Quantum Dynamic Programming

Haowei Deng. Kaiyan Shi

April 2021

1 Introduction

Dynamic programming is a widely used computer programming method. There is a collection of NP-hard problems(i.e. Traveling Salesman problem and Minimum set-cover problem) for which the best classical algorithms are exponentially expensive dynamic programming solutions. For these NP-hard problems, simply applying Grover's quantum search achieves a quadratic speedup over classical exhaustive search strategies. However, the naive exhaustive search is often not the fastest classical algorithm for these NP-hard problems. For example, for the Travelling Salesman Problem (TSP), exhaustive search over all possible routes runs in time $O^*(n!)$ and simply applying Grover's quantum search speeds the search up to $O^*(\sqrt{n!})$. This is still lower than the fastest classical algorithm based on dynamic programming [2] which runs in time $O^*(2^n)$.

Combining the quantum effects with classical problem-solving strategies for these NP-hard problems and achieving a quantum speedup over the fastest classical algorithm is an open problem. Ambainis et al. [1] proposed a divide-and-conquer idea with dynamic programming technique for some NP-hard problems, including the Travelling Salesman Problem (TSP), Minimum Set Cover and Hypercube Path. This idea can also be used in some other NP-hard problem including Graph-coloring[13], Minimum Steiner Tree[11].

In this paper, we first discuss Ambainis' idea base on several NP-hard problems. Then we discuss applying this idea to some further NP-hard problems, including graph coloring, minimum clique cover and minimum dominating set. We also compare the complexity of applying the different quantum methods on these NP-hard problems and cases where new ideas are needed to utilize the power of dynamic programming.

2 Previous work

For general NP-hard problem, Ambainis et al.[1] showed exponential-space exponential-time quantum algorithm using QRAM[5]. In this section, we discuss Ambainis' solution based on the Hamiltonian cycle problem, the Travelling Salesman Problem and the Minimum set cover problem.

2.1 Hamiltonian Cycle Problem

Hamiltonian Cycle problem is the special case of the Travelling Salesman problem. The Hamiltonian Cycle problem is defined as

- Hamiltonian cycle: For an undirected or directed graph, a cycle that visits each vertex exactly once.
- Hamiltonian cycle problem: Given a graph $G = (V, E)$, deciding whether it has a Hamiltonian Cycle

For a graph S and two different vertices u, v in S, let a Boolean function $f(S, u, v)$ be true iff there is a simple path that goes through all vertices of S, starts at u and ends at v. Let $|S|$ be the number of vertices in S. When discussing about complexity, O^* notation means

$$O^*(g(n)) = O(poly(n) * g(n))$$

Ambainis' algorithm first precomputes $f(S, u, v)$ for all subgraph $S \subset G$ such that $|S| = \frac{n}{4}+1$ using classical dynamic programming[]. The complexity this classical precomputation part is $O^*(\binom{n}{\frac{n}{4}}) = O^*(1.755^n)$.

Then the algorithm runs Grover search all subgraph $S \subset G$ such that $\frac{n}{2} + 1$ and all vertices pair $u, v \in S$ and checks whether $f(S, u, v)$ and $f(G\backslash S \cup \{u, v\}, v, u)$ are both true. In this case a the simple path that goes through all vertices in S (starts at u and ends at v) and the one in subgraph $G\backslash S \cup \{u, v\}$ (starts at v and ends at u) will construct a Hamiltonian cycle.

To calculate $f(S, u, v)$ for $|S| = \frac{n}{2} + 1$, the algorithm similarly runs another Grover's search over all subsets T of S such that $|T| = \frac{n}{4} + 1, u \in T, v \notin T$. For each T, search a vertex $t \in T$ to check whether $f(T, u, t)$ and $f((S\backslash T) \cup \{t\}, t, u)$ are both true. These result are known from the classical precomputation part and only needs $O(1)$ time. The first level search is searching for $\frac{n}{2}$ vertices subgraph from G so there are $\binom{n}{\frac{n}{2}}$ choices. Similarly the second level search has $\binom{\frac{n}{2}}{\frac{n}{4}}$ choices. So the complexity of the quantum part is

$$O^*(\sqrt{\binom{n}{\frac{n}{2}}\binom{\frac{n}{2}}{\frac{n}{4}}}) \tag{1}$$

Then the total complexity of the whole algorithm is the sum of two parts.

$$O^*(1.755^n + \sqrt{\binom{n}{\frac{n}{2}}\binom{\frac{n}{2}}{\frac{n}{4}}}) = O^*(1.755^n) \tag{2}$$

The complexity is bounded by the classical precomputation part. This idea for Hamiltonian Cycle problem can be extended to TSP easily and slightly optimized to $O^*(1.728^n)$ by making the classical and quantum parts have the same complexity.

2.2 Travelling Salesman Problem

This idea for the Hamiltonian Cycle problem can be immediately extended for TSP by replacing Grover's search with quantum minimum finding. The TSP problem is that given a weighted graph G, we need to find the length of the shortest simple cycle that visits each vertex. The fastest classical algorithm for TSP is Bellman-Held-Karp algorithm[6] in time $O(n^2 2^n)$. To solve TSP with quantum method, the quantum minimum finding technique is needed

Theorem 2.1 (Quantum Minimum Finding). *Let $a_1, ..., a_n$ be n integers, accessed by a procedure P. There is a quantum algorithm that finds $min_{i=1}^n\{a_i\}$ with success probability at least 2/3 using $O(\sqrt{n})$ applications of P.*

Suppose that integer a_i is given by the output of a subroutine A_i. If A_i can only returns the correct result with probability at least 2/3, the procedure P can be implemented as $O(log(n)$ repetitions of each A_i, to reduce the error probability of each P call to $O(\sqrt{n})$. Then with probability at least 2/3, all the calls to P will be correct, and quantum minimum finding will also be correct with probability at least 2/3, resulting in a total $O(\sqrt{n}log(n))$ runtime.

Now let's discribe Ambainis' quantum algorithm for Travelling Salesman which uses ideas from the Bellman-Held-Karp algorithm[6] which uses dynamic programming and runs in time $O(n^2 2^n)$. Let the given weighted graph be $G = (V, E, \omega)$ where $|V| = n, E \subset V^2$ and $\omega : E \to \mathbb{N}$ are the edge weights. Let $D = \{(S, u, v)|S \subset V, u, v \in S\}$ and a function $f : D \to \mathbb{N}$ and $f(S, u, v)$ is the length of the shortest simple path in subgragh of G induced by S that starts in u, ends at v and goes through all vertices in S. This function is similar to the function f in the algorithm for the Hamiltonian Cycle problem. The only difference is that it returns the length of the simple path rather than only its existence. In the quantum algorithm of Hamiltonian Cycle problem, the classical precomputation part is a bit longer than the quantum search part and causes the complexity of the whole algorithm larger than the quantum part. To optimize this, Ambainis et al.[1] introduce a parameter $\alpha \in (0, 1/3]$ to control the size of the subgraph that computed in the classical precomputation part. The whole algorithm is shown as below:

a. For a subgraph $S \subset V$, if $|S| \leq \frac{(1-\alpha)n}{4} + 1$, the algorithm calculate $f(S, u, v), u, v \in S$ by classical dynamic programming and store them in the memory. Thus the complexity of the classical part is $O^*(\binom{n}{\leq (1-\alpha)n/4})$ similar to the one of Hamiltonian Cycle problem.

b. Run quantum minimum finding over all subset $S \subset V$ such that $|S| = \frac{n}{2}+1$ to find the answer of

$$\min_{\substack{S \subset V \\ |S| = \frac{n}{2}+1}} \min_{\substack{u,v \in S \\ u \neq v}} f(S, u, v) + f((V \backslash S) \cup \{u, v\}, v, u) \tag{3}$$

This expression is similar to the the Grover search step in the algorithm of Hamiltonian cycle problem except using quantum minimum finding instead. To calculate $f(S, u, v)$ for $|S| = \frac{n}{2} + 1$, the algorithm runs quantum minimum finding for Eq 3 over all $T \subset S$ such that $|T| = \frac{n}{4} + 1$. To calculate $f(S, u, v)$ for $|S| = \frac{n}{4} + 1$, run quantum minimum finding for Eq 3 over all $T \subset S$ such that $|T| = \frac{\alpha n}{4} + 1$. For any S such that $|S| = \frac{\alpha n}{4} + 1$ or $|S| = \frac{n}{4} + 1$, the result is known from the classical precomputation. The complexity of this part is

$$O^*(\sqrt{\binom{n}{\frac{n}{2}}\binom{\frac{n}{2}}{\frac{n}{4}}\binom{\frac{n}{4}}{\frac{\alpha n}{4}}})$$

The total runtime of the algorithm is the sum of two parts. So the overall complexity is minimized when both parts' complexity are equal. The optimal choice for α then is approximately 0.055362. The running time of the algorithm then is $O^*(2^{0.788595n}) = O^*(1.727391^n)$.

2.3 Minimum Set Cover

Another famous NP-complete problem that can be speeded up using the Ambainis' idea is **Minimum Set Cover**. **Minimum Set Cover** problem is that given a collection $\mathcal{S}$ of subsets of an universe $\mathcal{U}$ where $|\mathcal{U}| = n$ and $|\mathcal{S}| = m$.

Let $\mathcal{C} \subset \mathcal{S}$ be a minimum set cover of $\mathcal{U}$. A set $\mathcal{P}$ is called a partial cover if $P = \bigcup_{S \in C'} S$ for some $C' \subset C$. Let $f(\mathcal{U}, \mathcal{S})$ be the size of the minimum set cover of $\mathcal{U}$ using sets of $\mathcal{S}$, a useful fact for the minimum set cover problem is that

Fact 1 (*Partial Cover Partition*).
If P is a partial cover of a minimum set cover of $\mathcal{U}$ using sets of $\mathcal{S}$, then

$$f(\mathcal{U}, \mathcal{S}) = f(P, \mathcal{S}) + f(\mathcal{U} \backslash P, S)$$

This expression is similar to the expression in TSP problem for which the algorithm uses quantum minimum finding to get the appropriate subset S. However, the difference is that such a P whose size is exactly $\frac{n}{2} + 1$ does not necessarily exist. So the algorithm for minimum set cover depends on the follow useful fact

Fact 2 (*Balanced Partial Cover Existence*).
For some $d = \beta n, 0 < \beta < 1$, if $|S| \leq d$ for all sets $S \in \mathcal{S}$, such a partial cover P using sets of $\mathcal{S}$ that $||P| - \frac{n}{2}| \leq \frac{d}{2}$ always exists.

Base on this fact, the minimum set cover problem can be solved with the similar idea in TSP with a slight modification. Given the universe $\mathcal{U}$ and the collection $\mathcal{S}$, the algorithm **MinCover** is below

a if $|\mathcal{U}| \leq cN$ where N is the number of elements of U in the very first call of the algorithm and c is a parameter, find the minimum set cover using classical dynamic programming and return the answer

b Iterate over all subsets $S \in \mathcal{S}$ such that $|S| \geq d = \beta n$, find **MinCover**($\mathcal{U} \backslash S, \{T \backslash S | T \in \mathcal{S}, T \backslash S \neq \phi\}$) and store the results for each S.

c Remove all sets S such that $|S| > d$ from $\mathcal{S}$.

d Now all sets $S \in \mathcal{S}$ has $|S| \leq d$. Run the same quantum minimum finding as in TSP for the following expression to find the minimum result

$$f(\mathcal{U}, \mathcal{S}) = \min_{P \in B} f(P, \mathcal{S}) + f(\mathcal{U} \backslash P, \mathcal{S})$$

where $B = \{P \subset \mathcal{U} | \; ||P| - \frac{n}{2}| \leq \frac{d}{2}$

e Return the best solutions from Steps b and d.

Each time the algorithm is called recursively in the Step b, the number of elements in the universe decreases for by $d = \beta n$. Hence the recursion depth is at most $log_{1-\beta} c = O(1)$ for some fixed β. At each recursive call, the algorithm will enumerate $poly(m, n)$ subsets from $\mathcal{S}$. So Step a, b, c only bring $poly(m, n)$ additional multiplicative complexity. So the complexity of the whole algorithm is still $O^*(1.728^n)$, the same as the algorithm for TSP.

2.4 Model

Ambainis' algorithms work in the commonly used QRAM (quantum random access memory)[5] model of computation which assumes quantum memory which can be accessed in a superposition. QRAM has the property that any time-T classical algorithm that uses random access memory can be invoked as a subroutine for a quantum algorithm in time $O(T)$. Thus it can be used for quantum search with condition checking which require data stored in a random data memory. However, while RAM is widely accepted model of classical computation, QRAM is sometimes criticized due to the difficulty of implementation. Recently, Shimizu et al[13] proposed several quantum dynamic programming algorithm not using QRAM. Can Ambainis' idea be used without QRAM is an open problem.

3 Graph Coloring

In graph theory, graph coloring is a way of coloring the vertices of a graph G such that no two adjacent vertices are of the same color. The minimum number of colors used to color G is called the chromatic number of G, i.e. $\chi(G)$. Graph coloring problem is to calculate such $\chi(G)$.

3.1 Classical Computational Complexity

Let $G = (V, E)$ be the target graph, and $G[S]$ be an induced subgraph S of G. With $\chi(G) = 0$ if G is an empty graph, dynamic programming can be formulated as below:

$$\chi(G[S]) = 1 + \min_{I \in \mathrm{MIS}(G)} f(G[V \backslash I]) \tag{4}$$

where $\mathrm{MIS}(G)$ denotes the set of all maximal independent set of G. An independent set is a set of vertices in a graph with no two of which are adjacent, and a maximal independent set of G is an independent set that is not a subset of any other independent set. $G[V\backslash I]$ is a subgraph obtained from G by deleting the vertices in I. There is a classical proof arguing that for G of $|V| = n$, $\mathrm{MIS}(G) \leq 3^{n/3}$ and also a classical algorithm enumerating all MISs with time $O^*(3^{n/3})$ [4]. The statement can be formally formulated below:

Theorem 3.1. *The maximum number of t-MISs of n-vertex graph is*

$$I(n,t) := \lfloor n/t \rfloor^{(\lfloor n/t \rfloor + 1)t - n} (\lfloor n/t \rfloor + 1)^{n - \lfloor n/t \rfloor t} == O^*(2^{n/3}).$$

Furthermore, there is a classical algorithm enumerating all $t-$MISs of $n-$ vertex graph in time $O^*(I(n,t)) = O^*(2^{n/3})$.

Therefore the time complexity of the classical dynamic programming algorithm of recursively applying the above equation for all possible subgraphs is

$$\sum_{s=0}^{n} \binom{n}{s} O^*(3^{s/3}) = O^*(1 + 3^{1/3})^n = O^*(2.4423^n) \tag{5}$$

3.2 Quantum Algorithm for Graph Coloring

There is one work strongly related to quantum dynamic programming of graph coloring [13]. They present an exponential-space quantum algorithm computing the chromatic number with running time $O^*(1.9140^n)$ using QRAM, and a polynomial-space quantum algorithm not using QRAM for the graph 20-coloring problem with running time $O^*(1.9575^*)$.

We are focusing on the QRAM case. Before reaching the algorithm with running time $O^*(1.9140^n)$, Kazuya and Ryuhei adopt the idea of Ambainis et al's [1] quantum dynamic programming algorithms of TSP with Grover's search on subprograms on graph coloring problems and give a relatively naive algorithm.

Ambainis et al.'s general idea of quantum dynamic programming divides the original graph $G = (V, E)$ into two subgraphs $G[S]$ and $G[V\backslash S]$ where S is some proper subset of V and has $|S| = \frac{n}{2} + 1$. It computes the target function of $G[S]$ and $G[V\backslash S]$ based on the construction of precomputed solutions of some smaller subgraph $G[S']$ with $|S| = \frac{n}{4} + 1$. Then it constructs the target function of G based on the solutions of $G[S]$ and $G[V\backslash S]$ with the same method. To utilize this idea in the graph coloring problem, we can rewrite the Equation 4 to

$$\chi(G) = \min_{S} \chi(G[S]) + \chi(G[V\backslash S]) \tag{6}$$

Equation 4 is exactly the special case of Equation 6 (When S is one of proper independent set in the maximal independent sets of G and we get $\chi(G[S]) = 1$). If one assumes $|S|$ is exactly $\lceil \frac{n}{2} \rceil$ or $\lfloor \frac{n}{2} \rfloor$, then one only need to run Grover's quantum search over all S such that $|S| = \frac{n}{2} + 1$ similar to Ambainis' algorithm. Then one can apply the Ambainis et al's quantum dynamic programming straightforwardly and obtain an $O^*(1.728^n)$-time quantum algorithm for the graph coloring problem. Unfortunately, the balance partition S satisfying $\chi(G) = \chi(G[S]) + \chi(G[V\backslash S])$ does not always exist and we cannot utilize Ambainis et al's idea directly. Hence, Kazuya and Ryuhei states use the following useful fact

Fact 3 (*Graph Partition*).

Let $a_1, a_2, ..., a_k$ be positive integers and $a_1 \geq a_i, \forall i = 1, 2, ..., k$. Let $n = \sum_{i=1}^{k} a_i$. Then, there exists $S \subset \{2, ..., k\}$ such that $\sum_{i \in S} a_i \leq \lceil \frac{n}{2} \rceil$ and $\sum_{i \in \{2,...,k\}\backslash S} a_i \leq \lfloor \frac{n}{2} \rfloor$.

Proof. Let $t = \max\{j \in \{2, ..., k\} | \sum_{i=2}^{j} a_j \leq \lceil \frac{n}{2} \rceil\}$. Let $S = \{2, 3, ..., t\}$. Then $\sum_{i \in \{2,...,k\}\backslash S} a_i = \sum_{i \in \{t+1,t+2,...,k\}} a_i \leq \sum_{i \in \{1,t+2,...,k\}} a_i = n - (\sum_{i \in S} a_i + a_{t+1}) \leq \lfloor \frac{n}{2} \rfloor$ since $\sum_{i \in S} a_i + a_{t+1} > \lceil \frac{n}{2} \rceil$. □

Given a graph G, suppose its minimum coloring divides it into $m = \chi(G)$ MISs $s_1, ..., s_m$ and their size are $a_1, ..., a_m$. Let s_1 be the maximum one over $s_1, ..., s_m$. The fact above suggests that if we remove s_1 from the graph G, the remained part of G can be divided into two parts constructed by $s_2, ..., s_m$, both with size under half of the size of G. In this way, Ambainis' algorithm can be kept being applied.

Then from fact 3, Kazuya and Ryuhei come up with the following algorithm. They first precomputes the chromatic number of all induced subgraphs with size at most $n/4$. This precomputation is based on classical recursive formula, i.e. Equation 4 with MIS enumeration. And this is where dividing happens.

They next argue in Section 3 in [13] that Byskov has shown an upper bound exists for the maximum number of t-MISs of n-vertex graph, i.e.

$$I(n,t) := \lfloor n/t \rfloor^{(\lfloor n/t \rfloor + 1)t - n} (\lfloor n/t \rfloor + 1)^{n - \lfloor n/t \rfloor t}, \tag{7}$$

and this satisfies the condition for the application of Grover's search. So they apply Grover's seach to Byskov's algorithm of MIS enumeration, which gives time complexity $O^*(3^{n/6})$.

Here, computed chromatic numbers are stored to QRAM. They apply Grover's search again for computing the minimum in Equation 4. Now note that the precomputation requires the running time $O^*\left(\sum_{i=1}^{n/4} \binom{n}{i} 3^{i/6}\right) = O^*(1.8370^n)$. Then, the main part of the algorithm computes $\chi(G)$ by using the formula

$$\chi(G) = 1 + \min_{I \in \mathrm{MIS}(G)} \min_{S \subseteq V\backslash I, |S| \leq n/2, |V\backslash I\backslash S| \leq n/2} \{\chi(G[S]) + \chi(G[V\backslash I\backslash S])\}$$

for $\chi(G) \geq 3$. The above formula is justified by Fact 3.

Now each graph at most has $n/4$ vertices, and they start the main part of the algorithm to do the conquering. They utilize these precomputations to come up with the chromatic number for larger graphs. The running time of this conquering part is

$$O^* \left(3^{n/6} \sqrt{\binom{n}{n/2}} 3^{n/12} \sqrt{\binom{n/2}{n/4}} \right) = O^*(2.2134^n).$$

Then the overall naive quantum dynamic programming algorithm for graph coloring is

$$O^*(2.2134^n) + O^*(1.8370^n) = O^*(2.2134^n).$$

But we would like to point out that if we directly adopt their idea of enumerating MIS with running time $O^* 3^{1/6}$ in the classical dynamic programming algorithm, and keep all left, we would have graph coloring solved in $O^*((1 + 3^{1/6})^n) = O^*(2.2009^n)$, which is smaller than the above time.

However later, Kazuya and Ryuhei improve their usage of divide-and-conquer algorithm described above. Their improved algorithm (stated in Theorem 1 in [13]) searches all MISs of size t for each $t \in [1, n]$ separately and calculates each enumeration of t-MISs in time $O^*(I(n, t))$ (defined in Equation 7), instead of directly using $O^*(3^{n/6})$.

Finally, they give a precise argument showing this variation improves the running time from $O^*(2.2134^n)$ to $O^*(1.9140^n)$.

4 Minimum Clique Cover

In graph theory, a **clique cover** of a given undirected graph is a partition of the vertices of the graph into cliques, subsets of vertices within which every two vertices are adjacent. Any undirected graph $G = (V, E)$ has a clique cover since G can be covered with $|V|$ cliques where each clique is a single vertex in V. A **minimum clique cover** of an undirected graph is a clique cover that uses as few cliques as possible. The minimum k for which a clique cover exists is called the clique cover number of the given graph.

4.1 Classical Computational Complexity

The clique cover problem in computational complexity theory is the algorithmic problem of finding a minimum clique cover, or finding a clique cover whose number of cliques is below a given threshold. Finding a minimum clique cover is NP-hard, and its decision version is NP-complete. It was one of Richard Karp's original NP-complete 21 problems[8]. It has been proved that minimum clique cover has the same complexity with the general graph coloring problem (chromatic number problem)[10]. Lawler[9] has shown that the chromatic number problem and so does the clique cover problem can be solved in $O(2.44^n)$ by dynamic programming.

Let $\theta(G)$ be the clique cover number of an undirected graph $G = (V, E)$ and $|V| = n$. Let $\theta(G) = 0$ if G is empty. The key idea of solving clique cover problem with dynamic programming is using the fact below

$$\theta(G) = 1 + \min_{S \in \mathcal{I}(G)} \theta(G[V \backslash S]) \tag{8}$$

where $\mathcal{I}(G)$ is the family of vertices subset of all maximal cliques in G. A clique of G is maximal if it is not a proper subset of a larger clique of G. $G[V \backslash I]$ is as defined above. For every subset $X \subset V$, we can thus compute the clique cover number $\theta(G[X])$ with Equation 8 recursively with dynamic programming until X is empty.

4.2 Quantum Algorithm for Clique Cover

Similar to the idea in graph coloring, the algorithm should be based on Equation 8. There also exists a classical proof proving that there are at most $O^*(2^{n/3})$ maximal cliques in a n-vertex graph [12], and a classical algorithm (Bron-Kerbosch algorithm) that enumerating all maximal cliques in $O^{n/3}$. Then we would like to investigate quantum speedup on maximal clique enumeration to find the clique cover number of G.

There is a paper designing a quantum circuit to construct all maximal cliques using Grover's search [15]. Its construction realizes maximal cliques enumeration in time $O^*(\sqrt{\frac{n^2 2^n}{M}})$, where M is the total number of maximal cliques. Note that unless we know M, we would always take $M = 1$ for upper bounds, and then it is the same as the quantum brute-force algorithm, i.e. $\sqrt{n^2 2^n}$.

We then run the above algorithm of maximal cliques enumeration for all induced subgraphs and then finally compute the clique cover number based on Equation 8. The time complexity for this naive idea is $O(\sum_{k=1}^{n} \binom{n}{k} \sqrt{n^2 2^n}) = O^*(2.414^n)$.

For other ideas of quantum dynamic programming, as a maximal clique of G is a maximal independent set of $\bar{G}$, all maximal cliques can be enumerated in time $O^*(3^{n/6})$ quantumly after finding $\bar{G}$ which is poly-time. Time complexity for this idea is $O^*(1.9140^n)$, the same as Kazuya et al.'s [13] (i.e. Ambainis et al.'s [1]) idea.

Also as $\chi(G) = \theta(\bar{G})$, the cliue cover number problem for G can be directly reduced to the graph coloring problem for $\bar{G}$, essentially the same as the idea above. So the time complexity is $O^*(1.9140^n)$ for this idea too.

5 Dominating Set

A dominating set for a graph $G = (V, E)$ is a subset D of V such that every vertex not in D is adjacent to at least one member of D. The task of the dominating set problem is to find the domination number $\gamma(G)$, i.e. the cardinality of a minimal dominating set of G.

5.1 Classical Time Complexity

First note that the brute-force algorithm inspecting all vertex subsets is able to find $\gamma(G)$ in running time $O(n2^n)$. Current state-of-art work algorithm, using $O(1.5048^n)$ time was found in [14], who also show that the number of minimum dominating sets can be computed in this time.

We now consider the time complexity of the classical dynamic programming algorithm. The method is based on the following equation

$$\gamma(G) = 1 + \min_{S=\{g \text{ and its induced vertices } | g \in V\}} \gamma(G[V \backslash S]). \tag{9}$$

There are n such S and enumeration is within $O(n)$. So the total time needed for the algorithm is $O(\sum_{k=1}^{n} \binom{n}{k} k) = O(n2^n)$.

5.2 Quantum Time Complexity

If we directly apply Grover's algorithm on the brute-force algorithm, i.e. using Grover's search to search over all 2^n subsets of vertices to find the smallest set of vertices that can dominate the graph, we will get the time complexity $O^*(\sqrt{2^n}) = O^*(1.414^n)$.

Also if we consider applying Grover in the searching in Equation 9, we would reduce the searching time over all $S = \{g$ and its induced vertices $|g \in V\}$ from $O(n)$ to $O(\sqrt{n})$. But we still need to do this step for all induced subgraphs with number of vertices $k \in [1, n]$, which gives the whole time complexity $O\left(\sum_{k=1}^{n} \binom{n}{k} \sqrt{k}\right) = O^*\left(2^n\right)$.

Besides the above two naive algorithms, we also consider applying Ambainis' divide-and-conquer dynamic programming ideas on the dominating set problem. Kann [7] has shown that the minimum dominating set problem and the minimum set cover problem are equivalent under L-reductions. From this fact and the corresponding construction, we can reduce the dominating set problem to the minimum set cover problem in polynomial time. This gives a bound on the time complexity, which is the same as that for the minimum set cover problem, $O^*(1.728^n)$.

So when we compare all these three algorithms, we observe that quantum brute-force algorithm has the smallest time complexity. Directly applying Ambainis' quantum dynamic programming idea would not help improve the time complexity.

Note that minimum set cover problem is actually harder than the dominating set problem even if they are equivalent under L-reductions. From the reduction of set covering to dominating set, for a collection S with $|S| = m$ and a universe U with $|U| = n$, it can be reduced in polynomial time to a dominating set problem of $G = (V, E)$ with $|V| = m + n$.

Therefore, although we can reduce a minimum set cover problem to a dominating set problem, the resulting brute-force algorithm for solving the dominating set problem will run in $O^*(2^{m+n})$. Then the quantum brute-force algorithm by applying Grover has time complexity $O^*(2^{\frac{m+n}{2}})$.

So the performance of Ambainis' dynamic programming algorithm and quantum brute-force algorithm depends on the value of m. By simple calculation, we would get that when $m > 0.5782n$, Ambainis' dynamic programming algorithm would win.

6 Cases where New Ideas are Needed

The dominating set problem described above is a case such that the direct application of divide-and-conquer dynamic programming performs worse than direct application of Grover's search on classical brute-force algorithm.

The minimum vertex cover problem is also such problem. A vertex cover of a graph is a set of vertices that includes at least one endpoint of every edge of the graph. Then a minimum vertex cover is a vertex cover of the smallest possible size. The task of this problem is to find this vertex cover.

There exists a polynomial time reduction from the minimum vertex cover problem to the dominating set problem, and also a polynomial reduction to the minimum set cover problem. This raises the same problem as discussed above and we need to adopt other quantum advantages and apply new ideas in dynamic programming to make improvements.

Even though we only have shown two cases, we would like to propose that if a problem can be reduced to the dominating set problem in polynomial time, then directly applying Ambainis' dynamic programming idea will not win over quantum brute-force algorithms.

7 Summary and Further Direction

In summary, we have reviewed Ambainis et al.'s divide and conquer dynamic programming idea on several NP-hard problem, including the Hamiltonian cycle problem, the TSP and the minimum set cover problem.

We then discuss about the application of this idea on some further NP-hard problem including graph coloring, minimum clique cover and minimum dominating set. Ambainis et al.'s idea can improve the time complexity for both the graph coloring problem [13] and the minimum clique cover problem from $O^*(2.4423^n)$ classically to $O^*(1.9140^n)$ quantumly.

We also have compared the complexity of applying different quantum methods on these NP-hard problems and proposed cases including the dominating set problem and the minimum vertex cover problem where new ideas are needed to utilize the power of dynamic programming.

For further direction, one might give a more general divide-and-conquer dynamic programming scheme in solving high-level graph numbering problems, i.e. chromatic number, clique cover number.

One could study how such divide and conquer dynamic programming idea can be applied to other NP-hard problems outside graph problems. Also, there exists some work proposing new quantum walks for subset-sum, which improves the time complexity [3]. Then similarly, one can further study any other adaptions of quantum ideas in dynamic programming besides Grover.

References

[1] Andris Ambainis, Kaspars Balodis, Jānis Iraids, Martins Kokainis, Krišjānis Prūsis, and Jevgēnijs Vihrovs. Quantum speedups for exponential-time dynamic programming algorithms. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1783–1793. SIAM, 2019.

[2] Richard Bellman. Dynamic programming treatment of the travelling salesman problem. *Journal of the ACM (JACM)*, 9(1):61–63, 1962.

[3] Xavier Bonnetain, Rémi Bricout, André Schrottenloher, and Yixin Shen. Improved classical and quantum algorithms for subset-sum. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 633–666. Springer, 2020.

[4] Jesper Makholm Byskov. Enumerating maximal independent sets with applications to graph colouring. *Operations Research Letters*, 32(6):547–556, 2004.

[5] Vittorio Giovannetti, Seth Lloyd, and Lorenzo Maccone. Quantum random access memory. *Physical review letters*, 100(16):160501, 2008.

[6] Michael Held and Richard M Karp. A dynamic programming approach to sequencing problems. *Journal of the Society for Industrial and Applied mathematics*, 10(1):196–210, 1962.

[7] Viggo Kann. *On the approximability of NP-complete optimization problems.* PhD thesis, Citeseer, 1992.

[8] Richard M Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. Springer, 1972.

[9] Eugene L Lawler and LAWLER EL. A note on the complexity of the chromatic number problem. 1976.

[10] Harry R Lewis. Computers and intractability. a guide to the theory of np-completeness, 1983.

[11] Masayuki Miyamoto, Masakazu Iwamura, Koichi Kise, and François Le Gall. Quantum speedup for the minimum steiner tree problem. *arXiv preprint arXiv:1904.03581*, 2019.

[12] John W Moon and Leo Moser. On cliques in graphs. *Israel journal of Mathematics*, 3(1):23–28, 1965.

[13] Kazuya Shimizu and Ryuhei Mori. Exponential-time quantum algorithms for graph coloring problems. In *Latin American Symposium on Theoretical Informatics*, pages 387–398. Springer, 2021.

[14] Johan MM van Rooij, Jesper Nederlof, and Thomas C van Dijk. Inclusion/exclusion meets measure and conquer. In *European Symposium on Algorithms*, pages 554–565. Springer, 2009.

[15] Chu Ryang Wie. A quantum circuit to construct all maximal cliques using grover search algorithm. *arXiv preprint arXiv:1711.06146*, 2017.