Aalto University
School of Science

Testing Spreading Behavior in Networks with Arbitrary Topologies

Augusto Modanese (with Yuichi Yoshida)

Department of Computer Science
augusto.modanese@aalto.fi

28 Nov 2023

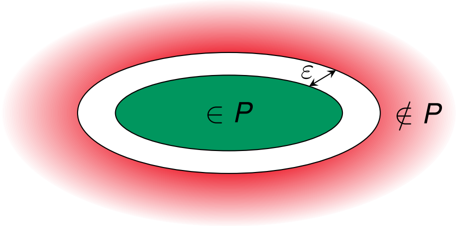
Motivation

- Imagine we have a *huge* network (with fixed connections)
- At every point in time we can *query* the state of any node
  - E.g., healthy/unhealthy (disease spreading), believes in rumor X (social networks), ...
- Unfeasible to keep track of every single node

- Hypothesis: states evolve following a fixed local rule
- Can we test this hypothesis? How?

Property Testing in a Nutshell

- Centralized, *randomized* machine with query access to input
- Task: Test if input is in property P or not
  - Formally property is just a set of “good” inputs (e.g., formal language)
- Concretely: Given parameter $\varepsilon > 0$, determine if input is in P or ε*-far* from being in P
  - “ε-far” $=$ has distance at least ε from P
  - Distance measure is context-specific



- We only look at *query* complexity (time, space, etc. irrelevant)
  - Best possible complexity is $O(1/\varepsilon)$ (for non-trivial properties)

Our Setting

- Underlying network is a graph $G = (V, E)$ with $|V| = n$ nodes
- Network runs for T steps
- The object we are testing is the *environment* $\texttt{ENV} : V \times [T] \to \{0, 1\}$
- Distance measure:

$$d(\texttt{ENV}, \texttt{ENV}') = \frac{|\{(v, t) \mid \texttt{ENV}(v, t) \neq \texttt{ENV}'(v, t)\}|}{nT}$$

Our Setting

- Underlying network is a graph $G = (V, E)$ with $|V| = n$ nodes
- Network runs for T steps
- The object we are testing is the *environment* $\texttt{ENV}: V \times [T] \to \{0, 1\}$
- Distance measure:

$$d(\texttt{ENV}, \texttt{ENV}') = \frac{|\{(v, t) \mid \texttt{ENV}(v, t) \neq \texttt{ENV}'(v, t)\}|}{nT}$$

- Previous work [GR17; NR21] on the case of cellular automata (i.e., G is a line graph)
- We study *only* the 1-BP a.k.a. OR rule:
  1. If $\texttt{ENV}(u, t) = 1$ for some $u \in N(v)$, then $\texttt{ENV}(v, t+1) = 1$
  2. If $\texttt{ENV}(u, t) = 0$ for all $u \in N(v)$, then $\texttt{ENV}(v, t+1) = 0$
- $\texttt{ENV}$ is ε-far from 1-BP if at least εnT bit flips are needed to make $\texttt{ENV}$ follow 1-BP rule

Some Properties of Algorithms

One- vs. Two-sided Error

- Algorithm A is a *one-sided error tester* for 1-BP if the following holds:
  1. If $\texttt{ENV} \in$ 1-BP, then always $A(\texttt{ENV}) = 1$
  2. If $\texttt{ENV}$ is ε-far from 1-BP, then $\Pr[A(\texttt{ENV}) = 1] < 1/2$

- Algorithm A is a *two-sided error tester* for 1-BP if the following holds:
  1. If $\texttt{ENV} \in$ 1-BP, then $\Pr[A(\texttt{ENV}) = 1] \geq 2/3$
  2. If $\texttt{ENV}$ is ε-far from 1-BP, then $\Pr[A(\texttt{ENV}) = 1] < 1/3$

Some Properties of Algorithms

Adaptiveness and Time-Conformability

- Property testing algorithms (in any context) are either *adaptive* or *non-adaptive*
- Algorithm A is non-adaptive if and only if it works as follows:
  1. A produces a set Q of queries (without looking at its input x)
  2. A receives the values of x for Q
  3. A then computes its decision based on these values (no further queries allowed)
- Otherwise A is adaptive

Some Properties of Algorithms

Adaptiveness and Time-Conformability

- Property testing algorithms (in any context) are either *adaptive* or *non-adaptive*
- Algorithm A is non-adaptive if and only if it works as follows:
  1. A produces a set Q of queries (without looking at its input x)
  2. A receives the values of x for Q
  3. A then computes its decision based on these values (no further queries allowed)
- Otherwise A is adaptive

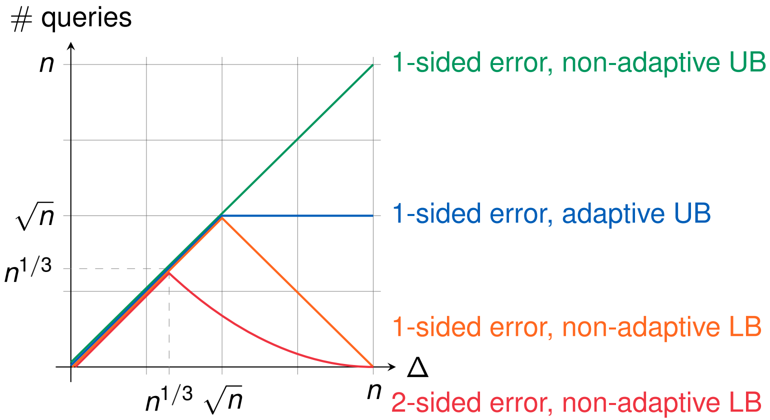
- In our context *time-conforming* also relevant
- A is time-conforming if it respects time:
  - If $(\cdot, t)$ query is made after $(\cdot, t')$ query, then necessarily $t > t'$ (even for distinct nodes)

Quick Overview of Results

- Case $T = 2$:
  - Two upper bounds $\leftarrow$ coming up next
  - Two lower bounds $\leftarrow$ not today
- Case $T > 2$:
  - Two upper bounds $\leftarrow$ coming up later

The Case $T = 2$

Results



Note: Diagram assumes ε constant, ignores polylog(n) factors

The Case $T = 2$

Techniques

- Fairly easy algorithm giving non-adaptive, 1-sided error $O(\Delta/\varepsilon)$ UB:
  1. Select a set Q of nodes uniformly at random ($|Q| = O(1/\varepsilon)$)
  2. Query all of Q and $N(Q)$ in 1st and 2nd steps
  3. If something is "bad", reject; otherwise accept

- LB shows this is optimal for $\Delta = \tilde{O}(\sqrt{n})$ and ε constant
  - Comes from expander graphs with the "right" expansion properties
- Adaptive 1-sided error $\tilde{O}(\sqrt{n}/\varepsilon)$ UB comes from a much more complex algorithm

The General Case $T > 2$

Results

- Trivial if $T \geq 2\,\mathrm{diam}(G)/\varepsilon$
  - Every connected component must be either black or white
- We give two 1-sided error, non-adaptive UBs:
  - First one is direct adaptation from $T = 2$ case
    - Query complexity $O(\Delta^{T-1}/\varepsilon T)$
  - Second one is inspired on idea of [NR21]
    - Assumes $T \geq 4/\varepsilon$
    - Query complexity $\tilde{O}(|E|/\varepsilon T)$
- Together we have non-trivial testing algorithms for $\Delta = o(\log n)$ in *any* graph
  - On graphs that exclude a fixed minor (e.g., planar graphs) we can strengthen this to $\Delta = o(\log^2 n)$

The General Case $T > 2$

The Method of [NR21]

Consider setting in a 1D cellular automaton

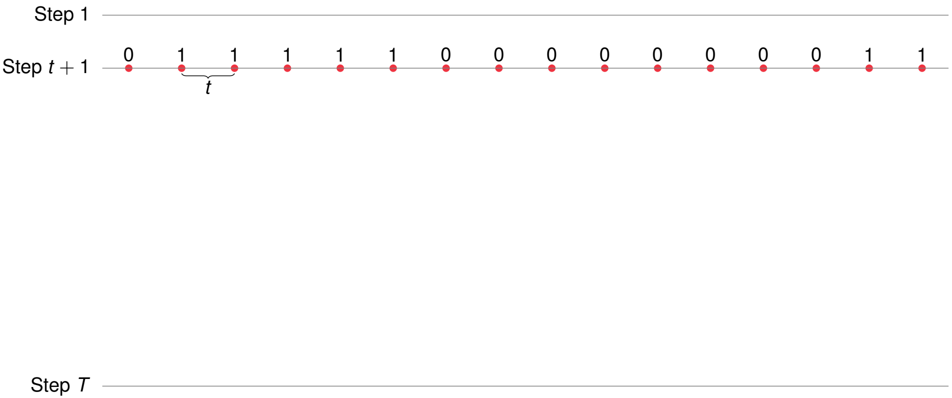
Step 1

Step T

The General Case $T > 2$

The Method of [NR21]

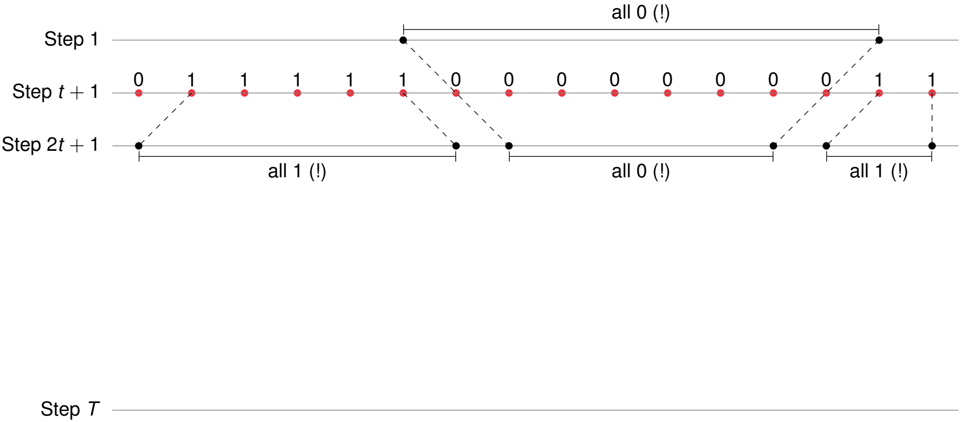
Consider setting in a 1D cellular automaton



The General Case $T > 2$

The Method of [NR21]

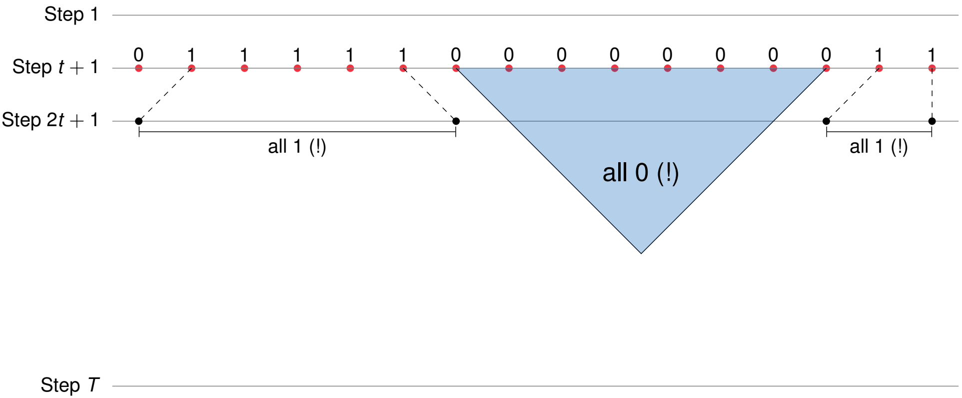
Consider setting in a 1D cellular automaton



The General Case $T > 2$

The Method of [NR21]

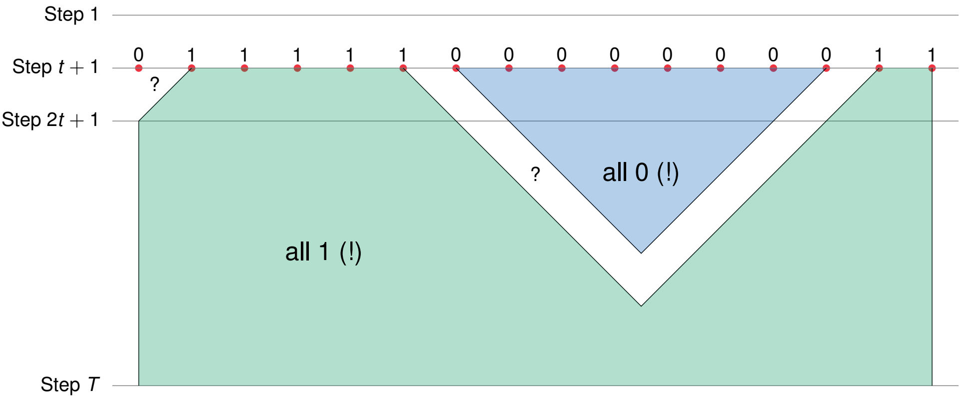
Consider setting in a 1D cellular automaton



The General Case $T > 2$

The Method of [NR21]

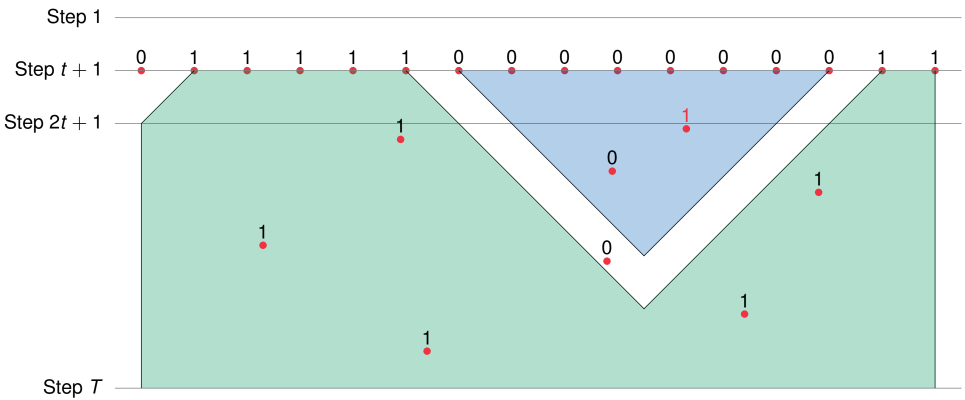
Consider setting in a 1D cellular automaton



The General Case $T > 2$

The Method of [NR21]

Consider setting in a 1D cellular automaton



The General Case $T > 2$

Our Algorithm

- Generalize the idea to arbitrary topologies:
  - Apply *low-diameter decomposition*: intervals "$=$" components
- For $d \in \mathbb{N}_+$ and $\alpha > 0$, a (d, α)*-decomposition* of $G = (V, E)$ is a set $C \subseteq E$ with $|C| \leq \alpha|E|$ and for which there is a partition $V = V_1 + \cdots + V_r$ such that:
  1. For $u, v \in V$, $uv \in C$ if and only if there are i and j with $i \neq j$ such that $u \in V_i$ and $v \in V_j$
  2. For every i, $\mathrm{diam}(V_i) \leq d$
- For any $d \in \mathbb{N}_+$, every graph admits a $(d, O(\log(n)/d))$-decomposition [Bar96]

The General Case $T > 2$

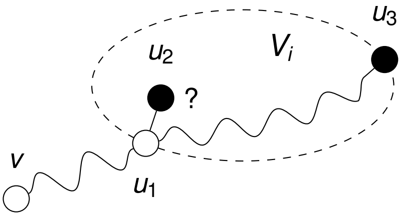
Our Algorithm

- Generalize the idea to arbitrary topologies:
  - Apply *low-diameter decomposition*: intervals "=" components
- For $d \in \mathbb{N}_+$ and $\alpha > 0$, a (d, α)-*decomposition* of $G = (V, E)$ is a set $C \subseteq E$ with $|C| \leq \alpha|E|$ and for which there is a partition $V = V_1 + \cdots + V_r$ such that:
  1. For $u, v \in V$, $uv \in C$ if and only if there are i and j with $i \neq j$ such that $u \in V_i$ and $v \in V_j$
  2. For every i, $\mathrm{diam}(V_i) \leq d$
- For any $d \in \mathbb{N}_+$, every graph admits a $(d, O(\log(n)/d))$-decomposition [Bar96]
- Algorithm sketch:
  1. Compute a decomposition of G (with adequate d and α)
  2. Query endpoints of C at a certain time step t
  3. Predict the states of every vertex as much as possible
  4. Perform random queries and reject if anything is not OK

The General Case $T > 2$

Our Algorithm—An Example

Suppose we have $\mathrm{dist}(v, u_1) \ll \mathrm{dist}(u_1, u_3) = \mathrm{diam}(V_i)$

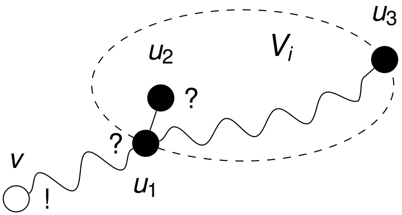


Step t

The General Case $T > 2$

Our Algorithm—An Example

Suppose we have $\mathrm{dist}(v, u_1) \ll \mathrm{dist}(u_1, u_3) = \mathrm{diam}(V_i)$

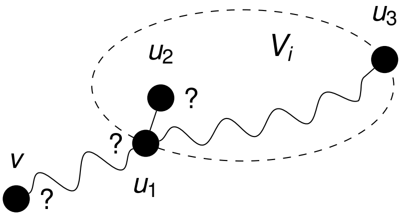


Step $t + \mathrm{dist}(v, u_1)$

The General Case $T > 2$

Our Algorithm—An Example

Suppose we have $\mathrm{dist}(v, u_1) \ll \mathrm{dist}(u_1, u_3) = \mathrm{diam}(V_i)$

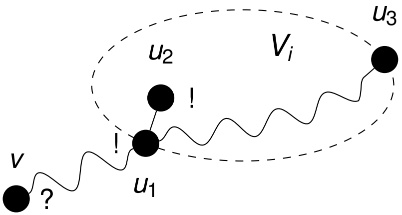


Step $t + \mathrm{dist}(v, u_1) + 1$

The General Case $T > 2$

Our Algorithm—An Example

Suppose we have $\mathrm{dist}(v, u_1) \ll \mathrm{dist}(u_1, u_3) = \mathrm{diam}(V_i)$

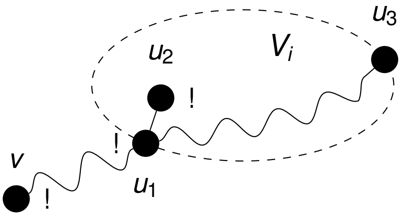


Step $t + \mathrm{diam}(V_i)$

The General Case $T > 2$

Our Algorithm—An Example

Suppose we have $\mathrm{dist}(v, u_1) \ll \mathrm{dist}(u_1, u_3) = \mathrm{diam}(V_i)$



Step $t + \mathrm{dist}(v, u_3)$

Wrap-Up and Outlook

- First foray into testing local rules in general graphs
- Focused on 1-BP aka OR rule

- Main results:
  - Upper and lower bounds for case $T = 2$
    - Tight up to $\Delta = O(n^{1/3})$
  - Two algorithms for case $T > 2$
    - Non-trivial testing possible up to $\Delta = o(\log n)$

- Still a lot left to explore:
  - Tighter results for $T = 2$ and large Δ
  - Lower bounds for $T > 2$ case
  - Other rules like XOR, majority, 2-BP and friends, ...

References

[1] Yair Bartal. "Probabilistic Approximations of Metric Spaces and Its Algorithmic Applications". In: *Proceedings of the 37th Annual Symposium on Foundations of Computer Science (FOCS)*. 1996, pp. 184–193.

[2] Oded Goldreich and Dana Ron. "On Learning and Testing Dynamic Environments". In: *J. ACM* 64.3 (2017), 21:1–21:90.

[3] Yonatan Nakar and Dana Ron. "Testing Dynamic Environments: Back to Basics". In: *Proceedings of the 48th International Colloquium on Automata, Languages, and Programming (ICALP)*. 2021, 98:1–98:20.