# A Study on Property Testing

*Ph.D. Dissertation Proposal of*

Joseph, Chuang-Chieh Lin

Supervisor: Professor Maw-Shang Chang

Computation Theory Laboratory
Department of Computer Science and Information Engineering
National Chung Cheng University, Taiwan

February 12, 2009

# Objectives of the dissertation

- Evolutionary tree reconstruction property testing:
    - **Testing quartet consistency.**

2. Graph property testing:
    a). **Testing if a graph is induced $P_4$-free.**
    b). **Testing if a graph is induced $C_4$-free.**

# Quartet consistency

**Property (Quartet consistency)**

**Input:** *A complete set of quartet topologies $Q$*

**Property:** *tree-consistency*

- Jobs:
  - Give a property tester for this property [preliminary results].
  - Prove its testability.

# Induced $C_4$-freeness & induced $P_4$-freeness

**Property (Induced $C_4$-freeness)**

**Input:** *A dense graph G represented by an adjacency-matrix*
**Property:** *Having no $C_4$ as an induced-subgraph*

- Jobs:
  - Give a property tester for this property.
  - Show that it is easily testable or not.

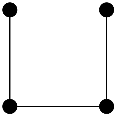
**Property (Induced $P_4$-freeness)**

**Input:** *A dense graph G represented by an adjacency-matrix*
**Property:** *Having no $P_4$ as an induced-subgraph*

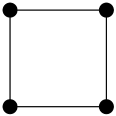
- Jobs:
  - Give a property tester for this property.
  - Show that it is easily testable or not.

**Open problems (Alon and Shapira; 2006)**

Is induced $C_4$-freeness **easily testable** in dense graphs?
Is induced $P_4$-freeness **easily testable** in dense graphs?



$P_4$



$C_4$

# Outline

1. Objectives of the dissertation
2. Brief introduction to property testing
   - Background of property testing
   - Previous results on testing graph properties
3. Preliminary results on quartet consistency

# Background of property testing

- In the real world nowadays, we are faced with imperious need to process increasing larger amounts of data in faster times.
- Many practical problems have inputs of very large size.
- Sometimes it is not realistic to solve a problem in the time even linear in the input size.
- Property testing is one of the possible approaches faster than linear time algorithms.

# Background of property testing (contd.)

- Try to answer “yes” or “no” for the following *relaxed* decision problems by observing only a small fraction of the input.
  - Does the input satisfy a designated property, or
  - is far from satisfying the property?

# Background of property testing (contd.)

- The general notion: Rubinfeld & Sudan [SIAM J. Comput. 1996].

Ronitt Rubinfeld

Madhu Sudan

- Testing *combinatorial objects*: Goldreich, Goldwasser, & Ron [J. ACM 1998].

Oded Goldreich

Shafi Goldwasser

Dana Ron

- Motivation: program checking.

# Background of property testing (contd.)

- The general notion: Rubinfeld & Sudan [SIAM J. Comput. 1996].


Ronitt Rubinfeld


Madhu Sudan

- Testing *combinatorial objects*: Goldreich, Goldwasser, & Ron [J. ACM 1998].


Oded Goldreich


Shafi Goldwasser


Dana Ron

- Motivation: program checking.

# Background of property testing (contd.)

- A program $M$: compute a function $f$.
  - Check if $M$ gives correct answers on most inputs of $f$.

Extension:

- Given a family of functions $\mathcal{F}$ over a domain $\mathcal{D}$ (e.g., all *linear* functions) and a program $M$.
  - Test if $\exists f \in \mathcal{F}$ such that $M$ has the same outputs as $f$ for most points of $\mathcal{D}$.

$M$ is regarded as a *black box*.

# Background of property testing (contd.)

- In property testing, we use $\epsilon$-far to say that the input is far from a certain property.
- $\epsilon$: the least fraction of the input needs to be modified.
- For example:
  - A sequence of integers $L = (0, 2, 3, 4, 1)$.
  - Allowed operations: integer deletions
  - $L$ is 0.2-far from being monotonically nondecreasing.

# Background of property testing (contd.)

- In property testing, we use $\epsilon$-far to say that the input is far from a certain property.
- $\epsilon$: the least fraction of the input needs to be modified.
- For example:
  - A sequence of integers $L = (0, 2, 3, 4, 1)$.
  - Allowed operations: integer deletions
  - $L$ is 0.2-far from being monotonically nondecreasing.

# Background of property testing (contd.)

- The commonly used complexity measure: queries.
- A query is like to probe, which is to examine certain value of the input.
  - to see if two vertices in a graph are adjacent under the adjacency-matrix model;
  - to know the $i$th neighbor of a vertex in a bounded-degree graph under the incidence-list model;
  - ...
- In property testing, the query complexity (say $q(n, \epsilon)$) is asked to be sublinear **in the input size** (say $f(n)$).
  - $q(n, \epsilon) = o(f(n))$ if $\lim_{n \to \infty} \frac{q(n,\epsilon)}{f(n)} \to 0$, where $\epsilon$ is viewed as a constant.

# Background of property testing (contd.)

- The commonly used complexity measure: queries.
- A query is like to probe, which is to examine certain value of the input.
  - to see if two vertices in a graph are adjacent under the adjacency-matrix model;
  - to know the $i$th neighbor of a vertex in a bounded-degree graph under the incidence-list model;
  - ...
- In property testing, the query complexity (say $q(n, \epsilon)$) is asked to be sublinear **in the input size** (say $f(n)$).
  - $q(n, \epsilon) = o(f(n))$ if $\lim_{n \to \infty} \frac{q(n,\epsilon)}{f(n)} \to 0$, where $\epsilon$ is viewed as a constant.

# Property testers

- A property tester for $\mathbb{P}$ is an algorithm utilizing sublinear queries such that:
  - $\triangleright$ if the input satisfies $\mathbb{P}$:
    answers "yes" with probability $\geq 2/3$ ($1 \longrightarrow$ one-sided error);
  - $\triangleright$ if the input is $\epsilon$-far from satisfying $\mathbb{P}$:
    answers "no" with probability $\geq 2/3$.

# Testabilities

- $\mathbb{P}$ is testable if
  - $\exists$ a property tester for $\mathbb{P}$ such that its query complexity is independent of the input size.
- $\mathbb{P}$ is easily testable if
  - $\exists$ a property tester of **one-sided error** for $\mathbb{P}$ such that its query complexity is $\mathbf{poly(1/\epsilon)}$.

# Testabilities

- $\mathbb{P}$ is testable if
  - $\exists$ a property tester for $\mathbb{P}$ such that its query complexity is independent of the input size.
- $\mathbb{P}$ is easily testable if
  - $\exists$ a property tester of **one-sided error** for $\mathbb{P}$ such that its query complexity is $\mathbf{poly(1/\epsilon)}$.

# An easy example of property testers

- Testing if a graph is empty
  - i.e., testing if a graph is induced $P_2$-free.
- Assume that the adjacency-matrix model is used to represent the input graph.
  - $O(1/\epsilon)$ queries are enough.
  - How can it be done?

# An easy example of property testers

- Testing if a graph is empty
  - i.e., testing if a graph is induced $P_2$-free.
- Assume that the adjacency-matrix model is used to represent the input graph.
  - $O(1/\epsilon)$ queries are enough.
  - How can it be done?

# An easy example of property testers

- Testing if a graph is empty
  - i.e., testing if a graph is induced $P_2$-free.
- Assume that the adjacency-matrix model is used to represent the input graph.
  - $O(1/\epsilon)$ queries are enough.
  - How can it be done?

# An easy example of property testers (contd.)

- $\epsilon$-far from being empty:
  - $\epsilon n^2$ pairs of vertices are *adjacent*.
- A property tester, say $\mathcal{A}$, works as follows.
  - Repeatedly, for $1/\epsilon$ times, pick two vertices uniformly at random and check if they are adjacent.
  - Once an edge is found, return "no",
  - otherwise (i.e., all of the chosen pairs of vertices are not adjacent) return "yes".

# An easy example of property testers (contd.)

- $\epsilon$-far from being empty:
  - $\epsilon n^2$ pairs of vertices are *adjacent*.
- A property tester, say $\mathcal{A}$, works as follows.
  - Repeatedly, for $1/\epsilon$ times, pick two vertices uniformly at random and check if they are adjacent.
  - Once an edge is found, return "no",
  - otherwise (i.e., all of the chosen pairs of vertices are not adjacent) return "yes".

# An easy example of property testers (contd.)

- $\epsilon$-far from being empty:
  - $\epsilon n^2$ pairs of vertices are *adjacent*.
- A property tester, say $\mathcal{A}$, works as follows.
  - Repeatedly, for $1/\epsilon$ times, pick two vertices uniformly at random and check if they are adjacent.
  - Once an edge is found, return "no",
  - otherwise (i.e., all of the chosen pairs of vertices are not adjacent) return "yes".

# An easy example of property testers (contd.)

- $\mathbf{Pr}[\mathcal{A}$ returns "yes" $|$G is empty $] = 1$.
- $\mathbf{Pr}[\mathcal{A}$ returns "yes" $|$G is $\epsilon$-far from being empty$] = (1 - \epsilon n^2/\binom{n}{2})^{1/\epsilon} < (1 - 2\epsilon)^{1/\epsilon} < e^{-2} < 1/3$.

# An easy example of property testers (contd.)

- $\mathbf{Pr}[\mathcal{A}$ returns "yes" $|$G is empty $] = 1$.
- $\mathbf{Pr}[\mathcal{A}$ returns "yes" $|$G is $\epsilon$-far from being empty$] = (1 - \epsilon n^2/\binom{n}{2})^{1/\epsilon} < (1-2\epsilon)^{1/\epsilon} < e^{-2} < 1/3$.

# Two commonly used models for graph property testing

# The model for dense graphs

- Graph representation: adjacency-matrix for a graph $G = (V, E)$.
    - undirected, no self-loops, $\leq 1$ edge between any $u, v \in V$.
    - $|V| = n$ vertices and $|E| = \Omega(n^2)$ edges.
    - A query: to see if two vertices $u$ and $v$ are adjacent or not.
- $\epsilon$-far from satisfying $\mathbb{P}$:
    - $\geq \epsilon n^2$ edges should be deleted or added to make $G$ satisfy $\mathbb{P}$.

# The model for sparse graphs

- Graph representation: incidence-list for a graph $G = (V, E)$ with bounded degree $d$.
  - undirected, no self-loops, $\leq 1$ edge between any $u, v \in V$.
  - $|V| = n$ vertices and $|E| = O(dn)$ edges.
  - A query: to see who is the $i$th neighbor of $v$.
- $\epsilon$-far from satisfying $\mathbb{P}$:
  - $\geq \epsilon dn$ edges should be deleted or added to make $G$ satisfy $\mathbb{P}$.

# Some important families of graph properties

# Some graph properties

△ Hereditary graph properties:

- closed under removal of vertices (taking induced subgraphs).

⋆ Monotone graph properties:

- closed under removal of vertices *and edges* (taking subgraphs).

---

△ $\mathbb{P}^*_H$: the property that a graph having no $H$ as an **induced** subgraph.

⋆ $\mathbb{P}_H$: the property that a graph having no $H$ as a subgraph.

# Previous results on testing graph properties in **dense** undirected graphs

| Property | Tester | Testable | Easily testable | Query |
|---|---|---|---|---|
| First-order graph properties without a quantifier alternation of type '$\forall\exists$' | Yes | Yes | No | $\star$ |
| First-order graph properties with a quantifier alternation of type '$\forall\exists$' | – | No | No | – |
| Monotone properties | Yes | Yes | No | $\star$ |
| Hereditary properties | Yes | Yes | No | $\star$ |

Table: '$\star$' stands for the bounds of the type towers of towers of exponents of height poly$(1/\epsilon)$; '–' means no explicit bound (or tester) is given.

$2^{2^{2^{2^{2}}}}$ : tower of 2's of height 5

| Property | Tester | Testable | Easily testable | Query |
|---|---|---|---|---|
| Bipartiteness | Yes | Yes | Yes | $O(\frac{\ln^8(1/\epsilon)\ln\ln^2(1/\epsilon)}{\epsilon^2})$ |
| $k$-colorability | Yes | Yes | Yes | $O(\frac{k^2\ln^2 k}{\epsilon^4})$ |
| Having a clique of size $\geq \rho n$ | Yes | Yes | No* | $O(\frac{\log^2(1/\epsilon)\rho^2}{\epsilon^6})$ |
| Having a cut of size $\geq \rho n^2$ | Yes | Yes | No* | $O(\frac{\log^2(1/\epsilon)}{\epsilon^7})$ |

Table: '*' stands for that only two-sided error property testers can be obtained.

| Property | Tester | Testable | Easily testable | Query |
|---|---|---|---|---|
| $\mathbb{P}_H$, $H$ is bipartite | Yes | Yes | Yes | $O(h^2 \left(\frac{1}{2\epsilon}\right)^{h^2/4})$ |
| $\mathbb{P}_H$, $H$ is not bipartite | Yes | Yes | No | $\Omega\left(\left(\frac{c}{\epsilon}\right)^{c\log(c/\epsilon)}\right)$ |
| $\mathbb{P}^*_H$, $H = P_2$ | Yes | Yes | Yes | $\Theta\left(\frac{1}{\epsilon}\right)$ |
| $\mathbb{P}^*_H$, $H = P_3$ | Yes | Yes | Yes | $O(\frac{\log(1/\epsilon)}{\epsilon})$ |
| $\mathbb{P}^*_H$, $H \neq P_2$, $P_3$, $P_4$, $C_4$ or their complements | Yes | Yes | No | $\Omega\left(\left(\frac{1}{\epsilon}\right)^{c\log(1/\epsilon)}\right)$ |
| $\mathbb{P}^*_H$, $H$ is $P_4$ | Yes | Yes | **?** | $\star$ |
| $\mathbb{P}^*_H$, $H$ is $C_4$ | Yes | Yes | **?** | $\star$ |

Table: '$\star$' stands for the bounds of the type towers of towers of exponents of height poly$(1/\epsilon)$; $c$ is a constant depending on $H$; '?' stands for an open question.

# Previous results on testing graph properties in **sparse** undirected graphs

| Property | Tester | Testable | Easily testable | Query |
|---|---|---|---|---|
| Hereditary properties in a hereditary and nonexpanding family of graphs | Yes | Yes | ? | $\star$ |
| Minor-closed properties | Yes | Yes | ? | $2^{2^{2^{\text{poly}(1/\epsilon)}}}$ |
| Bipartiteness | – | No | No | $\Omega(\sqrt{n})$ |
| Expansion | – | No | No | $\Omega(\sqrt{n})$ |
| 3-colorability | – | No | No | $\Omega(n)$ |

Table: '$\star$' stands for a bound in a not explicitly form yet it is independent of $n$; '?' stands for an open question; '–' means no explicit tester is given.

| Property | Tester | Testable | Easily testable | Query |
|---|---|---|---|---|
| Connectivity | Yes | Yes | Yes | $O(\frac{\log^2(1/\epsilon d)}{\epsilon})$ |
| $k$-edge-connectivity for $k = 1, 2$ | Yes | Yes | Yes | $O(\frac{\log^2(1/\epsilon d)}{\epsilon})$ |
| 3-edge-connectivity | Yes | Yes | Yes | $O(\frac{\log(1/\epsilon d)}{\epsilon^2 d})$ |
| $k$-edge-connectivity for $k \geq 4$ | Yes | Yes | Yes | $O(\frac{k^3 \log(1/(\epsilon d))}{\epsilon^{3-2/k} d^{2-2/k}})$ |
| Eulerian | Yes | Yes | Yes | $O(\frac{\log^2(1/\epsilon d)}{\epsilon})$ |
| Cycle-freeness | Yes | Yes | No | $O(\frac{1}{\epsilon^3})$* |

Table: '$\star$' stands for a bound in a not explicitly form yet it is independent of $n$; '*' stands for a result with two-sided error.

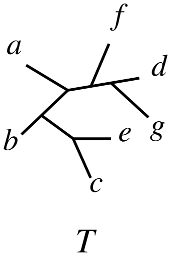
# As to our preliminary results..

| Property | Tester | Testable | Easily testable | Query |
|---|---|---|---|---|
| Quartet consistency | Yes | ? | ? | $O\left(\frac{n^3}{1-2(1-\epsilon)^{1/4}}\right)$ |

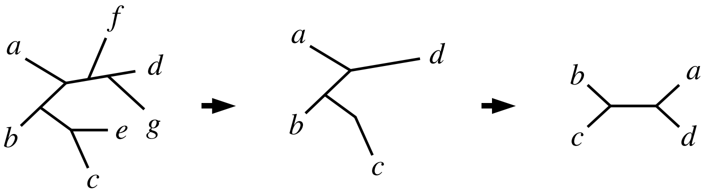
Table: Testing quartet consistency.

# Evolutionary trees

- $S$: a set of taxa; $|S| = n$.
- An evolutionary tree $T$ on $S$:
  - An *unrooted*, *leaf-labeled* tree
  - The leaves are bijectively labeled by the taxa in $S$
  - Each internal node has degree *three*



# Quartet topologies

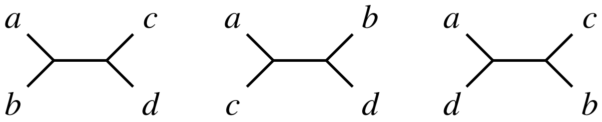



$T$

The path structure connecting $a, b, c, d$ in $T$

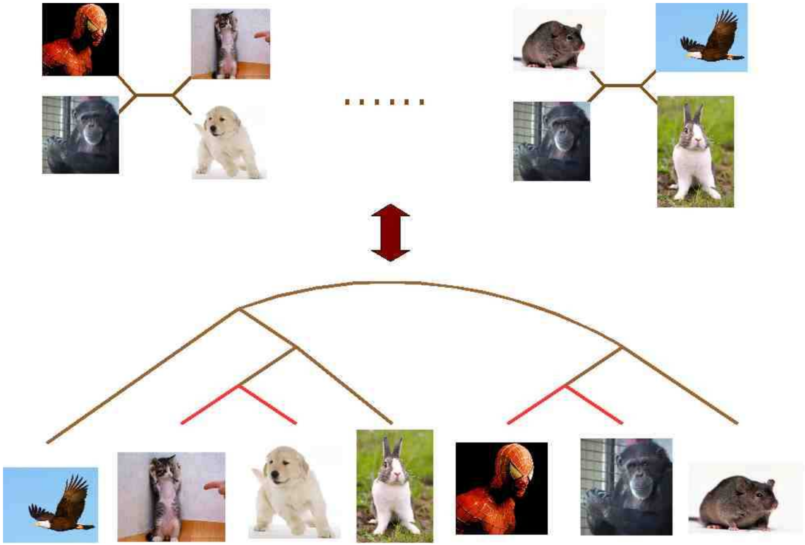
The topology of $\{a, b, c, d\}$

# Quartet topologies (contd.)




$[ab|cd]$ $[ac|bd]$ $[ad|bc]$

# Biological issue



# 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.

# 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.

# 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.

# 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.

# Quartet errors

- Given complete $Q$ and $Q^*$ (tree-like).
- **# quartet errors of** $Q$ **w.r.t.** $Q^*$:
  - $\delta(Q, Q^*)$.
- **# quartet errors of** $Q$:
  - $\Delta^*(Q) := \min\{\delta(Q, Q^*) : Q^* \text{ is tree-like}\}$.

# Quartet errors

- Given complete $Q$ and $Q^*$ (tree-like).
- **# quartet errors of $Q$ w.r.t. $Q^*$**:
  - $\delta(Q, Q^*)$.
- **# quartet errors of $Q$**:
  - $\Delta^*(Q) := \min\{\delta(Q, Q^*) : Q^* \text{ is tree-like}\}$.

# Quartet errors

- Given complete $Q$ and $Q^*$ (tree-like).
- **# quartet errors of** $Q$ **w.r.t.** $Q^*$:
  - $\delta(Q, Q^*)$.
- **# quartet errors of** $Q$:
  - $\Delta^*(Q) := \min\{\delta(Q, Q^*) : Q^* \text{ is tree-like}\}$.

## The parameterized MQI problem:

**Given:** a **complete** set of quartet topologies $Q$ and an integer $k$.

- The parameterized minimum quartet inconsistency problem:

> Determine whether there exists an evolutionary tree $T$ such that $\Delta(Q, Q_T) \leq k$.

- ⋆ **NP**-complete [Berry *et al.* 1999].
- ⋆ $O(4^k n + n^4)$ [Gramm and Niedermeier 2003].
- ⋆ $O^*(3.0446^k)$, $O^*(2.0162^k)$, and $O^*((1+\epsilon)^k)$ fixed-parameter algorithms [Chang, Lin, Rossmanith; IWPEC'08; to appear in *Theory of Computing Systems*].

# The parameterized MQI problem:

**Given:** a **complete** set of quartet topologies $Q$ and an integer $k$.

- The parameterized minimum quartet inconsistency problem:

> Determine whether there exists an evolutionary tree $T$ such that $\Delta(Q, Q_T) \leq k$.

- ⋆ **NP**-complete [Berry *et al.* 1999].
- ⋆ $O(4^k n + n^4)$ [Gramm and Niedermeier 2003].
- ⋆ $O^*(3.0446^k)$, $O^*(2.0162^k)$, and $O^*((1+\epsilon)^k)$ fixed-parameter algorithms [Chang, Lin, Rossmanith; IWPEC'08; to appear in *Theory of Computing Systems*].

# Related works (Constructing $T$ and QCP)

- Construct $T$ by a given tree-like $Q$:
  - ⋆ $O(n^4)$ [Berry and Gascuel 2000].
- The Quartet Compatibility Problem (QCP):

  > Determine whether there exists an evolutionary tree $T$ satisfying all quartet topologies in $Q$.

  - ⋆ **NP**-complete [Steel 1992].
  - ⋆ Polynomial time solvable if $Q$ is complete [Erdős et al. 1999].
- Consider the cases of **complete** $Q$.

# Related works (MQI & MQC)

Minimum Quartet Inconsistency Problem (MQI)

Construct an evolutionary tree $T$ s.t. $\Delta(Q, Q_T)$ is minimized.

- **NP**-hard [Berry *et al.* 1999].
- Approx. ratio: $O(n^2)$ [Jiang *et al.* 2000].
- $O(3^n n^4)$ dynamic programming [Ben-Dor *et al.* 1998].
- $O(n^4)$ if $\Delta^*(Q) < (n-3)/2$ [Berry *et al.* 1999].
- $O(n^5 + 2^{4c} n^{12c+2})$ if $\Delta^*(Q) < cn$ for some constant $c$ [Wu *et al.* 2006].

Maximum Quartet Consistency Problem (MQC)

Dual problem of MQI.

- **NP**-hard [Berry *et al.* 1999].
- PTAS [Jiang *et al.* 2001].

# Testing quartet consistency

- Now we consider property testing on the property that a complete $Q$ is tree-consistent (testing quartet consistency).
- The input size: $|Q| = \binom{n}{4}$.
- $Q$ is $\epsilon$-far from being tree-consistent: $Q$ is not tree-consistent unless at least $\epsilon n^4$ quartet topologies are changed.
- However, is it possible for $Q$ to have $\Omega(n^4)$ quartet errors?

# Testing quartet consistency

- Now we consider property testing on the property that a complete $Q$ is tree-consistent (testing quartet consistency).
- The input size: $|Q| = \binom{n}{4}$.
- $Q$ is $\epsilon$-far from being tree-consistent: $Q$ is not tree-consistent unless at least $\epsilon n^4$ quartet topologies are changed.
- However, is it possible for $Q$ to have $\Omega(n^4)$ quartet errors?

# Existence of $\Omega(n^4)$ quartet errors

- YES!

**Theorem (Chang, Lin, Rossmanith)**

*There exists a set of quartet topologies $Q$ which has $\Omega(n^4)$ quartet errors.*

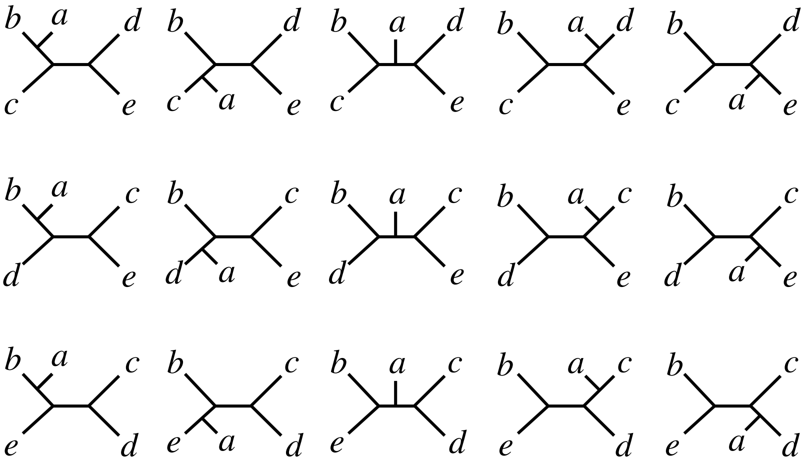
# Quintets

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

# Quintets

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

# Quintet topologies



## Consistent quintets

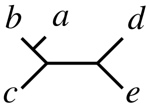
- What is a consistent quintet?

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

# Consistent quintets

- What is a consistent quintet?

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



# Tree consistency and quintets

**Theorem (Bandelt and Dress 1986)**

*$Q$ is tree-like $\Leftrightarrow$ every quintet containing $f$ is consistent.*

# The first property tester for quartet consistency

**Theorem (Chang, Lin, Rossmanith)**

*If $Q$ is $\epsilon$-far from satisfying quartet consistency, then there exist $\geq (1-2(1-\epsilon)^{1/4})n$ inconsistent quintets containing an arbitrary fixed taxon $f$.*

1. Pick an arbitrary taxon $f \in S$ and then repeat (a) and (b) for $\frac{2n^3}{1-2(1-\epsilon)^{1/4}}$ times.
   (a) Pick four taxa $s_1, s_2, s_3, s_4 \in S$ uniformly at random.
   (b) If the quintet $\{s_1, s_2, s_3, s_4, f\}$ is not consistent, then return "no".
2. Return "yes".

Table: Quartet Tester.

# The first property tester for quartet consistency (contd.)

**Theorem (Chang, Lin, Rossmanith)**

*Quartet Tester is a one-sided-error property tester for quartet consistency, which makes at most* $O\left(\frac{n^3}{1-2(1-\epsilon)^{1/4}}\right)$ *queries.*

- Our property tester is the first one for testing quartet consistency.
- Yet it is still open that whether this property is testable.

# Thank you!