Approximating vertex covers in anonymous networks

Jukka Suomela
Helsinki Institute for Information Technology HIIT
University of Helsinki, Finland
Joint work with Matti Åstrand
2 March 2010
Vertex cover problem

• **Vertex cover** for a graph $G$:
  • Subset $C$ of nodes that “covers” all edges: each edge incident to at least one node in $C$

• **Minimum vertex cover**:
  • Vertex cover with the smallest number of nodes

• **Minimum-weight vertex cover**:
  • Vertex cover with the smallest total weight
Vertex cover problem

- Classical NP-hard optimisation problem: given a graph $G$, find a minimum vertex cover

- Simple 2-approximation algorithm:
  - Find a maximal matching, output all endpoints
  - At most 2 times as large as minimum VC

- No polynomial-time algorithm with approximation factor $1.9999$ known
Research question

• Exactly how well can we approximate vertex cover in a **distributed setting**?

• **Focus:**
  • Fast, synchronous, **deterministic** distributed algorithms
  • Weakest possible models
Distributed algorithms

- Communication graph $G$
- Node = computer
  - e.g., Turing machine, finite state machine
- Edge = communication link
  - computers can exchange messages
Distributed algorithms

• All nodes are identical, run the same algorithm

• We can choose the algorithm

• An adversary chooses the structure of \( G \)

• Our algorithm must produce a valid vertex cover in any graph \( G \)
Synchronous distributed algorithms

1. Each node reads its own **local input**:
   - node identifier
   - if we assume unique node IDs
   - node weight
   - if we study weighted graphs
Synchronous distributed algorithms

1. Each node reads its own local input
2. Repeat synchronous communication rounds
   ...

Diagram showing a network of nodes connected by arrows.
Synchronous distributed algorithms

1. Each node reads its own local input
2. Repeat synchronous communication rounds until all nodes have announced their local outputs
   - $1 = \text{in vertex cover}$
Synchronous distributed algorithms

- Communication round: each node
  1. sends a message to each neighbour
Synchronous distributed algorithms

- Communication round: each node
  1. sends a message to each neighbour
     (message propagation...)
Synchronous distributed algorithms

- Communication round: each node
  1. sends a message to each neighbour
  2. receives a message from each neighbour
Synchronous distributed algorithms

- Communication round: each node
  1. sends a message to each neighbour
  2. receives a message from each neighbour
  3. updates its own state
Synchronous distributed algorithms

- Communication round: each node
  1. sends a message to each neighbour
  2. receives a message from each neighbour
  3. updates its own state
  4. possibly stops and announces its output
Synchronous distributed algorithms

- Communication rounds are repeated until all nodes have stopped and announced their outputs.
- Running time = \textit{number of rounds}.
- Worst-case analysis.
Distributed algorithms: three models

1. Unique identifiers
2. Port-numbering model
3. Broadcast model
Model 1: Unique identifiers

- Node identifiers are a permutation of 1, 2, ..., n
- Permutation chosen by adversary
Model 2: Port-numbering model

- No unique identifiers
- A node of degree $d$ can refer to its neighbours by integers 1, 2, ..., $d$
- Port-numbering chosen by adversary
Model 3: Broadcast model

- No identifiers, no port numbers
- A node has to send the same message to each neighbour
- A node does not know which message was received from which neighbour
Distributed algorithms: three models

1. Unique identifiers

2. Port-numbering model
   - Vector with deg(v) outgoing messages
   - Vector with deg(v) incoming messages

3. Broadcast model
   - Only one outgoing message
   - Multiset with deg(v) incoming messages
Deterministic distributed algorithms for vertex cover: approximation ratios

<table>
<thead>
<tr>
<th></th>
<th>lower</th>
<th>upper</th>
<th>lower</th>
<th>upper</th>
<th>lower</th>
<th>upper</th>
</tr>
</thead>
<tbody>
<tr>
<td>(O(n))</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(f(\Delta) + \text{polylog}(n))</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(f(\Delta) + O(\log^* n))</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(f(\Delta))</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Broadcast model | Port numbering | Unique identifiers

**log\(^*\) = iterated logarithm \(
\approx\text{inverse of power tower}\)**
Deterministic distributed algorithms for vertex cover: approximation ratios

<table>
<thead>
<tr>
<th></th>
<th>lower</th>
<th>upper</th>
<th>lower</th>
<th>upper</th>
<th>lower</th>
<th>upper</th>
</tr>
</thead>
<tbody>
<tr>
<td>$O(n)$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>$f(\Delta) + \text{polylog}(n)$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$f(\Delta) + O(\log^* n)$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$f(\Delta)$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Trivial</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>algorithm</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Broadcast model</th>
<th>Port numbering</th>
<th>Unique identifiers</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Trivial</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>algorithm</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Deterministic distributed algorithms for vertex cover: approximation ratios

<table>
<thead>
<tr>
<th></th>
<th>lower</th>
<th>upper</th>
<th>lower</th>
<th>upper</th>
<th>lower</th>
<th>upper</th>
</tr>
</thead>
<tbody>
<tr>
<td>$O(n)$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>$f(\Delta) + \text{polylog}(n)$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>$f(\Delta) + O(\log^* n)$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>$f(\Delta)$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Maximal matching**
(Panconesi & Rizzi 2001)

- Broadcast model
- Port numbering
- Unique identifiers
Deterministic distributed algorithms for vertex cover: approximation ratios

<table>
<thead>
<tr>
<th></th>
<th>lower</th>
<th>upper</th>
<th>lower</th>
<th>upper</th>
<th>lower</th>
<th>upper</th>
</tr>
</thead>
<tbody>
<tr>
<td>$O(n)$</td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>$f(\Delta) + \text{polylog}(n)$</td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>$f(\Delta) + O(\log^* n)$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>$f(\Delta)$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Near-maximal edge packing (Khuller et al. 1994)
- Broadcast model
- Port numbering
- Unique identifiers
## Deterministic distributed algorithms for vertex cover: approximation ratios

<table>
<thead>
<tr>
<th></th>
<th>lower</th>
<th>upper</th>
<th>lower</th>
<th>upper</th>
<th>lower</th>
<th>upper</th>
</tr>
</thead>
<tbody>
<tr>
<td>$O(n)$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>$f(\Delta) + \text{polylog}(n)$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>$f(\Delta) + O(\log^* n)$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2 + \epsilon</td>
<td>2</td>
</tr>
<tr>
<td>$f(\Delta)$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2 + \epsilon</td>
<td>2 + \epsilon</td>
</tr>
</tbody>
</table>

- **Deterministic LP rounding** (Kuhn et al. 2006)

<table>
<thead>
<tr>
<th></th>
<th>Broadcast model</th>
<th>Port numbering</th>
<th>Unique identifiers</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Deterministic distributed algorithms for vertex cover: approximation ratios

<table>
<thead>
<tr>
<th></th>
<th>lower</th>
<th>upper</th>
<th>lower</th>
<th>upper</th>
<th>lower</th>
<th>upper</th>
</tr>
</thead>
<tbody>
<tr>
<td>(O(n))</td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>(f(\Delta) + \text{polylog}(n))</td>
<td></td>
<td></td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(f(\Delta) + O(\log^* n))</td>
<td></td>
<td></td>
<td>2 + (\varepsilon)</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(f(\Delta))</td>
<td>2</td>
<td>2</td>
<td>2 + (\varepsilon)</td>
<td>2</td>
<td>2 + (\varepsilon)</td>
<td></td>
</tr>
</tbody>
</table>

- Czygrinow et al. 2008
- Lenzen & Wattenhofer 2008

- Broadcast model
- Port numbering
- Unique identifiers
Deterministic distributed algorithms for vertex cover: approximation ratios

<table>
<thead>
<tr>
<th></th>
<th>lower</th>
<th>upper</th>
<th>lower</th>
<th>upper</th>
<th>lower</th>
<th>upper</th>
</tr>
</thead>
<tbody>
<tr>
<td>$O(n)$</td>
<td>2</td>
<td></td>
<td>2</td>
<td>2</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>$f(\Delta) + \text{polylog}(n)$</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>$f(\Delta) + O(\log^* n)$</td>
<td>2</td>
<td>2</td>
<td>2 + $\varepsilon$</td>
<td>2</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>$f(\Delta)$</td>
<td>2</td>
<td>2</td>
<td>2 + $\varepsilon$</td>
<td>2</td>
<td>2 + $\varepsilon$</td>
<td></td>
</tr>
</tbody>
</table>

Trivial (cycles)

<table>
<thead>
<tr>
<th></th>
<th>Broadcast model</th>
<th>Port numbering</th>
<th>Unique identifiers</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Deterministic distributed algorithms for vertex cover: approximation ratios

<table>
<thead>
<tr>
<th></th>
<th>lower</th>
<th>upper</th>
<th>lower</th>
<th>upper</th>
<th>lower</th>
<th>upper</th>
</tr>
</thead>
<tbody>
<tr>
<td>$O(n)$</td>
<td>2</td>
<td></td>
<td>2</td>
<td>2</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>$f(\Delta) + \text{polylog}(n)$</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>$f(\Delta) + O(\log^* n)$</td>
<td>2</td>
<td>2</td>
<td>2 + $\varepsilon$</td>
<td>2</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>$f(\Delta)$</td>
<td>2</td>
<td>2</td>
<td>2 + $\varepsilon$</td>
<td>2</td>
<td>2 + $\varepsilon$</td>
<td>2</td>
</tr>
<tr>
<td>Broadcast model</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Unique identifiers</td>
</tr>
</tbody>
</table>
### Deterministic distributed algorithms for vertex cover: approximation ratios

<table>
<thead>
<tr>
<th></th>
<th>lower</th>
<th>upper</th>
<th>lower</th>
<th>upper</th>
<th>lower</th>
<th>upper</th>
</tr>
</thead>
<tbody>
<tr>
<td>( O(n) )</td>
<td>2</td>
<td></td>
<td>2</td>
<td>2</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>( f(\Delta) + \text{polylog}(n) )</td>
<td>2</td>
<td></td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( f(\Delta) + O(\text{log}* n) )</td>
<td>2</td>
<td></td>
<td>2</td>
<td>(2 + \varepsilon)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( f(\Delta) )</td>
<td>2</td>
<td></td>
<td>2</td>
<td>(2 + \varepsilon)</td>
<td>2</td>
<td>(2 + \varepsilon)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Broadcast model</th>
<th>Port numbering</th>
<th>Unique identifiers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Could we have 2?</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Deterministic distributed algorithms for vertex cover: approximation ratios

<table>
<thead>
<tr>
<th></th>
<th>lower</th>
<th>upper</th>
<th>lower</th>
<th>upper</th>
<th>lower</th>
<th>upper</th>
</tr>
</thead>
<tbody>
<tr>
<td>$O(n)$</td>
<td>2</td>
<td>?</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>$f(\Delta) + \text{polylog}(n)$</td>
<td>2</td>
<td>?</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$f(\Delta) + O(\log^* n)$</td>
<td>2</td>
<td>?</td>
<td>2</td>
<td>$2 + \varepsilon$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$f(\Delta)$</td>
<td>2</td>
<td>?</td>
<td>2</td>
<td>$2 + \varepsilon$</td>
<td>2</td>
<td>$2 + \varepsilon$</td>
</tr>
</tbody>
</table>

- **Could we have 2?**
- **Anything here?**
- **Broadcast model**
- **Port numbering**
- **Unique identifiers**
Deterministic distributed algorithms for vertex cover: approximation ratios

<table>
<thead>
<tr>
<th></th>
<th>lower</th>
<th>upper</th>
<th>lower</th>
<th>upper</th>
<th>lower</th>
<th>upper</th>
</tr>
</thead>
<tbody>
<tr>
<td>$O(n)$</td>
<td>2</td>
<td>?</td>
<td>2</td>
<td>2</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>$f(\Delta) + \text{polylog}(n)$</td>
<td>2</td>
<td>?</td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$f(\Delta) + O(\log^* n)$</td>
<td>2</td>
<td>?</td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$f(\Delta)$</td>
<td>2</td>
<td>?</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

DISC 2009

- Broadcast model
- Port numbering
- Unique identifiers
Deterministic distributed algorithms for vertex cover: approximation ratios

<table>
<thead>
<tr>
<th></th>
<th>lower</th>
<th>upper</th>
<th>lower</th>
<th>upper</th>
<th>lower</th>
<th>upper</th>
</tr>
</thead>
<tbody>
<tr>
<td>$O(n)$</td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$f(\Delta) + \text{polylog}(n)$</td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$f(\Delta) + O(\log^* n)$</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$f(\Delta)$</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

- **Broadcast model**
- **Port numbering**
- **Unique identifiers**

Latest results

DISC 2009
Deterministic distributed algorithms for vertex cover: approximation ratios

<table>
<thead>
<tr>
<th></th>
<th>lower</th>
<th>upper</th>
<th>lower</th>
<th>upper</th>
<th>lower</th>
<th>upper</th>
</tr>
</thead>
<tbody>
<tr>
<td>( O(n) )</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>( f(\Delta) + \text{polylog}(n) )</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>( f(\Delta) + O(\log^* n) )</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>( f(\Delta) )</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Broadcast model</td>
<td>Port numbering</td>
<td>Unique identifiers</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**Deterministic distributed algorithms for vertex cover: approximation ratios**

<table>
<thead>
<tr>
<th></th>
<th>lower</th>
<th>upper</th>
<th>lower</th>
<th>upper</th>
<th>lower</th>
<th>upper</th>
</tr>
</thead>
<tbody>
<tr>
<td>$O(n)$</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>$f(\Delta) + \text{polylog}(n)$</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$f(\Delta) + O(\log^* n)$</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$f(\Delta)$</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Broadcast model</th>
<th>Port numbering</th>
<th>Unique identifiers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Let’s study this case first...</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Vertex cover in the port-numbering model

• Convenient to study a more general problem: minimum-weight vertex cover
  • More general problems are sometimes easier to solve!

Notation:
\( w(v) = \text{weight of } v \)
Edge packings and vertex covers

• **Edge packing**: weight $y(e) \geq 0$ for each edge $e$

  • Packing constraint: for each node $v$, the total weight of edges incident to $v$ is at most $w(v)$
Edge packings and vertex covers

- **Edge packing**: weight $y(e) \geq 0$ for each edge $e$
  - Packing constraint: for each node $v$, the total weight of edges incident to $v$ is at most $w(v)$

\[3 + 0 + 4 + 0 + 0 + 2 \leq 9\]
Edge packings and vertex covers

- In linear programming, these are dual problems:
  - minimum-weight (fractional) vertex cover
  - maximum-weight edge packing
Edge packings and vertex covers

- **Saturated node** $v$: the total weight on edges incident to $v$ is equal to $w(v)$
Edge packings and vertex covers

- **Saturated edge** $e$:
  at least one endpoint of $e$ is saturated
  $\iff$ edge weight $y(e)$ can’t be increased

$2 + \epsilon$ would violate a packing constraint
Edge packings and vertex covers

- **Maximal edge packing**: all edges saturated
  \[\iff\] none of the edge weights \(y(e)\) can be increased
  \[\iff\] saturated nodes form a vertex cover
Edge packings and vertex covers

- Minimum-weight vertex cover $C^*$ difficult to find:
  - Centralised setting: NP-hard
  - Distributed setting: integer problem, symmetry-breaking issues

- Maximal edge packing $y$ easy to find:
  - Centralised setting: trivial greedy algorithm
  - Distributed setting: linear problem, no symmetry-breaking issues (?)
Edge packings and vertex covers

- Minimum-weight vertex cover $C^*$ difficult to find
- Maximal edge packing $y$ easy to find?
- Saturated nodes $C(y)$ in $y$: 2-approximation of $C^*$
  - $w(C(y)) \leq 2w(C^*)$
  - Notation: $w(C) =$ total weight of the nodes $v \in C$
  - Proof: LP-duality, relaxed complementary slackness
Edge packings and vertex covers

- Minimum-weight vertex cover $C^*$ difficult to find
- Maximal edge packing $y$ easy to find?
- Saturated nodes $C(y)$ in $y$: 2-approximation of $C^*$
  - $w(C(y)) \leq 2w(C^*)$
  - Constant 2: $C(y)$ covers edges at most twice, $C^*$ at least once
  - Immediate generalisation to hypergraphs

\[
    w(C(y)) = \sum_{v \in C(y)} y[v] = \sum_{e \in E} y(e) |e \cap C(y)| \leq 2 \sum_{e \in E} y(e) |e \cap C^*| = 2 \sum_{v \in C^*} y[v] \leq 2w(C^*)
\]
Finding a maximal edge packing

- Basic idea from Khuller et al. (1994) and Papadimitriou and Yannakakis (1993)
Finding a maximal edge packing: basic idea

- \( y[v] \) = total weight of edges incident to node \( v \)
- **Residual capacity** of node \( v \): \( r(v) = w(v) - y[v] \)
- Saturated node: \( r(v) = 0 \)
Finding a maximal edge packing: basic idea

Start with a trivial edge packing \( y(e) = 0 \)
Finding a maximal edge packing: basic idea

Each node \( v \) offers \( r(v)/\text{deg}(v) \) units to each incident edge.
Finding a maximal edge packing: basic idea

Each edge **accepts** the smallest of the 2 offers it received

Increase $y(e)$ by this amount

- Safe, can’t violate packing constraints
Finding a maximal edge packing: basic idea

Update residuals...
Finding a maximal edge packing: basic idea

Update residuals, discard saturated nodes and edges...
Finding a maximal edge packing: basic idea

Update residuals, discard saturated nodes and edges, repeat...

Offers...
Finding a maximal edge packing: basic idea

Update residuals, discard saturated nodes and edges, repeat...

Offers...

Increase weights...
Finding a maximal edge packing: basic idea

Update residuals, discard saturated nodes and edges, repeat...

Offers...
Increase weights...

Update residuals...
Finding a maximal edge packing: basic idea

Update residuals, discard saturated nodes and edges, repeat…

Offers…

Increase weights…

Update residuals and graph, etc.
Finding a maximal edge packing: basic idea

This is a simple deterministic distributed algorithm.

We are making some progress towards finding a maximal edge packing — but...
Finding a maximal edge packing: basic idea

This is a simple deterministic distributed algorithm.

We are making some progress towards finding a maximal edge packing — but this is too slow!
Finding a maximal edge packing: colouring trick

- Offer is a local minimum:
  - Node will be saturated
  - And all edges incident to it will be saturated as well

Residual capacity was 8, will be 0
Finding a maximal edge packing: colouring trick

- Offer is a local minimum:
  - Node will be saturated
- Otherwise there is a neighbour with a different offer:
  - Interpret the offer sequences as colours
  - Nodes $u$ and $v$ have different colours: \{u, v\} is multicoloured
Finding a maximal edge packing: colouring trick

- Progress guaranteed:
  - On each iteration, for each node, at least one incident edge becomes saturated or multicoloured
  - Such edges are be discarded; maximum degree $\Delta$ decreases by at least one
  - Hence in $\Delta$ rounds all edges are saturated or multicoloured
Finding a maximal edge packing: colouring trick

- In $\Delta$ rounds all edges are **saturated** or **multicoloured**
  - Saturated edges are good — we’re trying to construct a maximal edge packing
  - Why are the multicoloured edges useful?
Finding a maximal edge packing: colouring trick

• In $\Delta$ rounds all edges are **saturated** or **multicoloured**
  • Saturated edges are good — we’re trying to construct a maximal edge packing
  • Why are the multicoloured edges useful?
  • Let’s focus on unsaturated nodes and edges
Finding a maximal edge packing: colouring trick

- Colours are sequences of $\Delta$ rational numbers
  - Assume that node weights are integers 1, 2, ..., $W$
  - Then colours are rationals of the form $q/ (\Delta!)^\Delta$ with $q \in \{1, 2, ..., W\}$

![Graph Diagram]

(2, 2/3, 1/6, 1/12)

(2, 2/3, 1/6, 1/24)
Finding a maximal edge packing: colouring trick

- Colours are sequences of $\Delta$ rational numbers
  - Assume that node weights are integers $1, 2, \ldots, W$
  - Then colours are rationals of the form $q/(\Delta!)^\Delta$ with $q \in \{1, 2, \ldots, W\}$
  - $k = (W(\Delta!)^\Delta)^\Delta$ possible colours, replace with integers $1, 2, \ldots, k$
Finding a maximal edge packing: colouring trick

- Colours are sequences of $\Delta$ rational numbers
  - Assume that node weights are integers 1, 2, ..., $W$
  - Then colours are rationals of the form $q/(\Delta!)^\Delta$ with $q \in \{1, 2, ..., W\}$
  - $k = (W(\Delta!)^\Delta)^\Delta$ possible colours, replace with integers 1, 2, ..., $k$

Looks ugly, but don’t worry, in the end we will take log* of $k$
Finding a maximal edge packing: colouring trick

- We have a proper $k$-colouring of the unsaturated subgraph
- Orient from lower to higher colour (acyclic directed graph)
Finding a maximal edge packing: colouring trick

- We have a proper $k$-colouring of the unsaturated subgraph
- Orient from lower to higher colour (acyclic directed graph)
- Partition in $\Delta$ forests
  - Each node assigns its outgoing edges to different forests
Finding a maximal edge packing: colouring trick

- For each forest in parallel...
Finding a maximal edge packing: colouring trick

- For each forest in parallel:
  - Use Cole-Vishkin (1986) style colour reduction algorithm
  - Given a $k$-colouring, finds a 3-colouring in time $O(\log^* k)$
  - Bit manipulation trick: each step replaces a $k$-colouring with an $O(\log k)$-colouring
Finding a maximal edge packing: colouring trick

- For each forest and each colour $j = 1, 2, 3$ in sequence:
  - Saturate all outgoing edges of colour-$j$ nodes
  - Node-disjoint stars, easy to saturate in parallel
- In $O(\Delta)$ rounds we have saturated all edges
Finding a maximal edge packing: summary

• Total running time:
  • All edges are saturated or multicoloured: $O(\Delta)$
  • Multicoloured forests are 3-coloured: $O(\log^* k)$
  • 3-coloured forests are saturated: $O(\Delta)$
• $O(\Delta + \log^* k) = O(\Delta + \log^* W)$
  • $k$ is huge, but $\log^*$ grows slowly
Finding a maximal edge packing: summary

- Maximal edge packing and 2-approximation of vertex cover in time $O(\Delta + \log^* W)$
  - $W = \text{maximum node weight}$
- Unweighted graphs: running time simply $O(\Delta)$, independent of $n$
- Everything can be implemented in the port-numbering model
Vertex cover algorithms

• 2-approximation of vertex cover in time $O(\Delta)$ in the **port-numbering model**
  • Insight: consider a more general problem, minimum-*weight* vertex cover

• 2-approximation of vertex cover in time $\text{poly}(\Delta)$ in the **broadcast model**?
  • Insight: consider a more general problem, minimum-weight *set* cover!
Set cover algorithm

- Set covers in a distributed setting:
  - bipartite graph, “sets” and “elements”
- Degree bounds:
  - element frequency at most $f$
  - set size at most $k$
- Vertex cover:
  - edge $\approx$ element ($f = 2$)
  - node $\approx$ set ($k = \Delta$)
Set cover algorithm

- Similar techniques:
  - Find a **maximal fractional packing**
  - Generalisation of maximal edge packings
  - **Saturated sets**: $f$-approximation of minimum-weight set cover
Set cover algorithm

- Similar techniques:
  - Find a maximal fractional packing
  - “Greedy but safe” offer/accept rounds
  - Progress guaranteed: something is always saturated or multicoloured
Set cover algorithm

• Dissimilar techniques:
  • Repeated iterations of saturation + colouring phases
  • We don’t try to find a proper 3-colouring but a weak 3-colouring
    • Easier in the broadcast model, enough to make some progress
  • Lots of technicalities…
Set cover algorithm

- Maximal fractional packing
  in $O(f^2 k^2 + fk \log^* W)$ rounds, broadcast model
Set cover algorithm: application

• Use the set cover algorithm to find a vertex cover
  • In vertex cover instances, nodes have local state but edges are stateless
  • In set cover instances, both sets and elements have local state
  • Simulation possible, trick: pass around the full history of broadcasts, re-compute the states
  • Larger messages, but the same number of rounds
Set cover algorithm: application

- Use the set cover algorithm to find a vertex cover
- 2-approximation of unweighted vertex cover in $O(\Delta^2)$ rounds, broadcast model
Conclusions

• 2-approximation algorithms for vertex cover:
  • Time $O(\Delta)$, port-numbering model
  • Time $O(\Delta^2)$, broadcast model

• Research questions:
  • Can you do it faster, in any model?
  • What else can be solved in the broadcast model?
Conclusions

• 2-approximation algorithms for vertex cover:
  • Time $O(\Delta)$, *port-numbering model*
  • Time $O(\Delta^2)$, *broadcast model*

• Take-home message:
  • Sometimes more general problems are easier to solve!