

# Maintaining a large matching and a small vertex cover

Krzysztof Onak (MIT)

Ronitt Rubinfeld (Tel Aviv U. and MIT)

# Setting

- ~~Property testing?~~
- Dynamic graph algorithm
  - Updates: insert or delete edge
- Quantity to approximate:
  - Min vertex cover
  - Max matching – gives factor 2 approx to VC

# How to maintain?

- Exact maximum matching:  $n^{1.495}$  update [Sankowski]
- Is  $\text{polylog}(n)$  update time possible?
  - If  $o(\sqrt{n})$  then improve maximum matching algorithm of [Micali Vazirani 80]
- What about  $\text{polylog}(n)$  update time for *approximation*?
- Main result: Data structure for max matching and vertex cover
  - randomized
  - constant approximation
  - $\text{polylog}(n)$  amortized updated time

# Why this talk here?

- Use techniques from [Parnas Ron]
  - who show an interesting connection between distributed algorithms and sublinear time approximation algorithms

# Idea of Parnas Ron algorithm

- Parnas-Ron Vertex Partition algorithm:
  - $i \leftarrow 1$
  - While edges remain:
    - Remove vertices of degree  $> d_{\max} / 4^{i-1}$  and adjacent edges
    - Increment  $i$
  - Output *all* removed vertices as VC
- Yields  $O(\log d_{\max})$  approximation in  $O(\log d_{\max})$  phases
- For constant degree graphs, yields constant time approx algorithm

# Idea of our algorithm

- **Starting point of new data structure:**
  - Simulate Parnas-Ron partition with some laziness
  - Need to keep track of approximate number of preceding vertices
  - Gives  $O(\log n)$  approximation for VC (and max matching)
- **Better idea:**
  - Also remove a random large matching at each phase
  - Gives  $O(1)$  approximation for VC and max matching

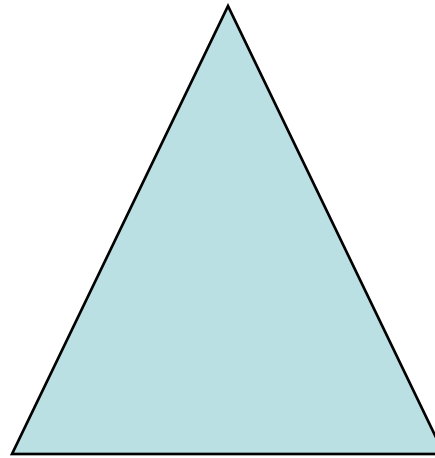
# Sublinear algorithms conquer the world?

Sublinear time algorithms

Parnas Ron



?



Distributed algorithms



Dynamic algorithms

here