Private Graph Algorithms in the Semi-Honest Model

Justin Brickell
November 24, 2004
Two Party Graph Algorithms

- Parties $P_1$ and $P_2$ own graphs $G_1$ and $G_2$
- $f$ is a *two-input* graph algorithm
- Compute $f(G_1,G_2)$ without revealing "unnecessary information"
Unnecessary Information

• Intuitively, the protocol should function as if a trusted third party computed the output
• We use *simulation* to prove that a protocol is private
The Semi-Honest Model

• A malicious adversary can alter his input
• A semi-honest adversary
  – adheres to protocol
  – tries to learn extra information from the message transcript
General Secure Two Party Computation

- *Any* polynomial sized functionality can be made private (in the semi-honest model)
  - Yao’s Method

- What are our goals?
  - Yao’s Method is inefficient
  - Efficient, private protocols to compute particular graph functionalities
  - Take advantage of information “leaked” by the result
Two-Input Graph Algorithms?

- Graph Isomorphism
- Comparison of graph statistics
  - $f(G_1) > f(G_2)$?
  - max flow, diameter, average degree
- Synthesized Graphs
  - $f(G_1 \cdot G_2)$
Graph Synthesis

• $G_1$ and $G_2$ are weighted complete graphs on the same vertex and edge set
  \[ G_1 = (V,E,w_1); \quad G_2 = (V,E,w_2) \]

• $g_{\text{max}}(G_1, G_2) = (V,E,w_{\text{max}})$
  \[ w_{\text{max}}(e) = \max(w_1(e), w_2(e)) \]

• $g_{\text{min}}(G_1, G_2) = (V,E,w_{\text{min}})$
  \[ w_{\text{min}}(e) = \min(w_1(e), w_2(e)) \]
Graph Synthesis

$G_1$

$G_2$

$g_{\text{max}}(G_1, G_2)$

$g_{\text{min}}(G_1, G_2)$
Graph Isomorphism

• Unlikely to find a private protocol
  – No known poly-time algorithm
Comparison of Graph Statistics

1. Compute statistic on own graph
   - Semi-honest participants can’t lie
2. Use a private comparison protocol
   - Yao’s Millionaire Protocol
   - Yao’s method (circuit protocol)
Synthesized Graphs

- This is the interesting case
- All Pairs Shortest Distance and Single Source Shortest Distance both “leak” significant useful information
  - Solved: APSD(gmin), SSSD(gmin)
  - Solved with leaks: APSD(gmax), SSSD(gmax)
APSD(gmin(G₁,G₂))

• Basic Idea: Add edges to the solution graph in order of smallest to largest
• Private, because we can recover the order from the final solution graph
Run APSD on $G_1$ and $G_2$
Initialize $G_0'$
Find shortest blue edge lengths

$G_1'$

$min1 = 2$

$G_2'$

$min2 = 2$

$G_0'$

$min0 = \infty$
Privately find global shortest length

\[\begin{align*}
G_1' & : 1 \rightarrow 2 : 7 \quad 1 \rightarrow 4 : 8 \\
G_2' & : 1 \rightarrow 2 : 2 \quad 1 \rightarrow 3 : 6 \\
G_0' & : 1 \rightarrow 2 \quad \infty \quad \infty
\end{align*}\]

\[\begin{align*}
\text{min}1 &= 2 \\
\text{min}2 &= 2 \\
\text{min}0 &= \infty
\end{align*}\]

\[\text{bluemin} = \min(\text{min}0, \text{min}1, \text{min}2) = 2\]
Find edges of length $\text{bluemin}$

$G_1'$

$G_2'$

$G_0'$

$S_1 = \{e_{23}\}$

$S_2 = \{e_{12}\}$

$S_0 = \{\}$

$\text{bluemin} = \min(\min0, \min1, \min2) = 2$
Privately find all edges of length bluemin

\[ S_1 = \{e_{23}\} \quad S_2 = \{e_{12}\} \quad S_0 = \{\} \]

\[ S = S_1 \cup S_2 \cup S_3 = \{e_{12}, e_{23}\} \]
Update $S$ edges in $G_0'$

$G_1'$

$S_1 = \{e_{23}\}$

$S = S_1 \cup S_2 \cup S_3 = \{e_{12}, e_{23}\}$
Run APSD on $G_0'$
Repeat!

\[ bluemin = \min(\min_0, \min_1, \min_2) = 4 \]

\[ S = S_1 \cup S_2 \cup S_3 = \{e_{13}, e_{24}\} \]
\[
\text{bluemin} = \min(\min_0, \min_1, \min_2) = 5
\]

\[
S = S_1 \cup S_2 \cup S_3 = \{e_{34}\}
\]
bluemin = min(min0, min1, min2) = 6

\[ S = S_1 \cup S_2 \cup S_3 = \{e_{14}\} \]
... until all edges are red
The solution is correct!

g_{\min}(G_1, G_2)

$G_0'$
Other Results

• A similar protocol for SSSD(gmin)
  – This isn’t free!
• Protocol for special case of APSD(gmax) and SSSD(gmax)
  – Input graphs obey triangle inequality
• “Leaky” protocol for APSD(gmax) and SSSD(gmax) in the general case
Final Thought

• Other graph algorithms don’t leak enough information
• Questions?