**Faster, Fault-resilient Sublinear Algorithms**

**Nithin M. Varma**

Department of Computer Science, Boston University

Collaborators: Kashyap Dixit, Ramesh Krishnan S. Pallavoor, Sofya Raskhodnikova, Abhradeep Thakurta

*Part of the work done at the Pennsylvania State University*

---

**Sublinear Algorithms**

- Limited computing power, memory and time.
- Approximate solutions enough.

**Fault-resilient Sublinear Algorithms: Why?**

- Limited computing power, memory and time.
- Approximate solutions enough.

**Property Testing**

**(Rubinfeld, Sudan `96, Goldreich, Goldwasser, Ron `98)**

- Input object, abstracted as a function.
- Query (Domain element) → Answer/Special symbol (Function value/s)
- Testing Algorithm
- Output YES or NO

**Input**

- \[ e \in (0,1), n \]

**Sample complexity of erasure-resilient testing**

- \[ \Theta \left( \frac{\log n}{e} \right) \] (monotonicity, convexity, Lipschitz properties)
- \[ \Theta \left( \frac{\log n}{1 - \alpha} \right) \] (approximation parameter & input size)

**Example Parameters**

- \( n \) is an additional parameter of the input

**Erasure-resilient Property Testing**

- Input object, abstracted as a function.
- Query (Domain element) → Answer/Special symbol (Function value/s)
- Testing Algorithm
- Output YES or NO

**Input**

- \[ e \in (0,1), n \]

**Sample complexity of testing**

- \( \Theta \left( \frac{\log n}{e} \right) \)

**Key Assumptions**

1. Worst-case erasures: All erasures made by an ‘adversary’ who has full knowledge of the algorithm.
2. Erasure-pattern unknown to algorithm: Algorithm knows whether a point is erased only after querying the point.

**Faster Sublinear Algorithms? Parameterize!**

**Big Idea:** Measure the complexity with respect to parameters other than the input size.

**Property Testing**

- \( n \) is an additional parameter of the input
- \( \Theta \left( \frac{\log n}{e} \right) \)

**Example Parameters**

- Number of distinct values in the image (image-size), Image diameter, Maximum degree (if the input is a graph)

**Ongoing Work and Future Directions**

- Compare the complexity of tolerant testing against erasure-resilient testing with worst-case erasures.
- Compare the complexity of testing with worst-case erasures and testing random erasures.
- Investigate other models of erasures and corruptions.
- Design erasure-resilient testers for other important properties such as that of being a low-degree polynomial and dictatorship.

---

(Published in the proceedings of ICALP `16. Joint work with Kashyap Dixit, Sofya Raskhodnikova and Abhradeep Thakurta.)

---

(Published in the proceedings of ITCS `17. Joint work with Ramesh Krishnan S. Pallavoor and Sofya Raskhodnikova.)