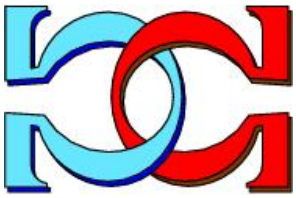

CDMTCS
Research
Report
Series

Solving the Hamiltonian Cycle Problem using a Quantum Computer

Michael J. Dinneen
Anuradha Mahasinghe
Richard Hua
Rajni Goyal

Department of Computer Science
University of Auckland
Auckland, New Zealand

CDMTCS-521
February 2018

Centre for Discrete Mathematics and
Theoretical Computer Science

Solving the Hamiltonian Cycle Problem using a Quantum Computer

Michael J. Dinneen, Anuradha Mahasinghe*, Richard Hua and Rajni Goyal†

Department of Computer Science, University of Auckland, New Zealand

Abstract

We review existing quantum computational methods for solving the Hamiltonian cycle problem in different computational frameworks such as quantum circuits, quantum walks and adiabatic quantum computation. Then we present a QUBO (quadratic unconstrained binary optimization) formulation, which is suitable for the adiabatic quantum computation for a D-Wave architecture. Further, we derive a physical Hamiltonian from the QUBO formulation and discuss its adequateness in the adiabatic framework. Finally, we discuss the complexity of running the Hamiltonian cycle QUBO on a D-Wave quantum computer, and compare it with existing quantum computational methods.

1 Introduction

The Hamiltonian cycle problem (HCP) is one of the extensively studied problems in graph theory and computer science. A Hamiltonian cycle is a cycle in a graph which passes through every vertex of it. Given a graph, determining whether the graph contains a Hamiltonian cycle is known as the HCP. This problem dates back to 1850's where its origin is considered the famous Icosian game proposed by Sir William Rowan Hamilton. The HCP is a well-known problem in the list of NP-complete problems [36]. The original HCP was proposed for undirected graphs, though the directed graph version is also extensively studied, and these two problems are closely related. For simplicity, we consider the undirected graph version of the HCP here.

Solving the HCP has always been a problem of interest. Different solution methods were proposed in different computational frameworks. In the scope of conventional computing, dynamic programming [6, 32], Markov chains [30] and Monte-Carlo methods [9] are noteworthy among these. On the other hand, The problem was also addressed in unconventional

*Visiting from Department of Mathematics, University of Colombo, Sri Lanka.
†Visiting from Department of Mathematics, University of Delhi, India.

computing paradigms [14] in the last decade. Accordingly, methods were proposed to solve the HCP in DNA computing [38], optical computing [43] and quantum computing [28, 39].

Amongst unconventional computing frameworks, quantum computing could be considered the field which drew the attention of the widest-ranging community, including computer scientists, physicists, mathematicians and engineers. Though it is still debatable whether quantum computers can solve NP-complete problems in polynomial time, they have provided exponentially faster solutions to several problems and led to a diverse range of applications [17, 29, 40]. Different quantum algorithms were developed in different quantum frameworks with promising solutions for several real-world problems. These include the quantum circuit framework, the quantum walk framework and the adiabatic quantum framework.

Since quantum circuit framework was the pioneering model for quantum computation, early quantum algorithms were designed over circuit architectures. Being the quantum counterpart of classical random walks, quantum walks also have provided bases for several quantum algorithms. However, solving NP-complete problems remained unanswered in both frameworks. A new paradigm of quantum computing, designed after the adiabatic theorem in quantum physics, was proposed in [25], which particularly aimed at solving NP-hard problems.

Considering the physical implementation of a quantum computer, significant progress was achieved after the invention of the D-Wave computer, which was modeled after the adiabatic quantum annealing. Despite the fact that the D-Wave machine is not a universal quantum computer, it has provided efficient solutions for several instances of NP-hard problems[15, 16]. The operation of this computer is based on simulating a physical Hamiltonian, which is derived from a quadratic unconstrained binary optimization (QUBO) problem. Therefore, problems intended to be solved by the D-Wave quantum computer are usually transformed into QUBO problems. However, not all QUBOs provide efficient solutions. Thus, given a problem, reformulating it as a D-Wave feasible QUBO becomes a non-trivial task.

Given that the HCP has been addressed previously in other unconventional computing paradigms including quantum computing, it is natural to ask how this problem should be formulated for the D-Wave quantum computer. One of the main tasks of this paper is to formulate the HCP as a QUBO which could efficiently work in the D-Wave architecture. Thus, we provide essential ingredients for a quantum algorithm for the HCP in the adiabatic quantum framework.

However, unlike other quantum computing frameworks, computational complexity of an adiabatic quantum algorithm is not explicitly pre-determined. It has been found that adiabatic algorithms could fail in certain circumstances [3, 24]. Also certain theoretical bounds for adiabatic algorithms were suggested [35]. Therefore, a QUBO formulation which perfectly suits the D-Wave architecture can be exponential in complexity due to physical effects. Thus, after formulating a QUBO, it is worthwhile to explore the corresponding physical Hamiltonian's capacity to produce efficient solutions. The second task of this paper is building a Hamiltonian based on our QUBO for the HCP in order to investigate its adequateness.

Having different quantum methods for solving the HCP, a comparative survey would provide some intuition on which framework is more suitable for solving it. However, comparing

adiabatic algorithms with other quantum algorithms could be questionable, as the computational complexity of the latter is defined explicitly while the former is not. A comparison is justifiable only with certain bounds on the complexity of the adiabatic method. Therefore, we use our QUBO formulation and its corresponding physical Hamiltonian to compare our method with already existing quantum methodsfor solving the HCP. Accordingly, the third task of this paper is putting together the existing quantum computational methods for solving the HCP along with our new method and making the way for an equitable comparison.

Since many quantum methods for HCP are based on the brute-force classical method, we describe the quantum version in Section 2.1 of this paper. Afterwards, starting in Section 2.2 we describe the standard quantum approach for HCP with its implementation in quantum circuit, quantum walk and adiabatic quantum frameworks. In Section 3 we present our QUBO formulation. Further, this section contains our results on the QUBO's capability of producing adequate physical Hamiltonians. Finally, we compare our results with other quantum computational methods for solving the HCP.

2 Existing quantum methods

2.1 Quantum brute-force method

The most known quantum method for solving the HCP is the quantum brute-force algorithm, which is known to provide a quadratic speed-up over the its classical counterpart. This method is based on famous Grover's algorithm which makes use of amplitude amplification as the key ingredient. As described in [42], it is can be easily reconstructed from Grover's algorithm for searching a database with a given number of entries. We briefly describe this method for completion.

Classical brute-force method:

It is straightforward to see that the HCP has a polynomial-time verifier. On an input graph G on n vertices, the brute-force method enumerates all cycles of this graph. There are $n!$ possible arrangements of vertices of the graph. Given such arrangement, verifying whether it contains the Hamiltonian cycle can be done using a $p(n)$-time polynomial verifier. Thus, the brute force method has complexity $p(n)n!$. In order to express this complexity as a power of 2 for the convenience of comparison, it is possible to use of Stirling's formula

$$\lg n! = n \lg n - (\lg e)\, n + O(\lg n)\,, \tag{1}$$

and to show that the complexity is $O\left(p(n)2^{n \lg n}\right)$.

Quantum speedup over classical brute-force method:

On an input graph G on n vertices, we index the vertices by $v_0, v_1, \ldots, v_{n-1}$. The state space in this quantum algorithm consists of all possible orderings of the vertices of this graph. In order to prepare this space, we represent each vertex by the quantum state $|v_i\rangle$, which is the ith element of the computational basis. Thus, each vertex can be stored using $\log n$ qubits.

For simplicity, we assume n to be a power of 2. If this is not the case, it is possible to prepare $2^{\lceil \lg n \rceil}$ states and make it a power of 2.

Now we permute the vertices and make the complete state space. More precisely, we make a composite system in which an element takes the form $|v_0, v_2, \ldots, v_{n-1}\rangle$, the tensor product of vertex states which is a Hilbert space spanned by $2^{n \log n}$ orthonormal states. Physically, each vertex state occupies $\log n$ qubits; making the size of the composite system equal to $n \log n$ qubits. This is the *search space* on which we apply Grover's algorithm.

We apply the brute-force search with polynomial verifier (with run-time $p(n)$) for the Hamiltonian cycle. That is, for each entry in the search space, we query an oracle which has access to the polynomial verifier for Hamiltonian cycles. This oracle returns different outputs based on the existence of Hamiltonian cycle in the relevant state. The action of the oracular operator is expressible as follows.

$$O|v_0, v_1, \ldots, v_{n-1}\rangle = \begin{cases} -|v_0, v_1, \ldots, v_{n-1}\rangle & \text{if } v_0, v_1, \ldots v_{n-1} \text{ is a Hamiltonian cycle} \\ |v_0, v_1, \ldots, v_{n-1}\rangle & \text{otherwise} \end{cases} . \quad (2)$$

It is a well-known fact that the speedup in Grover's algorithm over classical algorithms is achieved by simultaneous evaluation of a function at all entries in a search space. Thus, the amplitudes of the states in which the vertices make a Hamiltonian cycle can be increased by repeated application of Grover's diffusion operator and the oracular unitary operation.

Given a state $|x\rangle$ in the search space of $2^{n \lg n}$ entries, the action of Grover's diffusion operator on a superposition of states $\sum_{x=1}^{2^{n \lg n}} c_x |x\rangle$, can be expressed as follows.

$$U_s \left(\sum_{x=1}^{2^{n \lg n}} c_x |x\rangle \right) = \sum_{x=1}^{2^{n \lg n}} \left(\frac{2}{2^{n \lg n}} \left(\sum_{y=1}^{2^{n \lg n}} c_y \right) - c_x \right) |x\rangle \quad (3)$$

The operation in Equation (3) can be viewed as the rotation of the current state about the current mean (with respect to amplitude) state. On the other hand, Equation (2) can be considered a rotation about the solution; that is, rotation about an ordering of vertices which makes a Hamiltonian cycle in the graph.

In Grover's algorithm, a function is calculated on all values in the search space simultaneously, through a black box function operator. This is the operator given by Equation (2) for the HCP. Thus, an amplitude change of the states which contain a Hamiltonian cycle has been done by the oracular operator. Then, Grover's diffusion operator in Equation (3) rotates the state of the composite system about its mean, amplifying these amplitudes significantly. It has been proven by Lov Grover in [27] that it is sufficient to apply these black box and diffusion operators $O(\sqrt{N})$ times on a database with N elements, in order to find the searched-for element.

The amplitude amplification of the composite vertex states which contains a Hamiltonian cycle is followed by the well-known proof of Grover's algorithm. Recall the search space is spanned by $2^{n \lg n}$ orthonormal states. Therefore we apply Grover's iteration $O\left(2^{\frac{n \lg n}{2}}\right)$ times,

making the time complexity of the Hamiltonian cycle finding algorithm of the framework equal to $O\left(p(n)2^{\frac{n\lg n}{2}}\right)$. This is a quadratic speedup over classical brute-force algorithm.

Though exponential in complexity, this quantum brute-force method has been the motivation for a number of proposed quantum approaches and remains the standard quantum method for solving the HCP.

2.2 Quantum circuit implementation

Since quantum effects of this algorithm are entirely due to Grover's algorithm, the standard quantum circuit implementation for Grover's algorithm can be used to implement this in the circuit framework.

It must be noted that the problems of determining whether a given graph is Hamiltonian and finding all Hamiltonian cycles are related but slightly different in both classical and quantum frameworks. If the number of Hamiltonian cycles were known prior to implementation, standard Grover's algorithm can be used for amplitude amplification. Since this is not our case of interest, the unknown number of solutions version of Grover's algorithm (also known as *quantum counting*) in [11] must be used. This method provides a solution to the question whether the graph is Hamiltonian, in addition enumerates its Hamiltonian cycles. However, if we only need to determine whether the graph is Hamiltonian, quantum counting algorithm might not be optimal. The modification to Grover's search suggested as *quantum existence testing* in [34] could be used to circumvent this issue.

Though the quantum brute-force method has complexity $O\left(p(n)2^{\frac{n\lg n}{2}}\right)$ and it is implicitly designed for the quantum circuit framework, we found another work suggesting a method for finding the Hamiltonian cycle in [28] defined on the same framework. This method is based on polynomial reduction of 3-SAT to HCP. Based on Grover's algorithm, the authors claim a complexity $O\left(p(n)2^{\frac{n(n-1)}{2}}\right)$, which is however higher than the complexity of the standard quantum brute-force method.

2.3 Quantum walk implementation

The complexity in the circuit framework is defined in terms of the length of basic quantum gates involved. Thus, one undeniable advantage of the circuit framework is its explicit representation of the algorithm, which could easily be converted into mathematical expressions. This makes the complexity analysis easier than the other frameworks of quantum computation. On the other hand, it should also be mentioned that quantum circuits were experimented only on a few qubits [5]. Contrastly, physicists have progressed significantly in the quantum walk framework with more qubits, with strong experimental evidence on computational capacity [49].

Discrete-time quantum walk takes place in the Hilbert space $\mathcal{H}$, comprised of the tensor product of the space spanned by orthonormal set of vertex states and the coin Hilbert space spanned by orthonormal set of coin states, with d_i the degree of the ith vertex. The discrete-

time quantum walk on the graph is then achieved via repeated application of the unitary time-evolution operator

$$\hat{U} = \hat{S} \cdot \left(1 \otimes \hat{C}\right), \tag{4}$$

where C is the coin operator, and S is the shift operator acting on the Hilbert space $\mathcal{H}$ as

$$\hat{S}|v_i, c_j\rangle = |v_j, c_i\rangle. \tag{5}$$

The basic idea of quantum walk-based search is applying the coin version of Grover's diffusion operator in Equation (3). The d-degree Grover coin operator $\hat{C}_G^{(d)} \in \mathbb{R}^{d\times d}$ is defined as follows,

$$(\hat{C}_G^{(d)})_{ij} = \frac{2}{d} - \delta_{ij} \tag{6}$$

Applying the d-dimensional Grover coin operator, combined with the shift operator, to a d-degree regular graph. Then the discrete time quantum walk unitary evolution operator is expressible as,

$$(U)_{ij,kl} = \left[\hat{S} \cdot \left(1 \otimes \hat{C}_G^{(d)}\right)\right]_{ij,kl} = \begin{cases} \frac{2}{d_j} - \delta_{il}, & \text{if } j = k \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

where $(i, j), (k, l) \in E$ and d_j denotes the degree of the jth vertex. Quantum walk-based search could be considered a walk-version of Grover's search. In this perspective, solving the HCP in quantum walk framework was considered in [22] as an application of walk-based search.

Grover-based quantum search in a d-dimensional grid on N vertices has time complexity $O\left(N^{\frac{1}{2}+\frac{1}{d}}\right)$ as shown in [7]. This was improved to $O\left(N^{\frac{1}{2}}\right)$ in [1]. Quantum search over the hypercube was provable to be performed in $O\left(N^{\frac{1}{2}}\right)$ [31]. A particular class of graphs on which quantum walks were performed successfully is the Johnson graphs. The vertex set of the Johnson graph $J(n, r)$ represents the r-element subsets of an n-element set, and two vertices are adjacent if and only if corresponding intersection set has exactly r elements. The fact that Grover's quantum search can be performed faster on Johnson graphs has been identified in [4].

Based on this, improvement on the search has been suggested in [22] for the Hamiltonian cycle problem. Using previously known results on quantum walk-based search on Johnson graphs, it was suggested to use a Johnson graph as the search space in the brute-force search in order to achieve better speedups over the classical algorithms. The run-time of a quantum walk over some graph depends on the largest and second largest eigenvalue of an ergodic Markov chain. The main idea of the proposal in [22] is to make use of the eigenvalue results on Johnson graphs in [12] for quantum walk-based search. More precisely, if the quantum search for a Hamiltonian cycle takes place on Johnson graph $J\left(n^2, n^{\frac{2n}{n+1}}\right)$, then quantum query complexity can be reduced to $O\left(n^{\frac{2n}{n+1}}\right)$.

2.4 Adiabatic quantum implementation

The relatively new adiabatic quantum computing framework was introduced in [25] as an alternative to the circuit framework. This framework makes use of the adiabatic theorem in quantum mechanics as the key ingredient. While circuit framework is based on Heisenberg–Born picture of quantum mechanics, the adiabatic framework is based on Schrödinger picture. According to the Heisenberg–Born matrix formulation, the (discrete) time evolution of a quantum system is characterized by the application of a unitary matrix, which is an individual *gate* in a quantum circuit. Contrastly, the Schrödingers equation describes the (continuous) time evolution of a quantum system as the application of the Hermitian operator which acts as the energy Hamiltonian of the system.

The adiabatic framework is based on preparing this Hamiltonian. In this preparation process, it uses the adiabatic theorem [10] as the key ingredient. This theorem suggests that, if a quantum system is its ground (zero-energy) state and undergoes a sufficiently slow evolution, then the system will end up in high probability at a ground state. The adiabatic framework relies on preparing a ground state which encodes the solution to an NP-hard problem in a *problem* Hamiltonian. During the adiabatic process, an easy-to-prepare *initial* Hamiltonian is slowly evolved along an adiabatic path to the problem Hamiltonian. Thus, an adiabatic quantum algorithm consists of three components: an initial Hamiltonian H_I, a problem Hamiltonian H_P and an adiabatic evolution path $s(t)$ from the starting time $t = 0$ to terminating time $t = T$ [41]. The physical Hamiltonian $H(s)$ in the adiabatic evolution can be expressed as

$$H(s) = sH_I + (1 - s)H_P, \tag{8}$$

where $s(t)$ is a smooth function that decreases from 1 to 0 over $[0, T]$. The natural selection for the adiabatic path is the linear curve $1 - \frac{t}{T}$.

After the introduction of the adiabatic quantum framework, analogous algorithms were designed for certain quantum algorithms in the circuit framework such as adiabatic Grover's algorithm [26], period finding [33], Deutschs algorithm [20] and factorization [44].

Although polynomial equivalence of the circuit and adiabatic frameworks has been proved under certain conditions [2, 25], the run-time of adiabatic algorithms can become exponential due to physical effects. For instance, the quadratic speedup in Grover's algorithm in the circuit version does not apply equally in general for its analogous adiabatic version. It was proved in [45] that ordinary adiabatic Grover's algorithm has an exponentially small spectral gap, and hence ends up with no speedup over classical search.

Therefore, an attempt to solve for the HCP in the adiabatic framework using adiabatic Grover's algorithm has the above mentioned disadvantage, and ends up with complexity $O\left(2^{n \lg n}\right)$ unless the modification in [45] is applied to adiabatic Grover's algorithm. This is the same as classical brute-force complexity. Therefore, in order to reduce complexity of adiabatic HCP to $O\left(p(n)2^{\frac{n(n-1)}{2}}\right)$, the modified Grover's algorithm must be applied.

3 Solution for a D-Wave architecture

In the preceding section, we have discussed already proposed quantum methods for solving the HCP in the circuit, walk and adiabatic frameworks. Now, we propose a different quantum method for solving the HCP. Although our method is designed for the adiabatic framework, it is based on optimization rather than amplitude amplification. Our method is particularly designed for the D-Wave quantum setting, which belongs to the adiabatic framework. In the adiabatic framework, quantum annealing is a meta-heuristic for finding the global minimum of a given objective function over a given set of candidate solutions. The D-Wave quantum computer can be considered a hardware heuristic which minimizes an unconstrained objective function using a physically realized version of quantum annealing [37]. One form of problems the D-Wave quantum computer is claimed to solve efficiently is QUBO (quadratic unconstrained binary optimization) problems [21].

QUBO is an NP-hard mathematical problem consisting in the minimization of a quadratic objective function $f(\mathbf{x}) = \mathbf{x}^T Q \mathbf{x}$, where $\mathbf{x}$ is a n-vector of binary variables and Q is an upper-triangular $n \times n$ matrix:

$$x^* = \min_{\mathbf{x}} \sum_{i \leq j} x_i Q_{(i,j)} x_j, \text{ where } x_i \in \{0, 1\}. \tag{9}$$

For notational convenience and consistency, we will index i and j by integers in range from 0 to $n - 1$.

Since the invention of the D-Wave computer, several decision and optimization problems were reformulated in D-Wave feasible formats. Examples include broadcast-time [15], Boolean satisfiability [46], graph isomorphism [16], graph covering [21], graph partitioning [47] and job-shop scheduling [48] problems. It is well-known that many real problems have integer programming formulations. Further, it is possible to convert an integer programming problem to a QUBO problem by a simple transformation. However, not all QUBOs generate accurate results in D-Wave architecture. Adding slack variables to a QUBO problem could makes the computation inaccurate [13], thus the generic conversion from an integer program is very likely to produce inaccurate solutions. Also it is quite natural to try and use a small number of variables. Thus, it becomes a non-trivial task to reformulate a problem as a D-Wave feasible QUBO.

3.1 QUBO formulation of HCP

Given a graph $G = (V, E)$ of order n, a Hamiltonian cycle can be interpreted as a permutation of its n vertices, say $v_0, v_1, \ldots, v_{n-1}$, such that for all consecutive pairs of vertices v_i, v_{i+1} in the permutation, we have an edge (v_i, v_{i+1}) in E. We also require $\{v_0, v_{n-1}\}$ to be an edge in E so we could have a complete cycle. Since there are $n!$ permutations, if we assume that all variables are binary, then we need $\min\{k \mid 2^k \geq n!\} = \lceil \lg(n!) \rceil$ binary variables to represent any of the $n!$ permutations, that is about $n\lceil \lg n \rceil$ bits. However, this theoretical lower bound seems difficult to be realized and we provide a QUBO formulation of HCP with n^2 variables. Our construction is similar to the one sketched by Lucas in [39].

We consider the arrangements of vertices in graph G and define a binary variable $x_{i,j}$ with vertex index $i \in \{0, 1, \ldots n-1\}$ and position index $j \in \{0, 1, \ldots n-1\}$ as follows.

$$x_{i,j} = \begin{cases} 1 & \text{if vertex } i \text{ is at position } j \text{ of the sequence} \\ 0 & \text{otherwise} \end{cases}. \tag{10}$$

Let $\mathbf{x}$ denote the binary vector consisting of all n^2 variables. We also need some mechanism to map a given $\mathbf{x}$ to a vertex sequence by the definition of Equation (10). For notational convenience, we will use a mapping $d(\mathbf{x})$ to denote the sequence of vertices. This sequence will have an undefined value if $\mathbf{x}$ can not be decoded in a meaningful way (e.g. two vertices appearing at the same index).

The objective function to be minimized has the following form:

$$F(\mathbf{x}) = H(\mathbf{x}) + P_1(\mathbf{x}) + P_2(\mathbf{x}) \tag{11}$$

where,

$$H(\mathbf{x}) = \sum_{(i_1,i_2)\in V\times V - E(G)} \left(x_{i_1,0} x_{i_2,n-1} + \sum_{j=0}^{n-2} x_{i_1,j}\ x_{i_2,j+1} \right), \tag{12}$$

$$P_1(\mathbf{x}) = \sum_{i=0}^{n-1} \left(1 - \sum_{j=0}^{n-1} x_{i,j} \right)^2 \tag{13}$$

and

$$P_2(\mathbf{x}) = \sum_{j=0}^{n-1} \left(1 - \sum_{i=0}^{n-1} x_{i,j} \right)^2. \tag{14}$$

Each of the three terms of $F(\mathbf{x})$, serves as a penalty term which becomes strictly positive only when the condition of Hamiltonicity is violated. Equation (13) and (14) ensure that $d(\mathbf{x})$ is a permutation of n vertices, while Equation (12) check whether all pairs of consecutive vertices in the sequence have an edge connecting them. Together, they impose all the necessary conditions for a Hamiltonian cycle.

3.2 Proof of correctness

Given a graph G, if we assume $\mathbf{x}^*$ is the vector of n^2 binary variables which yields the optimal value $x^* = \min F(\mathbf{x})$, then G has a Hamiltonian cycle if and only if $x^* = 0$. If $x^* = 0$, we can also find the sequence of vertices which form the cycle by computing $d(\mathbf{x})$. Formally, we have the following theorem.

Theorem 1. *Assume $\mathbf{x}^*$ and x^* are the optimal variable assignments and its value in $F(\mathbf{x})$, respectively. Then, G has a Hamiltonian cycle if and only if $x^* = 0$. Furthermore, the sequence of vertices which forms the cycle is $d(\mathbf{x})$.*

It is easier to prove the correctness of Theorem 1 with the following two lemmata.

Lemma 1. *For each possible variable assignment* $\mathbf{x}$, $P_1(\mathbf{x}) + P_2(\mathbf{x}) = 0$ *if and only if* $d(\mathbf{x})$ *is a permutation of* n *vertices.*

Proof. Since both P_1 and P_2 are sums of quadratic functions, $P_1(\mathbf{x}) + P_2(\mathbf{x}) = 0$ if and only if $P_1(\mathbf{x}) = 0$ and $P_2(\mathbf{x}) = 0$.
First, by construction of $P_1(\mathbf{x})$, $\sum_{i=0}^{n-1}(1 - \sum_{j=0}^{n-1} x_{i,j})^2 = 0$ if and only if for each vertex i in range $0 \leq i \leq n-1$, $1 - \sum_{j=0}^{n-1} x_{i,j} = 0$. So it means that exactly one variable in the set $\{x_{i,j} \mid 0 \leq j \leq n-1\}$ has value 1. Hence each vertex must appear at exactly one position of the sequence defined by $d(\mathbf{x})$.

The second penalty term, $P_2(\mathbf{x})$ is of a similar structure. We have $\sum_{j=0}^{n-1}(1-\sum_{i=0}^{n-1} x_{i,j})^2 = 0$ if and only if for each j in range $0 \leq j \leq n-1$, there is exactly one variable in the set $\{x_{i,j} \mid 0 \leq i \leq n-1\}$ has a value of 1. Hence, there is only one vertex at each position index of the sequence defined by $d(\mathbf{x})$.
Together, the two conditions make sure that each vertex appears exactly once in the sequence and each index of the sequence has exactly one vertex assigned to it. By assumption, we assume that the number of vertices is the same as the length of $d(\mathbf{x})$ and therefore $d(\mathbf{x})$ is a permutation of n vertices. □

Lemma 2. *Let* $d(\mathbf{x})$ *be a permutation of* n *vertices, then* $H(\mathbf{x}) = 0$ *if and only if* $d(\mathbf{x})$ *is a cycle.*

Proof. Suppose $H(\mathbf{x}) = 0$ and consider two consecutive positions j and $j+1$. Let the vertices at these positions in the permutation $d(\mathbf{x})$ be i_1 and i_2 respectively. If $(i_1, i_2) \in V \times V - E(G)$, then $\sum_{j=0}^{n-2} x_{i_1,j}\ x_{i_2,j+1} > 0$. Then $H(\mathbf{x}) = 0$ by its definition, and leads to a contradiction. Therefore, we conclude that $(i_1, i_2) \in E(G)$.

A similar argument shows that the vertices at the two positions with indices 0 and $n-1$ also are adjacent. Therefore, it follows that the permutation $d(\mathbf{x})$ makes a cycle.

Now suppose $H(\mathbf{x}) \neq 0$. Therefore at least one term in the form $x_{i_1,j}x_{i_2,j+1}$ or $x_{i_1,0}x_{i_2,n-1}$ must be equal to one. That is, there exists at least two positions in the permutation $d(\mathbf{x})$ (including the first and last) such that the corresponding vertices $(i_1, i_2) \in V \times V - E(G)$. Thus, $d(\mathbf{x})$ is not a cycle. □

Now, we present the proof of Theorem 1.

Proof. The objective function $F(\mathbf{x})$ consists of three parts $H(\mathbf{x})$, $P_1(\mathbf{x})$ and $P_2(\mathbf{x})$. It is important to note that all three parts are made of a sum of quadratic terms. And since all variables are binary, we have $F(\mathbf{x}) = 0$ if and only if $H(\mathbf{x}) = P_1(\mathbf{x}) = P_2(\mathbf{x}) = 0$.

Given a graph G, let us construct the corresponding objective function $F(\mathbf{x})$. Suppose $x^* = 0$. This means that we must have $H(\mathbf{x}^*) = P_1(\mathbf{x}^*) = P_2(\mathbf{x}^*) = 0$ and hence by Lemma 1 and 2, we know $d(\mathbf{x}^*)$ must be a permutation of the n vertices of G which is also a cycle. That is, there exists a permutation $d(\mathbf{x}^*)$ of vertices of G which is a cycle. Therefore, the graph G is Hamiltonian and $d(\mathbf{x})$ is a Hamiltonian cycle.

Conversely, if $F(\mathbf{x}^*) \neq 0$ then at least one of the terms $H(\mathbf{x}^*)$, $P_1(\mathbf{x}^*)$ or $P_2(\mathbf{x}^*)$ is not equal to 0. If $P_1(\mathbf{x}^*)$ or $P_2(\mathbf{x}^*)$ is not equal to 0, then by Lemma 1 we know $d(\mathbf{x}^*)$ is not a

permutation of n vertices so it can not be a Hamiltonian cycle. If $P_1(\mathbf{x}^*) = P_2(\mathbf{x}^*) = 0$ and $H(\mathbf{x}^*) \neq 0$, then by Lemma 2, $d(\mathbf{x}^*)$ is not a Hamiltonian cycle again. □

3.2.1 A K_3 example

We now provide an example with the complete graph of order 3. The vertices of the graph are $V = \{0, 1, 2\}$ and the edges are $E = \{(0, 1), (0, 2), (1, 2)\}$, it can be visualized as a triangle where each corner is a vertex.

According to Equation (10), there will be nine variables $x_{i,j}$ for $0 \leq i \leq 2$ and $0 \leq j \leq 2$, so the first thing we do here is to map each variable to an index in range from 0 to 8. In theory, any arbitrary mapping is fine as long as the indexes and the variables have a one-to-one correspondence. Here, we define the mapping by a function $c(x_i, j)$ which maps variable $x_{i,j}$ to index $c(x_{i,j}) = 2i + j$.

Note that objective function (11) does not entirely fit the definition of the QUBO problem. Namely, there are two problems. First, there are constant terms in Equation (13) and (14). These constants can be safely ignored as removing a constant value from all different value assignment will not change the optimality of $\mathbf{x}^*$. However, it does affect x^*, and we get an offset of value 6. Second, there are linear terms in Equation (13) and (14). Since all variables are binary, we have $x_{i,j} = x_{i,j}^2$ so we can replace all linear terms by its square without changing the value of $F(\mathbf{x})$. We then compute the coefficient of all quadratic terms in the objective function. The coefficient of an quadratic term $x_1 x_2$ corresponds to the value of $Q_{(c(x_1),c(x_2))}$. The complete matrix is shown below.

$$Q(K_3) = \begin{bmatrix} -2 & 2 & 2 & 2 & 0 & 0 & 2 & 0 & 0 \\ 0 & -2 & 2 & 0 & 2 & 0 & 0 & 2 & 0 \\ 0 & 0 & -2 & 0 & 0 & 2 & 0 & 0 & 2 \\ 0 & 0 & 0 & -2 & 2 & 2 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & -2 & 2 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & -2 & 0 & 0 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & -2 & 2 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -2 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -2 \end{bmatrix}$$

Now K_3 obviously has a Hamiltonian cycle, however, if we consider the cycle as a permutation of the vertices then there are several different optimal solutions. Visually, the cycle can start at any corner of the triangle and then it can take two different orientations. Hence there are six different optimal solutions in this case. For example, the sequence $0 - 1 - 2$ would be encoded by the vector $\mathbf{x} = [1, 0, 0, 0, 1, 0, 0, 0, 1]$. Note that due to the value of the offset, $\mathbf{x}^T Q \mathbf{x} = -6$ in this case, since $-6 + 6 = 0$. A Python program that generates the QUBO matrix is given in Appendix A.

3.3 Adequateness of QUBO for the adiabatic framework

Since quantum annealing is based on the adiabatic quantum model, after forming a D-Wave feasible QUBO, it is worthwhile to examine how suitable that formulation is in producing capable physical Hamiltonians. That is, in order to produce efficient solutions, theoretically, a QUBO must produce a capable problem Hamiltonian H_P as in Equation (8).

Given a QUBO in the form $\mathbf{x}^T Q \mathbf{x}$, the binary decision variable x_i, which can take 0 or 1, can be directly transformed into the (Ising) spin decision variable s_i, through $s_i = 1 - 2x_i$. Thus, the spin variable s_i takes the value 1 or -1 as x_i is 0 or 1. According to Schrödinger picture of quantum mechanics, possible outcome of a measurement of the observable are the eigenvalues. Considering the spin, one could consider σ_z, Pauli z-matrix, which has eigenvalues ± 1. Accordingly, a physical Hamiltonain can be derived from the QUBO, in the form of interactions between the ith and jth qubits of the physical system [41]. This represents the relation between x_i and x_j of the original problem. Thus, it is possible to formulate the problem Hamiltonian H_P of dimensions $2^{n^2} \times 2^{n^2}$. Following the steps in [8], it is straightforward to see that the problem Hamiltonian can be expressed as follows.

$$H_P = \sum_i h_i \sigma_z^{(i)} + \sum_{i<j} J_{i,j} \sigma_z^{(i)} \sigma_z^{(j)}, \tag{15}$$

where,

$$h_i = \frac{1}{4} \sum_{k=1}^{n^2} \left(q_{i,k} + q_{k,i} \right), \tag{16}$$

$$J_{i,j} = \begin{cases} \frac{q_{i,j}+q_{j,i}}{4} & \text{if } i < j, \\ 0 & \text{otherwise} \end{cases} \tag{17}$$

and, $\sigma_z^{(i)} \sigma_z^{(i)}$ denotes that Pauli z-matrices $\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$ are acting on the ith and jth qubits of the physical system.

This problem Hamiltonian encodes the objective function so that the ground state of Q is an eigenstate of it, corresponding the minimum eigenvalue [41]. Recall the adiabatic process evolves from an initial problem Hamiltonian H_p along an adiabatic path. It is important to select an easy-to-prepare initial Hamiltonian. The standard choice for H_I is

$$H_I = \sum_i \sigma_x^{(i)}, \tag{18}$$

where σ_x is the Pauli x-matrix $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$.

3.3.1 The K_3 example revisited

Consider the QUBO matrix $Q(K_3)$ in Section 3.2.1. Following Equations (16) and (17), it can be seen that $h_i = 1$ for $i = 1, 2, \ldots, 9$. Also, $J_{1,2} = J_{1,3} = J_{1,4} = J_{1,7} = J_{2,3} = J_{2,5} =$

$J_{2,8} = J_{3,6} = J_{3,9} = J_{4,5} = J_{4,6} = J_{4,7} = J_{5,6} = J_{5,8} = J_{6,9} = J_{7,8} = J_{7,9} = J_{8,9} = 0.5$ and $J_{i,j} = 0$ for all other i, j's. The term in the Equation for problem Hamiltonian for K_3 which corresponds to Equation (15) indexed by $(2, 3)$ is $0.5I \otimes \sigma_z \otimes \sigma_z \otimes (I^{\otimes 6})$.

3.4 D-Wave experiments

We used a D-Wave 2X computer which consists of 1098 active *physical* qubits (out of a possible of 1152). We performed several trials of Hamiltonians on each graph with and without D-Wave post-processing optimization. That is, we considered the embeddings, physical QUBOs and D-Wave outputs, for which we used a post-processing script to check if the answers were compatible with our expectations. The D-Wave runs confirmed all our graph Hamiltonians were correct. For instance, we tested the Hamiltonians for all connected graphs on 4 vertices, which includes C_4 and K_4 with Hamiltonian cycles and the star graph S_3, Tadpole graph $T_{3,1}$ and the line graph P_4 as non-Hamiltonian graphs. Accordingly, the D-Wave runs did distinguish these graphs from the other graphs of the same order.

3.4.1 Remarks on complexity

There is no general theory to predict the runtime of a given quantum adiabatic algorithm. Rough estimates suggest the condition $T > \frac{1}{\Delta^2}$, where Δ is the minimum spectral gap, or, the gap between the two lowest eigenvalues of the physical Hamiltonian over the adiabatic path $s = 0$ to $s = 1$ [23]. Physically, the lowest eigenvalue corresponds the ground state energy and the second lowest eigenvalue the first excited state. In order to obtain the answer in adiabatic quantum computing, it is necessary for the process to remain in ground state throughout the whole time the computation takes place. This time is found to satisfy the condition $T > \frac{1}{\Delta^2}$, which is now considered as the condition for adiabatic computing [18].

Once a QUBO formulation for adiabatic framework is stated, it is worth to check this adiabatic condition. That is, the adequateness of the QUBO formulation to produce competent physical Hamiltonians for adiabatic quantum computation depends upon the minimum spectral gap over the interval $[0, 1]$. In other words, if the minimum spectral gap Δ becomes exponentially small with the size of the problem, the corresponding formulation is not suitable for adiabatic quantum computation. This has been the case for some adiabatic algorithms [45], which is discussed in detail in [45, 24].

Recall that the QUBO formulation must have minimum slack variables and constants. Even with a careful selection of parameters, there is a possibility for the adiabatic computation to take exponentially large runtime due to the spectral gap. Therefore, it is important to consider the spectrum of the final Hamiltonian. Accordingly, we evaluated the eigenvalues of the final Hamiltonian $H(s)$ over the standard adiabatic path for several instances of HCP.

Our results indicate that HCP Hamiltonian derived by our QUBO does not decrease exponentially with problem size. However, small spectral gaps were observed for some graphs, though the gap does not decrease exponentially with number of vertices. For example, the spectral gap for our QUBO Hamiltonian for the complete graph K_n decreases from $n = 1$ to $n = 6$ but it increases significantly for $n = 7$. However, the gap for $n = 6$

was considerably smaller, though the decrease does not show an exponential decay. See Table 1. We have tested several other graphs on small numbers of vertices, and observed no exponential decrease over the order (number of vertices) of the graph.

Table 1: Variation of the spectral gap for complete graphs K_3 to K_7.

Graph	Spectral gaps of Hamiltonian for										
	s=0	s=0.1	s=0.2	s=0.3	s=0.4	s=0.5	s=0.6	s=0.7	s=0.8	s=0.9	s=1.0
K_3	1.00	0.95	0.93	0.89	0.78	0.61	0.83	1.12	1.41	1.70	2.00
K_4	2.00	0.20	0.41	0.62	0.83	1.02	1.20	1.30	1.50	1.77	2.00
K_5	2.00	1.07	0.22	1.6	2.2	0.95	1.86	1.36	1.99	1.76	0.71
K_6	0.12	0.08	0.08	0.06	0.09	0.08	0.10	0.08	0.06	0.07	2.00
K_7	0	0.12	0.31	1.02	0.73	0.94	1.15	1.35	1.56	1.77	2.0

In certain adiabatic instances, a decrease of the minimum spectral gap occurred at a fixed point in the interval $[0, 1]$. For instance, in ordinary adiabatic Grover's algorithm, the minimum spectral gap occurs at $s = 0.5$ for all instances, irrespective of the size of the problem, which showed an exponential decrease with the size. Contrastingly, the point at which the minimum spectral gap occurs in our QUBO Hamiltonian changes with problem size. For instance, this occurs at 0.5 for K_3 and it shifts to $s = 0.1$ in K_4.

Further, our results indicate another distinct characteristic of the spectral gap of our QUBO Hamiltonian in contrast to adiabatic Grover and several other adiabatic quantum algorithms. That is, the spectral gap in our problem may not necessarily have a unique local minimum in the domain [0,1]. For instance, we observed that the spectral gap for K_6 reaches minimum at both $s = 0.3$ and $s = 0.8$. However, for many instances we tested numerically, the minimum occurred at a single point.

4 Discussion

We considered different solutions proposed for HCP in different quantum frameworks and presented a QUBO formulation, with proof of correctness, from which feasible Hamiltonians could be derived for the D-Wave architecture. Though designed upon different quantum computational frameworks, all existing quantum methods for solving HCP depend on brute-force approach, thus on enumerating all possible cases and checking for a Hamiltonian cycle. Contrastly, our QUBO formulation is derived using the notion of Hamiltonicity. Also, it was particularly designed to be feasible on the D-Wave architecture.

The best efficiency gained through existing quantum algorithms is the complexity of quantum brute-force method which has complexity $O\left(p(n)2^{\frac{n\log n}{2}}\right)$, where $p(n)$ is a polynomial of n. This is a polynomial speedup over classical brute-force method. We note that the Held-Karp algorithm [32] can be applied to find a Hamiltonian cycle in complexity $O\left(2^n n^2\right)$. Thus, brute-force quantum method and its implementations in different frameworks do not result in an efficiency gain for solving the HCP. The QUBO formulation for a HCP produces a Hamiltonian for the D-Wave architecture, which is designed over the adiabatic quantum

framework. Since there is no general theory to predetermine the complexity of adiabatic quantum algorithms, it is not straightforward to conclude an efficiency gain. One major instance where adiabatic algorithms fail is with the spectral gap, which might decrease exponentially with the size of the problem. We evaluated several Hamiltonians numerically, and our results indicate that the QUBO Hamiltonian for HCP does not possess an exponentially decreasing spectral gap, although very small spectral gaps were observed for several instances of HCP.

It must be noted that the spectral results derived numerically give a general overview on the adequateness of our QUBO formulation in adiabatic quantum computations. If one needs to consider the exact physical process that take place in a D-Wave quantum computer, certain assumptions must be lifted and further analysis is required. For example, our numerical computations are based on the assumption that the adiabatic process takes place over a linear path in the quantum computer. According to some recent works, D-Wave uses a different annealing path [41]. Further, D-Wave uses a Chimera graph as its hardware topology. Therefore, given a QUBO instance in n variables, it has to be minor-embedded onto the Chimera graph, effecting the complexity of the problem, as it requires additional qubits. An efficient technique for minor-embedding a complete graph onto Chimera was proposed in [19]. It would be an interesting task to take these physical and engineering aspects into account and to make a complete analysis to check the adequateness of the QUBO to be used on a realistic D-Wave platform.

5 Acknowledgements

The authors thank Cris Calude for helpful comments on an earlier draft of this paper. AM would like to thank André Nies for helpful discussions and also for the funds received through the Department of Computer Science of the University of Auckland and Marsden grant. This work was supported in part by the Quantum Computing Research Initiatives at Lockheed Martin.

References

[1] Scott Aaronson and Andris Ambainis. Quantum search of spatial regions. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, pages 200–209. IEEE, 2003.

[2] Dorit Aharonov, Wim Van Dam, Julia Kempe, Zeph Landau, Seth Lloyd, and Oded Regev. Adiabatic quantum computation is equivalent to standard quantum computation. *SIAM Review*, 50(4):755–787, 2008.

[3] Boris Altshuler, Hari Krovi, and Jérémie Roland. Anderson localization makes adiabatic quantum optimization fail. *Proceedings of the National Academy of Sciences*, 107(28):12446–12450, 2010.

[4] Andris Ambainis. Quantum walk algorithm for element distinctness. *SIAM Journal on Computing*, 37(1):210–239, 2007.

[5] Katie Barr and Viv Kendon. The expressive power of quantum walks in terms of language acceptance. In Ross Duncan and Prakash Panangaden, editors, *Proceedings 9th Workshop on Quantum Physics and Logic*, pages 39–51. ACM Press, October 2012.

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

[7] Paul Benioff. Space searches with a quantum robot. In *Quantum Computation and Information, Contemporary Mathematics Series*, volume 305, pages 1–12. AMS, 2002.

[8] Zhengbing Bian, Fabian Chudak, William G. Macready, and Geordie Rose. The Ising model: teaching an old problem new tricks. *D-Wave Systems*, 2, 2010.

[9] Andreas Bjorklund. Determinant sums for undirected hamiltonicity. In *Foundations of Computer Science (FOCS), 2010 51st Annual IEEE Symposium on*, pages 173–182. IEEE, 2010.

[10] Max Born and Vladimir Fock. Beweis des adiabatensatzes. *Zeitschrift für Physik A Hadrons and Nuclei*, 51(3):165–180, 1928.

[11] Michel Boyer, Gilles Brassard, Peter Høyer, and Alain Tapp. Tight bounds on quantum searching. *Fortschritte der Physik*, 46:493–505, 1998.

[12] Harry Buhrman and Robert Špalek. Quantum verification of matrix products. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete Algorithms*, pages 880–889. Society for Industrial and Applied Mathematics, 2006.

[13] Cristian S. Calude, Elena Calude, and Michael J. Dinneen. Guest column: Adiabatic quantum computing challenges. *ACM SIGACT News*, 46(1):40–61, 2015.

[14] Cristian S. Calude, John Casti, and Michael J. Dinneen. *Unconventional Models of Computation*. Springer-Verlag, 1998.

[15] Cristian S. Calude and Michael J. Dinneen. Solving the broadcast time problem using a D-Wave quantum computer. In *Advances in Unconventional Computing*, pages 439–453. Springer, 2017.

[16] Cristian S. Calude, Michael J. Dinneen, and Richard Hua. QUBO formulations for the graph isomorphism problem and related problems. *Theoretical Computer Science*, 701:54–69, 2017.

[17] Yudong Cao, Anargyros Papageorgiou, Iasonas Petras, Joseph Traub, and Sabre Kais. Quantum algorithm and circuit design solving the poisson equation. *New Journal of Physics*, 15(1):013021, 2013.

[18] Andrew Childs. Overview of adiabatic quantum computation.

[19] Vicky Choi. Minor-embedding in adiabatic quantum computation: I. the parameter setting problem. *Quantum Information Processing*, 7(5):193–209, 2008.

[20] Saurya Das, Randy Kobes, and Gabor Kunstatter. Adiabatic quantum computation and Deutsch's algorithm. *Physical Review A*, 65(6):062310, 2002.

[21] Michael J. Dinneen and Richard Hua. Formulating graph covering problems for adiabatic quantum computers. In *Proceedings of the Australasian Computer Science Week Multiconference*, ACSW '17, pages 18:1–18:10, New York, NY, USA, 2017. ACM.

[22] Sebastian Dörn. Quantum algorithms for graph traversals and related problems. In *Proceedings of CIE*, volume 7, pages 123–131, 2007.

[23] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. Quantum adiabatic evolution algorithms versus simulated annealing. *arXiv preprint quant-ph/0201031*, 2002.

[24] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, and Daniel Nagaj. How to make the quantum adiabatic algorithm fail. *International Journal of Quantum Information*, 6(03):503–516, 2008.

[25] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, and Michael Sipser. Quantum computation by adiabatic evolution. *arXiv preprint quant-ph/0001106*, 2000.

[26] Edward Farhi and Sam Gutmann. Analog analogue of a digital quantum computation. *Physical Review A*, 57(4):2403, 1998.

[27] Lov K. Grover. Quantum mechanics helps in searching for a needle in a haystack. *Physical Review Letters*, 79(2):325, 1997.

[28] Guo Hao, Long Gui-Lu, Sun Yang, and Xiu Xiao-Lin. A quantum algorithm for finding a Hamilton circuit. *Communications in Theoretical Physics*, 35(4):385, 2001.

[29] Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for linear systems of equations. *Physical Review Letters*, 103(15):150502, 2009.

[30] Michael Haythorpe. *Markov Chain Based Algorithms for the Hamiltonian Cycle Problem*. PhD thesis, University of South Australia, 2010.

[31] Birgit Hein and Gregor Tanner. Quantum search algorithms on the hypercube. *Journal of Physics A: Mathematical and Theoretical*, 42(8):085303, 2009.

[32] 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.

[33] Itay Hen. Period finding with adiabatic quantum computation. *EPL (Europhysics Letters)*, 105(5):50005, 2014.

[34] Sandor Imre and Ferenc Balazs. *Quantum Computing and Communications: an Engineering Approach*. John Wiley & Sons, 2005.

[35] Sabine Jansen, Mary-Beth Ruskai, and Ruedi Seiler. Bounds for the adiabatic approximation with applications to quantum computation. *Journal of Mathematical Physics*, 48(10):102111, 2007.

[36] Richard M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–103. Springer, 1972.

[37] James King, Sheir Yarkoni, Jack Raymond, Isil Ozfidan, Andrew D. King, Mayssam Mohammadi Nevisi, Jeremy P. Hilton, and Catherine C. McGeoch. Quantum annealing amid local ruggedness and global frustration. *Algorithms*, 5:7, 2017.

[38] Chang-Muk Lee, Sung Whan Kim, Sam Myo Kim, and Uik Sohn. DNA computing the hamiltonian path problem. *Molecules and Cells*, 9(5):464–469, 1999.

[39] Andrew Lucas. Ising formulations of many NP problems. *Frontiers in Physics*, 2:5, 2014.

[40] Anuradha Mahasinghe and Jingbo Wang. Efficient quantum circuits for Toeplitz and Hankel matrices. *Journal of Physics A: Mathematical and Theoretical*, 49(27):275301, 2016.

[41] Catherine C. McGeoch. Adiabatic quantum computation and quantum annealing: Theory and practice. *Synthesis Lectures on Quantum Computing*, 5(2):1–93, 2014.

[42] Michael A. Nielsen and Isaac Chuang. Quantum computation and quantum information, 2000.

[43] Mihai Oltean. Solving the Hamiltonian path problem with a light-based computer. *Natural Computing*, 7(1):57–70, 2008.

[44] Xinhua Peng, Zeyang Liao, Nanyang Xu, Gan Qin, Xianyi Zhou, Dieter Suter, and Jiangfeng Du. Quantum adiabatic algorithm for factorization and its experimental implementation. *Physical Review Letters*, 101(22):220405, 2008.

[45] Jérémie Roland and Nicolas J. Cerf. Quantum search by local adiabatic evolution. *Physical Review A*, 65(4):042308, 2002.

[46] Juexiao Su, Tianheng Tu, and Lei He. A quantum annealing approach for boolean satisfiability problem. In *Proceedings of the 53rd Annual Design Automation Conference*, page 148. ACM, 2016.

[47] Hayato Ushijima-Mwesigwa, Christian F.A. Negre, and Susan M. Mniszewski. Graph partitioning using quantum annealing on the d-wave system. *arXiv preprint arXiv:1705.03082*, 2017.

[48] Davide Venturelli, Dominic J.J. Marchand, and Galo Rojo. Job-shop scheduling solver based on quantum annealing. In *Proceedings of the 11th workshop on Constraint Satisfaction Techniques in Planning and Scheduling*, pages 25–34, 2016.

[49] Jingbo Wang and Kia Manouchehri. *Physical Implementation of Quantum Walks*. Springer, 2013.

A Python Program to Generate QUBO Formulation of the Hamilton Cycle Problem

```
import networkx as nx
import sys, math, time

# we assume the input graph is in standard adjacency list format
def read_input():
  n=int(sys.stdin.readline().strip())
    G1=nx.empty_graph(n,create_using=nx.Graph())
    for u in range(n):
        neighbors=sys.stdin.readline().split()
    for v in neighbors:
          G1.add_edge(u,int(v))
        return G1

def generate_qubo(G):
  n = G.order()
  varsDict = {}

  index = 0
  for i in range(n):
    for j in range(n):
      varsDict[(i,j)] = index
      index += 1

  # initialize Q
  Q = {}
  for i in range(n*n):
    for j in range(n*n):
      Q[i,j] = 0

  # p_1
  for i in range(n):
    for iprime in range(n):
      index = varsDict[(i,iprime)]
      Q[index,index] -= 2

    for iprime1 in range(n):
      for iprime2 in range(n):
        index1 = varsDict[(i,iprime1)]
        index2 = varsDict[(i,iprime2)]
        Q[index1, index2] += 1

  # p_2
  for iprime in range(n):
    for i in range(n):
      index = varsDict[(i,iprime)]
      Q[index,index] -= 2

    for i1 in range(n):
      for i2 in range(n):
```

```
            index1 = varsDict[(i1,iprime)]
            index2 = varsDict[(i2,iprime)]
            Q[index1, index2] += 1
            # f
            for i_1 in range(n):
                for i_2 in range(n):
                    if not(i_2 in G.neighbors(i_1)) and not(i_1 == i_2):
                        for j in range(n-1):
                            index_1 = varsDict[i_1,j]
                            index_2 = varsDict[i_2,j+1]
                            Q[index_1, index_2] += 1

                        index_1 = varsDict[i_1, n-1]
                        index_2 = varsDict[i_2, 0]

                        Q[index_1, index_2] += 1

        # Making Q uppertriangular
    for i in range(n*n):
        for j in range(n*n):
            if (i > j) and (not(Q[i,j]==0)):
                Q[j,i] += Q[i,j]
                Q[i,j] = 0

    return Q,varsDict,n*n

G = read_input()
qubo = generate_qubo(G)
Q = qubo[0]
n = qubo[2]

print n
for i in range(n):
    for j in range(n):
        print Q[i,j],
    print
print 'offset =', 2*G.order()
```

listings/hamiltonCycleQUBO.py