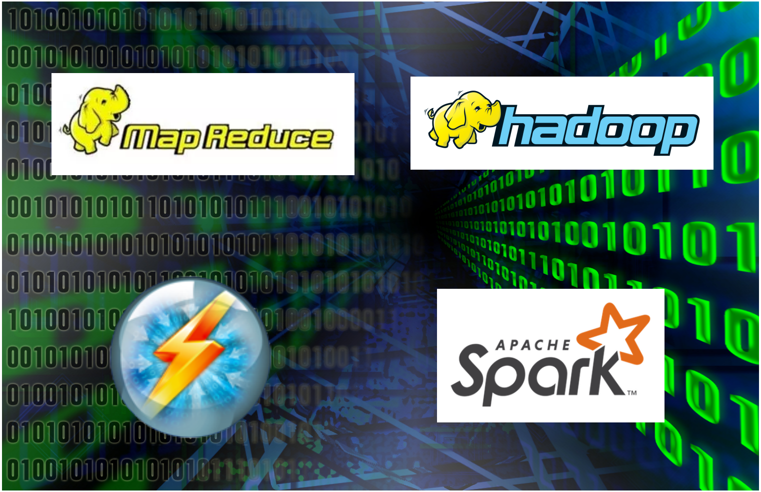
Round Compression for Parallel Matching Algorithms

Krzysztof Onak
IBM T.J. Watson Research Center

Joint work with **Artur Czumaj** (U of Warwick),
Jakub Łącki (Google), **Aleksander Mądry** (MIT),
Slobodan Mitrović (EPFL), and **Piotr Sankowski** (U of Warsaw)

Mandatory "Big Data" Slides



Massive Data Systems

Outline

1. Model of Computation
2. Graph Matchings in MPC
3. Review of Distributed Algorithms
4. Our Algorithm
5. Further Research

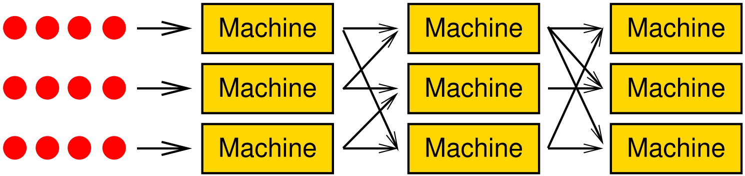
Outline

Model: Massive Parallel Computation (MPC)

Introduced by Karloff, Suri, and Vassilvitskii (2010)

M machines S space per machine

Input: N items

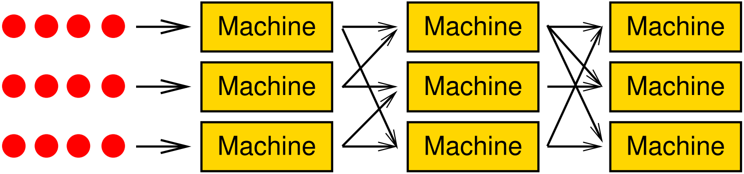


Model: Massive Parallel Computation (MPC)

Introduced by Karloff, Suri, and Vassilvitskii (2010)

M machines S space per machine

Input: N items



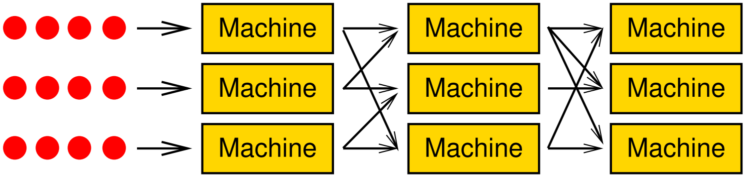
- Initially: each machine receives N/M items

Model: Massive Parallel Computation (MPC)

Introduced by Karloff, Suri, and Vassilvitskii (2010)

M machines S space per machine

Input: N items



- Initially: each machine receives N/M items
- Single round:
  1. Each machine performs computation
  2. Each machine sends and receives at most $O(S)$ data

Model: Massive Parallel Computation (MPC)

- Inspired by MapReduce [Dean, Ghemawat 2004]
- Sleak abstraction that hides details of MapReduce

Model: Massive Parallel Computation (MPC)

- Inspired by MapReduce [Dean, Ghemawat 2004]
- Sleak abstraction that hides details of MapReduce
- Total space considerations:

  [Beame 2009: Problem 27 at sublinear.info]

  [Beame, Koutris, Suciu 2013]

  [Andoni, Nikolov, Onak, Yaroslavtsev 2014]

  - Karloff et al. allow for $N^{1-\epsilon}$ machines with $N^{1-\epsilon}$ space

    $\Rightarrow$ near quadratic total space $N^{2-2\epsilon}$

Model: Massive Parallel Computation (MPC)

- Inspired by MapReduce [Dean, Ghemawat 2004]
- Sleak abstraction that hides details of MapReduce

- Total space considerations:

  [Beame 2009: Problem 27 at sublinear.info]

  [Beame, Koutris, Suciu 2013]

  [Andoni, Nikolov, Onak, Yaroslavtsev 2014]

  - Karloff et al. allow for $N^{1-\epsilon}$ machines with $N^{1-\epsilon}$ space

    $\Rightarrow$ near quadratic total space $N^{2-2\epsilon}$
  - A refined version asks for near-linear total space:

    $$M \times S = \tilde{O}(N).$$

Model: Massive Parallel Computation (MPC)

- Inspired by MapReduce [Dean, Ghemawat 2004]
- Sleak abstraction that hides details of MapReduce
- Total space considerations:

  [Beame 2009: Problem 27 at sublinear.info]

  [Beame, Koutris, Suciu 2013]

  [Andoni, Nikolov, Onak, Yaroslavtsev 2014]

  - Karloff et al. allow for $N^{1-\epsilon}$ machines with $N^{1-\epsilon}$ space

    $\Rightarrow$ near quadratic total space $N^{2-2\epsilon}$
  - A refined version asks for near-linear total space:

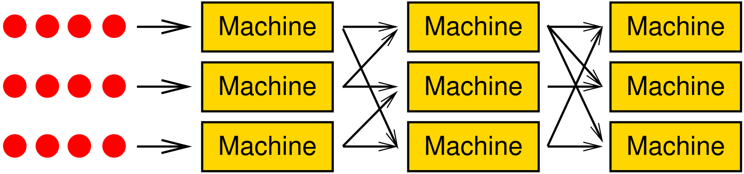
$$M \times S = \tilde{O}(N).$$

- Goals:
  - Small number of rounds
  - Small space per machine
  - Fast local computation

This Talk: MPC for Graphs

Input: edges of an m-edge graph on n vertices

S space per machine $\qquad$ $M = O(m/S)$ machines



This Talk: Matching Algorithms

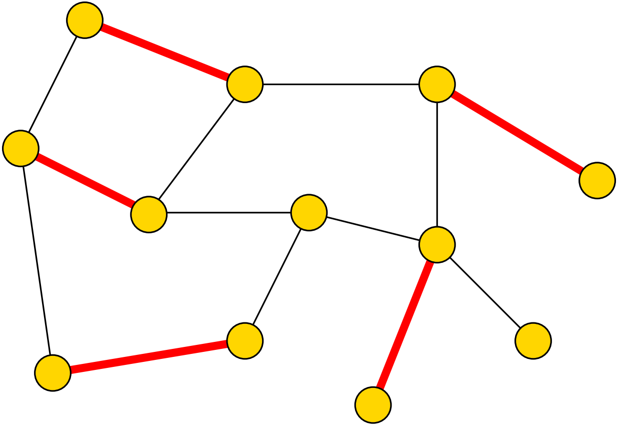
Why study graph matchings?

- Non-trivial appealing packing problem
- Great testbed for many new algorithmic ideas
- Helpful to understand the power of the model
- They have practical applications

Outline

Graph Problems

Maximum Matching: find maximum set of vertex disjoint edges



Large Matchings: Rounds vs. Space

- Space $n^{1+\Omega(1)}$: $(1+\epsilon)$-approximation in $O(1)$ rounds
  [Lattanzi, Moseley, Suri, Vassilvitskii 2011]
  [Ahn, Guha 2015]

Large Matchings: Rounds vs. Space

- Space $n^{1+\Omega(1)}$: $(1+\epsilon)$-approximation in $O(1)$ rounds
  [Lattanzi, Moseley, Suri, Vassilvitskii 2011]
  [Ahn, Guha 2015]
- Space $n^{\Omega(1)}$: 2-approximation in $O(\log n)$ rounds
  - Simulate classic MIS or Maximal Matching PRAM/distributed algorithms:
    - Luby (1986)
    - Alon, Babai, Itai (1986)
    - Israeli, Itai (1986)

Large Matchings: Rounds vs. Space

- Space $n^{1+\Omega(1)}$: $(1+\epsilon)$-approximation in $O(1)$ rounds

  [Lattanzi, Moseley, Suri, Vassilvitskii 2011]

  [Ahn, Guha 2015]

- Space $n^{\Omega(1)}$: 2-approximation in $O(\log n)$ rounds
  - Simulate classic MIS or Maximal Matching PRAM/distributed algorithms:
    - Luby (1986)
    - Alon, Babai, Itai (1986)
    - Israeli, Itai (1986)
  - Tools for simulation:
    - Karloff, Suri, and Vassilvitskii (2010)
    - Goodrich, Sitchinava, and Zhang (2011)

Large Matchings: Rounds vs. Space

- Space $n^{1+\Omega(1)}$: $(1+\epsilon)$-approximation in $O(1)$ rounds
  [Lattanzi, Moseley, Suri, Vassilvitskii 2011]
  [Ahn, Guha 2015]
- Space $n^{\Omega(1)}$: 2-approximation in $O(\log n)$ rounds
  - Simulate classic MIS or Maximal Matching PRAM/distributed algorithms:
    - Luby (1986)
    - Alon, Babai, Itai (1986)
    - Israeli, Itai (1986)
  - Tools for simulation:
    - Karloff, Suri, and Vassilvitskii (2010)
    - Goodrich, Sitchinava, and Zhang (2011)
- For $O(n)$ space, round complexity becomes $\Theta(\log n)$

Our Results

For $O(n)$ space, improve $O(1)$-approximation to $\mathrm{poly}(\log\log n)$ rounds

Our Results

For $O(n)$ space, improve $O(1)$-approximation to poly($\log\log n$) rounds

More detailed version:

For n/α space, $O(1)$-approximation in $O\left((\log\log n)^2 + \log\alpha\right)$ rounds

Our Results

For $O(n)$ space, improve $O(1)$-approximation
to poly$(\log\log n)$ rounds

More detailed version:

For n/α space, $O(1)$-approximation
in $O\left((\log\log n)^2 + \log\alpha\right)$ rounds

(Works well even if space slightly sublinear in n)

Our Results

For $O(n)$ space, improve $O(1)$-approximation to poly($\log \log n$) rounds

More detailed version:

For n/α space, $O(1)$-approximation in $O\left((\log \log n)^2 + \log \alpha\right)$ rounds

(Works well even if space slightly sublinear in n)

Interesting space regime:

- often just enough to fit a solution on a single machine
- gold standard for space in semi-streaming algorithms
- reasonable middle ground

Highlights of Our Approach

Starting point: $O(1)$-approximation distributed algorithm (in the LOCAL model)

- It uses $\Theta(\log n)$ rounds
- So would direct simulation

Highlights of Our Approach

Starting point: $O(1)$-approximation distributed algorithm (in the LOCAL model)

- It uses $\Theta(\log n)$ rounds
- So would direct simulation

Round compression:

Repeatedly compress a superconstant number of rounds of the original algorithm into a constant number of MPC rounds

Highlights of Our Approach

Starting point: $O(1)$-approximation distributed algorithm (in the LOCAL model)

- It uses $\Theta(\log n)$ rounds
- So would direct simulation

Round compression:
Repeatedly compress a superconstant number of rounds of the original algorithm into a constant number of MPC rounds

Vertex sampling:
Previous algorithms used edge sampling

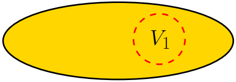
Outline

Vertex Cover [Parnas, Ron 2007]

- Distributed $O(\log n)$-approximation algorithm

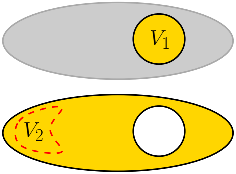
Vertex Cover [Parnas, Ron 2007]

- Distributed $O(\log n)$-approximation algorithm
- Algorithm:
  - Remove vertices of degree at least $n/2 \Rightarrow V_1$



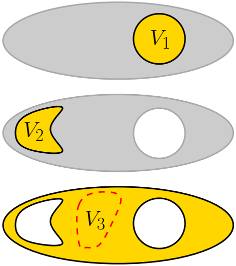
Vertex Cover [Parnas, Ron 2007]

- Distributed $O(\log n)$-approximation algorithm
- Algorithm:
  - Remove vertices of degree at least $n/2 \Rightarrow V_1$
  - Remove vertices of degree at least $n/4 \Rightarrow V_2$



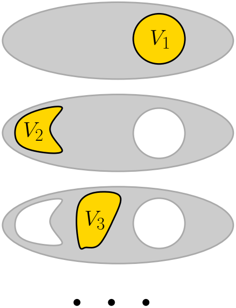
Vertex Cover [Parnas, Ron 2007]

- Distributed $O(\log n)$-approximation algorithm
- Algorithm:
  - Remove vertices of degree at least $n/2 \Rightarrow V_1$
  - Remove vertices of degree at least $n/4 \Rightarrow V_2$
  - Remove vertices of degree at least $n/8 \Rightarrow V_3$



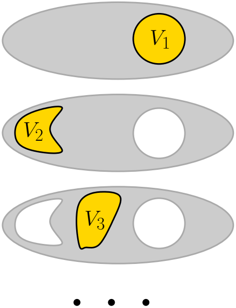
Vertex Cover [Parnas, Ron 2007]

- Distributed $O(\log n)$-approximation algorithm
- Algorithm:
  - Remove vertices of degree at least $n/2 \Rightarrow V_1$
  - Remove vertices of degree at least $n/4 \Rightarrow V_2$
  - Remove vertices of degree at least $n/8 \Rightarrow V_3$
  - ...
  - Remove vertices of degree at least $n/2^i \Rightarrow V_i$



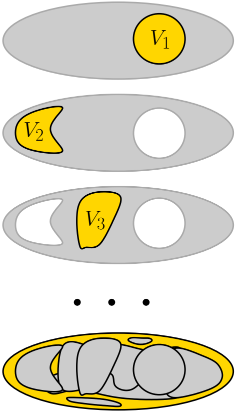
Vertex Cover [Parnas, Ron 2007]

- Distributed $O(\log n)$-approximation algorithm
- Algorithm:
  - Remove vertices of degree at least $n/2 \Rightarrow V_1$
  - Remove vertices of degree at least $n/4 \Rightarrow V_2$
  - Remove vertices of degree at least $n/8 \Rightarrow V_3$
  - ...
  - Remove vertices of degree at least $n/2^i \Rightarrow V_i$
  - ...
  - Remove vertices of degree at least $1 \Rightarrow V_{\log n}$



Vertex Cover [Parnas, Ron 2007]

- Distributed $O(\log n)$-approximation algorithm
- Algorithm:
  - Remove vertices of degree at least $n/2 \Rightarrow V_1$
  - Remove vertices of degree at least $n/4 \Rightarrow V_2$
  - Remove vertices of degree at least $n/8 \Rightarrow V_3$
  - ...
  - Remove vertices of degree at least $n/2^i \Rightarrow V_i$
  - ...
  - Remove vertices of degree at least $1 \Rightarrow V_{\log n}$
- $\mathcal{C} := \bigcup V_i$ is a vertex cover of size $O(\log n) \cdot \mathrm{OPT}$

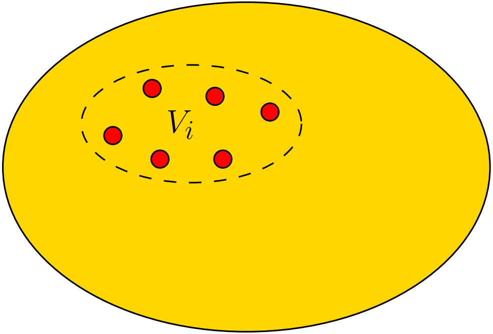


$O(1)$ Approximation [Onak, Rubinfeld 2010]

- Originally developed for dynamic graph algorithms

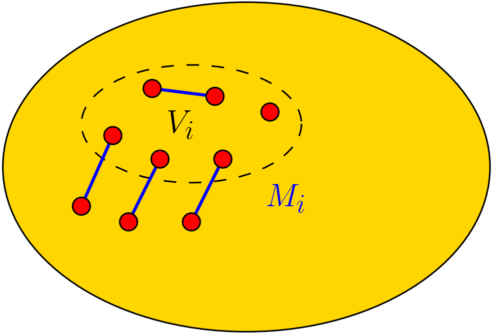
$O(1)$ Approximation [Onak, Rubinfeld 2010]

- Originally developed for dynamic graph algorithms
- Modified Parnas-Ron partition, in phase i:
  - Select V_i as before



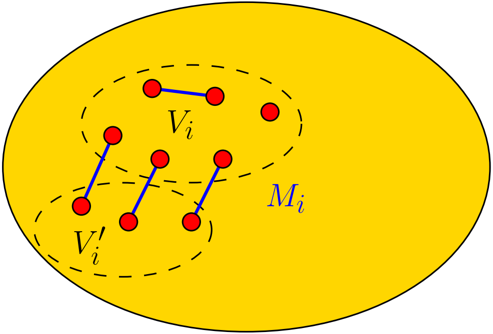
$O(1)$ Approximation [Onak, Rubinfeld 2010]

- Originally developed for dynamic graph algorithms
- Modified Parnas-Ron partition, in phase i:
  - Select V_i as before
  - Find a matching M_i of vertices V_i of size $\Omega(|V_i|)$



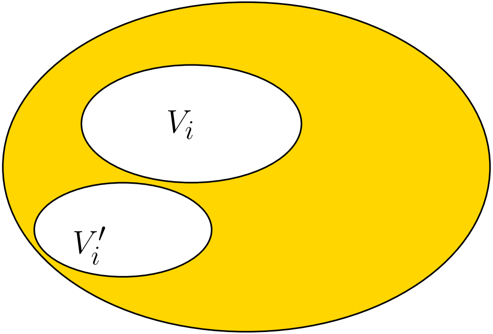
$O(1)$ Approximation [Onak, Rubinfeld 2010]

- Originally developed for dynamic graph algorithms
- Modified Parnas-Ron partition, in phase i:
  - Select V_i as before
  - Find a matching M_i of vertices V_i of size $\Omega(|V_i|)$
  - Let V_i' be the additionally matched vertices



$O(1)$ Approximation [Onak, Rubinfeld 2010]

- Originally developed for dynamic graph algorithms
- Modified Parnas-Ron partition, in phase i:
  - Select V_i as before
  - Find a matching M_i of vertices V_i of size $\Omega(|V_i|)$
  - Let V_i' be the additionally matched vertices
  - Remove from the graph both V_i and V_i'



$O(1)$ Approximation [Onak, Rubinfeld 2010]

- Originally developed for dynamic graph algorithms
- Modified Parnas-Ron partition, in phase i:
  - Select V_i as before
  - Find a matching M_i of vertices V_i of size $\Omega(|V_i|)$
  - Let V_i' be the additionally matched vertices
  - Remove from the graph both V_i and V_i'

- Output: vertex cover $\mathcal{C} = \bigcup_i (V_i \cup V_i')$
  matching $\mathcal{M} = \bigcup_i M_i$

$O(1)$ Approximation [Onak, Rubinfeld 2010]

- Originally developed for dynamic graph algorithms
- Modified Parnas-Ron partition, in phase i:
  - Select V_i as before
  - Find a matching M_i of vertices V_i of size $\Omega(|V_i|)$
  - Let V_i' be the additionally matched vertices
  - Remove from the graph both V_i and V_i'
- Output: vertex cover $\mathcal{C} = \bigcup_i (V_i \cup V_i')$
  matching $\mathcal{M} = \bigcup_i M_i$
- Analysis:
  - $|\mathcal{C}|$ and $|\mathcal{M}|$ are within a constant factor
  - minimum vertex cover size $\geq$ maximum matching size
  - $\mathcal{C}$ and $\mathcal{M}$ are constant-factor approximations

$O(1)$ Approximation [Onak, Rubinfeld 2010]

- Originally developed for dynamic graph algorithms
- Modified Parnas-Ron partition, in phase i:
  - Select V_i as before
  - Find a matching M_i of vertices V_i of size $\Omega(|V_i|)$
  - Let V_i' be the additionally matched vertices
  - Remove from the graph both V_i and V_i'
- Output: vertex cover $\mathcal{C} = \bigcup_i (V_i \cup V_i')$
  matching $\mathcal{M} = \bigcup_i M_i$
- Analysis:
  - $|\mathcal{C}|$ and $|\mathcal{M}|$ are within a constant factor
  - minimum vertex cover size $\geq$ maximum matching size
  - $\mathcal{C}$ and $\mathcal{M}$ are constant-factor approximations
- Goal: efficiently emulate this algorithm in MPC

Outline

Emulating the Peeling Algorithm

- Needed to emulate a phase of the peeling algorithm:
  1. (Approximate) vertex degrees
  2. A random neighbor for each high degree vertex

Emulating the Peeling Algorithm

- Needed to emulate a phase of the peeling algorithm:
  1. (Approximate) vertex degrees
  2. A random neighbor for each high degree vertex

- Then we can:
  1. Find the set of high degree vertices
  2. Find a matching for constant fraction of them

Emulating the Peeling Algorithm

- Needed to emulate a phase of the peeling algorithm:
  1. (Approximate) vertex degrees
  2. A random neighbor for each high degree vertex

- Then we can:
  1. Find the set of high degree vertices
  2. Find a matching for constant fraction of them

- Our plan:
  - Partition vertices at random into a number of groups
  - Ensure that graphs induced by each group fit onto a single machine
  - Ensure that enough neighbors on the machine to satisfy the properties above

Random Vertex Partitioning

- Phase 1:
  - Partition vertices at random into $\sqrt{n}$ groups
  - Each group should have $O((\sqrt{n})^2) = O(n)$ edges
  - In each group, degrees scale down by factor of $\sqrt{n}$
  - Can still find high degree vertices and their random neighbors

Random Vertex Partitioning

- Phase 1:
  - Partition vertices at random into $\sqrt{n}$ groups
  - Each group should have $O((\sqrt{n})^2) = O(n)$ edges
  - In each group, degrees scale down by factor of $\sqrt{n}$
  - Can still find high degree vertices and their random neighbors
- Phase 2:
  - Now maximum degree roughly $n/2$
  - Repeat the same by partitioning vertices into $\sqrt{n}$ groups

...

Random Vertex Partitioning

- Phase 1:
  - Partition vertices at random into $\sqrt{n}$ groups
  - Each group should have $O((\sqrt{n})^2) = O(n)$ edges
  - In each group, degrees scale down by factor of $\sqrt{n}$
  - Can still find high degree vertices and their random neighbors
- Phase 2:
  - Now maximum degree roughly $n/2$
  - Repeat the same by partitioning vertices into $\sqrt{n}$ groups

…

- Can do this for roughly $\log(\sqrt{n}) = \frac{1}{2} \log n$ phases:
  - Stuck when max degree gets roughly $\sqrt{n}$
  - Why? Current high degree vertices see no neighbors

Ideal Algorithm

- Maybe do not repartition each time?
  - No clear reason why this would not work
  - However, vertices are no longer randomly partitioned
  - Not clear how to analyze this

Ideal Algorithm

- Maybe do not repartition each time?
  - No clear reason why this would not work
  - However, vertices are no longer randomly partitioned
  - Not clear how to analyze this

- Where would this take us?
  - We would compress $\frac{1}{2} \log n$ phases into $O(1)$ MPC rounds

Ideal Algorithm

- Maybe do not repartition each time?
  - No clear reason why this would not work
  - However, vertices are no longer randomly partitioned
  - Not clear how to analyze this
- Where would this take us?
  - We would compress $\frac{1}{2}\log n$ phases into $O(1)$ MPC rounds
  - In the end, max degree at most $\sqrt{n}$
  - Can now partition into only $n^{1/4}$ groups and graphs induced by each group will fit onto a single machine
  - Would be able to simulate the next $\frac{1}{4}\log n$ phases

Ideal Algorithm

- Maybe do not repartition each time?
  - No clear reason why this would not work
  - However, vertices are no longer randomly partitioned
  - Not clear how to analyze this
- Where would this take us?
  - We would compress $\frac{1}{2}\log n$ phases into $O(1)$ MPC rounds
  - In the end, max degree at most $\sqrt{n}$
  - Can now partition into only $n^{1/4}$ groups and graphs induced by each group will fit onto a single machine
  - Would be able to simulate the next $\frac{1}{4}\log n$ phases
  - Then $\frac{1}{8}\log n$ phases, $\frac{1}{16}\log n$ phases, ...
  - After $O(\log\log n)$ MPC rounds, we would be done

Actual Solution

- We do not know how to analyze this approach directly
- We tweak the peeling algorithm
- Show independence and near-uniformity of surviving vertices
- We get $O\left((\log \log n)^2\right)$ rounds

Outline

Recent Follow-Up Work

Assadi (arXiv 2017)

- Round compression for the Parnas-Ron algorithm
- $O(\log n)$-approximation to vertex cover in $O(\log \log n)$ MPC round
- Bounding technique of Assadi and Khanna (2017)

Recent Follow-Up Work

Assadi (arXiv 2017)

Assadi, Bateni, Bernstein, Mirrokni, and Stein (arXiv 2017)

- Improve round complexity to $O(\log \log n)$
- Approximation improved to $1 + \epsilon$ [McGregor 2005]
- $(2 + \epsilon)$-approximation for vertex cover
- No round compression, but still vertex sampling
- Apply techniques developed for dynamic matching [Bernstein, Stein 2015]

Recent Follow-Up Work

Assadi (arXiv 2017)

Assadi, Bateni, Bernstein, Mirrokni, and Stein (arXiv 2017)

Ghaffari, Gouleakis, Konrad, Mitrović, Rubinfeld (PODC 2018)

- Improve round complexity to $O(\log \log n)$
- Simulate a parallel fractional algorithm
- Explore connections to congested clique model
- Also $O(\log \log n)$-round algorithm for Maximal Independent Set

Follow-Up Questions

- Round compression for other problems?

Follow-Up Questions

- Round compression for other problems?

- Any good reason why $\log\log n$ seems to be a barrier?
  - MPC hard to prove unconditional lower bounds
  - Show reductions to/from other problems?
  - Limitations of natural sampling techniques?

Follow-Up Questions

- Round compression for other problems?

- Any good reason why $\log \log n$ seems to be a barrier?

    - MPC hard to prove unconditional lower bounds
    - Show reductions to/from other problems?
    - Limitations of natural sampling techniques?

- Show that a simple very greedy algorithm just works?

Questions?

Full version: `https://arxiv.org/abs/1707.03478`