Improved Quantum Algorithm for Triangle Finding via Combinatorial Arguments

François Le Gall

Department of Computer Science
Graduate School of Information Science and Technology
The University of Tokyo

QIP 2015
12 January 2015

Triangle Finding

three vertices u, v, w such that $(u, v) \in E$
$(u, w) \in E$
$(v, w) \in E$

unweighted (and undirected)

Given a graph G=(V,E), decide if it contains a triangle

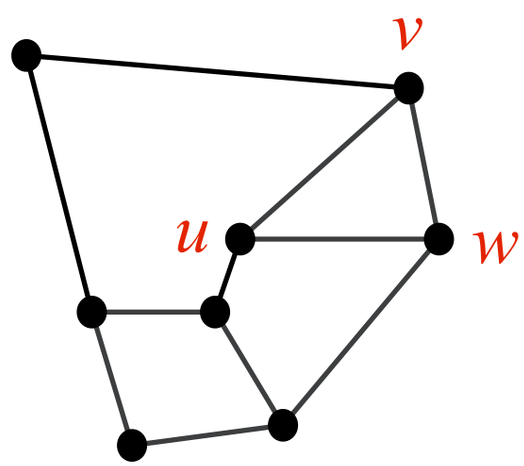
Triangle Finding

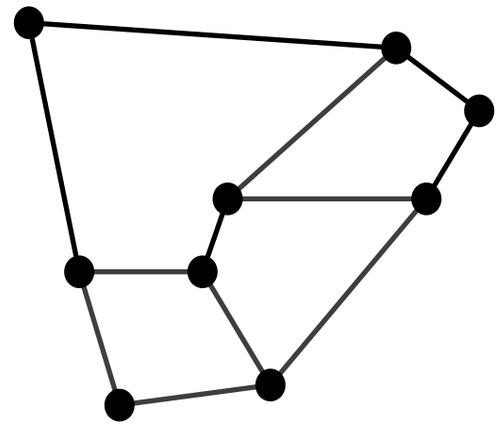
three vertices u, v, w such that $(u, v) \in E$
$(u, w) \in E$
$(v, w) \in E$

unweighted (and undirected)

Given a graph G=(V,E), decide if it contains a triangle

Examples:





no triangle

Triangle Finding

three vertices u, v, w such that $(u, v) \in E$
$(u, w) \in E$
$(v, w) \in E$

unweighted (and undirected)

Given a graph G=(V,E), decide if it contains a triangle

★ many algorithmic applications:

★ one of “most elementary” unsettled graph-theoretic problems

★ historically, the study of triangle finding has lead to the development of several quantum techniques (“showcase for new quantum techniques”)

Classical algorithms: trivial algorithm $O(n^3)$ time (here $n = |V|$)

try all $\binom{n}{3} = \Theta(n^3)$ triples of vertices

best known algorithm $O(n^{2.38})$ time

reduction to matrix multiplication

Trivial quantum algorithm: $\tilde{O}(n^{1.5})$ time by quantum search over triples of vertices

Query Complexity of Triangle Finding

undirected and unweighted

Given a graph G=(V,E), decide if it contains a triangle

query complexity of triangle finding

number of queries of the form "is (v_1,v_2) an edge?" needed to solve the problem

query complexity = number of queries made

time complexity = number of queries + number of other operations

$\binom{n}{2} = O(n^2)$ queries are obviously enough

trivial lower bounds

any classical algorithm requires $\Omega(n^2)$ queries

any quantum algorithm requires $\Omega(n)$ queries — best known lower bound

Quantum Query Complexity of Triangle Finding

Recent lower bound for triangle finding [Belovs, Rosmanis 13]:

any quantum algorithm based on (non-adaptive) learning graphs, or also solving weighted versions, must use $\Omega(n^{9/7}/\sqrt{\log n})$ queries

$\tilde{O}(n^{1.3})$-query quantum algorithm using quantum walks [Magniez, Santha, Szegedy 05]
(also solves weighted versions of triangle finding with the same complexity)

$O(n^{35/27})$-query quantum algorithm using learning graphs [Belovs 12]

35/27=1.296...

$O(n^{9/7})$-query quantum algorithm using learning graphs [Lee, Magniez, Santha 13]

same complexity also obtained by nested quantum walks [Jeffery, Kothari, Magniez 13]

9/7=1.285...

Our result: $\tilde{O}(n^{5/4})$-query quantum algorithm

Uses combinatorial properties of unweighted graphs, and (standard) quantum walks

Quantum Query Complexity of Triangle Finding

Recent lower bound for triangle finding [Belovs, Rosmanis 13]:

> any quantum algorithm based on (non-adaptive) learning graphs, or also solving weighted versions, must use $\Omega(n^{9/7}/\sqrt{\log n})$ queries

Our result proves that, in the quantum query complexity setting, unweighted triangle finding is easier than its weighted versions

results strongly suggesting similar separations are known in the classical time complexity setting

[Czumaj, Lingas 07] [Patrascu 10] [Vassilevska Williams, Williams 09&10]

Our result: $\tilde{O}(n^{5/4})$-query quantum algorithm

Uses combinatorial properties of unweighted graphs, and (standard) quantum walks

Quantum Walks for Graph Problems

Problem: Find a marked m-subset B of V

(m-subset: subset of size m; V: set of vertices of the graph; marked: satisfying some condition (e.g., contains an edge of a triangle))

Random sampling: $\frac{1}{\epsilon} \times \mathsf{C}$ queries

Grover search: $\frac{1}{\sqrt{\epsilon}} \times \mathsf{C}$ queries

$$\epsilon = \frac{\#\text{ marked } m\text{-subsets}}{\#\ m\text{-subsets}}$$

C: checking cost (checking if an m-subset B is marked)

Quantum walk search over the Johnson graph [Ambainis 04]

(each node of the Johnson graph represents an m-subset B)

perform the quantum walk while keeping a data structure $D(B)$ for the visited m-subset B (example: $D(B)$ = adjacency matrix of the graph induced by B)

$$\mathsf{S} + \frac{1}{\sqrt{\epsilon}} \left(\sqrt{m} \times \mathsf{U} + \mathsf{C} \right) \text{ queries}$$

C: checking cost (checking if B is marked given $D(B)$)

S: setup cost (creating the data structure)

U: update cost (updating the data structure)

Triangle Finding: Main

neighbors of u in the graph

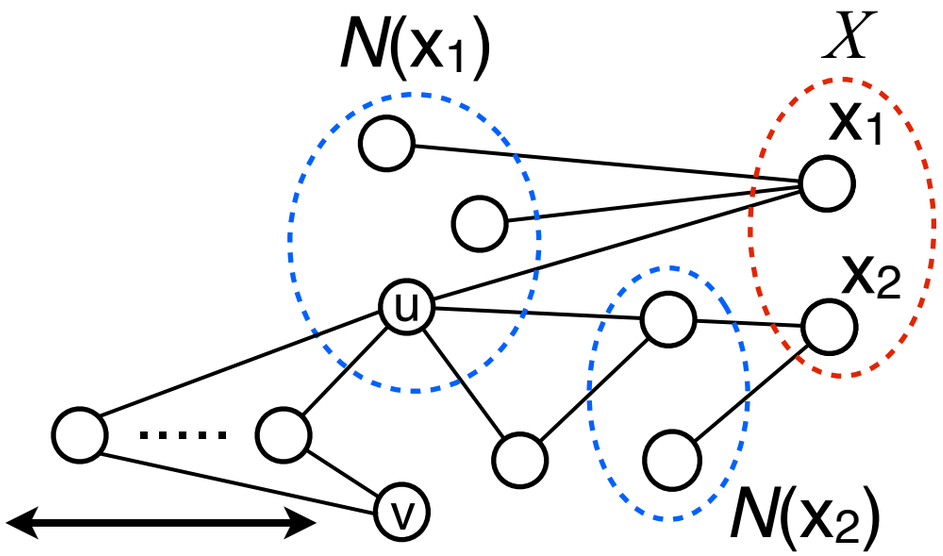
$(N(u) \times N(u)) \cap E = \emptyset$ for each $u \in X$

Take a random set $X \subseteq V$ of $\Theta(\sqrt{n} \log n)$ vertices

1. Check if the graph contains a triangle with a vertex in X

   $O(\sqrt{|X| \times |V \times V|}) = \tilde{O}(n^{5/4})$ queries using Grover search

2. If no triangle has been found, check if there exists a triangle with an edge in $(V \times V) \setminus S$, where $S = \bigcup_{u \in X} (N(u) \times N(u))$



at most $\sqrt{n}$ vertices

key property: sparsity

for each $(u, v) \in (V \times V) \setminus S$
there are (w.h.p) at most $\sqrt{n}$ vertices
$w \in V$ s.t. $(u, w) \in E$ and $(v, w) \in E$

instead of n^3

$\sum_{w \in V} |(N(w) \times N(w)) \setminus S| \leq n^{2.5}$ w.h.p.

Triangle Finding: Main Combinatorial Argument

Take a random set $X \subseteq V$ of $\Theta(\sqrt{n}\log n)$ vertices

1. Check if the graph contains a triangle with a vertex in X

   $O(\sqrt{|X| \times |V \times V|}) = \tilde{O}(n^{5/4})$ queries using Grover search

2. If no triangle has been found, check if there exists a triangle with an edge in $(V \times V) \setminus S$, where $S = \bigcup_{u \in X} (N(u) \times N(u))$

check if $\exists w \in V$ such that $(N(w) \times N(w)) \setminus S$ contains an edge

Step 2 can be implemented using $O\left(\sqrt{\sum_{w \in V} |(N(w) \times N(w)) \setminus S|}\right) = O(n^{5/4})$

queries by Grover search, if each set $(N(w) \times N(w)) \setminus S$ is known

→ $\sum_{w \in V} |(N(w) \times N(w)) \setminus S| \leq n^{2.5}$ w.h.p.

Problem with this strategy

Similar sparsity arguments have been used in prior works, with similar issues

- classical combinatorial algorithms for Boolean matrix multiplication [Bansal, Williams 09]
- quantum combinatorial algorithm for Triangle finding [Magniez, Santha, Szegedy 05]

Problems:

① Computing S requires $\Theta(|X| \times |V|) = \tilde{\Theta}(n^{1.5})$ queries
(we would like $\tilde{O}(n^{5/4})$ queries)

② Computing each $(N(w) \times N(w))$ requires $\Theta(|V|) = \Theta(n)$ queries
(we would like $\tilde{O}(n^{3/4})$ queries)

$$S = \bigcup_{u \in X} (N(u) \times N(u))$$

check if $\exists w \in V$ such that $(N(w) \times N(w)) \setminus S$ contains an edge

Step 2 can be implemented using $O\left(\sqrt{\sum_{w \in V} |(N(w) \times N(w)) \setminus S|}\right) = O(n^{5/4})$

queries by Grover search, if each set $(N(w) \times N(w)) \setminus S$ is known

→ $\sum_{w \in V} |(N(w) \times N(w)) \setminus S| \leq n^{2.5}$ w.h.p.

Problem with this strategy

Problems:

① Computing S requires $\Theta(|X| \times |V|) = \tilde{\Theta}(n^{1.5})$ queries
(we would like $\tilde{O}(n^{5/4})$ queries)

② Computing each $(N(w) \times N(w))$ requires $\Theta(|V|) = \Theta(n)$ queries
(we would like $\tilde{O}(n^{3/4})$ queries)

$$S = \bigcup_{u \in X} (N(u) \times N(u))$$

check if $\exists w \in V$ such that $(N(w) \times N(w)) \setminus S$ contains an edge

Step 2 can be implemented using $O\left(\sqrt{\sum_{w \in V} \left|(N(w) \times N(w)) \setminus S\right|}\right) = O(n^{5/4})$

queries by Grover search, if each set $(N(w) \times N(w)) \setminus S$ is known

Step 2 can be implemented, using quantum walks, without (completely) constructing these sets

Solution: our quantum algorithm

Current approach

Grover → check if $\exists w \in V$ such that

Grover ⟶ $(N(w) \times N(w)) \setminus S$ contains an edge

Problems for exploiting the sparsity:

① Computing S efficiently

② Computing each $(N(w) \times N(w))$ efficiently

First (incomplete) solution

Grover → check if $\exists w \in V$ such that

Q. walk ⟶ $\exists B \subseteq V$ of size $|B| = \sqrt{n}$ such that

with data structure

$D(B) = N(w) \cap B$

Grover ⟶ $((N(w) \cap B) \times (N(w) \cap B)) \setminus (S \cap (B \times B))$ contains an edge

known: this solves problem ②

problem ① remains: how to compute $S \cap (B \times B)$?

Final algorithm

Q. walk → check if $\exists A \subseteq V$ of size $|A| = n^{3/4}$ such that
with data structure
$D(A) = S \cap (A \times A)$
Grover → $\exists w \in V$ such that

Q. walk → $\exists B \subseteq A$ of size $|B| = \sqrt{n}$ such that
with data structure
$D(B) = N(w) \cap B$
Grover → $((N(w) \cap B) \times (N(w) \cap B)) \setminus (S \cap (B \times B))$ contains an edge

known

known, since
$S \cap (B \times B) \subseteq S \cap (A \times A)$
this solves problem ①

First (incomplete) solution

Grover → check if $\exists w \in V$ such that

Q. walk → $\exists B \subseteq V$ of size $|B| = \sqrt{n}$ such that
with data structure
$D(B) = N(w) \cap B$
Grover → $((N(w) \cap B) \times (N(w) \cap B)) \setminus (S \cap (B \times B))$ contains an edge

known: this solves problem ②
problem ① remains: how to compute $S \cap (B \times B)$?

Final algorithm

Q. walk → check if $\exists A \subseteq V$ of size $|A| = n^{3/4}$ such that

with data structure $D(A) = S \cap (A \times A)$

example: creating $D(A)$ uses $O(|X| \times |A|) = \tilde{O}(n^{5/4})$ queries

Grover → $\exists w \in V$ such that

Q. walk → $\exists B \subseteq A$ of size $|B| = \sqrt{n}$ such that

with data structure $D(B) = N(w) \cap B$

Grover → $((N(w) \cap B) \times (N(w) \cap B)) \setminus (S \cap (B \times B))$ contains an edge

known known

The overall query complexity of this 4-level algorithm is $\tilde{O}(n^{5/4})$

if all sets $((N(w) \cap B) \times (N(w) \cap B)) \setminus (S \cap (B \times B))$ are sparse

Technical difficulty:

Sparsity argument $\sum_{w \in V} |(N(w) \times N(w)) \setminus S| \leq n^{2.5}$ w.h.p.

Sparsity may not hold for each set $((N(w) \cap B) \times (N(w) \cap B)) \setminus (S \cap (B \times B))$

The same ideas still work by carefully defining the second walk and using "Grover search with variable checking costs ([Ambainis 08])" at the second level

Triangle Finding: Main Combinatorial Argument

Take a random set $X \subseteq V$ of $\Theta(\sqrt{n}\log n)$ vertices

1. Check if the graph contains a triangle with a vertex in X

   $O(\sqrt{|X| \times |V \times V|}) = \tilde{O}(n^{5/4})$ queries using Grover search

2. If no triangle has been found, check if there exists a triangle with an edge in $(V \times V) \setminus S$, where $S = \bigcup_{u \in X} (N(u) \times N(u))$

Step 2 can be implemented using $O\left(\sqrt{\sum_{w \in V} \left|(N(w) \times N(w)) \setminus S\right|}\right) = O(n^{5/4})$

queries by Grover search, if each set $(N(w) \times N(w)) \setminus S$ is known

Step 2 can be implemented using $\tilde{O}(n^{5/4})$ queries by our 4-level algorithm

Conclusion: Current Status of Triangle Finding

any quantum algorithm requires $\Omega(n)$ queries — best known lower bound

$\tilde{O}(n^{1.3})$-query quantum algorithm using quantum walks [Magniez, Santha, Szegedy 05]
(also solves weighted versions of triangle finding with the same complexity)

$O(n^{35/27})$-query quantum algorithm using learning graphs [Belovs 12]

35/27=1.296...

$O(n^{9/7})$-query quantum algorithm using learning graphs [Lee, Magniez, Santha 13]

9/7=1.285...

[Belovs, Rosmanis 13]:

$\Omega(n^{9/7}/\sqrt{\log n})$-query lower bound for weighted triangle finding and for the (non-adaptive) learning graphs approach

Our result: $\tilde{O}(n^{5/4})$-query quantum algorithm for unweighted triangle finding