Quantum Algorithms for Finding Constant-sized Sub-hypergraphs

Seiichiro Tani
(Joint work with François Le Gall and Harumichi Nishimura)

NTT Communication Science Labs., NTT Corporation, Japan.

The 20th International Computing and Combinatorics Conference

Outline

1. Introduction
2. Definition of Our Problem
3. Our Results
4. Technical Outlines
5. Conclusion

Graph Property Testing

Definition (Graph Property)

- Graph properties are those of graphs that are invariant under changing the labelings of vertices.
  (ex. connectedness, planarity)

Graph Property Testing

Definition (Graph Property)

- Graph properties are those of graphs that are invariant under changing the labelings of vertices.
  (ex. connectedness, planarity)
- If a simple graph G is given as its adjacency matrix A_G, then whether G has a certain graph property or not can be expressed as a (transitive) Boolean function over $\binom{n}{2}$ elements in A_G.

Graph Property Testing

Definition (Graph Property)

- Graph properties are those of graphs that are invariant under changing the labelings of vertices.
  (ex. connectedness, planarity)
- If a simple graph G is given as its adjacency matrix A_G, then whether G has a certain graph property or not can be expressed as a (transitive) Boolean function over $\binom{n}{2}$ elements in A_G.

Graph Property Testing

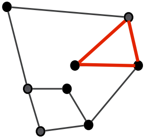
Decide if a graph $G = (V, E)$ has a graph property P with a minimum number of queries of the form "Is the pair (i,j) an edge of G?" (=$A_G[i, j]$)
(ignoring the cost of other operations.)

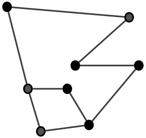
There are a long history of studies on this subject.

Triangle Finding

Triangle Finding Problem

Given a graph, decide with high probability if it contains a triangle as a subgraph by making a minimum number of queries.



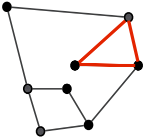


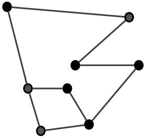
no triangle

Triangle Finding

Triangle Finding Problem

Given a graph, decide with high probability if it contains a triangle as a subgraph by making a minimum number of queries.





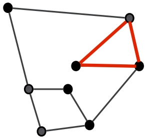
This is an particularly important problem well studied since a fast triangle finding algorithm in the sense of time complexity would compute/solve fast

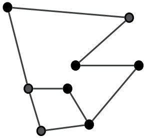
- Boolean matrix multiplication
- Max 2 -SAT

Triangle Finding

Triangle Finding Problem

Given a graph, decide with high probability if it contains a triangle as a subgraph by making a minimum number of queries.





no triangle

This is an particularly important problem well studied since a fast triangle finding algorithm in the sense of time complexity would compute/solve fast

- Boolean matrix multiplication
- Max 2 -SAT

As a first step, query-efficient algorithms are worth studying.

Triangle Finding as Graph Property Testing

Triangle Finding Problem

Given a graph, decide with at least probability $2/3$ if it contains a triangle as a subgraph by making a minimum number of queries.

Classical Case $\Omega(n^2)$ queries — we need to query almost all.

Triangle Finding as Graph Property Testing

Triangle Finding Problem

Given a graph, decide with at least probability $2/3$ if it contains a triangle as a subgraph by making a minimum number of queries.

Classical Case $\Omega(n^2)$ queries — we need to query almost all.

Quantum Case $O(\sqrt{\binom{n}{3}}) = O(n^{1.5})$ can obtained simply by applying Grover's quantum search algorithm.

Moreover, a series of improvements have been made by introducing novel general-purpose quantum techniques.

The triangle finding is one of the central problems that have advanced quantum algorithm/complexity theory.

Quantum Algorithms for Triangle Finding

The trivial quantum upper bound is $O(n^{1.5})$ queries.

Quantum Algorithms for Triangle Finding

The trivial quantum upper bound is $O(n^{1.5})$ queries.

- $\tilde{O}(n^{1.3})$ queries [Magniez-Santha-Szegedy, SODA2005]
  by a new application of quantum walk.

Quantum Algorithms for Triangle Finding

The trivial quantum upper bound is $O(n^{1.5})$ queries.

- $\tilde{O}(n^{1.3})$ queries [Magniez-Santha-Szegedy, SODA2005]
  by a new application of quantum walk.
- $O(n^{35/27})$ queries [Belovs, STOC2012] (35/27=1.296...)
  by introducing the learning graph technique.

Quantum Algorithms for Triangle Finding

The trivial quantum upper bound is $O(n^{1.5})$ queries.

- $\tilde{O}(n^{1.3})$ queries [Magniez-Santha-Szegedy, SODA2005]
  by a new application of quantum walk.
- $O(n^{35/27})$ queries [Belovs, STOC2012] (35/27=1.296...)
  by introducing the learning graph technique.
- $O(n^{9/7})$ queries [Lee-Magniez-Santha, SODA2013] (9/7=1.285...)
  by improving the learning graph technique.

Quantum Algorithms for Triangle Finding

The trivial quantum upper bound is $O(n^{1.5})$ queries.

- $\tilde{O}(n^{1.3})$ queries [Magniez-Santha-Szegedy, SODA2005]
  by a new application of quantum walk.
- $O(n^{35/27})$ queries [Belovs, STOC2012] (35/27=1.296...)
  by introducing the learning graph technique.
- $O(n^{9/7})$ queries [Lee-Magniez-Santha, SODA2013] (9/7=1.285...)
  by improving the learning graph technique.
- $\tilde{O}(n^{9/7})$ queries (simpler algorithm) [Jeffery-Kothari-Magniez, SODA2013]
  by introducing the concept of nested quantum walk.

Quantum Algorithms for Triangle Finding

The trivial quantum upper bound is $O(n^{1.5})$ queries.

- $\tilde{O}(n^{1.3})$ queries [Magniez-Santha-Szegedy, SODA2005]
  by a new application of quantum walk.
- $O(n^{35/27})$ queries [Belovs, STOC2012] (35/27=1.296...)
  by introducing the learning graph technique.
- $O(n^{9/7})$ queries [Lee-Magniez-Santha, SODA2013] (9/7=1.285...)
  by improving the learning graph technique.
- $\tilde{O}(n^{9/7})$ queries (simpler algorithm) [Jeffery-Kothari-Magniez, SODA2013]
  by introducing the concept of nested quantum walk.
- $O(n^{5/4})$ queries [LeGall, FOCS2014] (5/4=1.25)
  by combinatorial arguments + quantum walk.

Quantum Algorithms for Triangle Finding

The trivial quantum upper bound is $O(n^{1.5})$ queries.

- $\tilde{O}(n^{1.3})$ queries [Magniez-Santha-Szegedy, SODA2005]
  by a new application of quantum walk.
- $O(n^{35/27})$ queries [Belovs, STOC2012] (35/27=1.296...)
  by introducing the learning graph technique.
- $O(n^{9/7})$ queries [Lee-Magniez-Santha, SODA2013] (9/7=1.285...)
  by improving the learning graph technique.
- $\tilde{O}(n^{9/7})$ queries (simpler algorithm) [Jeffery-Kothari-Magniez, SODA2013]
  by introducing the concept of nested quantum walk.
- $O(n^{5/4})$ queries [LeGall, FOCS2014] (5/4=1.25)
  by combinatorial arguments + quantum walk.

This series of works have developed new quantum techniques for general purposes.

Quantum Algorithms for Triangle Finding

The trivial quantum upper bound is $O(n^{1.5})$ queries.

- $\tilde{O}(n^{1.3})$ queries [Magniez-Santha-Szegedy, SODA2005]
  by a new application of quantum walk.
- $O(n^{35/27})$ queries [Belovs, STOC2012] (35/27=1.296...)
  by introducing the learning graph technique.
- $O(n^{9/7})$ queries [Lee-Magniez-Santha, SODA2013] (9/7=1.285...)
  by improving the learning graph technique.
- $\tilde{O}(n^{9/7})$ queries (simpler algorithm) [Jeffery-Kothari-Magniez, SODA2013]
  by introducing the concept of nested quantum walk.
- $O(n^{5/4})$ queries [LeGall, FOCS2014] (5/4=1.25)
  by combinatorial arguments + quantum walk.

Along this line of research,
we consider a generalization of triangle finding to the hypergraph case.

Hypergraphs

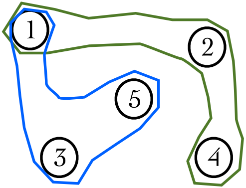
Definition (3-uniform Hypergraphs)

An undirected 3-uniform hypergraph is a pair (V, E), where

- V is a finite set (the set of vertices),
- $E \subseteq V \times V \times V$ is the set of hyperedges, i.e., unordered triples of elements in V.

Example

$V = \{1, 2, 3, 4, 5\}$
$E = \{\{1, 2, 4\}, \{1, 3, 5\}\}$



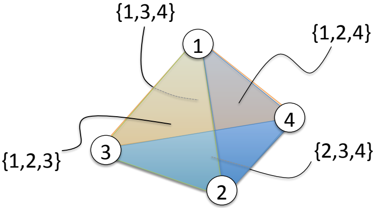
Note that we can define k-uniform hypergraphs, but we only deal with 3-uniform case in this talk.

4-Clique over a 3-Uniform Hypergraph

4-clique is a complete 3-uniform hypergraph on 4 vertices:
(a generalization of a triangle)

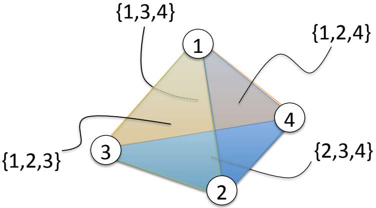
Example

Ex. $\{1,2,3\}, \{1,2,4\}, \{1,3,4\}, \{2,3,4\}$ are all hyperedges.



4-Clique over a 3-Uniform Hypergraph

4-clique is a complete 3-uniform hypergraph on 4 vertices:
(a generalization of a triangle)



Example

Ex. $\{1,2,3\},\{1,2,4\},\{1,3,4\},\{2,3,4\}$ are all hyperedges.

4-Clique Finding Problem

Given a hypergraph G, decide with high probability if it contains a 4-clique as a subhypergraph by making a minimum number of queries of the form: "Is the triple $\{i,j,k\}$ an hyperedge of G?"

This problem is closely related to Max-3SAT or multiplication of tensors.

Our Results: Finding 4-Clique in a 3-uniform Hypergraph

Theorem (4-clique Finding Quantum Algorithm)

There exists a quantum algorithm that detects with high probability if the input 3-uniform hypergraph on n vertices has a 4-clique as a subhypergraph (and finds a 4-clique if it exists), by making $\tilde{O}(n^{241/128}) = O(n^{1.883})$ *queries.*

Our Results: Finding 4-Clique in a 3-uniform Hypergraph

Theorem (4-clique Finding Quantum Algorithm)

There exists a quantum algorithm that detects with high probability if the input 3-uniform hypergraph on n vertices has a 4-clique as a subhypergraph (and finds a 4-clique if it exists), by making $\tilde{O}(n^{241/128}) = O(n^{1.883})$ queries.

- Better than naïve Grover search over the $\binom{n}{4}$ combinations of vertices, which only gives $O(n^2)$ queries.

Our Results: Finding 4-Clique in a 3-uniform Hypergraph

Theorem (4-clique Finding Quantum Algorithm)

There exists a quantum algorithm that detects with high probability if the input 3-uniform hypergraph on n vertices has a 4-clique as a subhypergraph (and finds a 4-clique if it exists),
by making $\tilde{O}(n^{241/128}) = O(n^{1.883})$ queries.

- Better than naïve Grover search over the $\binom{n}{4}$ combinations of vertices, which only gives $O(n^2)$ queries.
- Actually works for finding any constant-sized subhypergraph.

Our Results: Finding 4-Clique in a 3-uniform Hypergraph

Theorem (4-clique Finding Quantum Algorithm)

There exists a quantum algorithm that detects with high probability if the input 3-uniform hypergraph on n vertices has a 4-clique as a subhypergraph (and finds a 4-clique if it exists), by making $\tilde{O}(n^{241/128}) = O(n^{1.883})$ *queries.*

Technical outline

- Extend the idea of the triangle finding algorithm by [Lee-Magniez-Santha, SODA05] to the hypergraph case.

Our Results: Finding 4-Clique in a 3-uniform Hypergraph

Theorem (4-clique Finding Quantum Algorithm)

There exists a quantum algorithm that detects with high probability if the input 3-uniform hypergraph on n vertices has a 4-clique as a subhypergraph (and finds a 4-clique if it exists), by making $\tilde{O}(n^{241/128}) = O(n^{1.883})$ *queries.*

Technical outline

- Extend the idea of the triangle finding algorithm by [Lee-Magniez-Santha, SODA05] to the hypergraph case. But the analysis gets too complicated to be done.

Our Results: Finding 4-Clique in a 3-uniform Hypergraph

Theorem (4-clique Finding Quantum Algorithm)

There exists a quantum algorithm that detects with high probability if the input 3-uniform hypergraph on n vertices has a 4-clique as a subhypergraph (and finds a 4-clique if it exists), by making $\tilde{O}(n^{241/128}) = O(n^{1.883})$ *queries.*

Technical outline

- Extend the idea of the triangle finding algorithm by [Lee-Magniez-Santha, SODA05] to the hypergraph case. But the analysis gets too complicated to be done.
- Then cast the extended idea to the framework of nested quantum walk introduced by [Jeffery-Kothari-Magniez, SODA05].

Our Results: Finding 4-Clique in a 3-uniform Hypergraph

Theorem (4-clique Finding Quantum Algorithm)

There exists a quantum algorithm that detects with high probability if the input 3-uniform hypergraph on n vertices has a 4-clique as a subhypergraph (and finds a 4-clique if it exists),

by making $\tilde{O}(n^{241/128}) = O(n^{1.883})$ *queries.*

Technical outline

- Extend the idea of the triangle finding algorithm by [Lee-Magniez-Santha, SODA05] to the hypergraph case. But the analysis gets too complicated to be done.
- Then cast the extended idea to the framework of nested quantum walk introduced by [Jeffery-Kothari-Magniez, SODA05]. Still, need to somehow handle undesirable cases that is unique in the hypergraph case.

Our Results: Finding 4-Clique in a 3-uniform Hypergraph

Theorem (4-clique Finding Quantum Algorithm)

There exists a quantum algorithm that detects with high probability if the input 3-uniform hypergraph on n vertices has a 4-clique as a subhypergraph (and finds a 4-clique if it exists), by making $\tilde{O}(n^{241/128}) = O(n^{1.883})$ *queries.*

Technical outline

- Extend the idea of the triangle finding algorithm by [Lee-Magniez-Santha, SODA05] to the hypergraph case. But the analysis gets too complicated to be done.
- Then cast the extended idea to the framework of nested quantum walk introduced by [Jeffery-Kothari-Magniez, SODA05]. Still, need to somehow handle undesirable cases that is unique in the hypergraph case.
- Finally heavily use the concentration theorem over hypergeometric distribution to show that the undesirable cases rarely happen.

Applications

Definition (Ternary Associativity)

Let X be a finite set with $|X| = n$. A ternary operator $\mathcal{F}$ from $X \times X \times X$ to X is said to be *associative* if
$\mathcal{F}(\mathcal{F}(a,b,c),d,e) = \mathcal{F}(a,\mathcal{F}(b,c,d),e) = \mathcal{F}(a,b,\mathcal{F}(c,d,e))$
holds for every 5-tuple $(a,b,c,d,e) \in X^5$.

Theorem (Ternary Associativity Testing)

There exists a quantum algorithm that determines if $\mathcal{F}$ is associative with high probability using $\tilde{O}(n^{169/80}) = \tilde{O}(n^{2.1125})$ queries.

Proof.

First transform ternary associativity testing into the problem of finding a certain subhypegraph of constant size. The, we apply our algorithm. □

Quick Quantum Computing: one qubit case

- A quantum version of “a bit” is called a qubit.
- The quantum state of a qubit is a unit vector in a complex Euclidean space $\mathbb{C}^2$.

Quick Quantum Computing: one qubit case

- A quantum version of "a bit" is called a qubit.
- The quantum state of a qubit is a unit vector in a complex Euclidean space $\mathbb{C}^2$.
  - Take any orthonormal basis and let $\mathbf{e}_0 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and $\mathbf{e}_1 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$.

Quick Quantum Computing: one qubit case

- A quantum version of “a bit” is called a qubit.
- The quantum state of a qubit is a unit vector in a complex Euclidean space $\mathbb{C}^2$.
  - Take any orthonormal basis and let $\mathbf{e}_0 = \begin{pmatrix}1\\0\end{pmatrix}$ and $\mathbf{e}_1 = \begin{pmatrix}0\\1\end{pmatrix}$.
  - Any quantum state of a qubit is a linear combination of $\mathbf{e}_0$ and $\mathbf{e}_1$ over the complex field $\mathbb{C}$: $\alpha\mathbf{e}_0 + \beta\mathbf{e}_1$ with $\alpha, \beta \in \mathbb{C}$ and $|\alpha|^2 + |\beta|^2 = 1$.
  - We often regard this as “a superposition of ‘0’ and ‘1’”.

Quick Quantum Computing: one qubit case

- A quantum version of “a bit” is called a qubit.
- The quantum state of a qubit is a unit vector in a complex Euclidean space $\mathbb{C}^2$.
  - Take any orthonormal basis and let $\mathbf{e}_0 = \begin{pmatrix}1\\0\end{pmatrix}$ and $\mathbf{e}_1 = \begin{pmatrix}0\\1\end{pmatrix}$.
  - Any quantum state of a qubit is a linear combination of $\mathbf{e}_0$ and $\mathbf{e}_1$ over the complex field $\mathbb{C}$: $\alpha\mathbf{e}_0 + \beta\mathbf{e}_1$ with $\alpha, \beta \in \mathbb{C}$ and $|\alpha|^2 + |\beta|^2 = 1$.
  - We often regard this as “a superposition of ‘0’ and ‘1’”.
- Quantum operations on a qubit are unitary operators ($UU^* = I$) or orthogonal projectors ($PP = P$ and $P^*P = 0$).

Quick Quantum Computing: one qubit case

- A quantum version of “a bit” is called a qubit.
- The quantum state of a qubit is a unit vector in a complex Euclidean space $\mathbb{C}^2$.
  - Take any orthonormal basis and let $\mathbf{e}_0 = \begin{pmatrix}1\\0\end{pmatrix}$ and $\mathbf{e}_1 = \begin{pmatrix}0\\1\end{pmatrix}$.
  - Any quantum state of a qubit is a linear combination of $\mathbf{e}_0$ and $\mathbf{e}_1$ over the complex field $\mathbb{C}$: $\alpha\mathbf{e}_0 + \beta\mathbf{e}_1$ with $\alpha, \beta \in \mathbb{C}$ and $|\alpha|^2 + |\beta|^2 = 1$.
  - We often regard this as “a superposition of ‘0’ and ‘1’”.
- Quantum operations on a qubit are unitary operators ($UU^* = I$) or orthogonal projectors ($PP = P$ and $P^*P = 0$).
- Applying the set of orthogonal projectors summing to I is called measurement, which outputs a quantum state and a classical outcome.
  - To get classical results at the end of computation, we need to apply orthogonal projectors.

Quick Quantum Computing: one qubit case (example)

Let $\mathbf{e}_0 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and $\mathbf{e}_1 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$.

- $H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$ is unitary ($H^* H = I$), and

  $$H\mathbf{e}_0 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}}\mathbf{e}_0 + \frac{1}{\sqrt{2}}\mathbf{e}_1 .$$

Quick Quantum Computing: one qubit case (example)

Let $\mathbf{e}_0 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and $\mathbf{e}_1 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$.

- $H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$ is unitary ($H^* H = I$), and

  $$H\mathbf{e}_0 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}}\mathbf{e}_0 + \frac{1}{\sqrt{2}}\mathbf{e}_1.$$

- $P = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \begin{pmatrix} 1 & 0 \end{pmatrix} = \mathbf{e}_0 {\mathbf{e}_0}^*$ is the orthogonal projector onto the space spanned by $\mathbf{e}_0 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$. Note that $\mathrm{I} - P$ is $\mathbf{e}_1 {\mathbf{e}_1}^*$.

  Measurement $\{P, \mathrm{I} - P\}$ on $\alpha\mathbf{e}_0 + \beta\mathbf{e}_1$ outputs $\begin{cases} (\mathbf{e}_0, 0) & \text{with prob. } |\alpha|^2 \\ (\mathbf{e}_1, 1) & \text{with prob. } |\beta|^2 \end{cases}$

  (The resulting quantum state is normalized.)

Quick Quantum Computing: n-qubt case

- A quantum state of n qubits is a unit vector in a complex Euclidean space $\mathbb{C}^{2^n}$ of 2^n dimensions

Quick Quantum Computing: n-qubt case

- A quantum state of n qubits is a unit vector in a complex Euclidean space $\mathbb{C}^{2^n}$ of 2^n dimensions
- Let $\{\mathbf{e}_0, \ldots, \mathbf{e}_{2^n-1}\}$ be an orthonormal basis, where $\mathbf{e}_k$ is an 2^n dimensional unit vector that has 0 at all cordinates except the $(k+1)^{\text{st}}$ position.

Quick Quantum Computing: *n*-qubt case

- A quantum state of n qubits is a unit vector in a complex Euclidean space $\mathbb{C}^{2^n}$ of 2^n dimensions
- Let $\{\mathbf{e}_0, \ldots, \mathbf{e}_{2^n-1}\}$ be an orthonormal basis, where $\mathbf{e}_k$ is an 2^n dimensional unit vector that has 0 at all cordinates except the $(k+1)^{\mathrm{st}}$ position.
- The quantum state is represented as
  $\sum_{k=0}^{2^n-1} \alpha_k \mathbf{e}_k$ for $\alpha_k \in \mathbb{C}$ with $\sum_{k=1}^{2^n} |\alpha_k|^2 = 1$.

Quick Quantum Computing: *n*-qubt case

- A quantum state of n qubits is a unit vector in a complex Euclidean space $\mathbb{C}^{2^n}$ of 2^n dimensions
- Let $\{\mathbf{e}_0, \ldots, \mathbf{e}_{2^n-1}\}$ be an orthonormal basis, where $\mathbf{e}_k$ is an 2^n dimensional unit vector that has 0 at all cordinates except the $(k+1)^{\text{st}}$ position.
- The quantum state is represented as $\sum_{k=0}^{2^n-1} \alpha_k \mathbf{e}_k$ for $\alpha_k \in \mathbb{C}$ with $\sum_{k=1}^{2^n} |\alpha_k|^2 = 1$.
- Quantum operators are unitary operators or orthogonal projectors over the 2^n-dimensional space.

Quick Quantum Computing: *n*-qubt case

- A quantum state of n qubits is a unit vector in a complex Euclidean space $\mathbb{C}^{2^n}$ of 2^n dimensions
- Let $\{\mathbf{e}_0, \ldots, \mathbf{e}_{2^n-1}\}$ be an orthonormal basis, where $\mathbf{e}_k$ is an 2^n dimensional unit vector that has 0 at all cordinates except the $(k+1)^{\mathrm{st}}$ position.
- The quantum state is represented as $\sum_{k=0}^{2^n-1} \alpha_k \mathbf{e}_k$ for $\alpha_k \in \mathbb{C}$ with $\sum_{k=1}^{2^n} |\alpha_k|^2 = 1$.
- Quantum operators are unitary operators or orthogonal projectors over the 2^n-dimensional space.
  - These operators over a large space can be decomposed into some elemenatry operators acting on one or two qubits (similar to elementary gates in classical circuits).

Quick Quantum Computing: *n*-qubt case

- A quantum state of n qubits is a unit vector in a complex Euclidean space $\mathbb{C}^{2^n}$ of 2^n dimensions
- Let $\{\mathbf{e}_0, \ldots, \mathbf{e}_{2^n-1}\}$ be an orthonormal basis, where $\mathbf{e}_k$ is an 2^n dimensional unit vector that has 0 at all cordinates except the $(k+1)^{\text{st}}$ position.
- The quantum state is represented as $\sum_{k=0}^{2^n-1} \alpha_k \mathbf{e}_k$ for $\alpha_k \in \mathbb{C}$ with $\sum_{k=1}^{2^n} |\alpha_k|^2 = 1$.
- Quantum operators are unitary operators or orthogonal projectors over the 2^n-dimensional space.
  - These operators over a large space can be decomposed into some elemenatry operators acting on one or two qubits (similar to elementary gates in classical circuits).

Traditional Notation in Quantum Physics

Instead of $\mathbf{e}_k$, we will write $|k\rangle$ (pronounced "ket k").

Query Complexity Model (a.k.a. oracle model)

Definition (Classical Case)

- An input hypergraph $G = (V, E)$ is given as an oracle.

Our case

$$\text{Oracle} = \left\{ h_{ijk} \in \{T, F\} : i < j < k, (i, j, k) \in V \times V \times V \right\}.$$

Query Complexity Model (a.k.a. oracle model)

Definition (Classical Case)

- An input hypergraph $G = (V, E)$ is given as an oracle.

Our case

Oracle $= \left\{ h_{ijk} \in \{T, F\} \colon i < j < k, (i, j, k) \in V \times V \times V \right\}$.

- Algorithms need to make queries to the oracle to get input.

Our case

For the query $(\{i, j, k\}, ?)$, we receive the answer $(\{i, j, k\}, h_{ijk})$.

Algorithm $\xrightarrow{(\{i,j,k\},?)}$ Oracle $\xrightarrow{(\{i,j,k\},T)}$ Algorithm

Query Complexity Model (a.k.a. oracle model)

Definition (Classical Case)

- An input hypergraph $G = (V, E)$ is given as an oracle.

Our case

Oracle $= \left\{h_{ijk} \in \{T, F\} : i < j < k, (i, j, k) \in V \times V \times V\right\}$.

- Algorithms need to make queries to the oracle to get input.

Our case

For the query $(\{i, j, k\}, ?)$, we receive the answer $(\{i, j, k\}, h_{ijk})$.

Algorithm $\xrightarrow{(\{i,j,k\},?)}$ Oracle $\xrightarrow{(\{i,j,k\},T)}$ Algorithm

- Minimize # of queries, ignoring the cost of other operations.

Our case

The number of required queries is trivially at most $\binom{n}{3} = O(n^3)$.

Query Complexity Model (a.k.a. oracle model)

Definition (Quantum Case)

- An input hypergraph $G = (V, E)$ is given as an oracle.

Our case

$$\text{Oracle} = \left\{ h_{ijk} \in \{T, F\} : i < j < k, (i, j, k) \in V \times V \times V \right\}.$$

Query Complexity Model (a.k.a. oracle model)

Definition (Quantum Case)

- An input hypergraph $G = (V, E)$ is given as an oracle.

Our case

Oracle $= \left\{ h_{ijk} \in \{T, F\} \colon i < j < k, (i, j, k) \in V \times V \times V \right\}$.

- Algorithms need to make quantum queries to the oracle to get input.

Our case

- Quantum queries are superpositions of many classical queries, and the answers are those of the corresp. classical answers: a query $\sum \alpha_{i,j,k} |\{i, j, k\}, ?\rangle$, and the answer $\sum \alpha_{i,j,k} \left|\{i, j, k\}, h_{ijk}\right\rangle$.
- Note: a classical query can be simulated by a quantum query: Set $\alpha_{ijk} = 1$ and $\alpha_{pqr} = 0$ for all $(p, q, r) \neq (i, j, k)$.

Query Complexity Model (a.k.a. oracle model)

Definition (Quantum Case)

- An input hypergraph $G = (V, E)$ is given as an oracle.

Our case

Oracle $= \left\{h_{ijk} \in \{T, F\} : i < j < k, (i, j, k) \in V \times V \times V\right\}$.

- Algorithms need to make quantum queries to the oracle to get input.

Our case

- Quantum queries are superpositions of many classical queries, and the answers are those of the corresp. classical answers: a query $\sum \alpha_{i,j,k} |\{i, j, k\}, ?\rangle$, and the answer $\sum \alpha_{i,j,k} \left|\{i, j, k\}, h_{ijk}\right\rangle$.
- Note: a classical query can be simulated by a quantum query: Set $\alpha_{ijk} = 1$ and $\alpha_{pqr} = 0$ for all $(p, q, r) \neq (i, j, k)$.

- Minimize # of quantum queries, ignoring the cost of other operations.

Search with Random Walk

Search Problem

Given a Boolean function f over the domain X onto $\{0, 1\}$,
find a solution $x \in X$ such that $f(x) = 1$.

Search with Random Walk

Search Problem

Given a Boolean function f over the domain X onto $\{0, 1\}$, find a solution $x \in X$ such that $f(x) = 1$.

Simple Sampling Idea

- Sample a subset $Y_1 \subseteq X$ of size r.
- Check if Y_1 contains a solution; if it indeed does, we are done.

Search with Random Walk

Search Problem

Given a Boolean function f over the domain X onto $\{0, 1\}$, find a solution $x \in X$ such that $f(x) = 1$.

Simple Sampling Idea

- Sample a subset $Y_1 \subseteq X$ of size r.
- Check if Y_1 contains a solution; if it indeed does, we are done.
- Otherwise, we update Y_1 to Y_2 by replacing a random element in Y with a new element that is chosen at random from $X \setminus Y_1$.
  (Y_1 and Y_2 differ only by one element)

Let $Y_1 = \{1, 3, 5, 7\}$. If we pick out 5 from Y_1 and put in 9, then we have $Y_2 = \{1, 3, 7, 9\}$.

Search with Random Walk

Search Problem

Given a Boolean function f over the domain X onto $\{0, 1\}$,
find a solution $x \in X$ such that $f(x) = 1$.

Simple Sampling Idea

- Sample a subset $Y_1 \subseteq X$ of size r.
- Check if Y_1 contains a solution; if it indeed does, we are done.
- Otherwise, we update Y_1 to Y_2 by replacing a random element in Y with a new element that is chosen at random from $X \setminus Y_1$.
  (Y_1 and Y_2 differ only by one element)
- Check if Y_2 contains a solution; if it indeed does, we are done.

Search with Random Walk

Search Problem

Given a Boolean function f over the domain X onto $\{0, 1\}$, find a solution $x \in X$ such that $f(x) = 1$.

Simple Sampling Idea

- Sample a subset $Y_1 \subseteq X$ of size r.
- Check if Y_1 contains a solution; if it indeed does, we are done.
- Otherwise, we update Y_1 to Y_2 by replacing a random element in Y with a new element that is chosen at random from $X \setminus Y_1$.
  (Y_1 and Y_2 differ only by one element)
- Check if Y_2 contains a solution; if it indeed does, we are done.
- Otherwise, we update Y_2 to Y_3 by replacing...

We can regard the sequence $Y_1 \to Y_2 \to Y_3 \to \cdots$ as random walks over the graph whose nodes are subsets of size r of X.

Johnson Graph

Definition (Johnson graph $J(n, r) = (V, E)$)

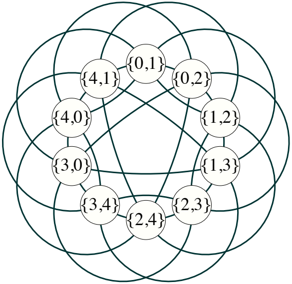
- V is the collection of all r-sized subsets of $[n]$, so that $|V| = \binom{n}{r}$.
  (Corresponding to sampling r-sized subsets from X with $|X| = n$).

Johnson Graph

Definition (Johnson graph $J(n, r) = (V, E)$)

- V is the collection of all r-sized subsets of $[n]$, so that $|V| = \binom{n}{r}$.
  (Corresponding to sampling r-sized subsets from X with $|X| = n$).
- For every vertex pairs $U, T \in V$, the pair $\{U, V\}$ is an edge (an element in E) if and only if U and T differ only by one element.

ex.) $J(5, 2)$ looks like $\longrightarrow$.

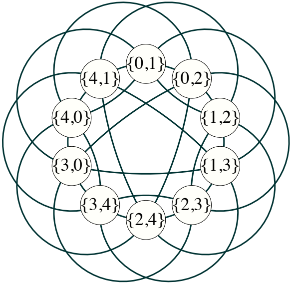


Johnson Graph

Definition (Johnson graph $J(n, r) = (V, E)$)

- V is the collection of all r-sized subsets of $[n]$, so that $|V| = \binom{n}{r}$.
  (Corresponding to sampling r-sized subsets from X with $|X| = n$).
- For every vertex pairs $U, T \in V$, the pair $\{U, V\}$ is an edge (an element in E) if and only if U and T differ only by one element.

ex.) $J(5, 2)$ looks like $\longrightarrow$.



Fact.

The spectral gap of $J(n, r)$ is $\Theta(1/r)$.

The spectral gap of the graph affects the hitting time of random walk over $J(n, r)$.

Search with Random Walk

Let us say that the nodes containing a solution is marked.

Fact.

If the underlying graph has spectral gap δ and the fraction of marked nodes is ϵ, then the hitting time (the number of steps required to find a marked node with high probability) is $O(\frac{1}{\delta \cdot \epsilon})$.

Search with Random Walk

Let us say that the nodes containing a solution is marked.

Fact.

If the underlying graph has spectral gap δ and the fraction of marked nodes is ϵ, then the hitting time (the number of steps required to find a marked node with high probability) is $O(\frac{1}{\delta \cdot \epsilon})$.

Corollary

The total cost for finding a solution is

$$\mathsf{S} + \frac{1}{\epsilon}\left(\frac{1}{\delta}\mathsf{U} + \mathsf{C}\right),$$

S: cost of initial sampling (initial queries)
U: cost of one step random walk (addition queries)
C: cost of checking if the node is marked. (additional queries).
(Here we perform checking procedure every $1/\delta$ steps.)

Search with Quantum Walk

Let us say that the nodes containing a solution is marked.

Fact.

If the underlying graph has spectral gap δ and the fraction of marked nodes is ϵ, then the number of steps required to find a marked node is ~~$O(\frac{1}{\delta \cdot \epsilon})$~~ $O(\sqrt{\frac{1}{\delta \cdot \epsilon}})$ with high probability. Note $\frac{1}{\delta \cdot \epsilon} \geq \sqrt{\frac{1}{\delta \cdot \epsilon}}$.

This implies that the total cost for finding a solution is

$$\cancel{\mathsf{S} + \frac{1}{\epsilon}\left(\frac{1}{\delta}\mathsf{U} + \mathsf{C}\right)} \quad \mathsf{S} + \frac{1}{\sqrt{\epsilon}}\left(\frac{1}{\sqrt{\delta}}\mathsf{U} + \mathsf{C}\right),$$

where
S: cost of initial sampling (initial queries)
U: cost of one step random walk (addition queries)
C: cost of checking if the node is marked. (additional queries).
(Here we perform checking procedure every $1/\delta$ steps.)

Our strategy for finding 4-clique

Let $\{a_1, a_2, a_3, a_4\}$ be a 4-clique. Sampling is actually recursive.

Our strategy for finding 4-clique

Let $\{a_1, a_2, a_3, a_4\}$ be a 4-clique. Sampling is actually recursive.

- Sample a set $V_1 \subseteq V$ with size v_1 of candidates for a_1.

Our strategy for finding 4-clique

Let $\{a_1, a_2, a_3, a_4\}$ be a 4-clique. Sampling is actually recursive.

- Sample a set $V_1 \subseteq V$ with size v_1 of candidates for a_1.
- To check if V_1 is marked,
  sample a set $V_2 \subseteq V$ with size v_2 of candidates for a_2.

Our strategy for finding 4-clique

Let $\{a_1, a_2, a_3, a_4\}$ be a 4-clique. Sampling is actually recursive.

- Sample a set $V_1 \subseteq V$ with size v_1 of candidates for a_1.
- To check if V_1 is marked,
  sample a set $V_2 \subseteq V$ with size v_2 of candidates for a_2.
- To check if V_2 is marked,
  sample a set $V_3 \subseteq V$ with size v_3 of candidates for a_3.

Our strategy for finding 4-clique

Let $\{a_1, a_2, a_3, a_4\}$ be a 4-clique. Sampling is actually recursive.

- Sample a set $V_1 \subseteq V$ with size v_1 of candidates for a_1.
- To check if V_1 is marked,
  sample a set $V_2 \subseteq V$ with size v_2 of candidates for a_2.
- To check if V_2 is marked,
  sample a set $V_3 \subseteq V$ with size v_3 of candidates for a_3.
- To check if V_3 is marked,
  sample a set $V_4 \subseteq V$ with size v_4 of candidates for a_4.

Our strategy for finding 4-clique

Let $\{a_1, a_2, a_3, a_4\}$ be a 4-clique. Sampling is actually recursive.

- Sample a set $V_1 \subseteq V$ with size v_1 of candidates for a_1.

- To check if V_4 is marked,
  sample a set of $F_{12} \subseteq V_1 \times V_2$ with size f_{12} of candidates for $\{a_1, a_2\}$.

Our strategy for finding 4-clique

Let $\{a_1, a_2, a_3, a_4\}$ be a 4-clique. Sampling is actually recursive.

- Sample a set $V_1 \subseteq V$ with size v_1 of candidates for a_1.
- To check if V_4 is marked,
  sample a set of $F_{12} \subseteq V_1 \times V_2$ with size f_{12} of candidates for $\{a_1, a_2\}$.
- To check if F_{12} is marked,
  sample a set of $F_{13} \subseteq V_1 \times V_3$ with size f_{13} of candidates for $\{a_1, a_3\}$.
- ...

Our strategy for finding 4-clique

Let $\{a_1, a_2, a_3, a_4\}$ be a 4-clique. Sampling is actually recursive.

- Sample a set $V_1 \subseteq V$ with size v_1 of candidates for a_1.

- To check if F_{34} is marked,
  sample a set of E_{123} with size e_{123} of candidates for $\{v_1, v_2, v_3\}$ by picking a pair from each of F_{12}, F_{23}, F_{13} to form a triple.
- ....

Our strategy for finding 4-clique

Let $\{a_1, a_2, a_3, a_4\}$ be a 4-clique. Sampling is actually recursive.

- Sample a set $V_1 \subseteq V$ with size v_1 of candidates for a_1.
- Check if $E_{123} \cup E_{124} \cup E_{134} \cup E_{234}$ contains a 4-clique.

Our strategy for finding 4-clique

Let $\{a_1, a_2, a_3, a_4\}$ be a 4-clique. Sampling is actually recursive.

- Sample a set $V_1 \subseteq V$ with size v_1 of candidates for a_1.

- Check if $E_{123} \cup E_{124} \cup E_{134} \cup E_{234}$ contains a 4-clique.

This sampling can be cast as recursive quantum-walk-based search.

Our strategy for finding 4-clique

Let $\{a_1, a_2, a_3, a_4\}$ be a 4-clique. Sampling is actually recursive.

- Sample a set $V_1 \subseteq V$ with size v_1 of candidates for a_1.
- Check if $E_{123} \cup E_{124} \cup E_{134} \cup E_{234}$ contains a 4-clique.

This sampling can be cast as recursive quantum-walk-based search.
Optimizing parameters v_i, f_{ij}, e_{ijk} gives $\tilde{O}(n^{241/128}) = O(n^{1.883})$ queries.

Conclusion

- We considered a generalization of Triangle Finding problem to the 3-uniform hypergraphs.
- For finding a 4-clique, we obtained a quantum algorithm with query complexity $O(n^{1.883})$, beating the $O(n^2)$-query trivial quantum algorithm.
- More generally, we developed a framework that give an efficient quantum algorithms for finding any constant-sized subhypergraph.
- For this, we designed a general technique for handling nested quantum walk over graphs of non-fixed size.

Open Problems

- Further improvements of our complexity?
- Can generalize our techniques to d-uniform hyper graphs ($d \geq 3$)?
- Other applications of our techniques?