A Study on Fixed-Parameter Algorithms and Property Testing

Student: Joseph Chuang-Chieh Lin

Advisors: Professor Maw-Shang Chang
Professor Peter Rossmanith

Department of Computer Science and Information Engineering
National Chung Cheng University
Taiwan

26 July 2011

Outline

Introduction:
Fixed-parameter algorithms

Fixed-parameter algorithms

A parameterized problem

A language $L \subseteq \Sigma^* \times \mathbb{Z}^+$ that consists of input pairs (I, k).

- The first component: problem instance (Σ: finite alphabet).
- The second component: parameter.

- Fixed-parameter algorithms:
  - Solving parameterized problems in $O(f(k) \cdot \text{poly}(n))$ time.
  - $f(k)$: an arbitrary function solely depending on k.
- Characteristics of fixed-parameter algorithms:
  - They can be used to solve **NP**-hard problems exactly.
  - Efficient when k is small.
- Problems admit such algorithms: **FPT** (fixed-parameter tractable).

Fixed-parameter algorithms (contd.)

- Common types of parameters:
  - Target size (e.g., the size of a vertex cover)
  - Structure of the input (e.g., the treewidth of a graph)

A parameter associated with the target: vertex cover

- Given a graph $G = (V, E)$,

  $C \subseteq V$ is a vertex cover:

  each edge in E has at least one of its endpoints in C.

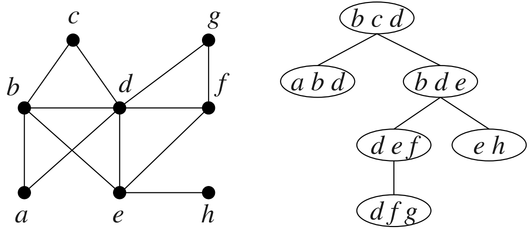
- The Vertex Cover problem:

  Determine if a graph has a vertex cover of size $\leq k$:
  - ▷ **FPT**: $O(1.2738^k + kn)$ [Chen *et al.* 2010].

⋆ It can be efficiently solved when k is small.

A parameter associated with the input structure: treewidth

- Tree-decomposition of a graph:



- The *width* of the tree-decomposition: $|\text{maximum bag}| - 1$.
- The treewidth of G: the minimum width over all tree-decompositions of G.

A parameter associated with the input structure: treewidth

- Roughly speaking, treewidth measures how close a graph is to being a tree.
- Many **NP**-hard graph problems can be solved in polynomial time or even linear time when the treewidth of the input graph is bounded.
  - the Maximum Independent Set problem
  - the Minimum Dominating Set problem
  - the Hamiltonian Cycle problem

    ⋮

Fixed-parameter *intractable*?

- $I \subseteq V$ is an independent set:
  - None of pairs of the vertices in I are adjacent.
- A well-known fact:
  - A graph G has a vertex cover of size $k \Leftrightarrow G$ has an independent set of size $k' = n - k$.
- $\exists$ an $O(f(k') \cdot \text{poly}(n))$ algorithm $\mathcal{A}$ for the Independent Set problem
  $\Rightarrow$ $\mathcal{A}$ can be used to solve the Vertex Cover problem efficiently even when $k = n - k'$ is large.
- Fixed-parameter intractable problems:

  The Independent Set problem, the Clique problem,
  the Dominating Set problem, . . ., etc.

Fixed-parameter *intractable*?

- $I \subseteq V$ is an independent set:
  - None of pairs of the vertices in I are adjacent.
- A well-known fact:
  - A graph G has a vertex cover of size $k \Leftrightarrow G$ has an independent set of size $k' = n - k$.
- $\exists$ an $O(f(k') \cdot \mathrm{poly}(n))$ algorithm $\mathcal{A}$ for the Independent Set problem
  $\Rightarrow \mathcal{A}$ can be used to solve the Vertex Cover problem efficiently even when $k = n - k'$ is large.
- Fixed-parameter intractable problems:
  The Independent Set problem, the Clique problem, the Dominating Set problem, . . ., etc.

Fixed-parameter *intractable*?

- $I \subseteq V$ is an independent set:
  - None of pairs of the vertices in I are adjacent.
- A well-known fact:
  - A graph G has a vertex cover of size $k \Leftrightarrow G$ has an independent set of size $k' = n - k$.
- $\exists$ an $O(f(k') \cdot \text{poly}(n))$ algorithm $\mathcal{A}$ for the Independent Set problem
  $\Rightarrow$ $\mathcal{A}$ can be used to solve the Vertex Cover problem efficiently even when $k = n - k'$ is large.
- Fixed-parameter intractable problems:

  The Independent Set problem, the Clique problem,
  the Dominating Set problem, ..., etc.

Introduction:
Property testing

Property testing

- General notion: Rubinfeld & Sudan (*SIAM J. Comput.* 1996).
  - Graph property testing: Goldreich, Goldwasser & Ron (*J. ACM* 1998).

Task of property testing

Input: An input I and a specified property $\mathcal{P}$

Task: Fulfill the following requirements in $o(|I|)$ time.

- If I satisfies $\mathcal{P}$ $\Rightarrow$ answer "yes" with prob. $\geq \frac{2}{3}$;
- If I is **far** from satisfying $\mathcal{P}$ $\Rightarrow$ answer "no" with prob. $\geq \frac{2}{3}$.

- A notion of "approximating" yes/no problems in sublinear time.
- **Property testers**: algorithms accomplishing the above task.

Property testing (ϵ-far)

- In property testing, we use ϵ-far to say that the input is far from a certain property.
- ϵ: the least fraction of the input that needs to be modified.
- For example, $L = (0, 2, 3, 4, 1)$ is 0.2-far from being monotonically increasing.

Property testing (ϵ-far)

- In property testing, we use ϵ-far to say that the input is far from a certain property.
- ϵ: the least fraction of the input that needs to be modified.
- For example, $L = (0, 2, 3, 4, 1)$ is 0.2-far from being monotonically increasing.

Motivations

Motivations

- Using property testing as preprocessing of running exact algorithms.
- Introducing parameters to the standard property testing:
  - Parameterized property testing

Parameterized property testing

Parameterized property testers

Input: An input object I, $\epsilon > 0$, and a parameter $k \in \mathbb{N}$.

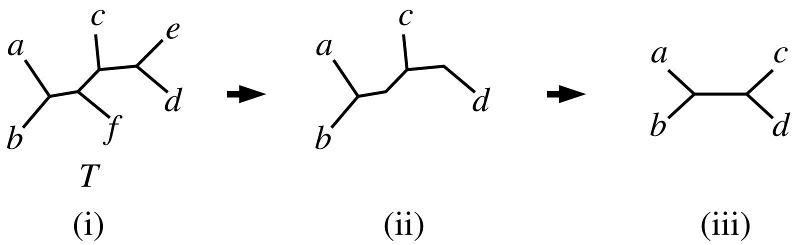
Property: $\mathcal{P}$.

Task: Testing if I has the property $\mathcal{P}$ in $O(f(k, 1/\epsilon))$ time.

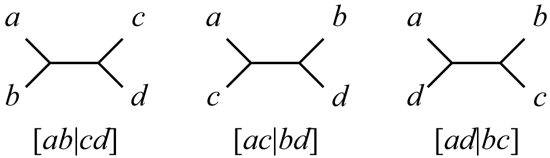
- $f(k, 1/\epsilon)$: a function solely depending on k and ϵ.
- The assumption: $|I|$ is sufficiently large.
  - $\star$ k and ϵ are regarded as constants with respect to $|I|$.

Our contributions

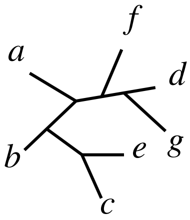
Quartets & evolutionary trees



Quartets & evolutionary trees



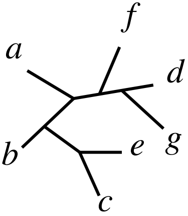
$[ab \mid cd]$,	$[ab \mid ce]$,	$[af \mid bc]$,	$[ag \mid bc]$,
$[ad \mid be]$,	$[ab \mid df]$,	$[ab \mid dg]$,	$[af \mid be]$,
$[ag \mid be]$,	$[ab \mid fg]$,	$[ad \mid ce]$,	$[ac \mid df]$,
$[ac \mid dg]$,	$[ac \mid ef]$,	$[ag \mid ce]$,	$[af \mid cg]$,
$[ae \mid df]$,	$[ae \mid dg]$,	$[af \mid dg]$,	$[ae \mid fg]$,
$[bd \mid ce]$,	$[bc \mid df]$,	$[bc \mid dg]$,	$[bf \mid ce]$,
$[bg \mid ce]$,	$[bc \mid fg]$,	$[be \mid df]$,	$[be \mid dg]$,
$[bd \mid fg]$,	$[be \mid fg]$,	$[ce \mid df]$,	$[ce \mid dg]$,
$[cf \mid dg]$,	$[ce \mid fg]$,	$[de \mid fg]$.	



T

Quartet errors w.r.t. T.

$[ab \mid cd]$, $[ab \mid ce]$, $[af \mid bc]$, $[ag \mid bc]$,
$[ad \mid be]$, $[ab \mid df]$, $[ab \mid dg]$, $[af \mid be]$,
$[ag \mid be]$, $[ab \mid fg]$, $[ad \mid ce]$, $[ac \mid df]$,
$[ac \mid dg]$, $[ac \mid ef]$, $[ag \mid ce]$, $[af \mid cg]$,
$[ae \mid df]$, $[ae \mid dg]$, $[af \mid dg]$, $[ae \mid fg]$,
$[bd \mid ce]$, $[bc \mid df]$, $[bc \mid dg]$, $[bf \mid ce]$,
$[bg \mid ce]$, $[bc \mid fg]$, $[be \mid df]$, $[be \mid dg]$,
$[bd \mid fg]$, $[be \mid fg]$, $[ce \mid df]$, $[ce \mid dg]$,
$[cf \mid dg]$, $[ce \mid fg]$, $[de \mid fg]$.



T

Quartet errors w.r.t. T.

I. Contribution in designing fixed-parameter algorithms

- MQI: minimum quartet inconsistency

The parameterized MQI problem

Input: A complete set Q of quartet topologies, $k \in \mathbb{N}$.

Task: Determine if Q has $\leq k$ quartet errors.

- Previous results:
  - ⋆ **NP**-complete [Berry *et al.* 1999];
  - ⋆ Approximation ratio: $O(n^2)$ [Jiang *et al.* 2000];
  - ⋆ $O(4^k n + n^4)$ [Gramm and Niedermeier 2003].
- Our results:
  - $O(3.0446^k n + n^4)$, $O(2.0162^k n^3 + n^5)$, $O^*((1+\varepsilon)^k)$.
    - ⋆ Chang, Lin, & Rossmanith: *Theory Comput. Syst.*, 2010.

I. Contribution in designing fixed-parameter algorithms

- MQI: minimum quartet inconsistency

The parameterized MQI problem

Input: A complete set Q of quartet topologies, $k \in \mathbb{N}$.

Task: Determine if Q has $\leq k$ quartet errors.

- Previous results:
  - ⋆ **NP**-complete [Berry *et al.* 1999];
  - ⋆ Approximation ratio: $O(n^2)$ [Jiang *et al.* 2000];
  - ⋆ $O(4^k n + n^4)$ [Gramm and Niedermeier 2003].
- Our results:
  - $O(3.0446^k n + n^4)$, $O(2.0162^k n^3 + n^5)$, $O^*((1+\varepsilon)^k)$.
    - ⋆ Chang, Lin, & Rossmanith: *Theory Comput. Syst.*, 2010.

I. Contribution in designing fixed-parameter algorithms

- MQI: minimum quartet inconsistency

The parameterized MQI problem

Input: A complete set Q of quartet topologies, $k \in \mathbb{N}$.

Task: Determine if Q has $\leq k$ quartet errors.

- Previous results:
  - ⋆ **NP**-complete [Berry *et al.* 1999];
  - ⋆ Approximation ratio: $O(n^2)$ [Jiang *et al.* 2000];
  - ⋆ $O(4^k n + n^4)$ [Gramm and Niedermeier 2003].
- Our results:
  - $O(3.0446^k n + n^4)$, $O(2.0162^k n^3 + n^5)$, $O^*((1+\varepsilon)^k)$.
    - ⋆ Chang, Lin, & Rossmanith: *Theory Comput. Syst.*, 2010.

II. Contribution in property testing

The property: tree-like ($\mathcal{P}_{TL}$)

- $\exists$ an evolutionary tree T consistent with a complete Q.

Input: A complete set Q of quartet topologies, $0 < \epsilon < 1$.
Task: Testing if Q is tree-like.

Our results:

- An $O(n^3/\epsilon)$ property tester.
  - Chang, Lin, & Rossmanith: *Theory Comput. Syst.*, online first.
- An $O(k3^k n^3/\epsilon)$ property tester with k missing quartets.
  - Lin: *Proc. Workshop on Combin. Math. Comput. Theory*, 2011.

II. Contribution in property testing

The property: tree-like ($\mathcal{P}_{TL}$)

- $\exists$ an evolutionary tree T consistent with a complete Q.

Input: A complete set Q of quartet topologies, $0 < \epsilon < 1$.
Task: Testing if Q is tree-like.

Our results:

- An $O(n^3/\epsilon)$ property tester.
  - ⋆ Chang, Lin, & Rossmanith: *Theory Comput. Syst.*, online first.
- An $O(k3^k n^3/\epsilon)$ property tester with k missing quartets.
  - ⋆ Lin: *Proc. Workshop on Combin. Math. Comput. Theory*, 2011.

The sparse / dense model for testing graph properties

⋆ **The sparse model**

- $f_G : V(G) \times [d] \mapsto V(G) \cup \varnothing$ (adjacency list)
  - $[d] := \{1, 2, \ldots, d\}$
  - $f_G(v, i) = u$: (u, v) is the ith edge incident to v.
  - $f_G(v, i) = \varnothing$: there is no such edge.
- ϵ-far:
  $\geq \epsilon dn$ edge insertions or removals are required.

⋆ **The dense model**

- $M_G : V(G) \times V(G) \mapsto \{0, 1\}$ (adjacency matrix)
  - $M_G(v, u) = 1$: u and v are adjacent.
- ϵ-far:
  $\geq \epsilon n^2$ edge insertions or removals are required.

III. Parameterized property testers (previous work)

Positive results:

- k-colorability in the dense model:
  - $O(k^2 \ln^2 k/\epsilon^4))$ [Alon & Krvivelevich 2002].
- being H-free (without having H as a subgraph; $|H| = h$) in the dense model:
  - $O(h^2(1/2\epsilon)^{h^2/4})$ [Alon 2002].
- k-connectivity in the sparse model:
  - $O(d(ck/\epsilon d)^k \log(k/\epsilon d)))$ [Yoshida & Ito 2008].

III. Parameterized property testers (previous work)

A negative result:

- k-colorability in the *sparse* model:
  - $\Omega(n)$ [Bogdanov, Obata & Trevisan 2002].

III. Parameterized property testers (trivial to test)

Properties that are trivial to test (sparse model):

- Having a simple k-path / k-cycle ($\in$ **FPT**).
  - Answer "yes" for any input graph.
- Having a dominating set of size $\leq k$ ($\notin$ **FPT**).
  - Answer "no" for any input graph.
- Having a clique / an independent set of size $\leq k$ ($\notin$ **FPT**).
  - Answer "yes" for any input graph.

III. Parameterized property testers (our positive results)

For the following two problems in **NP**-hard $\cap$ **FPT**:

- the Vertex Cover problem and
- the Treewidth problem,

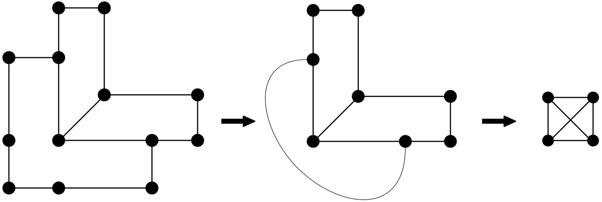
we propose parameterized property testers for their corresponding properties.

III. Parameterized property testers (Vertex Cover)

- The property: Having a vertex cover of size $\leq k$.
  - Denoted it by $\mathcal{P}_{VC\leq k}$.
- Previous results:
  - **FPT**: $O(1.2738^k + kn)$ [Chen *et al.* 2010]
  - **Property testing**:
    - $2^{2^{\cdot^{\cdot^{\cdot^{2}}}}} \Big\} O(\text{poly}(1/\epsilon))$ 2's (dense model) [Alon & Shapira 2008]
    - $\Omega(\sqrt{n})$ for $\mathcal{P}_{VC\leq \rho n}$ (sparse model) [Goldreich & Ron 2002]
- Our results:
  - ⋆ $O(kd/\epsilon)$ (sparse model).

III. Parameterized property testers (Treewidth)

- H is a minor of G:
  - H can be obtained from G by edge removals, vertex removals and edge *contractions*.



(a) (b) (c)

III. Parameterized property testers (Treewidth)

- Minor-closed properties: closed under taking minors.
- G satisfies a minor-closed property $\mathcal{P} \Leftrightarrow \exists$ a finite $\mathcal{F}$ s.t. H is not a minor of G for each $H \in \mathcal{F}$.
  [Robertson & Seymour 1983–2004]
- "Having treewidth bounded by k" is minor-closed [Kloks 1994].
  - Yet the obstruction set $\mathcal{F}$ is not explicitly known for $k > 3$.

III. Parameterized property testers (Treewidth)

- The property: Having treewidth $\leq k$.
  - Denoted it by $\mathcal{P}_{tw \leq k}$.
- Previous results:
  - **FPT**: $2^{\Theta(k^3)} \cdot k^{O(1)} \cdot n$ [Bodlaender 1996]
  - **Property testing**: $2^{\mathrm{poly}(1/\epsilon)}$ (sparse model) [Hassidim *et al.* 2009]
    - for minor-closed properties
- Our results: (sparse model)
  - $2^{d^{O(kd^3/\epsilon^2)}}$
  - $d^{(k/\epsilon)^{O(k^2)}} + 2^{\mathrm{poly}(k,d,1/\epsilon)}$.
    - ⋆ Both are **uniform** w.r.t. k.
    - ⋆ **Without** using the obstruction set of forbidden minors.

Summary of our contributions

Property (Problem)	PC	PT	PPT
MQI	$O(3.0446^k n + n^4)$ $O(2.0162^k n^3 + n^5)$ $O^*((1+\varepsilon)^k)$	–	–
$\mathcal{P}_{TL}$	–	$O(n^3/\epsilon)$	$O(k3^k n^3/\epsilon)$
$\mathcal{P}_{VC\leq k}$	$O(1.2738^k + kn)$	$\left.2^{2^{\cdot^{\cdot^{\cdot^{2}}}}}\right\}O(\text{poly}(1/\epsilon))\text{ 2's}$ †	$O(kd/\epsilon)$ ‡
$\mathcal{P}_{VC\leq\rho\cdot n}$	–	$\Omega(\sqrt{n})$ ‡	–
$\mathcal{P}_{tw\leq k}$	$2^{\Theta(k^3)}\cdot k^{O(1)}\cdot n$	$O(2^{\text{poly}(1/\epsilon)})$ ‡	$2^{d^{O(kd^3/\epsilon^2)}}$ ‡ $d^{(k/\epsilon)^{O(k^2)}} + 2^{\text{poly}(k,d,1/\epsilon)}$ ‡

Summary of our contributions (Testers)

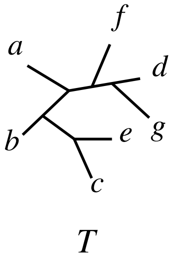
Property	Sublinear	Testable	Easily-testable	Non-adaptive/Adaptive	1/2-sided error
$\mathcal{P}_{TL}$	Yes	?	?	Non-adaptive	1
$\mathcal{P}_{VC\leq k}$	Yes	Yes	Yes	Adaptive	1
$\mathcal{P}_{tw\leq k}$	Yes	Yes	?	Adaptive	2

Part I:

Efficient fixed-parameter algorithms for the MQI problem

Tree-consistency

- Q_T : the set of quartet topologies induced by T.
  - $|Q_T| = \binom{n}{4}$.
- Q is tree-consistent (with T):
  - $\exists T$ s.t. $Q \subseteq Q_T$.
  - ▷ **tree-like** if $Q = Q_T$.
- Q is called complete:
  - Exactly one topology for every quartet;
  - Otherwise, *incomplete*.



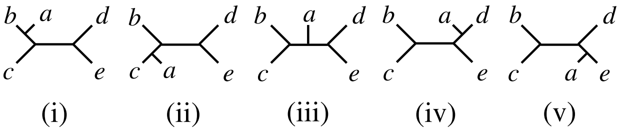
An evolutionary tree T.

Quintets & quintet topologies

- A quintet is a set of five taxa in S.
- Quintet topologies:

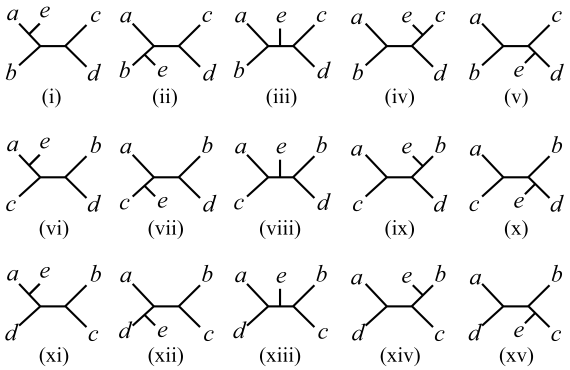
Quintets & quintet topologies

- A quintet is a set of five taxa in S.
- Quintet topologies:



Quintets & quintet topologies

- A quintet is a set of five taxa in S.
- Quintet topologies:



Resolved quintets

- A resolved quintet:
  - $[ab|cd], [ab|ce], [ab|de], [ac|de], [bc|de]$.

Resolved quintets

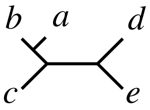
- A resolved quintet:

▷ $[ab|cd], [ab|ce], [ab|de], [ac|de],$
$[bc|de].$

Resolved quintets

- A resolved quintet:

▷ $[ab|cd]$, $[ab|ce]$, $[ab|de]$, $[ac|de]$, $[bc|de]$.



The crucial theorem

Theorem 2.1 (Bandel & Dress 1986; Chang, Lin & Rossmanith 2010)

Given

- Q: *a complete set* Q *of quartet topologies over* S
- $\ell \in S$: *an arbitrarily fixed taxon*

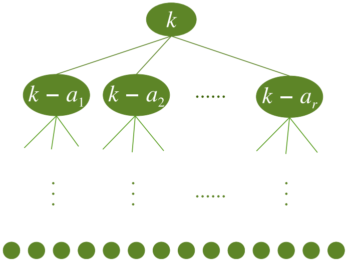
Q *is tree-like* $\Leftrightarrow$ *every quintet containing* ℓ *is resolved.*

The approach: depth-bounded search trees

- Given

  $$T(k) = T(k - a_1) + T(k - a_2) + \ldots + T(k - a_r),$$

  $$T(0) = T(1) = \ldots = T(a'_r - 1) = 1,$$
  where $a'_r = \min\{a_1, a_2, \ldots, a_r\}$.



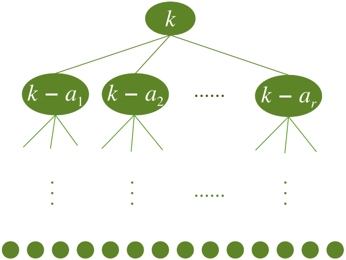
- **Goal:** find a number $\alpha > 0$, such that $T(k) = O^*(\alpha^k)$.
  - O^*: complexity without polynomial terms.
  - Trick: Solving $\alpha^k = \alpha^{k-a_1} + \alpha^{k-a_2} + \ldots + \alpha^{k-a_r}$.
    - e.g., for $T(k) = T(k-1) + T(k-2)$, we can derive $T(k) = O^*(1.62^k)$.
  - $(a_1, a_2, \ldots, a_r)$: branching vector; α: branching number.

The approach: depth-bounded search trees

- Given

  $T(k) = T(k - a_1) + T(k - a_2)$
  $+ \ldots + T(k - a_r)$,

  $T(0) = T(1) = \ldots = T(a'_r - 1) = 1$,
  where $a'_r = \min\{a_1, a_2, \ldots, a_r\}$.



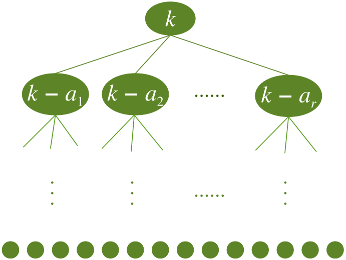
- **Goal:** find a number $\alpha > 0$, such that $T(k) = O^*(\alpha^k)$.
  - O^*: complexity without polynomial terms.
  - Trick: Solving $\alpha^k = \alpha^{k-a_1} + \alpha^{k-a_2} + \ldots + \alpha^{k-a_r}$.
    - e.g., for $T(k) = T(k-1) + T(k-2)$,
      we can derive $T(k) = O^*(1.62^k)$.
  - $(a_1, a_2, \ldots, a_r)$: branching vector; α: branching number.

The approach: depth-bounded search trees

- Given

  $T(k) = T(k - a_1) + T(k - a_2)$
  $+ \ldots + T(k - a_r)$,

  $T(0) = T(1) = \ldots = T(a'_r - 1) = 1$,
  where $a'_r = \min\{a_1, a_2, \ldots, a_r\}$.



- **Goal:** find a number $\alpha > 0$, such that $T(k) = O^*(\alpha^k)$.
  - O^*: complexity without polynomial terms.
  - Trick: Solving $\alpha^k = \alpha^{k-a_1} + \alpha^{k-a_2} + \ldots + \alpha^{k-a_r}$.
    - e.g., for $T(k) = T(k-1) + T(k-2)$, we can derive $T(k) = O^*(1.62^k)$.
  - $(a_1, a_2, \ldots, a_r)$: branching vector; α: branching number.

Idea of our first fixed-parameter algorithm

- Using the depth-bounded search tree strategy.
- Eliminate unresolved quintets.
- 15 branches for each node of the search tree.

The first algorithm (contd.)

- For the quintet $\{a, b, c, d, e\}$:
  - $[ab|cd]$, $[ac|be]$, $[ae|bd]$, $[ad|ce]$, $[bc|de] \in Q$.
- Consider the (first) quintet topology:
  - $[ab|cd]$, $[ab|ce]$, $[ab|de]$, $[ac|de]$, $[bc|de]$.

branching vector	branching number
$(3, 3, 4, 3, 3, 3, 4, 3, 3, 4, 4, 3, 3, 4, 3)$	$2.30042\ldots$
$(2, 4, 4, 4, 5, 2, 2, 3, 3, 4, 3, 4, 3, 3, 4)$	$2.46596\ldots$
$(2, 4, 4, 4, 5, 3, 3, 4, 4, 5, 2, 3, 2, 2, 3)$	$2.54314\ldots$
$(1, 3, 3, 3, 4, 3, 3, 4, 4, 5, 3, 4, 3, 3, 4)$	$2.55234\ldots$
$(3, 5, 5, 3, 5, 2, 2, 3, 5, 5, 2, 3, 2, 3, 2)$	$2.67102\ldots$
$(1, 3, 3, 5, 5, 1, 3, 3, 3, 4, 2, 4, 4, 4, 5)$	$3.04454\ldots$

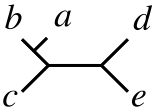
The first algorithm (contd.)

- For the quintet $\{a, b, c, d, e\}$:
  - ▷ $[ab|cd]$, $[ac|be]$, $[ae|bd]$, $[ad|ce]$, $[bc|de] \in Q$.
- Consider the (first) quintet topology:
  - ▷ $[ab|cd]$, $[ab|ce]$, $[ab|de]$, $[ac|de]$, $[bc|de]$.

branching vector	branching number
$(3, 3, 4, 3, 3, 3, 4, 3, 3, 4, 4, 3, 3, 4, 3)$	2.30042...
$(2, 4, 4, 4, 5, 2, 2, 3, 3, 4, 3, 4, 3, 3, 4)$	2.46596...
$(2, 4, 4, 4, 5, 3, 3, 4, 4, 5, 2, 3, 2, 2, 3)$	2.54314...
$(1, 3, 3, 3, 4, 3, 3, 4, 4, 5, 3, 4, 3, 3, 4)$	2.55234...
$(3, 5, 5, 3, 5, 2, 2, 3, 5, 5, 2, 3, 2, 3, 2)$	2.67102...
$(1, 3, 3, 5, 5, 1, 3, 3, 3, 4, 2, 4, 4, 4, 5)$	3.04454...

The first algorithm (contd.)

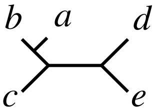
- For the quintet $\{a, b, c, d, e\}$:
  - ▷ $[ab|cd]$, $[ac|be]$, $[ae|bd]$, $[ad|ce]$, $[bc|de] \in Q$.
- Consider the (first) quintet topology:
  - ▷ $[ab|cd]$, $[ab|ce]$, $[ab|de]$, $[ac|de]$, $[bc|de]$.



branching vector	branching number
(3, 3, 4, 3, 3, 3, 4, 3, 3, 4, 4, 3, 3, 4, 3)	2.30042...
(2, 4, 4, 4, 5, 2, 2, 3, 3, 4, 3, 4, 3, 3, 4)	2.46596...
(2, 4, 4, 4, 5, 3, 3, 4, 4, 5, 2, 3, 2, 2, 3)	2.54314...
(1, 3, 3, 3, 4, 3, 3, 4, 4, 5, 3, 4, 3, 3, 4)	2.55234...
(3, 5, 5, 3, 5, 2, 2, 3, 5, 5, 2, 3, 2, 3, 2)	2.67102...
(1, 3, 3, 5, 5, 1, 3, 3, 3, 4, 2, 4, 4, 4, 5)	3.04454...

The first algorithm (contd.)

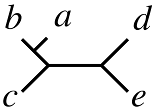
- For the quintet $\{a, b, c, d, e\}$:
  - ▷ $[ab|cd]$, $[ac|be]$, $[ae|bd]$, $[ad|ce]$, $[bc|de] \in Q$.
- Consider the (first) quintet topology:
  - ▷ $[ab|cd]$, $[ab|ce]$, $[ab|de]$, $[ac|de]$, $[bc|de]$.



branching vector	branching number
(**3**, 3, 4, 3, 3, 3, 4, 3, 3, 4, 4, 3, 3, 4, 3)	2.30042...
(2, 4, 4, 4, 5, 2, 2, 3, 3, 4, 3, 4, 3, 3, 4)	2.46596...
(2, 4, 4, 4, 5, 3, 3, 4, 4, 5, 2, 3, 2, 2, 3)	2.54314...
(1, 3, 3, 3, 4, 3, 3, 4, 4, 5, 3, 4, 3, 3, 4)	2.55234...
(3, 5, 5, 3, 5, 2, 2, 3, 5, 5, 2, 3, 2, 3, 2)	2.67102...
(1, 3, 3, 5, 5, 1, 3, 3, 3, 4, 2, 4, 4, 4, 5)	3.04454...

The first algorithm (contd.)

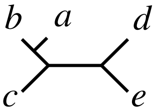
- For the quintet $\{a, b, c, d, e\}$:
  - $[ab|cd]$, $[ac|be]$, $[ae|bd]$, $[ad|ce]$, $[bc|de] \in Q$.
- Consider the (first) quintet topology:
  - $[ab|cd]$, $[ab|ce]$, $[ab|de]$, $[ac|de]$, $[bc|de]$.



branching vector	branching number
$(3, 3, 4, 3, 3, 3, 4, 3, 3, 4, 4, 3, 3, 4, 3)$	2.30042...
$(2, 4, 4, 4, 5, 2, 2, 3, 3, 4, 3, 4, 3, 3, 4)$	2.46596...
$(2, 4, 4, 4, 5, 3, 3, 4, 4, 5, 2, 3, 2, 2, 3)$	2.54314...
$(1, 3, 3, 3, 4, 3, 3, 4, 4, 5, 3, 4, 3, 3, 4)$	2.55234...
$(3, 5, 5, 3, 5, 2, 2, 3, 5, 5, 2, 3, 2, 3, 2)$	2.67102...
$(1, 3, 3, 5, 5, 1, 3, 3, 3, 4, 2, 4, 4, 4, 5)$	3.04454...

The first algorithm (contd.)

- For the quintet $\{a, b, c, d, e\}$:
  - $[ab|cd]$, $[ac|be]$, $[ae|bd]$, $[ad|ce]$, $[bc|de] \in Q$.
- Consider the (first) quintet topology:
  - $[ab|cd]$, $[ab|ce]$, $[ab|de]$, $[ac|de]$, $[bc|de]$.



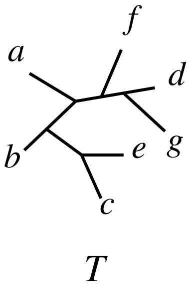
branching vector	branching number
$(3, 3, 4, 3, 3, 3, 4, 3, 3, 4, 4, 3, 3, 4, 3)$	2.30042...
$(2, 4, 4, 4, 5, 2, 2, 3, 3, 4, 3, 4, 3, 3, 4)$	2.46596...
$(2, 4, 4, 4, 5, 3, 3, 4, 4, 5, 2, 3, 2, 2, 3)$	2.54314...
$(1, 3, 3, 3, 4, 3, 3, 4, 4, 5, 3, 4, 3, 3, 4)$	2.55234...
$(3, 5, 5, 3, 5, 2, 2, 3, 5, 5, 2, 3, 2, 3, 2)$	2.67102...
$(1, 3, 3, 5, 5, 1, 3, 3, 3, 4, 2, 4, 4, 4, 5)$	3.04454...

The first algorithm (contd.)

Theorem 2.2

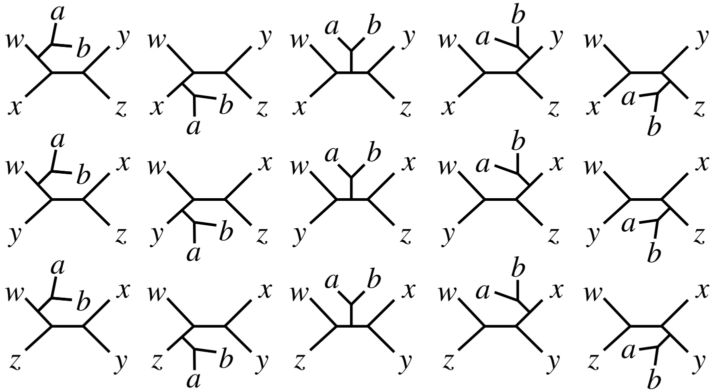
There exists an $O(3.0446^k n + n^4)$ fixed-parameter algorithm for the parameterized MQI problem.

Siblings



- Siblings: $\{c, e\}$ and $\{d, g\}$.

Sextet topologies & a fixed pair of siblings



Sextet topologies & a fixed pair of siblings (contd.)

- $\{a, b, w, x\}$, $\{a, b, w, y\}$, $\{a, b, w, z\}$, $\{a, b, x, y\}$, $\{a, b, x, z\}$, $\{a, b, y, z\}$ have determined topologies.
  - ▷ $[ab|wx]$, $[ab|wy]$, $[ab|wz]$, $[ab|xy]$, $[ab|xz]$, $[ab|yz]$.
- 9 quartet topologies undetermined.

Sextet topologies & a fixed pair of siblings (contd.)

- $\{a, b, w, x\}$, $\{a, b, w, y\}$, $\{a, b, w, z\}$, $\{a, b, x, y\}$, $\{a, b, x, z\}$, $\{a, b, y, z\}$ have determined topologies.
  - ▷ $[ab|wx]$, $[ab|wy]$, $[ab|wz]$, $[ab|xy]$, $[ab|xz]$, $[ab|yz]$.
- 9 quartet topologies undetermined.

The second algorithm

branching vector	branching number
$(6,6,8,6,6,6,6,5,6,6,6,6,5,6,6)$	1.58005...
$(5,6,6,5,6,6,6,5,6,6,7,6,7,6,7)$	1.58142...
...	...
$(1,5,5,7,8,2,6,6,8,9,3,7,7,8,8)$	2.00904...
$(1,5,5,9,9,2,6,6,6,8,3,7,7,7,9)$	2.01615...

Theorem 2.3

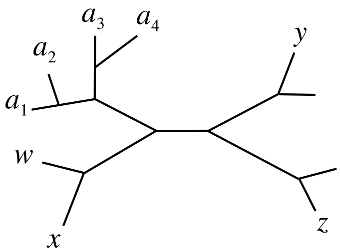
There exists an $O(2.0162^k n^3 + n^5)$ fixed-parameter algorithm for the parameterized MQI problem.

Idea of the third algorithm

- Generalized from the second algorithm.
- Siblings $\Rightarrow$ adjacent taxa.

Adjacent taxa

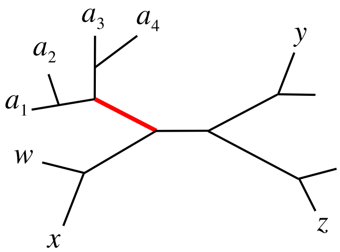
- Adjacent $m \geq 2$ taxa $a_1, \ldots, a_m$:
- $(\{a_1, \ldots, a_m\}, S \setminus \{a_1, \ldots, a_m\})$.



Given a number $2 \leq \omega \leq n/2$, there must be m adjacent taxa, where $\omega \leq m \leq 2\omega - 2$.

Adjacent taxa

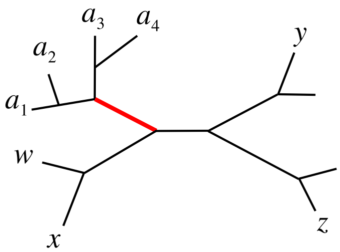
- Adjacent $m \geq 2$ taxa $a_1, \ldots, a_m$:
- $(\{a_1, \ldots, a_m\}, S \setminus \{a_1, \ldots, a_m\})$.



Given a number $2 \leq \omega \leq n/2$, there must be m adjacent taxa, where $\omega \leq m \leq 2\omega - 2$.

Adjacent taxa

- Adjacent $m \geq 2$ taxa $a_1, \ldots, a_m$:
- $(\{a_1, \ldots, a_m\}, S \setminus \{a_1, \ldots, a_m\})$.

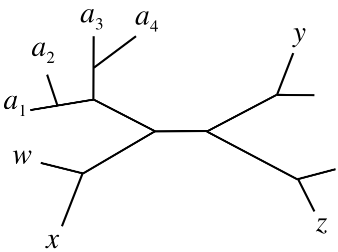


Given a number $2 \leq \omega \leq n/2$, there must be m adjacent taxa, where $\omega \leq m \leq 2\omega - 2$.

The third algorithm

- Change the topology of $\{a_1, w, x, y\}$.

▷ Change the topologies of $\{a_2, w, x, y\}$, $\{a_3, w, x, y\}$, $\{a_4, w, x, y\}$ as well.



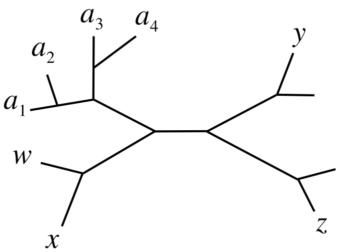
Theorem 2.4

There exists an $O^*((1+\varepsilon)^k)$ fixed-parameter algorithm for the parameterized MQI problem.

The third algorithm

- Change the topology of $\{a_1, w, x, y\}$.

▷ Change the topologies of $\{a_2, w, x, y\}$, $\{a_3, w, x, y\}$, $\{a_4, w, x, y\}$ as well.



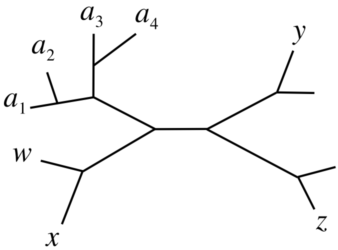
Theorem 2.4

There exists an $O^*((1+\varepsilon)^k)$ fixed-parameter algorithm for the parameterized MQI problem.

The third algorithm

- Change the topology of $\{a_1, w, x, y\}$.

▷ Change the topologies of $\{a_2, w, x, y\}$, $\{a_3, w, x, y\}$, $\{a_4, w, x, y\}$ as well.



Theorem 2.4

There exists an $O^*((1+\varepsilon)^k)$ fixed-parameter algorithm for the parameterized MQI problem.

Part II:

Property testing for tree-likeness of quartet topologies

Testing tree-likeness of quartet topologies

- The input size: $|Q| = \binom{n}{4}$.
- $\mathcal{P}_{TL}$: tree-likeness of quartet topologies.
- Q is ϵ-far from being tree-like:
  - $\geq \epsilon\binom{n}{4}$ quartet topologies are required to be modified.

The first property tester for $\mathcal{P}_{TL}$

- $\mathcal{P}_{TL}$: tree-likeness of quartet topologies

Theorem 3.1

If Q is ϵ-far from $\mathcal{P}_{TL}$

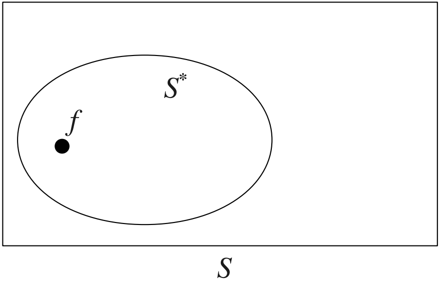
$$\Downarrow$$

$\geq \epsilon n/36$ *unresolved quintets containing f for arbitrary $f \in S$* .

- $\mathbf{Pr}$[a random quintet containing f is unresolved] $\geq \frac{\epsilon/36}{n^3}$.

The first property tester for $\mathcal{P}_{TL}$ (contd.)

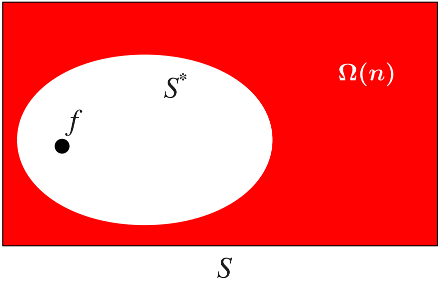
- f: an arbitrary fixed taxon in S.
- S^*: maximal subset of S containing f such that Q over S^* is tree-like.



- $|S \setminus S^*| \neq o(n)$.
  - Otherwise, changing $\leq \binom{n}{3} \cdot \binom{o(n)}{1} + \binom{n}{2} \cdot \binom{o(n)}{2} + \binom{n}{1} \cdot \binom{o(n)}{3} + \binom{o(n)}{4} = o(n^4)$ quartet topologies of Q makes it tree-like.

The first property tester for $\mathcal{P}_{TL}$ (contd.)

- f: an arbitrary fixed taxon in S.
- S^*: maximal subset of S containing f such that Q over S^* is tree-like.



- $|S \setminus S^*| \neq o(n)$.
  - Otherwise, changing $\leq \binom{n}{3} \cdot \binom{o(n)}{1} + \binom{n}{2} \cdot \binom{o(n)}{2} + \binom{n}{1} \cdot \binom{o(n)}{3} + \binom{o(n)}{4} = o(n^4)$ quartet topologies of Q makes it tree-like.

The first property tester for $\mathcal{P}_{TL}$ (contd.)

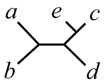
`Tree-Like-Tester`(Q): /* a property tester for $\mathcal{P}_{TL}$ */
/* Q: a complete set of quartet topologies. */

1. pick an arbitrary taxon $\ell \in S$;
2. **repeat**
   a. pick $s_1, s_2, s_3, s_4 \in S \setminus \{\ell\}$ uniformly at random;
   b. **if** $\{s_1, s_2, s_3, s_4, \ell\}$ is not resolved **then return** "no";
   c. **end if**
3. **until** the loop iterates for $\frac{72}{\epsilon} n^3$ times
4. "yes".

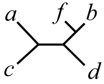
- $O(n^3/\epsilon)$ time.
- One-sided error & non-adaptive.

Dealing with incomplete Q's

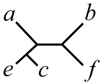
- Extending our results to incomplete Q's seems impossible...



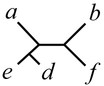
$[ab|ce]$, $[ab|de]$, $[ad|ce]$, $[bd|ce]$



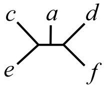
$[ac|bf]$, $[ad|bf]$, $[ac|df]$, $[bf|cd]$



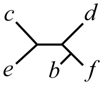
$[ab|ce]$, $[ac|bf]$, $[af|ce]$, $[ae|bf]$, $[bf|ce]$



$[ab|de]$, $[ad|bf]$, $[ae|bf]$, $[bf|de]$



$[ad|ce]$, $[ac|df]$, $[af|ce]$, $[ce|df]$

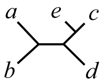


$[bd|ce]$, $[bf|cd]$, $[bf|ce]$, $[bf|de]$, $[ce|df]$

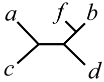
- What if Q is *almost complete*?

Dealing with incomplete Q's

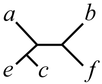
- Extending our results to incomplete Q's seems impossible...



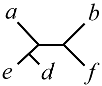
$[ab|ce]$, $[ab|de]$, $[ad|ce]$, $[bd|ce]$



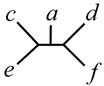
$[ac|bf]$, $[ad|bf]$, $[ac|df]$, $[bf|cd]$



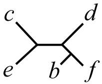
$[ab|ce]$, $[ac|bf]$, $[af|ce]$, $[ae|bf]$, $[bf|ce]$



$[ab|de]$, $[ad|bf]$, $[ae|bf]$, $[bf|de]$



$[ad|ce]$, $[ac|df]$, $[af|ce]$, $[ce|df]$



$[bd|ce]$, $[bf|cd]$, $[bf|ce]$, $[bf|de]$, $[ce|df]$

- What if Q is *almost complete*?

Testing tree-consistency with k missing quartets

- Q: a set of $\binom{n}{4} - k$ quartet topologies.
- $T_{miss} = \{\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_k\}$: the set of k missing quartets.

⋆ There are 3^k possible assignments of topologies of the k missing quartets in T_{miss}.

A modified tree-likeness tester

`Tree-Like-Tester2`(Q)
/* Q: A complete set of quartet topologies. */

1. pick an arbitrary taxon $\ell \in S$;
2. **repeat**
   a. pick four taxa $s_1, s_2, s_3, s_4 \in S \setminus \{\ell\}$ uniformly at random;
   b. **if** the quintet $\{s_1, s_2, s_3, s_4, \ell\}$ is not resolved **then return** "no";
3. **until** the loop iterates for $\frac{72(k+1)}{\epsilon} n^3$ times
4. **return** "yes";

Corollary 3.2

Algorithm `Tree-Like-Tester2`*:*

- *time complexity:* $O(kn^3/\epsilon)$.
- *one-sided-error.*
- *if* Q *is* ϵ*-far from being tree-like, it returns "yes" with probability* $< 1/3^{k+1}$.

- $Q_{miss}(i) = \{q_1(i), q_2(i), \ldots, q_k(i)\}$:
  the ith assignment of topologies of the k missing quartets.
  - $q_j(i)$: the topology of the quartet $\mathbf{t}_j$ in the ith assignment.

`Dense-Consistency-Tester`(Q)

1. **for** $i \leftarrow 1$ to 3^k
   a. **if** `Tree-Like-Tester2`$(Q \cup Q_{miss}(i))$ returns "yes" **then return** "yes";
2. **end for**
3. **return** "no".

- Time complexity: $O(k3^k n^3/\epsilon)$.
- Error probability: $< \frac{1}{3^{k+1}} \cdot 3^k = \frac{1}{3}$ (union bound).

Parameterization helps testing here.

- $Q_{miss}(i) = \{q_1(i), q_2(i), \ldots, q_k(i)\}$:
  the ith assignment of topologies of the k missing quartets.
  - $q_j(i)$: the topology of the quartet $\mathbf{t}_j$ in the ith assignment.

`Dense-Consistency-Tester`(Q)

1. **for** $i \leftarrow 1$ to 3^k
   a. **if** `Tree-Like-Tester2`($Q \cup Q_{miss}(i)$) returns "yes" **then** **return** "yes";
2. **end for**
3. **return** "no".

- Time complexity: $O(k3^k n^3/\epsilon)$.
- Error probability: $< \frac{1}{3^{k+1}} \cdot 3^k = \frac{1}{3}$ (union bound).

Parameterization helps testing here.

- $Q_{miss}(i) = \{q_1(i), q_2(i), \ldots, q_k(i)\}$:
  the ith assignment of topologies of the k missing quartets.
  - $q_j(i)$: the topology of the quartet $\mathbf{t}_j$ in the ith assignment.

`Dense-Consistency-Tester`(Q)

1. **for** $i \leftarrow 1$ to 3^k
   a. **if** `Tree-Like-Tester2`($Q \cup Q_{miss}(i)$) returns "yes" **then** **return** "yes";
2. **end for**
3. **return** "no".

- Time complexity: $O(k3^k n^3/\epsilon)$.
- Error probability: $< \frac{1}{3^{k+1}} \cdot 3^k = \frac{1}{3}$ (union bound).

Parameterization helps testing here.

Part III:

Parameterized property testers

Parameterized property testing for $\mathcal{P}_{VC_{\leq k}}$

A simple observation

Observation

In the sparse model,

- $G = (V, E)$ *satisfies* $\mathcal{P}_{VC \leq k} \Rightarrow |E| \leq kd$.
- $G = (V, E)$ *is* ϵ*-far from* $\mathcal{P}_{VC \leq k} \Rightarrow |E| \geq \epsilon dn$.

- This leads to an $O(d/\epsilon)$ tester for $\mathcal{P}_{VC \leq k}$ of two-sided error.
- Is it possible to obtain a tester with one-sided error?
  - ▷ Yes! Only an additional factor k is required.
    - **The key idea:** Use the size of a matching of a graph as a *lower bound* on the size of a vertex cover.

A simple observation

Observation

In the sparse model,

- $G = (V, E)$ *satisfies* $\mathcal{P}_{VC \le k}$ $\Rightarrow$ $|E| \le kd$.
- $G = (V, E)$ *is* ϵ*-far from* $\mathcal{P}_{VC \le k}$ $\Rightarrow$ $|E| \ge \epsilon d n$.

- This leads to an $O(d/\epsilon)$ tester for $\mathcal{P}_{VC \le k}$ of two-sided error.
- Is it possible to obtain a tester with one-sided error?
  - ▷ Yes! Only an additional factor k is required.
  - **The key idea:** Use the size of a matching of a graph as a *lower bound* on the size of a vertex cover.

A simple observation

Observation

In the sparse model,

- $G = (V, E)$ *satisfies* $\mathcal{P}_{VC \le k} \;\Rightarrow\; |E| \le kd$.
- $G = (V, E)$ *is* ϵ*-far from* $\mathcal{P}_{VC \le k} \;\Rightarrow\; |E| \ge \epsilon d n$.

- This leads to an $O(d/\epsilon)$ tester for $\mathcal{P}_{VC \le k}$ of two-sided error.
- Is it possible to obtain a tester with one-sided error?
  - ▷ Yes! Only an additional factor k is required.
  - **The key idea:** Use the size of a matching of a graph as a *lower bound* on the size of a vertex cover.

A simple observation

Observation

In the sparse model,

- $G = (V, E)$ *satisfies* $\mathcal{P}_{VC \leq k} \quad \Rightarrow \quad |E| \leq kd$.
- $G = (V, E)$ *is* ϵ*-far from* $\mathcal{P}_{VC \leq k} \quad \Rightarrow \quad |E| \geq \epsilon d n$.

- This leads to an $O(d/\epsilon)$ tester for $\mathcal{P}_{VC \leq k}$ of two-sided error.
- Is it possible to obtain a tester with one-sided error?
  - ▷ Yes! Only an additional factor k is required.
  - **The key idea:** Use the size of a matching of a graph as a *lower bound* on the size of a vertex cover.

An $O(kd/\epsilon)$ tester with one-sided error

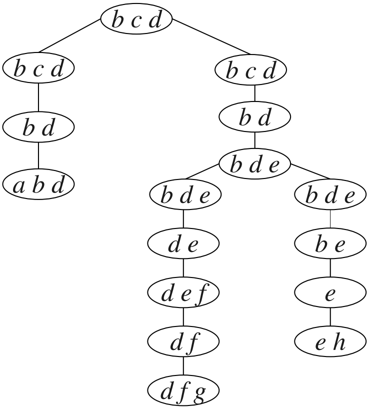
```
VC-FPT-Tester(G, k)
1. t ← 0; /* t: the size of the found matching */
2. repeat /* Sampling disjoint edges */
   a. choose a vertex v ∈ V uniformly at random;
   b. if v is marked then continue;
   c. for i ← 1 to d do
        if f_G(v, i) ≠ ∅ and f_G(v, i) is not marked then
          i.  t ← t + 1 and mark i and f_G(v, i);
          ii. break; /* exit the for-loop */
        end if
   d. end for
3. until ⌈10k/ϵ⌉ times
4. if t ≥ k + 1 then return "no";
5. else return "yes";
```

- $\mathbf{E}[\#\text{ iterations for } t \geq k+1 \mid G \text{ is } \epsilon\text{-far from } \mathcal{P}_{VC_{\leq k}}] \leq (2k+1)/\epsilon$.

Parameterized property testing for $\mathcal{P}_{tw \leq k}$

Nice tree-decomposition

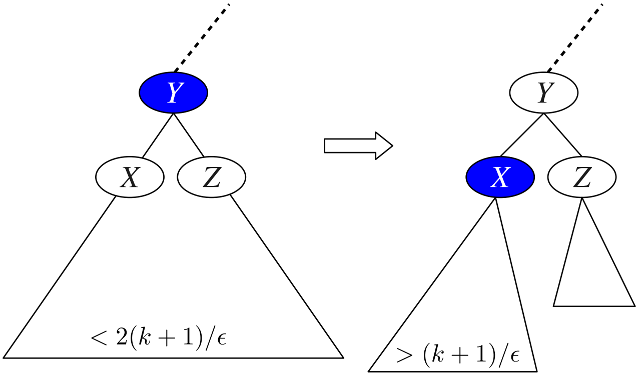
- Every graph with treewidth k has a nice tree-decomposition of width k. [Kloks 1994]



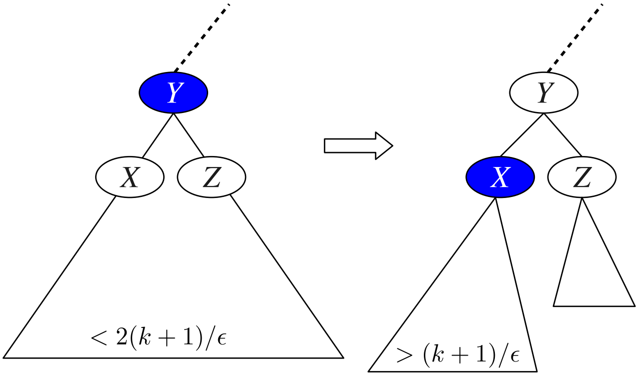
Proposition 1

Let $G = (V, E)$ be a graph with $tw(G) \leq k$. Then for any $\epsilon \in (0, 1)$, there is a set $U \subseteq V$ such that

- $|U| \leq \epsilon n$
- *each connected component of $G - U$ is of size $\leq 2(k + 1)/\epsilon$.*



- $|U| \leq \frac{n}{(k+1)/\epsilon} \cdot (k+1) \leq \epsilon n.$



- $|U| \leq \frac{n}{(k+1)/\epsilon} \cdot (k+1) \leq \epsilon n.$

The general idea

Apply the ideas in [Hassidim *et al.* 2009] & [Onak 2010].

- Construct an oracle $\mathcal{O}$ that determines U and the resulting connected components.

Proposition 2

There exists an oracle $\mathcal{O}$ such that

$G \in \mathcal{P}_{tw \leq k} \Rightarrow$ with prob. $\geq 82/90$, it computes $U \subseteq V$ with $|U| \leq \epsilon n/4$ such that removing U results in connected components of size bounded by $O(kd^3/\epsilon^2)$.

⋆ Computation for each query to the oracle $\mathcal{O}$:

- $2^{d^{O(kd^3/\epsilon^2)}}$ (extending the result in [Hassidim *et al.* 2009])
- $d^{(k/\epsilon)^{O(k^2)}}$ (extending the result in [Onak 2010])

The general idea

Apply the ideas in [Hassidim *et al.* 2009] & [Onak 2010].

- Construct an oracle $\mathcal{O}$ that determines U and the resulting connected components.

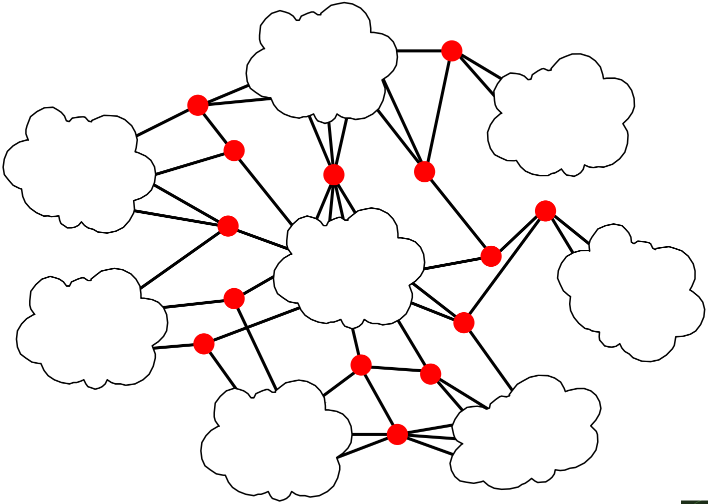
Proposition 2

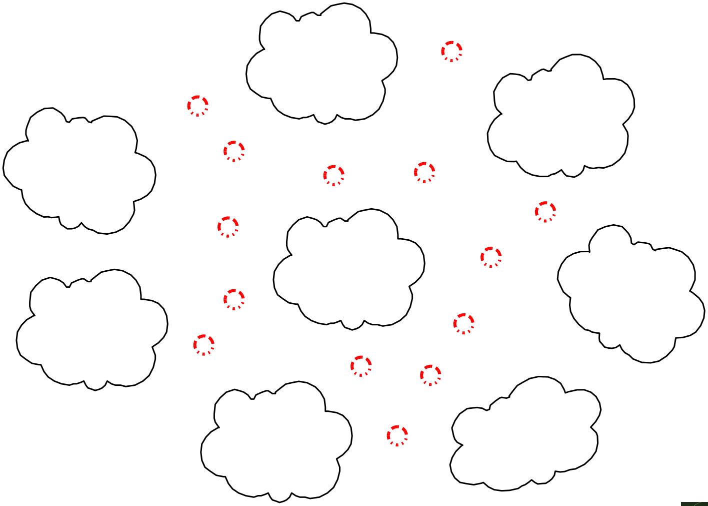
There exists an oracle $\mathcal{O}$ such that

$G \in \mathcal{P}_{tw \leq k} \Rightarrow$ with prob. $\geq 82/90$, it computes $U \subseteq V$ with $|U| \leq \epsilon n/4$ such that removing U results in connected components of size bounded by $O(kd^3/\epsilon^2)$.

⋆ Computation for each query to the oracle $\mathcal{O}$:

- $2^{d^{O(kd^3/\epsilon^2)}}$ (extending the result in [Hassidim *et al.* 2009])
- $d^{(k/\epsilon)^{O(k^2)}}$ (extending the result in [Onak 2010])

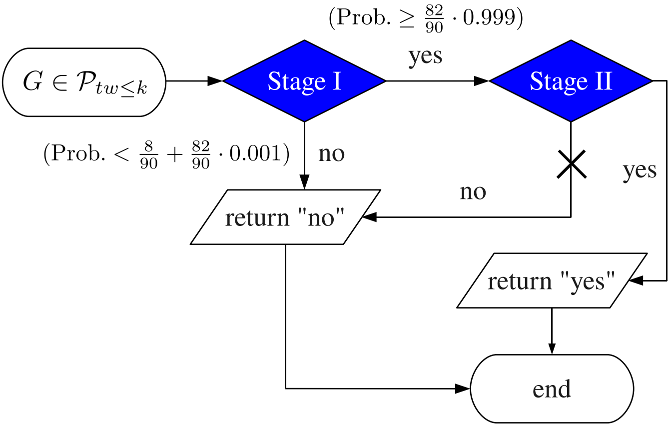




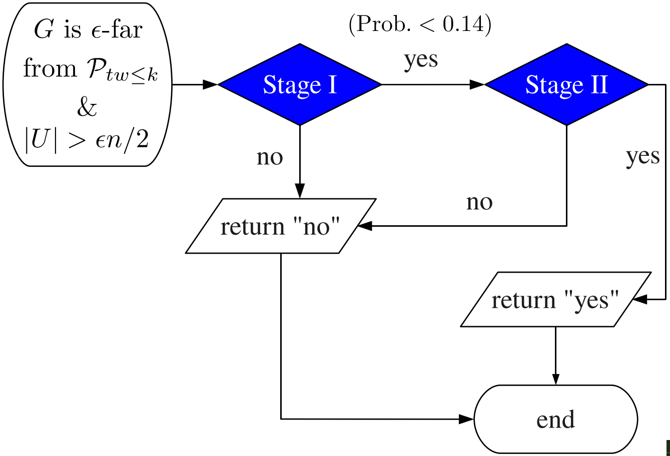
The general idea (contd.)

- Then, test if the size of U is bounded by $\epsilon n/4$ (Stage I).
  - Randomly select $256/\epsilon^2$ vertices and use the oracle $\mathcal{O}$ to compute the number of sampled vertices that belong to U.
  - ▷ Return "no" if the above number is more than $96/\epsilon$.
- Otherwise, randomly select $4/\epsilon$ vertices from the rest of the graph to check if any of them belongs to a connected component of treewidth greater than k (Stage II).

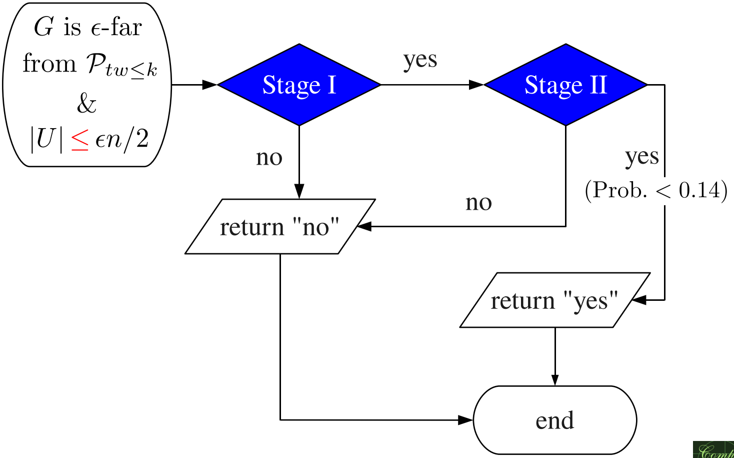
A sketch by the flow charts



A sketch by the flow charts



A sketch by the flow charts



Thanks for your attention.

Appendix

A simple error reduction method: Majority votes

- Assume that we have:
  - A property tester $\mathcal{A}$ for property $\mathcal{P}$
  - Input: I

such that

  - $\mathbf{Pr}[\mathcal{A} \text{ answers "yes"} \mid I \text{ satisfies } \mathcal{P}] = \frac{2}{3}$;
  - $\mathbf{Pr}[\mathcal{A} \text{ answers "no"} \mid I \text{ is } \epsilon\text{-far from } \mathcal{P}] = \frac{2}{3}$;

A simple error reduction method: Majority votes (contd.)

- An enhanced property tester $\mathcal{A}'$:
  - Run $\mathcal{A}$ for **three** times independently;
  - If $\mathcal{A}$ answers "yes" at least twice then answer "yes"; Otherwise, answer "no".
- We then have
  - $\mathbf{Pr}[\mathcal{A}' \text{ answers "yes"} \mid I \text{ satisfies } \mathcal{P}] = \binom{3}{2}\left(\frac{2}{3}\right)^2 + \binom{3}{3}\left(\frac{2}{3}\right)^3 = \frac{20}{27} > 0.74.$
  - $\mathbf{Pr}[\mathcal{A}' \text{ answers "no"} \mid I \text{ is } \epsilon\text{-far from } \mathcal{P}] = \binom{3}{2}\left(\frac{2}{3}\right)^2 + \binom{3}{3}\left(\frac{2}{3}\right)^3 > 0.74.$

Back

A simple $\mathcal{P}_{VC\leq k}$ tester with two-sided error

`Simple-VC-Tester`(G)

1. **repeat**
   a. choose a vertex $v \in V$ uniformly at random;
   b. **for** $i \leftarrow 1$ to d **do**
      i. **if** $f_G(v, i) \neq \emptyset$ **return** "no";
   c. **end for**
2. **until** $2/\epsilon$ times;
3. **return** "yes";

A simple $\mathcal{P}_{VC \leq k}$ tester with two-sided error (contd.)

- G satisfies $\mathcal{P}_{VC \leq k}$:
  - The algorithm returns "no" with probability $\leq 2kd/n < 1/3$ ($\because$ n is sufficiently large).
- G is ϵ-far from $\mathcal{P}_{VC \leq k}$:
  - The algorithm returns "yes" with probability $\leq (1 - \epsilon dn/dn)^{2/\epsilon} = (1 - \epsilon)^{2/\epsilon} < e^{-2} < 1/3$.

Back

- Consider the case that G is ϵ-far from $\mathcal{P}_{VC\leq k}$.
  - $|E(G)| \geq \epsilon dn$.
- We may assume that $k < \epsilon n/4$.
- A_i: be the number of finished iterations of the loop such that i disjoint edges are found.
- X_i : $A_i - A_{i-1}$. Hence, $A_i = \sum_{j=0}^{i} X_j$, where $X_0 = A_0 = 0$.
- Y_i: the event that a new edge is found whose endpoints are not in the previous found $i-1$ disjoint edges.
  - $\mathbf{Pr}[Y_1] \geq \epsilon n/n = \epsilon$. $\mathbf{Pr}[Y_2] \geq (\epsilon n - 2)/n = \epsilon - 2/n$.
  - Similarly, $\mathbf{Pr}[Y_i] \geq (\epsilon n - 2(i-1))/n = \epsilon - 2(i-1)/n$.

- Thus, $\mathbf{E}[X_i] \leq 1/(\epsilon - 2(i-1)/n)$, and then we obtain that

$$\begin{aligned}\mathbf{E}[A_{k+1}] &\leq \frac{1}{\epsilon} + \frac{1}{\epsilon - 2/n} + \ldots + \frac{1}{\epsilon - 2k/n} \\ &\leq \frac{1}{\epsilon} + \frac{k}{\epsilon - 2k/n} \\ &< \frac{1}{\epsilon} + \frac{k}{\epsilon - \epsilon/2} \quad (\because k < \epsilon n/4) \\ &\leq \frac{2k+1}{\epsilon}.\end{aligned}$$

- Thus, the probability that Algorithm `VC-FPT-Tester` returns “yes” is

$$\mathbf{Pr}\left[A_{k+1} > \left\lceil \frac{10k}{\epsilon} \right\rceil\right] \leq \mathbf{Pr}\left[A_{k+1} \geq \frac{10k}{\epsilon}\right] \leq \frac{(2k+1)/\epsilon}{10k/\epsilon} \leq \frac{3k}{10k} < \frac{1}{3},$$

where the second inequality follows by Markov’s inequality.

Back

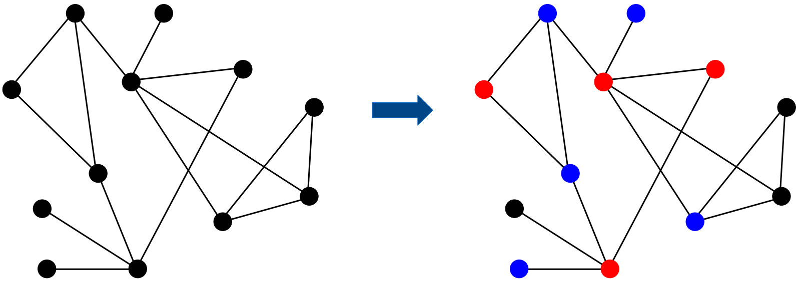
Sketch of the proof of Proposition 2

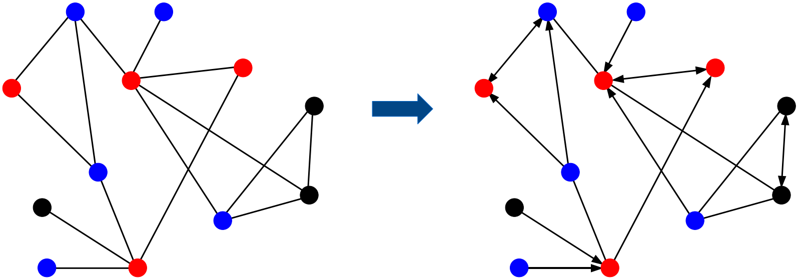
Fact 6.1 (Dujmović & Wood 2007 + Nash-Williams 1964)

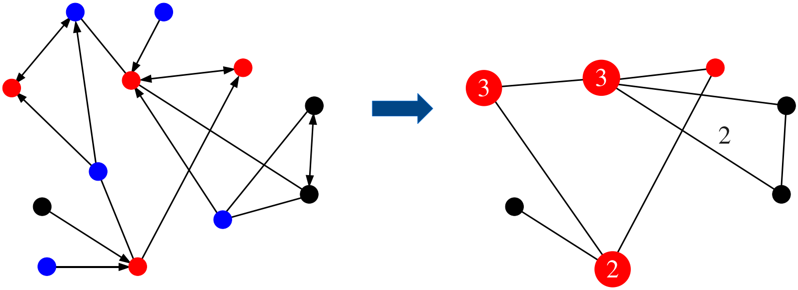
For every finite graph $G = (V, E) \in \mathcal{P}_{tw \leq k}$, E can be partitioned into $\leq k$ forests.

- $G = (V, E, w) \in \mathcal{P}_{tw \leq k}$: each edge has integer weight ≥ 1.
- W: the sum of all the edge weights.

$\star$ At least one of these forests has weight $\geq W/k$.






3
3
2
2

Sketch of the proof of Proposition 2 (contd.)

- The above process contracts edges with total weight $\geq (W/2k)/9 = W/18k$ expectedly.
- With probability $\geq 82/90$, the sum of the remaining edges becomes $\leq \epsilon n$ after running the process for $\Omega(k \log_{1+1/(36k-1)}(k/\epsilon))$ times.
- The diameter of each derived component: $(k/\epsilon)^{O(k^2)}$.
- The total amount of computation for each vertex v: $d^{(k/\epsilon)^{O(k^2)}}$.
  - The computation can be simulated locally by running the algorithm on the subgraph induced by the vertices of distance $(k/\epsilon)^{O(k^2)}$ from v [cf. Parnas & Ron 2007].

Back to Proposition 2

Tail probabilities of the testing of $|U|$

The case: $|U| \leq \frac{\epsilon n}{4}$.

- $\star$ f: the number of sampled vertices that belong to U (computed by the oracle $\mathcal{O}$).
- $\mu = \mathbf{E}[f] \leq \frac{\epsilon}{4} \cdot \frac{256}{\epsilon^2} = \frac{64}{\epsilon}$.

  By the Chernoff bounds, we have

$$
\begin{aligned}
\mathbf{Pr}\left[f \geq \frac{96}{\epsilon}\right] &\leq \mathbf{Pr}\left[f \geq \left(1 + \frac{1}{2}\right) \cdot \mu\right] \\
&\leq \left(\frac{e^{1/2}}{(1+1/2)^{(1+1/2)}}\right)^{64/\epsilon} \\
&< 0.001.
\end{aligned}
$$

Back to the general idea of treewidth testing

Tail probabilities of the testing of $|U|$

The case: $|U| > \frac{\epsilon n}{2}$.

- $\mu' = \mathbf{E}[f] > \frac{\epsilon}{2} \cdot \frac{256}{\epsilon^2} = \frac{128}{\epsilon}$.

  By the Chernoff bounds, we have

$$\begin{aligned}
\mathbf{Pr}\left[f < \frac{96}{\epsilon}\right] &= \mathbf{Pr}\left[f < \frac{3}{4} \cdot \frac{128}{\epsilon}\right] \\
&\leq \mathbf{Pr}\left[f \leq \left(1 - \frac{1}{4}\right) \mu'\right] \\
&\leq e^{-\mu' \cdot \left(\frac{1}{4}\right)^2 \cdot \frac{1}{2}} \\
&< 0.14,
\end{aligned}$$

Back to the general idea of treewidth testing

The testing in Stage II

The case: G is ϵ-far from $\mathcal{P}_{tw \leq k}$ & $|U| \leq \frac{\epsilon n}{2}$.

- One still has to remove $\geq \epsilon dn - \epsilon dn/2$ edges from $G - U$ to make it satisfy $\mathcal{P}_{tw \leq k}$.
- $\geq \epsilon n/2$ vertices belong to components of treewidth $> k$.
- "No such a vertex is selected" $\Rightarrow$ the probability is less than $< (1 - \epsilon/2)^{4/\epsilon} < e^{-2} < 0.14$.

Back to the general idea of treewidth testing

The tail probabilities

Markov's inequality

Let X be a nonnegative random variable. Then for any $a > 0$,

$$\mathbf{Pr}[X \geq a] \leq \frac{\mathbf{E}[X]}{a}.$$

The tail probabilities (contd.)

Chernoff bounds

Let $X_1, \ldots, X_n$ be mutually independent 0–1 random variables such that $\mathbf{Pr}[X_i] = p_i$. Let $S = \sum_{i=1}^{n} X_i$ and $\mu = \mathbf{E}[S]$. Then the following inequalities holds.

- for any $\delta > 0$,
$$\mathbf{Pr}[S \geq (1+\delta)\mu] \leq \left(\frac{e^{\delta}}{(1+\delta)^{(1+\delta)}}\right)^{\mu};$$
- for any $0 < \delta < 1$,
$$\mathbf{Pr}[S \geq (1+\delta)\mu] \leq e^{-\mu\delta^2/3},$$
$$\mathbf{Pr}[S \leq (1-\delta)\mu] \leq e^{-\mu\delta^2/2}.$$

Back to tail probabilities of testing of $|U|$