Distributed computation and sublinear time algorithms

Lecture 13
Ronitt Rubinfeld
Distributed computation vs. sublinear time algorithms

Connection between sublinear time algorithms and local distributed algorithms (constant number of communication rounds) on sparse graphs

[Parnas Ron]
Sparse Graphs

• Representation:
  – Max degree $d$
  – Adjacency list format

• Problems of interest:
  – Vertex cover
  – Matching
  – Dominating set
  – Set cover (on sparse set systems)
Vertex Cover (VC)

Def. $V' \subseteq V$ is a **vertex cover** if $\forall \ e=(u,v) \in E$, either $u \in V'$ or $v \in V'$

- In a degree $\leq d$ graph:
  - Vizing’s theorem says every graph is edge-colorable with $\leq d+1$ colors
  - Picking color with most edges gives matching of size $\geq m/(d+1)$
  - So $VC \geq m/(d+1)$ (since for each matching edge must put at least one of endpoints in VC)
Approximation for VC

• Multiplicative?
  – VC of graph with no edges vs. graph with 1 edge

• Additive?
  – Need to allow some multiplicative error: Computationally hard to approximate to better than 1.36 factor

• Combination?
  – Def. \( y' \) is \((\alpha, \epsilon)\)-estimate of \( y \) if
    \[ y \leq y' \leq \alpha \cdot y + \epsilon \cdot n \]
    Good for minimization problems
Distributed Algorithms (simple version)

• Network
  – Processors
  – Links
  – (assume maximum degree is known to all)

• Communication round
  – Each node sends message to each neighbor

• Vertex Cover Problem:
  – Network graph = input graph
  – After k rounds, each node knows if it is in VC
Connection for Vertex Cover

Thm [Parnas Ron]: $t$-round distributed algorithm for vertex cover yields $d_{\text{max}}^{O(t)}$ sequential query approximation algorithm for vertex cover.

Reduction idea:

• Sample vertices of graph
• For each sampled vertex $v$, simulate distributed algorithm to see if $v$ is in VC
• Output $(\text{fraction in VC}) \cdot n$
A “local algorithm” for vertex cover

• Vertex Cover algorithm: \((\text{max degree } d)\)
  – \(i \leftarrow 1\)
  – While edges remain:
    • Remove vertices of degree > \(d / 2^i\) and adjacent edges
    • Update degrees of remaining nodes
    • Increment \(i\)
  – Output \textit{all removed vertices} as VC

• How many rounds?
  \(\log d\)
Example run of Parnas-Ron

Remove vertices of degree $\geq 8$
Remove vertices of degree $\geq 4$
Why a Vertex Cover?

• Vertex Cover algorithm:
  – $i \leftarrow 1$
  – While edges remain:
    • Remove vertices of degree $> d_{\text{max}} / 2^i$ and adjacent edges
    • Update degrees of remaining nodes
    • Increment $i$
  – Output *all removed vertices* as VC

• No edges remain at end – all removed along with adjacent vertex
Why a good approximation?

Let $VC_G =$ size of min vertex cover of $G$

**Theorem:** $VC_G \leq |C| \leq (2\log d + 1) \ VC_G$

**Proof:**

$VC_G \leq |C|$:  
- Algorithm removes edges only if at least one endpoint placed in cover
- All edges gone at end
Why a good approximation? (cont.)

Theorem: $\text{VC}_G \leq |C| \leq (2\log d + 1) \text{VC}_G$

Proof:

$|C| \leq (2\log d + 1) \text{VC}_G$:

– See board
Approximation algorithm for vertex cover

On input $G$, with max degree $d$, there is an $O(d^{O(\log d) / \epsilon^2})$ (sequential) time algorithm which outputs $\beta$ such that

$$VC_G \leq \beta \leq (2\log d + 1) VC_G + \epsilon n$$

- Proof: $O(\log d)$ round distributed algorithm + Parnas-Ron theorem
- No dependence on $n$
- Can get $O(1)$ multiplicative estimates and faster runtimes in terms of $d, \epsilon$
Constant time approximation algorithms for sparse graphs

• Paradigm + local algorithms yield constant time approximation algorithms for bounded degree graph problems  [Parnas Ron] [Marko Ron] [Nguyen Onak] [Yoshida Yamamoto Ito] [Hasidim Kelner Nguyen Onak]…
  – Applies to vertex cover, maximum matching, dominating set, sparse set cover,…
  – Various algorithmic ideas to create local algorithms
    • Next lecture: locally simulating greedy algorithms
  – For some problems, runtimes polynomial in $d$ and $\varepsilon$