236358
Distributed Graph Algorithms

Spring 2017
Class by Keren Censor-Hillel
Vertex Cover

• \( U \subseteq V \) such that for every edge \( e=(u,v) \in E \)
  – \( u \) or \( v \) is in \( U \)

• **Minimum vertex cover (MVC):**
  – Find \( U \) of smallest size
  – **Weighted**: \( G=(V,w,E) \) with weights \( w:V \rightarrow W \),
    Find \( U \) of smallest **weight** \( w(U)=\Sigma_{v \in U} w(v) \)
Minimum Vertex Cover

• NP-hard

• Approximation algorithms:
  – Find a vertex cover \( U: w(U) \leq \alpha w(U_{opt}) \)
  – \( U_{opt} \) is the minimum vertex cover

• 2-approximation for unweighted:
  endpoints of greedy maximal matching

• Weighted: UGC \( \rightarrow \) no polynomial \((2-\varepsilon)\)-approximation
Local Ratio for MVC

• Consider a vertex cover $U$, with weight $w(U)$

• Instead of going over all nodes in $U$ and buying their weights $w(U) = \sum_{v \in U} w(v)$, consider:

• Illustration
Local Ratio for MVC

• Going over edges \( e \) in \( E \), and buying \( z_e = \min(w(u), w(v)) \) from both endpoints \( e = (u, v) \).
  – Remaining weight

• When a node remains with zero weight we bought it entirely so we can take it into \( U \)

• When for all edges \( e \) in \( E \) we bought an endpoint, \( U \) is a vertex cover
Local Ratio for MVC

• How much did we pay for our solution?

• Consider edge $e=(u,v)$
  – We pay at most $2z_e$
    • recall $z_e = \min(w(u),w(v))$
  – An optimal $\text{OPT}$ solution has to pay $z_e$
    • Because $e$ has to be covered
Local Ratio for MVC

• We pay at most $2z_e$
• An optimal $\text{OPT}$ solution has to pay $z_e$

• For $e$, the local ratio between the costs is $\leq 2$

• For $E$, the ratio between the costs is $\leq 2$
Local Ratio

• In a sequential implementation, this requires polynomial time

• This is a much more general framework
  – For approximations of many additional problems

• We will see: A distributed local-ratio implementation of a \((2+\varepsilon)\)-approximation for minimum vertex cover
(2+\(\varepsilon\))-approximation

• Going over edges \(e\) in \(E\), and buying \(z'_e < \min(w(u), w(v))\) from both endpoints \(e=(u, v)\).

• Still correct, possibly slower
(2+\(\varepsilon\))-approximation

• Going over edges \(e\) in \(E\), and buying \(z'_e < \min(w(u), w(v))\) from both endpoints \(e=(u,v)\).

• When a node \(v\) remains with \(\varepsilon'w(v)\) weight we take it into \(U\), paying the remaining weight
  • \(\varepsilon' = \varepsilon/(2+\varepsilon)\)

• When for all edges \(e\) in \(E\) we bought an endpoint, \(U\) is a vertex cover
(2+\(\varepsilon\))-approximation

• How much do we pay?

• Consider edge \(e=(u,v)\)
  - Assign it weight \(z(e)=\min(w(u), w(v))\)
    • Remaining weights
  - \textbf{OPT} pays at least \(\sum_{e \in E} z(e)\)

• We pay \(\sum_{v \in U} w(v)\)

• How much is it compared to \(\sum_{e \in E} z(e)\)?
(2+ε)-approximation

- We buy a node after paying \( \Sigma_{e: v \in e} z(e) \) even if its remaining weight is not 0, but \( \epsilon'w(v) \)
- \( w(v) \leq \epsilon'w(v) + \Sigma_{e: v \in e} z(e) \)
- \( \Sigma_{v \in U} w(v) \leq \left(\frac{1}{1-\epsilon'}\right) \Sigma_{v \in U} \Sigma_{e: v \in e} z(e) \)
  \[ \leq \left(\frac{1}{1-\epsilon'}\right) 2\Sigma_{e \in E} z(e) \]
  \[ \leq (2+\epsilon)OPT \]
Distributed Implementation

- Seems local: can work in parallel on edges that do not share endpoints
- But can create conflicts for edges that share an endpoint
  - Weight cannot become negative

- Illustration

- How do we coordinate this?
Distributed Implementation

• High level description:
  – A node sends a request $X$ to its neighbor
  – The neighbor responds with a budget $Y$
    • $Y \leq X$
  – Both nodes reduce $Y$ from their weight

• Invariants:
  – Weights always remain non-negative
**Distributed Implementation**

- **Initial weight**: $w_0(v)$
- **Divide weight**: The current weight $w_i(v)$ is split:
  - $\text{Vault}_i(v) \leftarrow \epsilon'w_0(v)$
  - $\text{Bank}_i(v) \leftarrow w_i(v) - \text{Vault}_i(v)$

- **Vault** is for making requests, **Bank** is for responding
Distributed Implementation

• **Initial weight:** $w_0(v)$
  – $\text{Vault}_i(v) \leftarrow \epsilon'w_0(v)$
  – $\text{Bank}_i(v) \leftarrow w_i(v) - \text{Vault}_i(v)$

• In iteration $i$:
  – Node $v$ sends $\text{request}_i(v) \leftarrow \text{Vault}_i(v)/d_i(v)$ to each of its neighbors
  – Node $v$ responds to $\text{request}_i(u)$ from $\text{Bank}_i(v)$
Distributed Implementation

• Node $v$ responds to $\text{request}_i(u)$ from $\text{Bank}_i(v)$:
  – Sorts neighbors $u_1, \ldots, u_{d(v)}$
  – Responds to $u_1$ with
    $\text{budget}_{i,u_1}(v) \leftarrow \min(\text{request}_i(u_1), \text{Bank}_i(v))$
    and updates $\text{Bank}_i(v) \leftarrow \text{Bank}_i(v) - \text{budget}_{i,u_1}(v)$
  – Responds to $u_2$ with
    $\text{budget}_{i,u_2}(v) \leftarrow \min(\text{request}_i(u_2), \text{Bank}_i(v))$
    and updates $\text{Bank}_i(v)$
  – Etc.
Distributed Implementation

• Illustration

• Node $v$ receives $\text{budget}_{i,v}(u_j)$ from each neighbor $u_j$ and updates

$$\text{Vault}_i(v) \leftarrow \text{Vault}_i(v) - \sum_{j=1}^{d_i(v)} \text{budget}_{i,v}(u_j)$$

• Node $v$ updates:
  – Weight: $w_{i+1}(v) \leftarrow \text{Vault}_i(v) + \text{Bank}_i(v)$
  – Neighbors: $d_{i+1}(v) \leftarrow d_i(v) - \left| \{j: \text{budget}_{i,v}(u_j) < \text{request}_i(v) \} \right|$
Distributed Implementation

• If $w_{i+1}(v) \leq \varepsilon' w_0(v)$ then $v$ enters the cover $U$
  – Sends messages to neighbors
  – Outputs InCover

• If $d_{i+1}(v) = 0$ then $v$ does not enter the cover $U$
  – Outputs NotInCover

• Otherwise, $v$ continues to iteration $i+1$:
  – $\text{Vault}_{i+1}(v) \leftarrow \varepsilon' w_0(v)$
  – $\text{Bank}_{i+1}(v) \leftarrow w_{i+1}(v) - \text{Vault}_{i+1}(v)$
Analysis

• **Claim 1:** If $\text{budget}_{i,v}(u_j) < \text{request}_i(v)$ then $u_j$ terminates at the end of the iteration.

• **Proof:** If $\text{budget}_{i,v}(u_j) < \text{request}_i(v)$ then $\text{Bank}_i(u_j) = 0$ and hence

$$w_{i+1}(u_j) \leq \text{Vault}_i(u_j) \leq \varepsilon'w_0(u_j)$$

so $u_j$ outputs InCover
Analysis

- **Claim 2**: Either $d_{i+1}(v) \leq d_i(v)/2$, or $w_{i+1}(v) \leq w_i(v) - \epsilon'w_0(v)/2$

- **Proof**: If $d_{i+1}(v) > d_i(v)/2$ then, by Claim 1, for at least $d_i(v)/2$ neighbors it holds that $\text{budget}_{i,v}(u_j) = \text{request}_i(v)$.

- Since $\text{request}_i(v) = \epsilon'w_0(v)/d_i(v)$, this means that $w_{i+1}(v) \leq w_i(v) - \epsilon'w_0(v)/2$. 

Keren Censor-Hillel, Spring 2017
Analysis

• **Theorem 1**: The algorithm gives a \((2+\varepsilon)\)-approximation for minimum vertex cover in \(O(2/\varepsilon'+\log\Delta)\) rounds

• **Proof**: Applying Claim 2 for node \(v\) gives:
  • There can be at most \(O(\log\Delta)\) rounds for which \(d_{i+1}(v) \leq d_i(v)/2\) happens, and
  • There can be at most \(O(2/\varepsilon')\) rounds for which \(w_{i+1}(v) \leq w_i(v) - \varepsilon' w_0(v)/2\) happens.
Improved analysis

• **Claim 2’**: Either \( d_{i+1}(v) \leq d_i(v)/K \),
  or \( w_{i+1}(v) \leq w_i(v) - \varepsilon'w_0(v)/K \)

• **Proof**: If \( d_{i+1}(v) > d_i(v)/K \) then, by Claim 1, for at least \( d_i(v)/K \) neighbors it holds that \( \text{budget}_{i,v}(u_j) = \text{request}_i(v) \).

• Since \( \text{request}_i(v) = \varepsilon'w_0(v)/d_i(v) \), this means that \( w_{i+1}(v) \leq w_i(v) - \varepsilon'w_0(v)/K \).
Analysis

• **Theorem 1’**: The algorithm gives a \((2+\varepsilon)\)-approximation for minimum vertex cover in \(O(\log\Delta/\log\log\Delta)\) rounds

• **Proof**: Applying Claim 2’ for node \(v\) gives:
  
  • That there can be at most \(O(\log_k \Delta)\) rounds for which \(d_{i+1}(v) \leq d_i(v)/K\) happens, and
  
  • There can be at most \(O(K/\varepsilon')\) rounds for which \(w_{i+1}(v) \leq w_i(v) - \varepsilon'w_0(v)/K\) happens.

• To minimize \(O(K/\varepsilon'+\log_k \Delta)\) we choose \(K = \log \Delta/\log\log\Delta\)
CONGEST?

- Even if original weights are $O(\log n)$-bit values in $W=\{1,\ldots,\text{poly}(n)\}$, fractional values may not be so.
CONGEST?

• Instead of sending requests:
  – Send $w_0(v)$ and $d_i(v)$ to all neighbors
  – Nodes locally compute requests $\varepsilon'w_0(v)/d_i(v)$
CONGEST?

• Instead of sending budgets:
  – Vault is modified to $\varepsilon'w_0(v)/2$
  – If $\text{budget}_{i,v}(u_j) = \text{request}_i(v)$ then $u_j$ sends “accept”
  – Otherwise, respond with maximal integer $t$ for which
    $$t\varepsilon'w_0(v)/2 \leq \text{budget}_{i,u}(v).$$
  – The amount $t\varepsilon'w_0(v)/2$ is locally computed by $u$. 
CONGEST?

• Instead of sending budgets:
  – Vault is modified to $\varepsilon'w_0(v)/2$

  – The remainder of weight in vertex $v$ is another value of at most $\varepsilon'w_0(v)/2$ on top of the at most $\varepsilon'w_0(v)/2$ value which might remain in $\text{Vault}_i(v)$.
  – This sums to no more than $\varepsilon'w_0(v)/2$.
  – So, indeed $v$ returns $\text{InCover}$.