Testing of ‘massively parametrized problems’ -

Ilan Newman
Haifa University

Based on joint work with:
Sourav Chakraborty, Eldar Fischer, Shirley Halevi, Oded Lachish, Arie Matsliah, Eyal Rozenberg, Dekel Tzur, Orly Yahalom.

Standard Models

- A Fixed underlying structure. Inputs: a set of ‘vectors’ assigned with this structure. E.g., a coloring of the points. Property: a collection of ‘vectors’, : E.g.,
- Graph properties: Structure is K_n, input (vectors): Boolean assignment on edges. Property: e.g., connected graphs, bipartite graphs...

- Properties of Boolean functions: Structure: the Boolean cube. Inputs: Boolean assignment of vertices. Property: e.g., monotone, linear,….

- Here: Structure is not fixed in advance !
  E.g., Structure: a given undirected graph, inputs: all 0/1 assignments to its edges, property: the subgraph is Eulerian, connected,….
- Strongly connected, DAG, having a di-path of length k….
- Structure: A given graph, inputs: all 0/1 assignments to its vertices. Properties: graph properties of the induced subgraph.

- Structure: A Boolean circuit/ formula/ branching program…, inputs: Boolean assignment to the variables. Property: the 1-inputs of the computation.
- There are many more examples….

Comments on 'standard' models, e.g., graph properties

- [GT01]: Every 1-sided error testable property is testable by a generic algorithm: An algorithm that queries at random a subgraph of a given size and accept/reject only based on it.
- Thus, algorithm are somewhat 'not interesting'.

- [AFNS] A characterization of all testable graph properties in terms of regular partitions.

- In massively parametrized graph properties:
- Typically, there is a 'significant' place for preprocessing the structure.
- Algorithms turns out to be quite different from the 'standard' sampling.

Some ‘old’ results

- [N00] testing membership in read-once constant width Branching programs.
- [FLNRRS02] – testing monotonicity in ‘general’ posets.

Subgraphs porperties

- Structure: A given arbitrary underlying graph $G=(V,E)$. Algorithm has full knowledge of G.
- Inputs: (Boolean) assignment on the edges (vertices). Hence a property P is a subset of $\{0,1\}^E$.

P can be interpreted is several ways:

subgraph porperties

The edge assignment is interpreted as its existence /non existence. Thus an input defines a subgraph G containing the edges of value '1'.

Hence, a property is a collection of subgraphs, e.g:

Being bipartite (k-colorable), Eulearian, Hamiltonian, being acyclic etc.

Orientations porperties

The edge assignment is interpreted as an orientation of it. Hence, a property is a collection of directed graphs obtained by orienting the edges of G in certain ways.

e.g:

Being strongly connected, Eulearian, having an s-t path, being acyclic, excluding a forbidden subgraph etc.

Properties of constraint graphs

Structure: An arbitrary undirected graph, and Boolean formulae φ_v, for every vertex v in G, on variables that are indexed by the adjacent edges to v.

Inputs: Boolean assignment to the variables.

Property: assignments that satisfy φ_v for every vertex v.

Examples

- the vertex formulae assert that the number of '1'-edges is even (Eulerian).
- A 2-coloring of the edges s.t not all edges adjacent to a vertex have the same value.

Motivation

- The constraint graph model is fairly general, any property problem can be cast in this way.
- The subgraph model directly generalizes the dense graph model. Gives the possibility to consider sparse graphs in a way that the representation remains simple.
- One can pose interesting problems.
- The algorithms are interesting (not just sampling, not just local search).

Connection to other testing problems: Testing satisfying assignment of CNF formulae.

- [BHR] 3CNF are generally hard to test, even if every variable appears O(1) times.
- [FLNRRS] 2CNF are also hard, even if monotone (By testing monotonicity).
- If monotone and every variable appears O(1) times – testable.
- Read-twice CNF are testable – reduction from a result on orientation/constraint graphs.

This works for the combination of: every monotone variable appears O(1) times and every non-monotone appears 2 times.

Read-O(1)-times is not testable in general.

Testing constraint graphs

[HLNT CCC07]

- Every property can be cast in this way (star).
- A constraint graph is in LD_3 if for every vertex with degree at least 3, the hamming distance between any two assignments not satisfying φ_v is at least 3. e.g: φ_v is a clause of size 3 or more.
- Thm: Every LD_3 has an $(\varepsilon, \exp(1/\varepsilon))$ 1-sided error test.

- Cor: Every read-twice CNF formula is testable.

- Algorithm: non-trivial sampling. Proof is quite technical.
- Best possible; there are properties in which two non-sat assignments have dist=2 and are highly non-testable. Similarly for read-3-times CNF's.

- Cor: the property of orientation of having no source vertex is testable.

  The property of edge 2-coloring in which not all edges have the same color is testable.

Algorithm flavour

- Define a suitable neighborhood B(z), around each vertex z.
- Algorithm for the 'generic' case:
  - Select a random edge e.
  - for each vertex z such that e is in B(z), and z has suitably bounded degree, test all edges adjacent to z and reject if z is not satisfied.

Testing of Orientations

[HLNT ECCC06, CFLMN Random07, FLMNY Random08].

Testing H-freeness

- For underlying graphs with bounded degree, being H-free is testable for any fixed forbidden directed graph H, that has no source or has no drain.
- For forbidden graphs with sources and drains: P2-free is testable while P3-free is highly non-testable.

- What about testing H-freeness in input graphs of unbounded average degree ?

- If testable, algorithm is not poly(1/ ε).

Testing strong connectivity

Easy cases:

- G has $\omega(n)$ edges.
- The DAG of components has $\Omega(n)$ sources.

- Def: An undirected graph $G=(V,E)$ is called δ -efficiently-Steiner connected if for every $S \subseteq V$, $|S| < \delta^2 n$ there is a connected subgraph $T=(V,E')$ of G spanning S, with $|E'| < 10\,\delta n$.

- Thm: If G is $1/\log n$ -efficiently Steiner connected then strong conn. is testable for G.
- SC is testable for $n x n$ grid.
- SC is testable on expanders.

Testing s-t connectivity

- Testing s-t connectivity can be efficiently done for any underlying graph.
  - Algorithm is non-trivial. It uses several reduction steps to testing small width branching programs.

- Testing Eulerianity: Not testable in general. However, there are sublinear testing algorithms and quite efficient for certain classes of graphs.

Some general lower bounds for non-adaptive 1-sided error algorithms

[FLNR on-going work]

Consider the property of subgraphs of being bipartite. A 1-sided error algorithm needs to find a refutation in order to reject. Here a witness is an odd-cycle.

Hence, the size of the refutation is a lower bound. However, this is quite weak.

- Let $G=(V,E)$ be an expander graph, with girth = $\Omega(\log n)$.

- Refutation size is $O(\log n)$.

- Can prove: non-adaptive lower bound of $\Omega(n^{\delta})$, for some fixed $\delta>0$.

This is quite general; the same technique gives lower bound for testing acyclicity, testing any property in which a refutation contains a 'large' path, or a cycle.

E.g., any (non-trivial) minor-H-free graph for a given H, e.g., planarity.

- [FL..... – on *going*]: membership in read-once formulae is testable.
- Extensions to non-boolean case