CDMTCS
Research
Report
Series

A New QUBO Objective Function For Solving the Maximum Common Subgraph Isomorphism Problem Via Quantum Annealing

Nan Huang, Dominik Roje
School of Computer Science
The University of Auckland
Auckland, New Zealand

CDMTCS-544
June 2020

Centre for Discrete Mathematics and
Theoretical Computer Science

A New QUBO Objective Function For Solving the Maximum Common Subgraph Isomorphism Problem Via Quantum Annealing

Nan Huang[*], Dominik Roje[†]

October 23, 2020

Abstract

We present QUBO objective functions for the maximum common sub-graph isomorphism problem, proved their correctness and illustrate them with a few toy problems. Very small scale problems can be solved by the D-Wave 2X machine with high enough accurate rates; however, for larger scale problems, the accurate rate drops significantly.

1 Introduction

Quantum computing is a field of increasing interest to computer scientists that involves taking advantage of quantum effects to perform computations that differ significantly from those of classical computers. Certain types of problems, such as simulations of quantum systems, which are considered to be too hard for classical computers, may be solved by quantum computers within a reasonable time. Even though it is still debatable if exponential speed-up on certain types of problems can be achieved by quantum computers, the possibility of a huge advantage still drives people to develop new types of quantum machines. In this area, developing quantum algorithms is essential.

D-Wave computers are based on a process known as quantum annealing. These quantum computers solve discrete optimisations problems like Ising problem or the Quadratic Unconstrained Binary Optimisation (QUBO) problem. D-Wave computers take advantage of quantum tunnelling to find the global minimum of objective functions. Various problems can be reformulated as QUBOs, so they can solved by these machines [3, 18, 19]. This method has the potential to solve hard graph problems, such as the graph isomorphism problem [26], graph partitioning [21] and many others [8, 13, 20].

[*]School of Computer Science, The University of Auckland, Auckland, New Zealand.
ORCID: 0000-0003-3760-496X
Email: nhua630@aucklanduni.ac.nz

[†]School of Computer Science, The University of Auckland, Auckland, New Zealand

The maximum common subgraph isomorphism problem is one of the NP-hard graph related problems that has a wide impact in bio-informatics [4], alignment of chemical structures [14], computer vision and pattern matching [9, 12], etc. By solving this problem, we get the largest common subgraph of the two input graphs. There are two variations of this problem that have been studied a lot [4, 14]: the maximum common induced subgraph isomorphism (MCISI) problem and the maximum common edge subgraph isomorphism (MCESI) problem.

In this paper we propose QUBO formulations for the problems MCISI and MCESI, so we can solve them on a D-Wave computer. We first give their definitions and their QUBO formulations. Then we prove the correctness of these formulations. Finally, we run instances of the problems for some small organic molecules on a real D-Wave 2X machine and analyse the experimental data.

2 QUBO formulations for the maximum common subgraph isomorphism problem

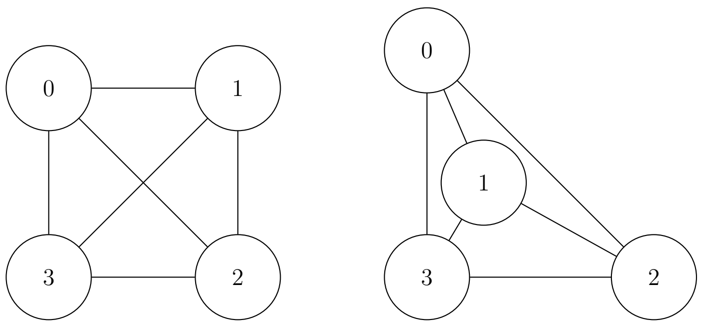
In [6] QUBO formulations for the subgraph isomorphism problem and the induced subgraph isomorphism problem are provided. An improved QUBO formulation that reduces the number of variables was proposed in [25]. Solving the graph isomorphism problem decides whether the two graphs are isomorphic. Solving the subgraph (induced subgraph) isomorphism problem decides whether one graph is isomorphic to a subgraph (induced subgraph) of the other graph. A simple example of these three problems are shown in Figure 1. In Figure 1(a), the two graphs are isomorphic to each other. In Figure 1(b), there is an isomorphism between the left graph and an induced subgraph of the right graph. In Figure 1(c), the left graph is isomorphic to a subgraph of the right graph. However, the solutions of all three problems cannot show the common sub-structure shared by two graphs of arbitrary sizes. In order to obtain this piece of information, we can solve the maximum common subgraph isomorphism problem to determine the largest shared sub-structure. We will build our QUBO formulation for the maximum common subgraph isomorphism problem by extending the idea in [6]. Before we show our QUBO formulation, we first describe the notation that we will use, then we give a detailed definition of the problems.

2.1 Definition and notation

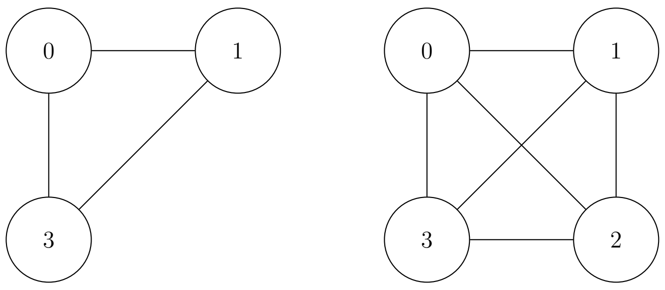
We denote the integers modulo n by $\mathbb{Z}_n = \{0, \dots, n-1\}$ and a repeated cross product by

$$\underbrace{\mathbb{Z}_n \times \cdots \times \mathbb{Z}_n}_{m \text{ times}} = \mathbb{Z}_n^m.$$

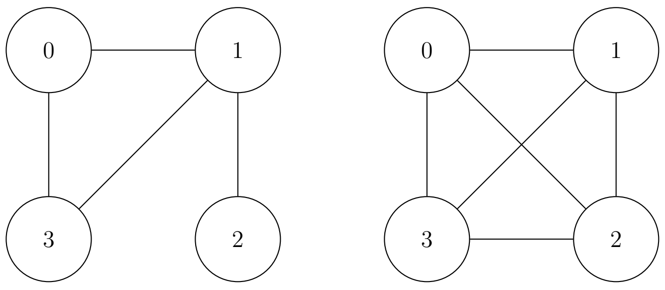
A partial function $f : X \dashrightarrow Y$ is a function which is undefined for some values of $x \in X$. The domain of f, denoted $\mathrm{Dom}(f)$, is all $x \in X$ where $f(x)$ is defined and the image of f, denoted $\mathrm{Im}(f)$, is the set of $f(x)$ for all $x \in \mathrm{Dom}(f)$.



(a) Graph isomorphism



(b) Induced subgraph isomorphism



(c) Subgraph isomorphism

Figure 1: Examples of graph isomorphism, induced subgraph isomorphism and subgraph isomorphism

Definition 1. A *labeled graph* (adding weights to a simple graph) can be represented by a four-tuple $G = (V, E, \alpha, \beta)$, where

- V is the set of vertices,
- $E \subseteq V \times V$ is the set of edges,
- $\alpha : V \to L_V$ is a function assigning labels to vertices,
- $\beta : E \to L_E$ is a function assigning labels to edges.

If $u, v \in V$ then we denote an edge connecting u and v by uv. Alternatively, we say $(u, v) \in E$.

Definition 2. Given a graph $G = (V, E, \alpha, \beta)$, a *subgraph* of G is a graph $S = (V_S, E_S, \alpha_S, \beta_S)$, where

- $V_S \subseteq V$,
- $E_S \subseteq E \cap (V_S \times V_S)$,
- $\alpha_S = \alpha \mid_{V_S}$,
- $\beta_S = \beta \mid_{E_S}$.

Definition 3. Given a graph $G = (V, E, \alpha, \beta)$, an *induced subgraph* of G is a graph $I = (V_I, E_I, \alpha_I, \beta_I)$, where

- $V_I \subseteq V$,
- $E_I = E \cap (V_I \times V_I)$,
- $\alpha_I = \alpha \mid_{V_I}$,
- $\beta_I = \beta \mid_{E_I}$.

Definition 4. A bijective function $\psi : V_1 \to V_2$ is a *graph isomorphism* between graphs $G_1 = (V_1, E_1, \alpha_1, \beta_1)$ and $G_2 = (V_2, E_2, \alpha_2, \beta_2)$ if

- $\alpha_1(v_1) = \alpha_2(\psi(v_1))$ for all $v_1 \in V_1$ and $\alpha_2(v_2) = \alpha_1(\psi^{-1}(v_2))$ for all $v_2 \in V_2$,
- for any edge $e_1 = (v_1, v_1') \in E_1$, there exists an edge $e_2 = (\psi(v_1), \psi(v_1')) \in E_2$, such that $\beta_1(e_1) = \beta_2(e_2)$, and for any edge $e_2 = (v_2, v_2') \in E_2$, there exists an edge $e_1 = (\psi^{-1}(v_2), \psi^{-1}(v_2')) \in E_1$, such that $\beta_1(e_1) = \beta_2(e_2)$

Definition 5. An injective function $\phi : V_1 \to V_2$ is a *subgraph isomorphism* between graphs $G_1 = (V_1, E_1, \alpha_1, \beta_1)$ and $G_2 = (V_2, E_2, \alpha_2, \beta_2)$ if there exists a subgraph S of the graph G_2, such that ϕ is a graph isomorphism between G_1 and S.

Definition 6. An injective function $\Phi : V_1 \to V_2$ is an *induced subgraph isomorphism* between graphs $G_1 = (V_1, E_1, \alpha_1, \beta_1)$ and $G_2 = (V_2, E_2, \alpha_2, \beta_2)$ if there exists an induced subgraph I of the graph G_2, such that Φ is a graph isomorphism between G_1 and I.

Definition 7. Given graphs $G_1 = (V_1, E_1, \alpha_1, \beta_1)$ and $G_2 = (V_2, E_2, \alpha_2, \beta_2)$, S is a *common subgraph* of both G_1 and G_2 if there exists a subgraph isomorphism between S and G_1, and a subgraph isomorphism between S and G_2.

Definition 8. Given graphs $G_1 = (V_1, E_1, \alpha_1, \beta_1)$ and $G_2 = (V_2, E_2, \alpha_2, \beta_2)$, I is a *common induced subgraph* of both G_1 and G_2 if there exists an induced subgraph isomorphism Φ between I and G_1, and an induced subgraph isomorphism between I and G_2.

Next, we give the definition for the maximum common subgraph isomorphism. Since it is both meaningful to find the maximum common induced subgraph and the maximum common subgraph, we give the definitions for both choices. For the maximum common induced subgraph isomorphism, it is enough to maximise the number of vertices. For the maximum common subgraph isomorphism, it is more meaningful to maximise the number of edges, otherwise we are likely to end up with a solution that contains only isolated vertices.

Definition 9. The *maximum common induced subgraph* (MCIS) is the common induced subgraph between $G_1 = (V_1, E_1, \alpha_1, \beta_1)$ and $G_2 = (V_2, E_2, \alpha_2, \beta_2)$ with the largest number of vertices compared to the other common induced subgraphs of G_1 and G_2.

Definition 10. The *maximum common edge subgraph* (MCES) is the common subgraph between $G_1 = (V_1, E_1, \alpha_1, \beta_1)$ and $G_2 = (V_2, E_2, \alpha_2, \beta_2)$ with the largest number of edges compared to the other common subgraphs of G_1 and G_2.

Observation 1. *If the graphs $G_1 = (V_1, E_1, \alpha_1, \beta_1)$ and $G_2 = (V_2, E_2, \alpha_2, \beta_2)$ have a maximum common induced subgraph $I = (V_I, E_I, \alpha_I, \beta_I)$, Φ_1 is an induced subgraph isomorphism from I to G_1 and Φ_2 is an induced subgraph isomorphism from I to G_2, then the maximum common induced subgraph isomorphism (MCISI) is a bijective partial function $\gamma : V_1 \dashrightarrow V_2$. For every $v_1 \in V_1$, we have*

$$\gamma(v_1) = \begin{cases} \Phi_2(\Phi_1^{-1}(v_1)), & \text{if } v_1 \in \operatorname{Im}(\Phi_1), \\ \text{undefined}, & \text{otherwise.} \end{cases} \tag{1}$$

Observation 2. *If the graphs $G_1 = (V_1, E_1, \alpha_1, \beta_1)$ and $G_2 = (V_2, E_2, \alpha_2, \beta_2)$ have a maximum common edge subgraph $S = (V_S, E_S, \alpha_S, \beta_S)$, ϕ_1 is a subgraph isomorphism from S to G_1 and ϕ_2 is a subgraph isomorphism from S to G_2, then the maximum common edge subgraph isomorphism (MCESI) is a bijective partial function $\gamma : V_1 \to V_2$. For every $v_1 \in V_1$, we have*

$$\gamma(v_1) = \begin{cases} \phi_2(\phi_1^{-1}(v_1)), & \text{if } v_1 \in \operatorname{Im}(\phi_1), \\ \text{undefined}, & \text{otherwise.} \end{cases} \tag{2}$$

Next we give some examples and explain when MCISI or MCESI should be used. In Figure 2 we have two input graphs, so solving the MCISI may give us a mapping from 0 to a, 1 to b and 3 to d, while solving the MCESI may give us a mapping from 0 to a, 1 to b, 2 to c and 3 to d. Note that the above solutions are not unique. The solutions are different because MCISI requests the common subgraph to be an induced subgraph of both input graphs and MCESI does not. MCISI has a tighter restriction due to the requirement of being induced subgraph. MCISI can be used when we are looking for exactly same sub-structure while we will use MCESI for detecting a similar sub-structure.

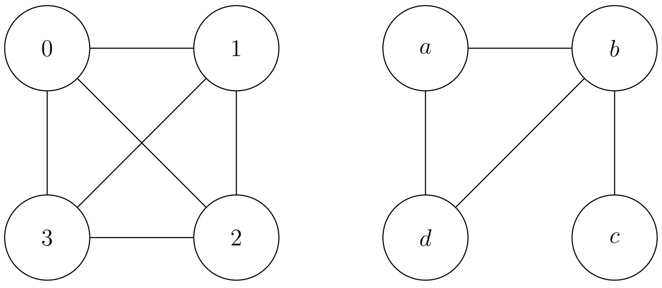


Figure 2: An example of MCISI and MCESI.

2.2 Classical algorithms to solve the maximum common subgraph isomorphism problem

Since the maximum common subgraph isomorphism problem can be reduced to the maximal clique problem, a lot of work has been done via this reduction. A clique of a graph is a subgraph of that graph, such that all pairs of vertices in the subgraph have an edge to connect them. A maximal clique is a clique of the graph that has the largest number of vertices. In order to reduce the maximum common subgraph isomorphism to the maximal clique problem, a compatibility graph needs to be constructed first. A compatibility graph is a graph generated from two input graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$. The vertex set is the set $V_1 \times V_2$. Two vertices (v_1, u_1), (v_2, u_2), $v_1, u_1 \in V_1, v_2, u_2 \in V_2$ are adjacent when $v_1u_1 \in E_1$ and $v_2u_2 \in E_2$, or $v_1u_1 \notin E_1$ and $v_2u_2 \notin E_2$. Then by solving the maximal clique problem of the compatibility graph gives us the solution for the maximum common subgraph isomorphism problem [2, 17].

Another class of algorithms uses backtracking algorithms to solve the problem. Since the isomorphism can be represented by vertex mappings, if we build up the mapping by adding one vertex at a time, then a tree structure can be generated that backtracking algorithms help reduce the time of traversal. Algorithms based on backtracking are discussed in [7, 23].

The works above are exact solvers for the maximum common subgraph problem. There are also heuristic solvers that are faster but only give approximate results. Genetic algorithms are one important class in this paradigm. A genetic algorithm uses tools such as crossover, mutation, clone, etc. to generate a new population from the old ones. A fitness function is defined to make sure the new population has higher fitness. The speed and performance of this approach depend on the design of the fitness function. Different fitness functions are developed in [24, 5, 22]. Funabiki and Kitamichi [15] have used another combinatorial optimisation method that they call two-Stage Discrete Optimisation Method. They have used a greedy algorithm to find candidates and then a randomised discrete method to refine the result. Simulated annealer has been used by Barakat and Dean in [1].

2.3 QUBO formulation for MCISI

Starting with the definition of MCISI, we will design penalty terms in the QUBO objective function according to the features of MCISI. Let us work with the two graphs $G_1 = (V_1, E_1, \alpha_1, \beta_1)$ and $G_2 = (V_2, E_2, \alpha_2, \beta_2)$. We denote the MCIS of G_1 and G_2 as $I = (V_I, E_I, \alpha_I, \beta_I)$ and the MCISI from G_1 to G_2 as γ. That means there exists an induced subgraph $I_1 = (V_{I_1}, E_{I_1}, \alpha_{I_1}, \beta_{I_1})$ of G_1 and an induced subgraph $I_2 = (V_{I_2}, E_{I_2}, \alpha_{I_2}, \beta_{I_2})$ of G_2, such that:

- $\alpha_{I_1}(v_1) = \alpha_{I_2}(\gamma(v_1))$ for all $v_1 \in V_{I_1}$, and $\alpha_{I_2}(v_2) = \alpha_{I_1}(\gamma^{-1}(v_2))$ for all $v_2 \in V_{I_2}$ (injective);
- for any edge $e_1 = (v_1, v_1') \in E_{I_1}$, there exists an edge $e_2 = (\gamma(v_1), \gamma(v_1')) \in E_{I_2}$, such that $\beta_{I_1}(e_1) = \beta_{I_2}(e_2)$, and for any edge $e_2 = (v_2, v_2') \in E_{I_2}$, there exists an edge $e_1 = (\gamma^{-1}(v_2), \gamma^{-1}(v_2')) \in E_{I_1}$, such that $\beta_{I_1}(e_1) = \beta_{I_2}(e_2)$ (edge-preserving);
- if any pair of vertices $(v_1, v_1') \notin E_{I_1}$, then $(\gamma(v_1), \gamma(v_1')) \notin E_{I_2}$, and if any pair of vertices $(v_2, v_2') \notin E_{I_2}$, then $(\gamma^{-1}(v_2), \gamma^{-1}(v_2')) \notin E_{I_1}$ (non-edge-preserving).

We then build the QUBO objective function of MCISI by translating those three features into penalty terms, which are terms with positive coefficient. We design penalty terms so that any assignments that do not fit into these features will add a positive value into the final result; they will not be picked as solution of the problem since a D-Wave machines will choose the assignments that give the minimum value of the objective function. Let $V_1 = \{u_0, u_1, \ldots, u_{n_1-1}\}$ and $V_2 = \{v_1, v_2, v_{n_2-1}\}$. For every $u \in V_1$, $v \in V_2$, a variable $x_{u,v}$ is used to represent the possible vertex mapping between u and v. Let $\mathbf{x} \in \mathbb{Z}_2^{n_1 n_2}$ with

$$\mathbf{x} = (x_{u_0,v_0}, x_{u_0,v_1}, \ldots, x_{u_0,v_{n_2-1}}, x_{u_1,v_0}, \ldots, x_{u_{n_1-1},v_{n_2-2}}, x_{u_{n_1-1},v_{n_2-1}}).$$

Then we define a function $\tau : V_1 \times V_1 \times V_2 \times V_2 \to \mathbb{Z}$ to identify the possible mapping between an edge in G_1 to an edge in G_2 with respect to the vertex labels and edge labels. Let $e_1 = (u, u')$, $u, u' \in V_1$ and $e_2 = (v, v')$, $v, v' \in V_2$,

$$\tau(u, u', v, v') = \begin{cases} 0, & \alpha_1(u) = \alpha_2(v), \alpha_1(u') = \alpha_2(v') \text{ and } \beta_1(e_1) = \beta_2(e_2), \\ 1, & \text{otherwise.} \end{cases} \tag{3}$$

We define the objective function by

$$F : \mathbb{Z}_2^{n_1 n_2} \to \mathbb{R},$$

$$F(\mathbf{x}) = A[H(\mathbf{x}) + P(\mathbf{x}) + N(\mathbf{x})] - BM(\mathbf{x}), \quad A, B \in \mathbb{R}^+, \tag{4}$$

where

$$H(\mathbf{x}) = \sum_{u\in V_1}\sum_{v\in V_2}\left(x_{u,v}\sum_{\substack{v'\in V_2\\ v'\neq v}} x_{u,v'}\right) + \sum_{v\in V_2}\sum_{u\in V_1}\left(x_{u,v}\sum_{\substack{u'\in V_1\\ u'\neq u}} x_{u',v}\right), \tag{5}$$

$$P(\mathbf{x}) = \sum_{uu'\in E_1}\sum_{v\in V_2}\left(x_{u,v}\sum_{v'\in V_2} x_{u',v'}\tau(u,u',v,v')\right), \tag{6}$$

$$N(\mathbf{x}) = \sum_{uu'\notin E_1}\sum_{v\in V_2}\left(x_{u,v}\sum_{v'\in V_2} x_{u',v'}e_{v,v'}\right), \tag{7}$$

$$M(\mathbf{x}) = \sum_{u\in V_1}\sum_{v\in V_2} x_{u,v}. \tag{8}$$

We request B to be sufficiently smaller than A, which is needed in later proofs. Here $e_{v,v'}$ can be calculated by $e_{v,v'} = 1$ if $vv' \in E_2$ and $e_{v,v'} = 0$ otherwise.

The function $H(\mathbf{x})$ is designed to ensure that the mapping is injective. $P(\mathbf{x})$ ensures that the mapping is edge-preserving. Penalties are only applied to impossible-edge mappings with respect to both the vertex label and the edge label. With the help of τ, some of the entries of the QUBO matrix may be updated to zero. $N(\mathbf{x})$ ensures that the mapping is non-edge-preserving. $M(\mathbf{x})$ ensures that the number of vertices is maximised. All these claims will be showed in later proofs.

The following part is to show that this objective function indeed solves the maximum common induced subgraph isomorphism problem. We define $\mathcal{F}$ to be the set of all common subgraph isomorphisms between G_1 and G_2. We now specify a decoder function

$$D : \mathbb{Z}_2^{n_1 n_2} \dashrightarrow \mathcal{F},$$

the purpose of which is to interpret the isomorphism encoded in $\mathbf{x}$ so that we can obtain a vertex mapping. The domain of D contains all vectors $\mathbf{x} \in \mathbb{Z}_2^{n_1 n_2}$ that can be 'decoded' into such functions. That is

$$\mathrm{Dom}(D) = \left\{ \mathbf{x} \in \mathbb{Z}_2^{n_1 n_2} \;\middle|\; \sum_{v\in V_2} x_{u,v} \le 1, \text{ for all } u \in V_1 \right. \\ \left. \text{and } \sum_{u\in V_1} x_{u,v} \le 1, \text{ for all } v \in V_2 \right\}$$

and

$$D(\mathbf{x}) = \begin{cases} \psi, & \text{if } \mathbf{x} \in \mathrm{Dom}(D), \\ \text{undefined}, & \text{otherwise.} \end{cases}$$

Here $\psi : V_1 \dashrightarrow V_2$ is a partial function that for every $u \in V_1$, $\psi(u) = v$, $u \in V_1$ and $v \in V_2$, if and only if $x_{u,v} = 1$.

Lemma 1. *For all $\mathbf{x} \in \mathbb{Z}_2^{n_1 n_2}$, $H(\mathbf{x}) = 0$ if and only if $D(\mathbf{x})$ is injective.*

Proof. We first show that if $H(\mathbf{x}) = 0$, then $D(\mathbf{x})$ is injective.

Since $H(\mathbf{x})$ only contains sum of products of two binary variables, which are non-negative, $H(\mathbf{x}) = 0$ if and only if all products have value zero.

The first part of $H(\mathbf{x})$, $\sum_{u \in V_1} \sum_{v \in V_2} \left(x_{u,v} \sum_{\substack{v' \in V_2 \\ v' \neq v}} x_{u,v'} \right) = 0$ if and only if all products have value zero. For every fixed u, the products can only be in the following three cases:

1. all elements of $\{x_{u,v} | v \in V_2\}$ have value zero;
2. exactly one element of $\{x_{u,v} | v \in V_2\}$ has value one;
3. more than one element of $\{x_{u,v} | v \in V_2\}$ have value one.

In the first case: $\sum_{v \in V_2} \left(x_{u,v} \sum_{\substack{v' \in V_2 \\ v' \neq v}} x_{u,v'} \right) = 0$, since all elements $\{x_{u,v} \mid v \in V_2\}$ have value zero. In the second cases: since $v' \neq v$ and there is only one $x_{u,v}$ can have value one, $x_{u,v}$ and $x_{u,v'}$ cannot simultaneously have value one. That means every product in $\sum_{v \in V_2} \left(x_{u,v} \sum_{\substack{v' \in V_2 \\ v' \neq v}} x_{u,v'} \right)$ has value zero. In the third case: since at least two elements of $\{x_{u,v} \mid v \in V_2\}$ have value one, there exists an $x_{u,v} = 1$ and an $x_{u,v'} = 1$ while $v \neq v'$. We then have at least one product in $\sum_{v \in V_2} \left(x_{u,v} \sum_{\substack{v' \in V_2 \\ v' \neq v}} x_{u,v'} \right)$ has value one.

In order to have $H(\mathbf{x}) = 0$, for every $u \in V_1$, the variables' assignment should be in either the first case or the second one. If it is in the first case, then that u is mapped to no vertex in V_2. If it is in the second case, then that u is mapped to exactly one vertex in V_2.

Similarly, we have the second part of $H(\mathbf{x}) = 0$ if and only if no two elements of V_1 is mapped to the same element in V_2 by $D(\mathbf{x})$. That means $\mathbf{x} \in \mathrm{Dom}(D)$. Therefore, we have if $H(\mathbf{x}) = 0$, then $D(\mathbf{x})$ is injective.

We then show that if $D(\mathbf{x})$ is injective, then $H(\mathbf{x}) = 0$.

Let us assume $D(\mathbf{x})$ is injective. Then for each $u \in V_1$, the value of all elements of $\{x_{u,v} | v \in V_2\}$ will fall into the first two cases. Hence, the sum of all these products are zero. That means $H(\mathbf{x}) = 0$.

Therefore, $H(\mathbf{x}) = 0$ if and only if $D(\mathbf{x})$ is injective.

□

Lemma 2. *If $H(\mathbf{x}) = 0$ then $P(\mathbf{x}) = 0$ if and only if $D(\mathbf{x})$ is edge-preserving.*

Proof. Assume $H(\mathbf{x}) = 0$ and let $\psi = D(\mathbf{x})$, we know that ψ is an injective function by Lemma 1. For brevity, define

$$P_{u,u'}(\mathbf{x}) = \sum_{v \in V_2} \left(x_{u,v} \sum_{v' \in V_2} x_{u',v'} \tau(u, u', v, v') \right).$$

Then

$$P(\mathbf{x}) = \sum_{uu' \in E_1} P_{u,u'}(\mathbf{x}).$$

Now $\tau(u, u', v, v')$ is a pre-computed constant that is either 0 or 1. Therefore $P_{u,u'}(\mathbf{x})$ is a linear combination of non-negative terms and is thus non-negative. Then it follows that

$$P(\mathbf{x}) = 0 \text{ if and only if } P_{u,u'}(\mathbf{x}) = 0, \text{ for all } uu' \in E_1.$$

Let us assume that $P(\mathbf{x}) = 0$. We know that $P_{u,u'}(\mathbf{x}) = 0$ and we can expand $P_{u,u'}(\mathbf{x})$ to obtain

$$\begin{aligned} P_{u,u'}(\mathbf{x}) &= \sum_{v \in V_2} x_{u,v} \Big(x_{u',v_0}\tau(u, u', v, v_0) + x_{u',v_1}\tau(u, u', v, v_1) + \cdots \\ &\quad + x_{u,v_{n_2-1}}\tau(u, u', v, v_{n_2-1}) \Big) \\ &= 0. \end{aligned}$$

Because ψ is injective we have two cases:

1. $x_{u,v} = 0$ for all $v \in V_2$, so uu' is not an edge in the common subgraph.
2. there exists a unique element $x^*_{u,v}$ with value one in $\{x_{u,v} \mid v \in V_2\}$ and similarly we have $x^*_{u',v'}$ in $\{x_{u',v'} \mid v' \in V_2\}$.

It then follows that we have

$$\begin{aligned} P_{u,u'}(\mathbf{x}) &= x^*_{u,v} \Big(x_{u',v_0}\tau(u, u', v, v_0) + x_{u',v_1}\tau(u, u', v, v_1) + \cdots \\ &\quad + x_{u,v_{n_2-1}}\tau(u, u', v, v_{n_2-1}) \Big) \\ &= x^*_{u,v} \Big(x^*_{u',v'}\tau(u, u', v, v')) \Big) \\ &= \tau(u, u', v, v'). \end{aligned}$$

We know $P_{u,u'}(\mathbf{x}) = 0$, thus $\tau(u, u', v, v') = 0$. From the definition of $\tau(u, u', v, v') = 0$, we know a possible edge mapping exists between uu' and vv' for all $uu' \in E_1$. Similarly, a possible edge mapping exists between uu' and vv' for all $vv' \in E_2$. Thus $D(x)$ is edge-preserving.

Conversely, assume that $D(x)$ is edge-preserving and $H(\mathbf{x}) = 0$ while $P(\mathbf{x}) \neq 0$. Since $H(\mathbf{x}) = 0$, ψ is injective and we have the same two cases as above. There then exists a uu' that is an edge of the common subgraph, such that $P_{u,u'}(\mathbf{x}) \neq 0$. Because it is in the second case, we have

$$P_{u,u'}(\mathbf{x}) \Leftrightarrow x^*_{u,v}x^*_{u',v'}\tau(u, u', v, v') \neq 0 \Leftrightarrow \tau(u, u', v, v') \neq 0.$$

It follows that there exists no $vv' \in E_2$, such that a possible edge mapping exists between uu' and vv'. Hence ψ is not edge-preserving. A contradiction arises.

Therefore we have $P(\mathbf{x}) = 0$ if and only if $D(\mathbf{x})$ is edge-preserving. □

Lemma 3. *If $H(\mathbf{x}) = 0$ then $N(\mathbf{x}) = 0$ if and only if $D(\mathbf{x})$ is non-edge-preserving.*

Proof. An analogue of the proof of Lemma 2 proves this lemma. □

Corollary 1. *For all $\mathbf{x} \in \mathbb{Z}_2^{n_1 n_2}$, $A[H(\mathbf{x}) + P(\mathbf{x}) + N(\mathbf{x}))] = 0$ if and only if $D(\mathbf{x})$ is an injective, edge-invariant function.*

Proof. All three components, $H(\mathbf{x}), P(\mathbf{x}), N(\mathbf{x})$ are non-negative. We also know from Lemma 1 and 2 that $H(\mathbf{x}) = 0$ if and only if $D(\mathbf{x})$ is injective and $P(\mathbf{x}) = 0$ if and only if $D(\mathbf{x})$ is edge-preserving. Lemma 3 shows $N(\mathbf{x}) = 0$ if and only if $D(\mathbf{x})$ is non-edge-preserving. Hence for all $\mathbf{x} \in \mathbb{Z}_2^{n_1 n_2}$, $A[H(\mathbf{x}) + P(\mathbf{x}) + N(\mathbf{x}))] = 0$ if and only if $D(\mathbf{x})$ is an injective, edge-invariant function. □

Lemma 4. $\min(\mathrm{Im}(F)) \leq 0$.

Proof. If $\mathbf{x} = (0, \ldots, 0)$, then $F(\mathbf{x}) = 0$. Hence $\min(\mathrm{Im}(F)) \leq 0$. □

Lemma 5. *If $\mathbf{x} = \min_{\mathbf{x}} F(\mathbf{x})$ then $D(\mathbf{x})$ is an edge-invariant injection.*

Proof. Assume $F(\mathbf{x})$ is the minimum value of F. By Lemma 4, we have $F(\mathbf{x}) \leq 0$. Suppose that $D(\mathbf{x})$ is not an injective, edge-invariant function. That is, $A[H(\mathbf{x}) + P(\mathbf{x}) + N(\mathbf{x})] > 0$. For B sufficiently smaller than A we have $F(\mathbf{x}) > 0$. A contradiction arises. Therefore, we have $D(\mathbf{x})$ is an edge-invariant injection. □

Theorem 1. *If $\mathbf{x} = \min_{\mathbf{x}} F(\mathbf{x})$ then $D(\mathbf{x})$ is a solution to MCISI.*

Proof. By Lemma 5 we know that $D(\mathbf{x})$ is an injective, edge-invariant function. Suppose that $D(\mathbf{x})$ is not a solution to MCISI and instead we have an $\mathbf{x}'$ such that $D(\mathbf{x}')$ is. Since $D(\mathbf{x}')$ is a solution of MCISI, it must be an injective, edge-invariant function. Hence $A[H(\mathbf{x}') + P(\mathbf{x}') + N(\mathbf{x}')] = 0$. However, $M(\mathbf{x}') > M(\mathbf{x})$ as $D(\mathbf{x}')$ is a solution to MCISI and thus the corresponding common induced subgraph of $D(\mathbf{x})'$ has more vertices than the corresponding common induced subgraph of $D(\mathbf{x})$. So we obtain $F(\mathbf{x}') < F(\mathbf{x})$. However, this is not possible, since $F(\mathbf{x})$ is the minimal value of F. Hence $D(\mathbf{x})$ is a solution to MCISI. □

2.4 QUBO formulation for MCESI

Following a similar path, we will design penalty terms in the QUBO objective function according to the features of MCESI. Let us work with the two graphs $G_1 = (V_1, E_1, \alpha_1, \beta_1)$ and $G_2 = (V_2, E_2, \alpha_2, \beta_2)$. We denote the MCES of G_1 and G_2 as $S = (V_S, E_S, \alpha_S, \beta_S)$ and the MCESI from G_1 to G_2 as γ. This means there exists a subgraph $S_1 = (V_{S_1}, E_{S_1}, \alpha_{S_1}, \beta_{S_1})$ of G_1 and a subgraph $S_2 = (S_{S_2}, E_{S_2}, \alpha_{S_2}, \beta_{S_2})$ of G_2, such that:

- $\alpha_{S_1}(v_1) = \alpha_{S_2}(\gamma(v_1))$ for all $v_1 \in V_{S_1}$, and $\alpha_{S_2}(v_2) = \alpha_{S_1}(\gamma^{-1}(v_2))$ for all $v_2 \in V_{S_2}$ (injective);
- for any edge $e_1 = (v_1, v_1') \in E_{S_1}$, there exists an edge $e_2 = (\gamma(v_1), \gamma(v_1')) \in E_{S_2}$, such that $\beta_{S_1}(e_1) = \beta_{S_2}(e_2)$, and for any edge $e_2 = (v_2, v_2') \in E_{S_2}$, there exists an edge $e_1 = (\gamma^{-1}(v_2), \gamma^{-1}(v_2')) \in E_{S_1}$, such that $\beta_{S_1}(e_1) = \beta_{S_2}(e_2)$ (edge-preserve).

Then we use the same variable system and function τ to encode these features into the QUBO objective function. We define the objective function of MCESI as

$$F : \mathbb{Z}_2^{n_1 n_2} \to \mathbb{R},$$
$$F(\mathbf{x}) = A[H(\mathbf{x}) + P(\mathbf{x})] - BM(\mathbf{x}), A, B \in \mathbb{R}^+, \tag{9}$$

by

$$H(\mathbf{x}) = \sum_{u \in V_1} \sum_{v \in V_2} \left(x_{u,v} \sum_{\substack{v' \in V_2 \\ v' \neq v}} x_{u,v'} \right) + \sum_{v \in V_2} \sum_{u \in V_1} \left(x_{u,v} \sum_{\substack{u' \in V_1 \\ u' \neq u}} x_{u',v} \right), \tag{10}$$

$$P(\mathbf{x}) = \sum_{uu' \in E_1} \sum_{v \in V_2} \left(x_{u,v} \sum_{v' \in V_2} x_{u',v'} \tau(u, u', v, v') \right), \tag{11}$$

$$M(\mathbf{x}) = \sum_{uu' \in E_1} \sum_{vv' \in E_2} \left(x_{u,v} x_{u',v'} \right). \tag{12}$$

We request B to be sufficiently smaller than A.

The function $H(\mathbf{x})$ is designed to add penalty when the mapping is not injective. $P(\mathbf{x})$ adds penalty when the mapping is not edge-preserving. $M(\mathbf{x})$ adds penalty when the number of edges is not maximised.

We then prove the correctness of this objective function.

Corollary 2. *For all $\mathbf{x} \in \mathbb{Z}_2^{n_1 n_2}$, $A[H(\mathbf{x}) + P(\mathbf{x}))] = 0$ if and only if $D(\mathbf{x})$ is an injective, edge-preserving function.*

Proof. Both $H(\mathbf{x})$ and $P(\mathbf{x})$ are non-negative functions. We also know from Lemma 1 and 2 that $H(\mathbf{x}) = 0$ if and only if $D(\mathbf{x})$ is injective and $P(\mathbf{x}) = 0$ if and only if $D(\mathbf{x})$ is edge-preserving. Hence it follows that $A[H(\mathbf{x}) + P(\mathbf{x})] = 0$ if and only if $D(\mathbf{x})$ is an injective, edge-preserving function. □

Lemma 6. *If $\mathbf{x} = \min_{\mathbf{x}} F(\mathbf{x})$ then $D(\mathbf{x})$ is an edge-preserving injection.*

Proof. Assume $F(\mathbf{x})$ is the minimum value of F. By Lemma 4, we have $F(\mathbf{x}) \leq 0$. Suppose that $D(\mathbf{x})$ is not an injective, edge-preserving function. That is, $A[H(\mathbf{x}) + P(\mathbf{x})] > 0$. For B sufficiently smaller than A we have $F(\mathbf{x}) > 0$. A contradiction arises. Therefore, we have $D(\mathbf{x})$ is an edge-preserving injection.

□

Theorem 2. *If $\mathbf{x} = \min_{\mathbf{x}} F(\mathbf{x})$ then $D(\mathbf{x})$ is a solution to MCESI.*

Proof. By Lemma 6 we know that $D(\mathbf{x})$ is an injective, edge-preserving function. Suppose that $D(\mathbf{x})$ is not a solution to MCESI and instead we have an $\mathbf{x}'$ such that $D(\mathbf{x}')$ is. Since $D(\mathbf{x}')$ is a solution of MCESI, it must be an injective, edge-preserving function. Hence $A[H(\mathbf{x}') + P(\mathbf{x}')] = 0$. However, $M(\mathbf{x}') > M(\mathbf{x})$ as $D(\mathbf{x}')$ is a solution to MCESI and thus the corresponding common edge subgraph of $D(\mathbf{x}')$ has more edges than the corresponding common edge subgraph of $D(\mathbf{x})$. So we obtain $F(\mathbf{x}') < F(\mathbf{x})$. However, this is not possible, since $F(\mathbf{x})$ is the minimal value of F. Hence $D(\mathbf{x})$ is a solution to MCESI. □

3 Experiments on a D-Wave 2X computer

We have used a D-Wave 2X system to test the performance of our QUBO formulations. We first used a classical solver to find the optimal answers for the QUBO matrices and decode them into the solutions of corresponding maximum common subgraph isomorphism problems. Then we compared them with the solutions from a classical maximum common subgraph isomorphism solver. After these checks, we run these instances on a D-Wave 2X system and collected data for further analysis.

D-Wave 2X systems use Chimera graph architecture, an example is showed in Figure 3. It is not a complete graph. Therefore, we used the default minor-embedding algorithm that D-Wave company provides in their user interface [11] to find a subgraph of the machine's hardware graph, such that the QUBO graph is a graph minor of it.

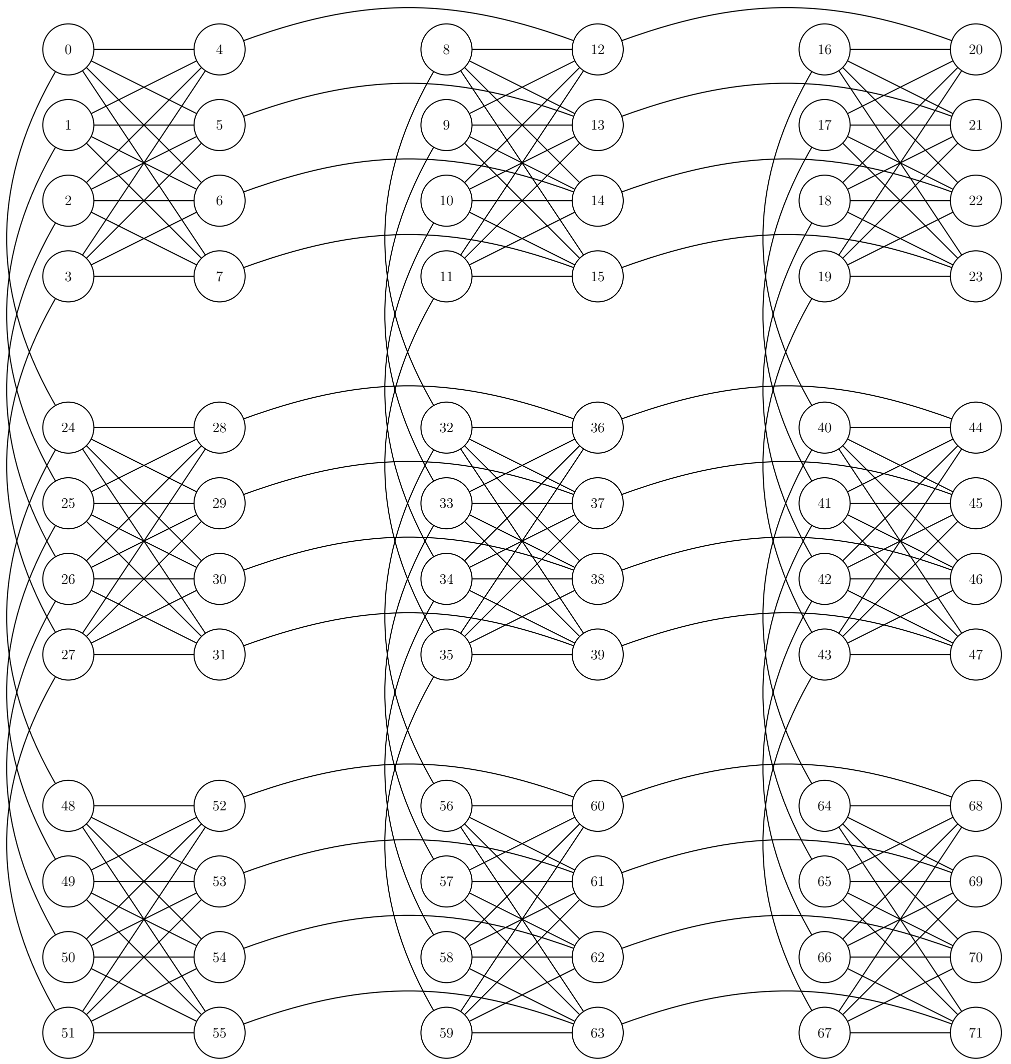
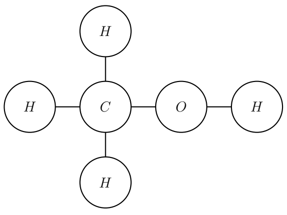
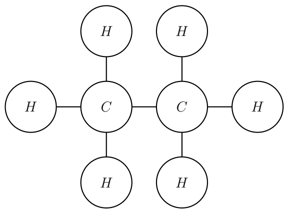


Figure 3: Chimera architecture example.

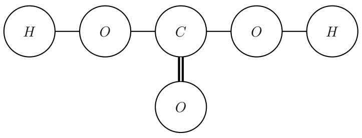
Since our QUBO formulation for both MCISI and MCESI needs to have a logical variable for each possible vertex mapping and the QUBO matrix is quite dense, we do not have enough resources (qubits) to run for large graphs. Hence, only small graphs that represent some small organic molecules are used to generate instances for the D-Wave 2X system. We represent some simple molecules as graphs in Figure 4. All these molecules are very small; they contain no more than nine atoms. Different atoms are given different labels, such as C, O and H. Different covalent bonds are labelled as single lines and double lines.



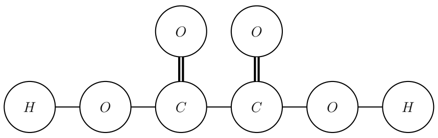
(a) Methanol



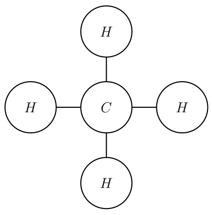
(b) Ethane



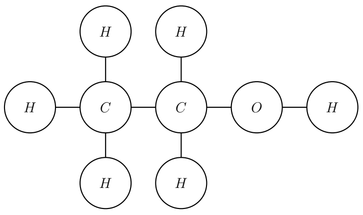
(c) Carbonic acid



(d) Oxalic acid



(e) Methane



(f) Ethanol

Figure 4: Input graphs

We have used the QUBO formulations in Section 2 to generate the corresponding QUBO matrices for MCISI and MCESI corresponding to the chosen graphs. Then the QUBO matrices were used by the default embedding program from D-Wave to generate the corresponding embedding. We repeated this process for all possible pairs of graphs in list in Figure 4. Finally, we have sent the batch of problems to the D-Wave computer for processing. The following table shows the performance of the D-Wave 2X runs.

In the experiments, we used the majority voting feature that the D-Wave API provides. When a broken chain appears, it will force the qubits in the same chain to have the same value as that of more than half the qubits. The tables show how effective the algorithm works on D-Wave 2X. The first column shows which pair of molecule graphs was used as input. The second column shows the length of the max chain after we have performed minor embedding. The third column shows the average chain length of all logical variables. The fourth column is the count of correct solutions (compare with the

input pair of graphs	max chain length	average chain length	count of correct answers with post-process	count of correct answers without post-process
CarbonicAcid_CarbonicAcid	4	2.57	4855	3070
CarbonicAcid_Ethane	4	3.14	0	532
CarbonicAcid_Ethanol	6	4.47	0	1
CarbonicAcid_Methane	3	2.44	0	11
CarbonicAcid_Methanol	3	2.41	5000	4505
CarbonicAcid_OxalicAcid	6	4.61	0	0
Ethane_CarbonicAcid	5	3.64	1	12
Ethane_Ethane	16	12.05	0	0
Ethane_Ethanol	14	10.77	0	0
Ethane_Methane	8	6.5	402	255
Ethane_Methanol	9	7.03	0	0
Ethane_OxalicAcid	5	3.75	2820	533
Ethanol_CarbonicAcid	6	4.35	0	0
Ethanol_Ethane	18	12.17	0	0
Ethanol_Ethanol	18	13.41	0	0
Ethanol_Methane	9	7.19	53	86
Ethanol_Methanol	12	7.74	0	0
Ethanol_OxalicAcid	6	5.3	6	0
Methane_CarbonicAcid	3	2.44	4955	4662
Methane_Ethane	9	6.42	0	0
Methane_Ethanol	9	7.11	0	0
Methane_Methane	5	3.58	1371	545
Methane_Methanol	5	3.94	499	506
Methane_OxalicAcid	4	3.1	4466	3744
Methanol_CarbonicAcid	5	3.25	5000	3423
Methanol_Ethane	9	7.11	0	0
Methanol_Ethanol	10	7.51	0	0
Methanol_Methane	5	3.70	179	55
Methanol_Methanol	5	4.05	12	14
Methanol_OxalicAcid	5	4.0	4952	3763
OxalicAcid_CarbonicAcid	7	5.0	0	0
OxalicAcid_Ethane	5	3.81	7	21
OxalicAcid_Ethanol	6	5.05	1	3
OxalicAcid_Methane	3	2.7	0	28
OxalicAcid_Methanol	6	4.35	3609	1132
OxalicAcid_OxalicAcid	9	7.25	0	0

Figure 5: Maximum common induced subgraph isomorphism results.

input pair of graphs	max chain length	average chain length	count of correct answers with post-process	count of correct answers without post-process
CarbonicAcid_CarbonicAcid	6	3.28	308	217
CarbonicAcid_Ethane	3	2.71	5000	4998
CarbonicAcid_Ethanol	4	3.11	0	8
CarbonicAcid_Methane	2	1.66	5000	4978
CarbonicAcid_Methanol	3	2.25	24	1196
CarbonicAcid_OxalicAcid	7	4.22	0	0
Ethane_CarbonicAcid	4	3.64	4004	3513
Ethane_Ethane	19	11.67	0	0
Ethane_Ethanol	14	10.32	0	0
Ethane_Methane	10	7.53	0	0
Ethane_Methanol	8	6.61	0	0
Ethane_OxalicAcid	5	4.43	1821	1170
Ethanol_CarbonicAcid	5	4.23	58	1
Ethanol_Ethane	14	10.82	0	0
Ethanol_Ethanol	18	11.0	0	0
Ethanol_Methane	10	6.34	26	10
Ethanol_Methanol	10	7.59	0	0
Ethanol_OxalicAcid	7	4.55	0	0
Methane_CarbonicAcid	4	2.55	4995	4966
Methane_Ethane	10	8.03	0	0
Methane_Ethanol	9	7.42	0	0
Methane_Methane	6	4.70	167	20
Methane_Methanol	6	4.58	99	287
Methane_OxalicAcid	4	2.7	5000	4978
Methanol_CarbonicAcid	4	2.91	3559	147
Methanol_Ethane	10	6.92	0	0
Methanol_Ethanol	10	7.66	0	0
Methanol_Methane	6	4.64	34	253
Methanol_Methanol	5	4.27	22	2
Methanol_OxalicAcid	6	3.42	1012	53
OxalicAcid_CarbonicAcid	5	3.83	62	3
OxalicAcid_Ethane	10	3.0	5000	3259
OxalicAcid_Ethanol	8	3.15	0	0
OxalicAcid_Methane	2	1.6	5000	4963
OxalicAcid_Methanol	5	2.57	1384	550
OxalicAcid_OxalicAcid	9	5.70	0	0

Figure 6: Maximum common edge subgraph isomorphism results.

results from a classical QUBO solver) out of 5000 rounds with post-processing. The fifth column is the count of correct solutions (compare with the results from a classical QUBO solver) out of 5000 rounds without post-processing. Figure 5 and Figure 6 show that when both the maximum chain length and average chain length are small enough, the performance is quite good: a majority of the graph pairs have a high count of correct answers. However, when chain length grows, the performance drops significantly. This is consistent with other experiments [16].

D-Wave API provides a post-process function that classically optimises the quantum machine's output [10]. According to this document, a D-Wave computer will break the embedded graph into several low tree-width subgraphs, then it will run local search for those subgraphs and combine the results to update the global result. Since the embedded graph contains all constraints for the problem, while the subgraphs only contain part of the constraints, it is possible that the post-processing may give a wrong answer. We run the same examples with the post-processing on and off to see whether this explanation makes sense. Figure 5 and Figure 6 show that when the chain length is long but not too long, without post-optimisation, we may get a bit more correct answers. However, when the chain length is short, the post-processing improves the quality of the answers. These facts give some support to the explanation. More importantly, one has to be careful about when to turn the post-processing on. For the current problem, verifying answers is as difficult as finding them. Therefore, even when we know that some of the correct answers may be discarded by the post-processing, we may still want to apply the post-processing, as it is too time consuming to investigate when the post-processing makes mistakes. However, for the problems like integer factoring, where it is hard to find the answers, but very easy to verify them, the situation is different. For such cases it is advisable to turn the post-processing off to avoid discarding correct answers.

It would be interesting to compare the run times obtained with the quantum solution with those obtained with the state of art heuristic classical solvers. However, the comparison would not make much sense unless larger scale problems could be solved on the quantum annealing machine (the heuristic classical solvers mentioned in Section 2.2 are capable of solving much larger scale problems).

4 Conclusion and future work

We have proposed QUBO formulations for the problems MCISI and MCESI, proved their correctness, Section 2, then solved instances of the problems for some small organic molecules that on a real D-Wave computer and analyse the experimental data, Section 3. Very small scale problems can be solved by the D-Wave 2X machine with high enough accurate rates; however, for larger scale problems, the accurate rate drops significantly. With 5,000+ qubits and a 15-way qubit connectivity for its Pegasus architecture, the D-Wave Advantage can embed larger scale problems and has a better technology of managing the broken chain. The accuracy of the results reported in this paper could be better on this quantum computer.

In the current version of our algorithm, we have set no restrictions on the connectivity of the common subgraphs. The density of a graph can be defined as the ratio of the size of the graph (number of edges) and the size of the complete graph with the same order (number of vertices). Since we are investigating the common sub-structure shared by two graphs, a denser common subgraph may have a better chance to give the relevant information. Therefore, one possible future step is to add in some constrains that will force the answer to be denser. We could also investigate methods that require fewer logical variables and/or fewer interactions among the logical variables. The former may help to extract the information of interest and the latter may improve chances to solve larger problems on the same machine. If larger scale problems could be solved on future quantum annealing machines, then meaningful comparisons between the proposed quantum algorithms and the state of art heuristic classical ones would be possible.

5 Acknowledgement

The authors thank Cristian Calude, Michael Dinneen and Richard Hua for helpful comments. The authors also thank Michael Dinneen for running the experiments on the D-Wave 2X system and the anonymous referees for critical comments which improved the presentation. This work was supported in part by the Quantum Computing Research Initiatives at Lockheed Martin.

6 Conflict of Interest

The authors declare that they have no conflict of interest.

References

[1] Maha T Barakat and Philip M Dean. Molecular structure matching by simulated annealing. iii. the incorporation of null correspondences into the matching problem. *Journal of computer-aided molecular design*, 5(2):107–117, 1991.

[2] Harry G Barrow and BURSTALL RM. Subgraph isomorphism, matching relational structures and maximal cliques. 1976.

[3] Zhengbing Bian, Fabian Chudak, William Macready, Aidan Roy, Roberto Sebastiani, and Stefano Varotti. Solving sat (and maxsat) with a quantum annealer: Foundations, encodings, and preliminary results. *Information and Computation*, page 104609, 2020.

[4] Vincenzo Bonnici, Rosalba Giugno, Alfredo Pulvirenti, Dennis Shasha, and Alfredo Ferro. A subgraph isomorphism algorithm and its application to biochemical data. *BMC bioinformatics*, 14(7):S13, 2013.

[5] Robert D Brown, Gareth Jones, Peter Willett, and Robert C Glen. Matching two-dimensional chemical graphs using genetic algorithms. *Journal of Chemical Information and Computer Sciences*, 34(1):63–70, 1994.

[6] 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.

[7] Yiqun Cao, Tao Jiang, and Thomas Girke. A maximum common substructure-based algorithm for searching and predicting drug-like compounds. *Bioinformatics*, 24(13):i366–i374, 2008.

[8] Guillaume Chapuis, Hristo Djidjev, Georg Hahn, and Guillaume Rizk. Finding maximum cliques on the d-wave quantum annealer. *Journal of Signal Processing Systems*, 91(3-4):363–377, 2019.

[9] Donatello Conte, Pasquale Foggia, Carlo Sansone, and Mario Vento. Thirty years of graph matching in pattern recognition. *International journal of pattern recognition and artificial intelligence*, 18(03):265–298, 2004.

[10] D-Wave. Postprocessing methods.

[11] D-Wave. Developer's guide. *D-Wave system Inc.*, 2016.

[12] Guillaume Damiand, Christine Solnon, Colin De La Higuera, Jean-Christophe Janodet, and Émilie Samuel. Polynomial algorithms for subisomorphism of nd open combinatorial maps. *Computer Vision and Image Understanding*, 115(7):996–1010, 2011.

[13] Michael J Dinneen, Mohammad Reza Hooshmandasl, and Richard Hua. Formulating mixed dominating set problems for adiabatic quantum computers. Technical report, Department of Computer Science, The University of Auckland, New Zealand, 2017.

[14] Edmund Duesbury, John Holliday, and Peter Willett. Comparison of maximum common subgraph isomorphism algorithms for the alignment of 2d chemical structures. *ChemMedChem*, 13(6):588–598, 2018.

[15] Nobuo Funabiki and Junji Kitamichi. A two-stage discrete optimization method for largest common subgraph problems. *IEICE TRANSACTIONS on Information and Systems*, 82(8):1145–1153, 1999.

[16] Richard Hua and Michael J. Dinneen. Improved qubo formulation of the graph isomorphism problem. *SN Computer Science*, 1(1):19, Sep 2019.

[17] Giorgio Levi. A note on the derivation of maximal common subgraphs of two directed or undirected graphs. *Calcolo*, 9(4):341, 1973.

[18] Alex Mott, Joshua Job, Jean-Roch Vlimant, Daniel Lidar, and Maria Spiropulu. Solving a higgs optimization problem with quantum annealing for machine learning. *Nature*, 550(7676):375–379, 2017.

[19] Daniel O'Malley, Velimir V Vesselinov, Boian S Alexandrov, and Ludmil B Alexandrov. Nonnegative/binary matrix factorization with a d-wave quantum annealer. *PloS one*, 13(12):e0206653, 2018.

[20] Olawale Titiloye and Alan Crispin. Quantum annealing of the graph coloring problem. *Discrete Optimization*, 8(2):376–384, 2011.

[21] Hayato Ushijima-Mwesigwa, Christian FA Negre, and Susan M Mniszewski. Graph partitioning using quantum annealing on the d-wave system. In *Proceedings of the Second International Workshop on Post Moores Era Supercomputing*, pages 22–29. ACM, 2017.

[22] Cesare Valenti. A genetic approach to the maximum common subgraph problem. In *Proceedings of the 20th International Conference on Computer Systems and Technologies*, pages 98–104, 2019.

[23] Rogier JP Van Berlo, Wynand Winterbach, Marco JL De Groot, Andreas Bender, Peter JT Verheijen, Marcel JT Reinders, and Dick De Ridder. Efficient calculation of compound similarity based on maximum common subgraphs and its application to prediction of gene transcript levels. *International journal of bioinformatics research and applications*, 9(4):407–432, 2013.

[24] Markus Wagener and Johann Gasteiger. The determination of maximum common substructures by a genetic algorithm: Application in synthesis design and for the structural analysis of biological activity. *Angewandte Chemie International Edition in English*, 33(11):1189–1192, 1994.

[25] Natsuhito Yoshimura, Masashi Tawada, Shu Tanaka, Junya Arai, Satoshi Yagi, Hiroyuki Uchiyama, and Nozomu Togawa. Efficient ising model mapping for induced subgraph isomorphism problems using ising machines. In *2019 IEEE 9th International Conference on Consumer Electronics (ICCE-Berlin)*, pages 227–232. IEEE, 2019.

[26] Kenneth M Zick, Omar Shehab, and Matthew French. Experimental quantum annealing: case study involving the graph isomorphism problem. *Scientific reports*, 5:11168, 2015.