MEMORY AND QUERY TRADE-OFFS FOR GRAPH PROPERTY TESTING

Benjamín Benčík, Sumegha Garg

DIMCAS – REU 2024

GRAPH PROPERTY TESTING

PROPERTY TESTING

Definition (ϵ distance)
For any graph a property $\mathcal{P}$ and $0 \leq \epsilon \leq 1$, we say that a graph G is ϵ-far from having $\mathcal{P}$ if it has a distance greater than ϵ from every graph with $\mathcal{P}$. Otherwise, the graph is ϵ-close.

Definition (Graph property tester)
An algorithm testing property $\mathcal{P}$ is given a distance parameter ϵ and can perform queries concerning the existence of edges between any pair of vertices. If the tested graph has the property, the algorithm should accept with probability at least 2/3. If it is ϵ-far from having the property, the algorithm should reject at least with probability 2/3.

TWO GRAPH MODELS

1. Adjacency matrix model
   - Tester given access to an oracle $g : V \times V \to \{0, 1\}$
   - $dist(G_1, G_2)$ is fraction of pairs (u, v) such that $g_1(u, v) \neq g_2(u, v)$
2. Incidence function model
   - Degree of vertices is bounded by d
   - Tester is given access to $g : V \times [d] \to V \cup \{\texttt{null}\}$
   - $dist(G_1, G_2)$ is fraction of pairs (u, i) such that $g_1(u, i) \neq g_2(u, i)$

MODEL CASE - K-EDGE CONNECTIVITY

1: **for** $i = 1$ to $log(2/\tilde{\epsilon})$ **do**
2: Uniformly and independently sample $m_i = \frac{8 \log(2/\tilde{\epsilon})}{2^i \tilde{\epsilon}}$ vertices
3: For each sampled vertex v, perform 2^i BFS iteration
4: **if** If any search finds small connected component **then**
5: `reject`
6: **end if**
7: **end for**
8: `accept`

Query complexity: $\mathcal{O}\left(\frac{\log^2 1/\tilde{\epsilon}}{\tilde{\epsilon}^2}\right)$

METHODOLOGY

STUDYING MEMORY IN PROPERTY TESTING SETTINGS

- we studied $\mathcal{M}$: *memory complexity* $\times$ *query complexity* of testers for a given property
- goal is to provide lower bound
- methodology of the study:
  1. pick a property $\mathcal{P}$
  2. come up with low query algorithm
  3. come up with low memory algorithm
  4. conjecture the complexity lower bound of $\mathcal{M}$
  5. come up with proof of lower bound

DETECTING K-STAR

Consider two graph families $\mathcal{G}_1, \mathcal{G}_2$ defined as:

- $\mathcal{G}_1$: family of random graphs with n vertices of degree bound $3/4n$
- $\mathcal{G}_2$: same as $\mathcal{G}_1$ but one vertex is picked at random and connected to the rest of the graph to create a star of degree $n - 1$

Low Memory algorithm: *Naive degree checking*

- Query complexity: $\mathcal{O}(n \log(n))$
- Memory complexity: $\mathcal{O}(\log(n))$

DETECTING K-STAR

Consider two graph families $\mathcal{G}_1, \mathcal{G}_2$ defined as:

- $\mathcal{G}_1$: family of random graphs with n vertices of degree bound $3/4n$
- $\mathcal{G}_2$: same as $\mathcal{G}_1$ but one vertex is picked at random and connected to the rest of the graph to create a star of degree $n-1$

Low Memory algorithm: *Naive degree checking*

- Query complexity: $\mathcal{O}(n \log(n))$
- Memory complexity: $\mathcal{O}(\log(n))$

Query optimal algorithm: *Degree incrementing*

- Query complexity: $\mathcal{O}(n)$
- Memory complexity: $\mathcal{O}(\log(n)^2)$

LOW QUERY ALGORITHM

what should be fkn k and how is this memory log2(n)

1: $S = \{1, \ldots, n\}$
2: **repeat** $\log(n)$ **times**
3: $\quad A \leftarrow V : |A| = k$
4: $\quad$ Query (u, v) for all $u \in A$ and $v \in S$
5: $\quad S = \{v | (u, v) \in E \forall u \in A\}$
6: **end**
7: **if** $S = \varnothing$ **then**
8: $\quad$ `accept`
9: **else**
10: $\quad$ `reject`
11: **end if**

STAR-FREENESS TESTER

1: $S = \varnothing$
2: **repeat** $2n$ **times**
3: $\quad A \leftarrow V : |A| = k$
4: $\quad B \leftarrow V : |B| = k$
5: $\quad S = \{v \in S \cup A : v \text{ is connected to all vertices in } B\}$
6: **end**
7: **if** S contain vertex of degree $n - 1$ **then**
8: $\quad$ `accept`
9: **else**
10: $\quad$ `reject`
11: **end if**

ALGORITHM THE ANALYSIS

Lemma (Soundness of the algorithm)

The algorithm with constant parameter k rejects graphs from the family $\mathcal{G}_2$ *with probability* $\geq 2/3$.

Lemma (Constant size of S**)**

$max|S| \log(n) = \mathcal{O}(\log(n))$.

PROOF OF LEMMA 1

Once the star vertex s is added into the set S it will not be deleted. If $s \in S$ the algorithm correctly rejects graphs from $\mathbb{G}_2$. Therefore we want $\Pr[s \notin S] \leq 1/3$.

$$\left(\frac{n-1}{n}\right)^{k2n} \leq 1/3$$

$$k2n \log\left(\frac{n-1}{n}\right) \leq \log(1/3)$$

$$k \geq \frac{-\log(3)}{2n \log\left(1-\frac{1}{n}\right)} \approx \frac{-\log(3)}{2n\frac{-1}{n}} = \frac{\log(3)}{2}$$

PROOF OF LEMMA 2

- consider $3/4n$ regular graph (hard instance)
- denote S_i as the state of the set S in i-th iteration

$$|S_i| \sim Bin\left(|S_i|, \left(\frac{3}{4}\right)^k\right) + Bin\left(k, \left(\frac{3}{4}\right)^k\right)$$

$$\mathbb{E}[|S_i|] = k\left(\frac{3}{4}\right)^k \cdot \sum_{j=0}^{j-1}\left(\frac{3}{4}\right)^{jk} \leq \frac{k3^k}{4^k - 3^k}$$

- Apply the Chernoff inequality

GRAPH MODELS WITH RANDOMNESS

TYPES OF RANDOMNESS

- data streaming: edges are received in a random order
- semi-random queries: query to a vertex v gives random neighbour of v

CLIQUE IN THE SEMI-RANDOM MODEL

- Easy instance:
  - $\mathcal{G}_1$: there are two isolated cliques of size $n/2$
  - $\mathcal{G}_2$: random graph where each edge has probability $1/2$
- Harder instance
  - $\mathcal{G}_1$: there is a clique of size at least $n/10$ rest of the vertices form isolated random graph
  - $\mathcal{G}_1$: random graph where each edge has probability $1/2$

PROJECT CONTINUES VERY INTERESTING RESULTS COMING SOON!

ACKNOWLEDGEMENTS

This project has received funding from the European Union's Horizon 2020 research and innovation program under the Marie Skłodowska-Curie grant agreement No 823748.