236610
Distributed Graph Algorithms

Spring 2018
Keren Censor-Hillel
Decision Tasks

• Is there a triangle in the graph?
• Is there a 4-cycle in the graph?
• Is there a \( k \)-cycle in the graph?
Distributed Decision

• **Distributed Decision:**
  – If $G$ satisfies $P$, all nodes output **true**
  – If $G$ does not satisfy $P$, at least one node outputs **false**

• Significance of allowing only one node to indicate false output (compare to global tasks)
Decision Tasks

• Is there a triangle in the graph?
• Is there a $4$-cycle in the graph?
• Is there a $k$-cycle in the graph?

• In LOCAL?
  – $O(k)$ rounds
• In CONGEST?
  – depends
3,4-cycles

4-cycles
• Can be solved in $O(n)$ rounds
• Admits an $\Omega(n/\log n)$ lower bound

3-cycles
• Seems hard
• Distributed property testing
4-Cycle Detection

**Fact 1**: There is a constant $c$ such that every graph with $cn^{3/2}$ edges contains a 4-cycle

**Theorem**: There is an $O(\sqrt{n})$-round algorithm for detecting 4-cycle freeness
4-Cycle Detection Algorithm

• \( T = a\sqrt{n} \), for \( a \geq 2c+2 \) (\( c \) from Fact 1)
• \( N^{\text{heavy}}(v) = \{u \in N(v) \mid d(u) > T\} \)
• \( h(v) = |N^{\text{heavy}}(v)| \)

1. If \( d(v) \leq T \), send \( N(v) \) to \( N(v) \)
2. If \( d(v) > T \)
3. If \( h(v) > T \), output \text{false} 
4. Else, \( h(v) \leq T \), send \( N^{\text{heavy}}(v) \) to \( N(v) \)
5. If received \( z \) from \( u, w \) in \( N(v) \), output \text{false} 
6. Output \text{true}
Complexity

The only communication is in Lines 1 and 4, each requires $T = O(\sqrt{n})$ rounds.
Correctness

Claim 1: If $h(v) > T$ then there is a 4-cycle.

Proof:

• Number of edges touching $N_{\text{heavy}}(v)$ is at least $h(v)T/2$
• By Fact 1, if there are $c \cdot h(v)^{3/2}$ edges within $N_{\text{heavy}}(v)$ then there is a 4-cycle
• Otherwise, at least $h(v)T/2 - c \cdot h(v)^{3/2}$ edges are from a node in $N_{\text{heavy}}(v)$ to a node outside of $N_{\text{heavy}}(v)$
Correctness

Claim 1: If $h(v) > T$ then there is a 4-cycle.

Proof: Otherwise, at least $h(v)T/2-c\cdot h(v)^{3/2}$ edges are from a node in $N^{\text{heavy}}(v)$ to a node outside of $N^{\text{heavy}}(v)$

- $h(v)T/2-c\cdot h(v)^{3/2} = h(v)(T/2-c\cdot h(v)^{1/2})$
  
  \[ \geq h(v)(a\sqrt{n}/2-c\sqrt{n}) \]
  
  \[ \geq a\sqrt{n} (a\sqrt{n}/2-c\sqrt{n}) \]
  
  \[ = (a^2/2-a\cdot c)n > 2n \]

- So there is at least one shared neighbor that is not $v$ itself
Correctness

• **Proof of Theorem**: By Claim 1, there is a node that outputs true only if there is a 4-cycle.

• Need to show that if there is a 4-cycle, there is a node that outputs true. By a case analysis.

heavy/light?

Keren Censor-Hillel, Spring 2018
Correctness

A light node sends all of its neighbors

A heavy node sends all of its heavy neighbors
Lower Bound for Detecting 4-Cycles

**Fact 2:** There is a constant $c'$ and a graph $G_{-4}$ with $c'n^{3/2}$ edges that does not contain a 4-cycle

**Theorem:** Every algorithm for detecting 4-cycles requires $\Omega(\sqrt{n}/\log n)$ rounds

- Add edges of $G_{-4}$ depending on input
- 4-cycle iff inputs not disjoint
- Input size $k = \Theta(n^{3/2})$
- Cut size $C = \Theta(n)$
- Requires $R = \Omega(\sqrt{n}/\log n)$ rounds
3-Cycle Detection

• Can solve in $\Delta$ rounds
  – Exchange list of neighbors
  – Linear in $n$, in the worst case

• Faster solutions?
  – New randomized sub-linear algorithm

• Lower bounds?
  – Alice-Bob framework does not work (each triangle is known to a player having two of its vertices)
Property Testing

• LCA – local computation algorithms
• Provide answers without reading the entire input

• Property testing:
  – Make queries
  – Output true if $G$ satisfies property $P$ holds
  – Output false if $G$ is far from satisfying property $P$
  – Otherwise, arbitrary output allowed
Triangle-Freeness

• $G$ is triangle-free if there are no triangles in $G$

• Different than the property of having triangles
  – Which is harder
When is $G$ far from satisfying $P$?

- Two graphs $G$ and $G'$ obtained from each other by inserting or deleting $\varepsilon m$ edges are $\varepsilon$-far

- $G$ is $\varepsilon$-far from satisfying $P$ if $G'$ does not satisfy $P$, for every $G'$ that is $\varepsilon$-far from $G$
Distributed Property Testing

• **Distributed Property Testing:**
  – If $G$ satisfies $P$, all nodes output true
  – If $G$ is $\epsilon$-far from satisfying $P$, at least one node outputs false
  – Otherwise, arbitrary outputs allowed

• Significance of allowing only one node to indicate false output (compare to global tasks)
Triangle-Freeness

**Theorem:** There is a distributed triangle-freeness testing algorithm that completes in $O(1/\varepsilon)$ rounds

**Algorithm:**
- For $\Theta(1/\varepsilon)$ times
  - Send random in $N(v) \setminus \{u\}$ to each $u$ in $N(v)$
  - If received neighbor: output **false**
- Output **true**
Distributed triangle-freeness testing

Proof:
- If triangle-free: all nodes output \textbf{TRUE}
Distributed triangle-freeness testing

**Proof:**

- If **triangle-free**: all nodes output **TRUE**
- If **$\epsilon$-far** from being **triangle-free**: At least $\epsilon m$ edges belong to triangles.
Distributed triangle-freeness testing

**Proof:**
- If triangle-free: all nodes output TRUE
- If $\varepsilon$-far from being triangle-free:
  At least $\varepsilon m$ edges belong to triangles.
  If $\{v, w\}$ is in triangle then:
  $\Pr[v$ sends $w$ node in $N(w)] \geq 1/d(v) \geq 1/m$
Distributed triangle-freeness testing

Proof:

• If \textbf{triangle-free}: all nodes output \textbf{TRUE}

• If \(\varepsilon\)-far from being \textbf{triangle-free}:

At least \(\varepsilon m\) edges belong to triangles.

If \(\{v,w\}\) is in triangle then:

\[
\Pr[v \text{ sends } w \text{ node in } N(w)] \geq 1/d(v) \geq 1/m
\]

\[
\Rightarrow \Pr[\text{no output } \text{FALSE in first round}] \leq (1-1/m)^{\varepsilon m}
\]
Distributed triangle-freeness testing

Proof:
• If triangle-free: all nodes output TRUE
• If ε-far from being triangle-free:

At least $\epsilon m$ edges belong to triangles.

If $\{v,w\}$ is in triangle then:

$\Pr[v$ sends $w$ node in $N(w)] \geq 1/d(v) \geq 1/m$

$\implies \Pr[\text{no output FALSE in first round}] \leq (1-1/m)^{\epsilon m}$

$\implies \Pr[\text{no output FALSE in } O(1/\epsilon) \text{ rounds}] \leq 1/3$
Distributed tree-freeness detection
Distributed tree-freeness detection

Color coding: Alon, Yuster, Zwick (1994)
Distributed tree-freeness testing

Color coding: Alon, Yuster, Zwick (1994)
Distributed tree-freeness detection
Distributed tree-freeness detection
Distributed tree-freeness detection
Distributed tree-freeness detection
Distributed tree-freeness detection
Distributed tree-freeness detection

• If there is no copy of the tree in G, every node outputs true.
• If there is a copy, then with probability $(1/k)^k$ it is colored according to the tree and hence will be found.

\[ \text{Repeat } O(k^k) \text{ independent times: success probability } > \frac{2}{3} \]
Subgraph testing

• H: a subgraph of an edge connected to a tree

• **Claim**: If $G$ is $\varepsilon$-far from being $H$ free then $G$ contains at least $\frac{\varepsilon m}{|E(H)|}$ edge-disjoint copies of $H$
Subgraph testing

Algorithm:
• Sample edge $e$ with orientation
• Detect tree
Subgraph testing

Algorithm:
• Sample edge $e$ with orientation
• Detect tree

How do we sample?
Each oriented edge $e$ chooses $r(e)$ in $[1,\ldots,n^4]$
For $e$, $e'$: $\Pr[r(e)=r(e')] \leq 1/n^4$
So, $\Pr[\text{all unique}] \geq 1/2$
Take $e$ with minimum $r(e)$
Subgraph testing

Algorithm:
• Sample edge $e$ with orientation
• Detect tree

$\Pr[e \text{ is unique}] \geq 1/2$

By claim, $\Pr[e \text{ in } H] \geq \varepsilon/2 |E(H)|$

Hence:
$\Pr[e \text{ is unique, in } H, H \text{ is well-colored}] \geq \varepsilon/4 |E(H)| k^k$

Repeat $O(1/\varepsilon)$ times for probability $2/3$ of success