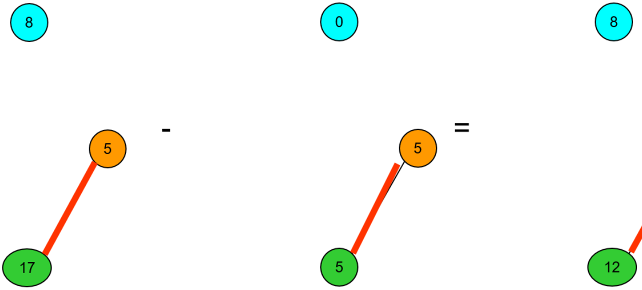


Sequential Local Ratio - Weighted Vertex Cover

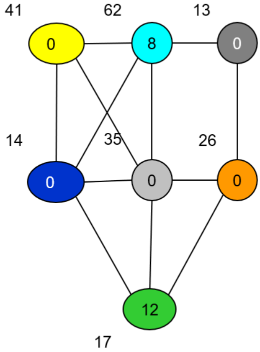
Select **any** uncovered edge e whose vertices have positive weight. Reduce the minimum of the two vertex weights from the two vertices. Remove all vertices with weight 0 and add them to the cover. Repeat while edges remain.



Select **any** uncovered edge *e* whose vertices have positive weight. Reduce the minimum of the two vertex weights from the two vertices. Remove all vertices with weight 0 and add them to the cover. Repeat while edges remain.



Select **any** uncovered edge e whose vertices have positive weight. Reduce the minimum of the two vertex weights from the two vertices. Remove all vertices with weight 0 and add them to the cover. Repeat while edges remain.



Select **any** uncovered edge e whose vertices have positive weight. Reduce the minimum of the two vertex weights from the two vertices. Remove all vertices with weight 0 and add them to the cover. Repeat while edges remain.

Theorem ([Bar-Yehuda and Even, 1985])

The sequential local ratio algorithm gives a 2-approximation for weighted vertex cover.

Local Ratio in MapReduce - Weighted Vertex Cover

Repeat while E is not empty:

- Sample each edge in E with probability $2n^{1+\mu}/|E|$
- Run local ratio on sampled edges
- Add all vertices of zero weight to cover and update E

Theorem (Our paper)

Each iteration, the number of edges decrease by a factor of n^{μ}. With $|E| = n^{1+c}$ edges, the entire algorithm terminates in $O(c/\mu)$ iterations.

Local Ratio in MapReduce - Weighted Vertex Cover

Repeat while E is not empty:

- Sample $2n^{1+\mu}$ edges from E uniformly
- Run local ratio on sampled edges
- Add all vertices of zero weight to cover and update E

Intuition:

- Edges left for the next iteration must have both endpoints with positive weight.
- If there are more than $|E|/n^{\mu}$ such edges, then sampling $2n^{1+\mu}$ edges should have sampled all of them with high probability.
- If an edge is sampled, at least one of its endpoints must be zero weight, so it couldn't have been in the next iteration

Local Ratio in MapReduce - Weighted Vertex Cover

Repeat while E is not empty:

- Sample $2n^{1+\mu}$ edges from E uniformly
- Run local ratio on sampled edges
- Add all vertices of zero weight to cover and update E

Can be further generalized to an f-approximation for set cover, where each element is in at most f sets.

[Ahn and Guha, 2015] Ahn, K. J. and Guha, S. (2015).
Access to data and number of iterations: Dual primal algorithms for maximum matching under resource constraints.
In *Proceedings of the 27th ACM symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 202–211.

[Bar-Yehuda and Even, 1985] Bar-Yehuda, R. and Even, S. (1985).
A local-ratio theorem for approximating the weighted vertex cover problem.
Annals of Discrete Mathematics, 25:27–45.

[Crouch and Stubbs, 2014] Crouch, M. and Stubbs, D. S. (2014).
Improved streaming algorithms for weighted matching, via unweighted matching.
In *International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*, pages 96–104.

[Grigorescu et al., 2016] Grigorescu, E., Monemizadeh, M., and Zhou, S. (2016).
Streaming weighted matchings: Optimal meets greedy.

[Karloff et al., 2010] Karloff, H. J., Suri, S., and Vassilvitskii, S. (2010).
A model of computation for MapReduce.
In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 938–948.

[Lattanzi et al., 2011] Lattanzi, S., Moseley, B., Suri, S., and Vassilvitskii, S. (2011).
Filtering: a method for solving graph problems in MapReduce.
In *Proceedings of the 23rd Annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 85–94.

References (cont.)

[Leskovec et al., 2005] Leskovec, J., Kleinberg, J. M., and Faloutsos, C. (2005).
Graphs over time: densification laws, shrinking diameters and possible explanations.
In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Chicago, Illinois, USA, August 21-24, 2005*, pages 177–187.

[Paz and Schwartzman, 2017] Paz, A. and Schwartzman, G. (2017).
A $(2 + \epsilon)$-approximation for maximum weight matching in the semi-streaming model.
In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2153–2161.

[Vassilvitskii, 2012] Vassilvitskii, S. (2012).
CSCI 8980: Algorithmic techniques for big data analysis.