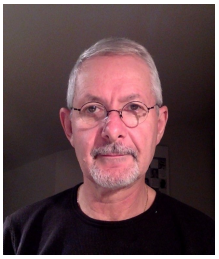
Guest Column: Adiabatic Quantum Computing Challenges[1]

Cristian S. Calude[2] *Elena Calude*[3] *Michael J. Dinneen*[2]



Abstract

The paper presents a brief introduction to quantum computing with focus on the adiabatic model which is illustrated with the commercial D-Wave computer. We also include new theory and experimental work done on the D-Wave computer. Finally we discuss a hybrid method of combining classical and quantum computing and a few open problems.

1 Introduction

The proliferation of personal computers and intelligent gadgets, the Internet and industries' large amounts of data, and modern scientific challenges, all fuel a huge need of computing power.

If we believe in Moore's Law—which predicts that the number of transistors on a microprocessor will continue to double every 18 months—then around the years 2020–2030 the circuits on a microprocessor will be on an atomic scale. Accordingly, new types of computers will have to be designed. Will the new computer be quantum, a machine theorized for more than 30 years? Will the quantum computer kill the silicon-based computer?

In what follows we present a few elementary facts about quantum computing followed by the adiabatic model illustrated with D-Wave, the first commercialized adiabatic quantum computer. Two NP-complete problems, the independent set and the network broadcast problems, are used to show how D-Wave works. We finish with "quassical computing," a hybrid method of combining classical and quantum computing, and a few open problems.

2 A glimpse of quantum computing

Quantum computing was first introduced by Yuri I. Manin [39, 40] in 1980 and Richard Feynman [27] in 1982 and intensively researched afterwards. In this section we briefly present the basics of quantum computing, milestone results, their meaning, power and limits.

In 1985 Deutsch [21] constructed the first model of quantum computer by quantization of the universal Turing machine. In 1994 Shor [56] designed a quantum algorithm for factoring integers

 Supported in part by the Quantum Computing Research Initiatives at Lockheed Martin.

[2]Department of Computer Science, University of Auckland, Auckland, New Zealand. {cristian,mjd}@cs.auckland.ac.nz

[3]Institute of Natural and Mathematical Sciences, Massey University at Albany, Auckland, New Zealand. e.calude@massey.ac.nz.

in polynomial (quantum) time in the size of the input; the problem whether there is a classical polynomial algorithm for factoring is still open (see more in Section 3). Two years later Grover [31] discovered a quantum algorithm for searching an unsorted N-entry database in $O(\sqrt{N})$ time and $O(\log N)$ space: this algorithm is optimal within the quantum computing model for black box oracles [7]. Quantum algorithms are probabilistic and give the correct answer with high probability; the probability of failure can be decreased by repeating the algorithm.

The most popular model of quantum computing is probably the *circuit (gate) model* in which quantum algorithms are built from a small set of quantum gates. *Adiabatic quantum computing*—proposed in 2000 [25]—relies on the adiabatic theorem (see Section 4) to do calculations [18].

According to [61] (see also [29]),

> a quantum computer is a computation system that makes direct use of quantum-mechanical phenomena, such as superposition and entanglement, to perform operations on data.

Classical computers encode data into binary digits (bits) using classical physical systems which are always in one of two definite states (0 or 1). Quantum computers operate with qubits (quantum bits) which are described as quantum states (i.e. abstract, in Hilbert space) and physically implemented by quantum systems (e.g., an atom) which are in one of two definite states, denoted by $|0\rangle$ or $|1\rangle$. Unlike classical bits, qubits can be in a *superposition* of states represented as a linear combination of $|0\rangle$ and $|1\rangle$, e.g., $\alpha|0\rangle + \beta|1\rangle$, $|\alpha|^2 + |\beta|^2 = 1$. Here α and β are complex numbers representing amplitudes. When we *measure* a qubit, the probability of outcome $|0\rangle$ is $|\alpha|^2$ and the probability of outcome $|1\rangle$ is $|\beta|^2$. For example, in the quantum state $\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$, $|0\rangle$ and $|1\rangle$ have the same probability $\frac{1}{2}$. While a quantum computer can work on all states within a superposition, we only get the results for one state (or a property of all the states) at measurement time. A quantum computer with two qubits can be in all superpositions of the states $((|0\rangle, |0\rangle), (|0\rangle, |1\rangle), (|1\rangle, |0\rangle)$ and $(|1\rangle, |1\rangle))$, each pair requiring two coefficients, one for $|0\rangle$ and another one for $|1\rangle$; in general, n qubits need $2n$ numbers.

Classical parallel computers operate simultaneously with *many* processing units. In contrast, *superposition* allows a quantum computer to "run" on all its possible quantum states simultaneously, thus naturally operate in parallel with a *single* processing unit. Other quantum effects useful for quantum computation are: a) *entanglement*, the quantum phenomenon that occurs when the quantum state of each particle in the group cannot be described independently from the quantum state of the group as a whole (for example, the state $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ is entangled as it cannot be separated as a tensor product of any two states), b) *tunneling*, the quantum phenomenon where a particle tunnels through a barrier, classically an impossibility.

The quantum evolution (quantum transformation, operator) of (on) a qubit is described by a "unitary operator," that is an operator induced by a unitary matrix. In this way we can construct quantum gates which operate on qubits, the building blocks of the quantum gate model. For example, the NOT transformation interchanging the vectors $|0\rangle$ and $|1\rangle$, that is the matrix $\text{NOT} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$: NOT $|0\rangle = |1\rangle$ and NOT $|1\rangle = |0\rangle$. The square-root of NOT operator, defined by

$$\sqrt{\text{NOT}} = \frac{1}{2}\begin{pmatrix} 1+i & 1-i \\ 1-i & 1+i \end{pmatrix}; (i = \sqrt{-1})$$

satisfies the equality

$$\sqrt{\text{NOT}} \cdot \sqrt{\text{NOT}} = \text{NOT} . \tag{1}$$

This operator is a typical "quantum gate" in the sense that *it is impossible to have a single-input/single-output classical binary logic gate that satisfies* (1). Quantum computers solve problems probabilistically, generally with high probability. For more details see [32, 15].

2.1 Deutsch's problem

Probably the simplest and most frequently used way to illustrate the power of quantum computing is to solve *Deutsch's problem*: *Given a (classical) black box that computes an unknown binary function* $f\colon \{0,1\} \to \{0,1\}$, *decide whether* f *is* constant ($f(0) = f(1)$) or balanced ($f(0) \neq f(1)$).

Classically, to solve the problem seems to require the computation of $f(0)$ and $f(1)$, and comparing the results. Is it possible to solve the problem with *only one* computation of f? In a famous paper [21], Deutsch posed the problem and obtained a "quantum" partial affirmative answer. A complete, probability-one solution was presented in [17]; see [32, 47, 43].

2.2 Deutsch's problem: the quantum solution

Suppose that we have a quantum black box which computes a function that coherently extends f from $\{0,1\}$ to the quantum (Hilbert) space generated by the base $\{|0\rangle, |1\rangle\}$, i.e. on input $|x\rangle$ it produces $|f(x)\rangle$, $x \in \{0,1\}$. The quantum transformation U_f maps two qubits, $|x\rangle$ and $|y\rangle$, to $|x\rangle|y \oplus f(x)\rangle$ ($\oplus$ denotes the sum modulo 2). We start the computation with $|x\rangle$ and $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$:

$$U_f\left(|x\rangle\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)\right) = |x\rangle\frac{1}{\sqrt{2}}(|0 \oplus f(x)\rangle - |1 \oplus f(x)\rangle) = (-1)^{f(x)}|x\rangle\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle).$$

With $x = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ we get:

$$U_f\left(\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)\right) = \frac{1}{2}(-1)^{f(0)}(|0\rangle + (-1)^{f(0)\oplus f(1)}|1\rangle)(|0\rangle - |1\rangle). \tag{2}$$

If we perform a measurement that projects the first qubit onto the basis $\frac{1}{\sqrt{2}}(|0\rangle+|1\rangle)$, $\frac{1}{\sqrt{2}}(|0\rangle-|1\rangle)$ we will obtain $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ if the function f is balanced and $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ in the opposite case.

The action of (2) can be presented in matrix form as:

$$U_f = \begin{pmatrix} 1-f(0) & f(0) & 0 & 0 \\ f(0) & 1-f(0) & 0 & 0 \\ 0 & 0 & 1-f(1) & f(1) \\ 0 & 0 & f(1) & 1-f(1) \end{pmatrix}.$$

The quantum algorithm solving Deutsch's problem, presented on the next page, uses the Hadamard transformation H to generate a superposition of states.

$$H = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & -\frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & \frac{1}{2} \end{pmatrix}.$$

1. Start with a closed physical system prepared in the quantum state $|01\rangle$.
2. Evolve the system according to H.
3. Evolve the system according to U_f.
4. Evolve the system according to H.
5. Measure the system.
6. If the result is the second possible output, then f is constant; if the result is the fourth possible output, then f is balanced.

The Deutsch's quantum algorithm can be presented mathematically as follows:

$$HU_fH|01\rangle = HU_fH \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 - f(0) - f(1) \\ 0 \\ f(1) - f(0) \end{pmatrix}.$$

If we *measure* the state $HU_fH|01\rangle$ we get: a) output 1 with probability $p_1 = 0$, b) output 2 with probability $p_2 = (1 - f_Q(|0\rangle) - f_Q(|1\rangle))^2$, c) output 3 with probability $p_3 = 0$, d) output 4 with probability $p_4 = (f_Q(|1\rangle) - f_Q(|0\rangle))^2 = 1 - p_2$. By magic,[4] the quantum algorithm has solved, with probability one, the problem with just one computation of the black box. *There was a widespread belief that this cannot be done classically*, see [21, 32, 15, 43].

2.3 Deutsch's problem: a classical solution

The success of the quantum solution relies on the following three facts: (a) the "embedding"[5] of f into f_Q (see also the discussion in [42]), (b) the ability of the quantum computer to be in a superposition of states: we can check whether $f_Q(|0\rangle)$ is equal or not to $f_Q(|1\rangle)$ not by computing f_Q on $|0\rangle$ and $|1\rangle$, but on a superposition of $|0\rangle$ and $|1\rangle$, and (c) the possibility to extract the required information with just one measurement.

Assuming that we are allowed to use an "embedding," then we *de-quantize*[6] Deutsch's algorithm in the following way [12]. Let $\mathbf{Q}$ be the set of rationals and $\mathbf{Q}[i] = \{a + bi \mid a, b \in \mathbf{Q}\}$. We embed the original function f in $\mathbf{Q}[i]$ and we define the classical analogue C_f of the quantum evolution U_f acting from $\mathbf{Q}[i]$ to itself as follows (compare with the formula (2)): $C_f(a + bi) = (-1)^{0 \oplus f(0)}a + (-1)^{1 \oplus f(1)}bi$.

There are four different possible bit-functions C_f from $\mathbf{Q}[i]$ to $\mathbf{Q}[i]$ ($\bar{x}$ is the complex conjugate of x): $C_{00}(x) = \bar{x}$, if $f(0) = 0, f(1) = 0, C_{01}(x) = x$, if $f(0) = 0, f(1) = 1, C_{10}(x) = -x$, if $f(0) = 1, f(1) = 0, C_{11}(x) = -\bar{x}$, if $f(0) = 1, f(1) = 1$. Deutsch's problem becomes: *Given a function f in the set $\{C_{00}, C_{01}, C_{10}, C_{11}\}$ determine whether the function is balanced or constant.*

A simple verification shows the correctness of the following *deterministic* classical algorithm which solves the Deutsch's problem with our computation of C_f:

Given f, calculate $(i - 1) \times C_f(1 + i)$. If the result is real, then the function is balanced; otherwise, the function is constant.

[4] "Any sufficiently advanced technology is indistinguishable from magic," A. C. Clarke.

[5] This is controversial as it changes the problem: instead of the classical box we use the quantum box which has far more computing power.

[6] De-quantization, a term coined in [12], is the process of producing a classical algorithm which is as efficient as its quantum counterpart.

Here the power of superposition can be exploited classically in a simpler way: there is no need of quantum physics.

3 What is a quantum computer?

The definition of quantum computer reproduced in Section 2.3 refers to the "direct use of quantum-mechanical phenomena, such as superposition and entanglement, to perform operations on data." The reference to quantum phenomena appears also in Webster dictionary's definition of quantum computer [59]: "A type of computer which uses the ability of quantum systems, such as a collection of atoms, to be in many different states at once. In theory, such superpositions allow the computer to perform many different computations simultaneously."

The first informal definitions of quantum computer/computation were proposed in [23, 24] and have been formalized (to some degree) ten years later in [48]. They resulted in four general requirements—the availability of a quantum memory, the ability to induce a (near) unitary evolution, the ability to implement (information-theoretic) cooling and readout of the quantum memory. Variations of these criteria have been proposed by M. Mosca and discussed in [45].

The quantum computing community is an interdisciplinary one, including among others, physicists, engineers, computer scientists and mathematicians. Different people interpret the requirements like "using intrinsically quantum effects" or "to be able to capture the full computational power of quantum mechanics" differently.

Still, why is it difficult to decide whether a given computer is quantum or not? The first idea is to run the quantum machine alongside a classical computer and see how they compete—in time and accuracy of results—for solving well-chosen hard problems. And, indeed, this has been done many times for D-Wave machines. There are many problems with this approach, including the lack of (agreed) uniform criteria of comparison and the fact that the choice of problems can favor in subtle ways one or another machine.

Are there more "objective" obstacles? Yes, there are. A natural idea—to observe how the machine works—fails because of the famous Schrödinger's cat paradox: one cannot look at a quantum system without stopping its quantum behavior. Further, if we observe, i.e. measure, an arbitrary qubit—which is in a superposition of $|0\rangle$ and $|1\rangle$—we get either 0 or 1. The information on intermediate states is lost in the process of measurement, and even worse, almost all observables are *value indefinite*, that is, they do not have a value before measurement [3].

Another obstacle is related to the recurrent idea that *a truly quantum computation uses intrinsically quantum effects*—that cannot naturally be modeled by classical physics—*to the aim of getting an asymptotic speed-up over the best classical algorithms for some problem.*[7] We have seen that the quantum algorithm for Deutsch's problem uses essentially superpositions of qubits and it was claimed to be better than any classical solution. However, this is not the case as we showed in Section 2.3. But maybe this toy example is not significant?

The quantum Fourier transform (QFT)—the operation transforming a quantum state vector into its Fourier representation—is a more interesting example.[8] Given an N qubit state $|\psi\rangle \in \mathbb{C}^N$

[7] Even with unbounded entanglement this is no guarantee that an algorithm can't be matched classically, see [2].

[8] The QFT is often an intermediate step in quantum algorithms (e.g., in Shor's algorithm), so its de-quantization can have implications for other quantum algorithms [32].

(where $N = 2^n$) the QFT performs the transformation F_N:

$$F_N \sum_{a=0}^{N-1} f(a)|a\rangle = \sum_{c=0}^{N-1} \hat{f}(c)|c\rangle, \text{ where } \hat{f}(c) = \frac{1}{\sqrt{N}} \sum_{a=0}^{N-1} e^{2\pi iac/N} f(a).$$

The QFT can be implemented with n^2 2-qubit unitary gates. The quantum algorithm is defined for the basis states, i.e. the action $F_N|a\rangle$ is implemented for a basis vector $|a\rangle$, where no entanglement is present in the initial or final state.

The QFT acts on a quantum state, whereas the classical Fourier transform acts on a vector, so, like in the case of Grover's algorithm, not every task that uses the classical Fourier transform can take advantage of the quantum exponential speed-up. In fact this advantage has been challenged: shortly after the discovery of the Shor's algorithm a semiclassical algorithm for the QFT was proposed in [30]; see also [5, 11]. By characterizing carefully when a separable, i.e. not entangled, state remains separable under the QFT, the following de-quantization result was proved in [1].

Theorem 1 *(1) Let* $|\psi_N\rangle = \begin{pmatrix}\alpha_1\\ \beta_1\end{pmatrix} \otimes \begin{pmatrix}\alpha_2\\ \beta_2\end{pmatrix} \otimes \cdots \otimes \begin{pmatrix}\alpha_n\\ \beta_n\end{pmatrix}$ *be a separable n-bit quantum state. Then the state* $F_N|\psi_N\rangle = |\hat{\psi}_N\rangle$ *is separable if and only if there exists an integer* k, $0 \le k \le n$, *and a binary string* $a_1a_2\ldots a_k \in \{0,1\}^k$ *such that the following conditions are satisfied:*

i) Each of the first $j \le k$ *qubits satisfies the following phase relationship:*

$$\alpha_j = e^{\pi i \sum_{l=1}^{j} a_l/2^{j-l}} \beta_j.$$

ii) Qubits $k+2$ *to* n *are not in a superposition, i.e.* $\alpha_j\beta_j = 0$ *for* $j > k+1$. *(The qubit* $k+1$ *can be in an arbitrary superposition state.)*

(2) In such a case, the state vector $|\hat{\psi}_N\rangle$ *can be computed classically (in its product representation) in time* $O(n)$ *using complex classical bits.*

Separable quantum states are important for de-quantization because one can store the state of each qubit individually (represented by $2n$ complex numbers for n qubits); this is not possible for entangled states which in general need 2^n complex numbers. More importantly, the de-quantized algorithm is *provably faster* than the QFT.

De-quantization points to the fact that genuinely fast quantum algorithms may not automatically have asymptotic speed-ups over classical algorithms solving the same problems [14, 53].

4 Adiabatic computing

We focus on the adiabatic model of computing which uses the propensity of physical systems—classical or quantum—to minimize their free energy. *Quantum annealing* is free energy minimization in a quantum system.

An *adiabatic quantum computation* (AQC) is an algorithm[9] that computes an exact or approximate solution of an optimization problem encoded in the ground state—its lowest-energy state—of

[9]Its precursors are the Monte Carlo methods [58], the Metropolis-Hastings algorithm [44] and the simulated annealing [36].

a Hamiltonian (the operator corresponding to the total energy of the system). The algorithm starts at an initial state H_I that is easily obtained, then evolves adiabatically, i.e. by slowly changing to the Hamiltonian H_P. An example of evolution is $H = (1-t)H_I + tH_P$ as the time t increases monotonically from 0 to 1. During the entire computation, the system must stay in a valid ground state. If the system can reach its ground state we get an exact solution; if it can only reach a local minimum, then we get an approximate solution. The slower the evolution process the better the approximate (possibly exact) solution is obtained.

AQC is based on the Born-Fock adiabatic theorem [9] which accounts for the adiabatic evolution of quantum states when the change in the time-dependent Hamiltonian is sufficiently slow [25]: *A physical system remains in its instantaneous eigenstate if a given perturbation is acting on it slowly enough and if there is a gap between the eigenvalue and the rest of the Hamiltonian's spectrum.*

The quantum adiabatic computation model and the gate quantum computation model (see Section 2) are polynomially time equivalent [4].

5 Ising and QUBO specifications

Discrete optimization problems for the adiabatic quantum (annealing) computing model are specified using the Ising or QUBO models [41]; they are the standard representation for problems for the D-Wave. These equivalent models are usually chosen by personal preferences: the Ising model tends to be favored by physicists while the QUBO model is preferred by computer scientists.

The *Ising* model works with variables $x_i \in \{-1, +1\}$ that represent magnetic dipole moments of atomic spins. These variables interact with neighbors in a graph $G = (V, E)$, where each edge $(i, j) \in E$ has a non-zero energy interaction $J_{(i,j)}$ and each vertex $i \in V$ has an external energy h_i. The Ising (minimization) problem of $n = |V|$ variables $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ has the form:

$$x^* = \min_{\mathbf{x}} \sum_{(i,j)\in E} x_i J_{(i,j)} x_j + \sum_{i \in V} h_i x_i, \text{ where } x_i \in \{-1, 1\}.$$

The *Quadratic Unconstrained Binary Optimization* (QUBO) is an NP-hard mathematical problem consisting in the minimization of a quadratic objective function $z = \mathbf{x}^T Q \mathbf{x}$, where $\mathbf{x}$ is a n-vector of binary variables and Q is a symmetric $n \times n$ matrix:[10]

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

There is a simple transformation between the spin variables of the Ising model and the binary variables of the QUBO model: $x' = 2x - 1$ (QUBO to Ising) and $x' = (x + 1)/2$ (Ising to QUBO).

6 D-Wave

The D-Wave computers are produced by the Canadian company D-Wave Systems[11] which describes them as using quantum annealing to improve convergence of the system's energy towards the ground state energy of an Ising/QUBO problem [54]. The computer architecture consists of qubits arranged

[10] The study of integer and Boolean optimization was introduced in [33] (see also [10]).

[11] D-Wave One (2011) operates on a 128-qubit chipset. D-Wave Two (2013) works with 512 qubits [19].

with a host configuration as a subgraph of a *Chimera graph.*[12] A Chimera graph consists of an $M \times N$ two-dimensional lattice of blocks, with each block consisting of $2L$ vertices (a complete bipartite graph $K_{L,L}$), in total $2MNL$ variables. The D-Wave One has $M = N = L = 4$ for a maximum of 128 qubits. D-Wave qubits are loops of superconducting wire, the coupling between qubits is magnetic wiring and the machine itself is supercooled.

To index a qubit we use four numbers (i, j, u, k), where (i, j) indexes the (row, column) of the block, $u \in \{0, 1\}$ is the left/right bipartite half of $K_{L,L}$ and $0 < k < L$ is the offset within the bipartite half. Qubits indexed by (i, j, u, k) and (i', j', u', k') are neighbors if and only if

1. $i = i'$ and $j = j'$ and $[(u, u') = (0, 1)$ or $(u, u') = (1, 0)]$ or
2. $i = i' \pm 1$ and $j = j'$ and $u = u'$ and $u = 0$ and $k = k'$ or
3. $i = i'$ and $j = j' \pm 1$ and $u = u'$ and $u = 1$ and $k = k'$.

Figure 1 shows for $L = N = 4$ (and $M > 2$) the structure of an initial part of a Chimera graph where the two half partitions of the bipartite graphs $K_{L,L}$ (blocks) are drawn horizontally and vertically, respectively. The linear index (qubit id of the vertices) from the four tuple (i, j, u, k) is the value $2NLi + 2Lj + Lu + k$.

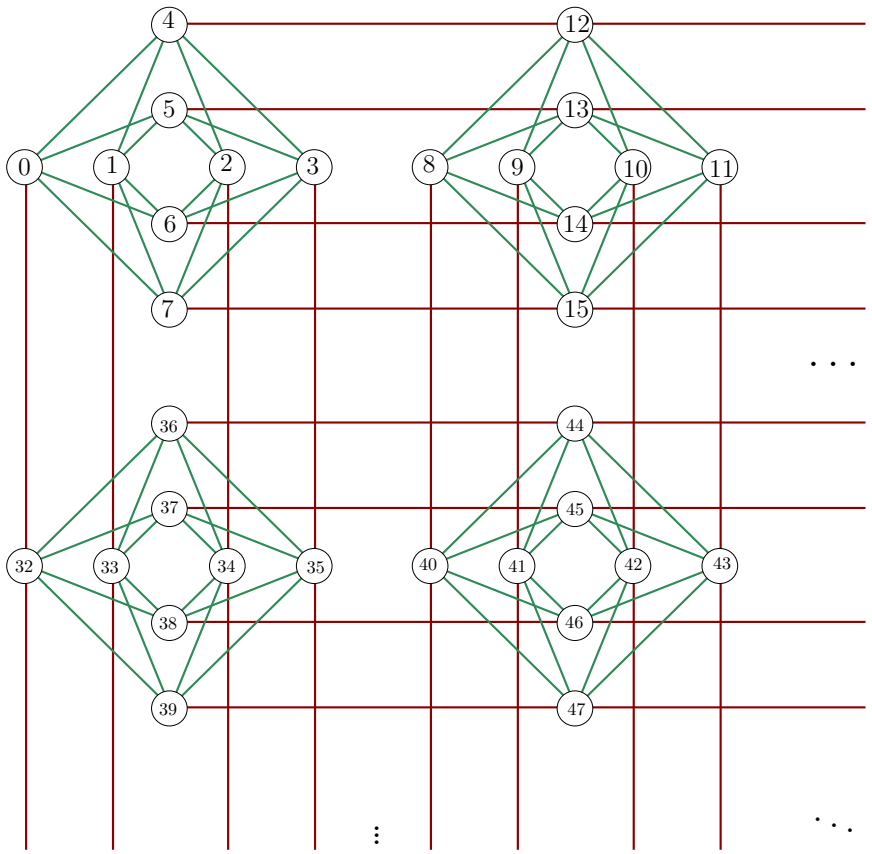


Figure 1: D-Wave architecture: a subgraph of a Chimera graph with $L = N = 4$.

[12] A monster from Greek mythology that breathes fire, has a lion's head, a goat's body and a snake's tail.

The questions about whether D-Wave machines perform quantum computation and allow a quantum speed-up have been heatedly debated [8, 53, 55]. Note that part of the difficulties in answering these questions are theoretical, as previously discussed in Section 3.

7 The independent set problem

We now show the QUBO formulation of an NP-hard problem, namely the *Maximum Independent Set* problem for graphs [28]: *Given a graph $G = (V, E)$, what is the largest set $V' \subseteq V$ such that for all $u, v \in V'$ we have $(u, v) \notin E$?*

For a graph G we consider all possible subsets of a n-vector $\mathbf{x}$ of length $n = |V|$. We take $x_i = 1$ if and only if $i \in V$ is selected as an element of a potential independent set. Finally we constrain the maximum number (and legality) of variables $x_i = 1$ by giving a large *penalty* to any edge $(u, v) \in E$ in which both $x_u = x_v = 1$:

$$Q_{(i,j)} = \begin{cases} -1, & \text{if } i = j, \\ \geq 2, & \text{if } i < j \text{ and } (i,j) \in E, \\ 0, & \text{if } i < j \text{ and } (i,j) \notin E. \end{cases}$$

The calculation of the maximum independent set of an example graph (C6 with a chord as shown in Figure 2) that embeds in the D-Wave Chimera architecture is shown in Figure 3. Here we used the penalty value of 6 for each edge. Note that qubit 3 is not used because of the requirement to embed this graph into the Chimera graph, which is a restriction imposed by the use of a D-Wave computer.

Our example's vertex labeling is just a subgraph of Figure 1. This particular program, when run on a local D-Wave simulator [20], returns two maximum solutions ($\{4, 5, 6\}$ and $\{0, 1, 2\}$) of maximum size 3 (i.e. global optimum) and one maximal solution $\{2, 5\}$ of size 2 (i.e. local optimum), corresponding to final annealing energies -3 and -2, respectively.

More information about this problem, including a weighted version and finding embeddings in Chimera graphs, is in [16]. Direct formulations of NP-hard problems for the adiabatic quantum computing are presented in [38].

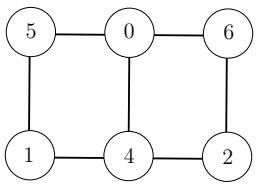


Figure 2: Graph input used in our Maximum Independent Set program.

```
import dwave_sapi

# use a local solver
solver=dwave_sapi.local_connection.get_solver("c4-sw_sample")

print "Number of qubits", solver.properties["num_qubits"]
print "Working qubits: ", solver.properties["qubits"]
print "Working couplers: ", solver.properties["couplers"]

Q = {}
Q[(0, 4)] = Q[(0, 5)] = Q[(0, 6)] = Q[(1, 4)] = 6
Q[(1, 5)] = Q[(2, 4)] = Q[(2, 6)] = 6
Q[(0, 0)] = Q[(1, 1)] = Q[(2, 2)] = -1
Q[(3, 3)] = 10 # unused qubit/vertex
Q[(4, 4)] = Q[(5, 5)] = Q[(6, 6)] = -1

answer = solver.solve_qubo(Q, num_reads=10)
print "QUBO answer (using \"optimized\" method): ", answer
```

```
Number of qubits 128
Working qubits:  (0, 1, 2, 3, 4, 5, 6, 7, ...
Working couplers:  ((0, 4), (0, 5), (0, 6), (0, 7),
(0, 32), (1, 4), (1, 5), (1, 6), (1, 7), (1, 33),
(2, 4), (2, 5), (2, 6), (2, 7), (2, 34), ...

QUBO answer (using "optimized" method):
{'energies': [-3.0, -3.0, -2.0],
'num_occurrences': [2, 7, 1],
'solutions': [ [0, 0, 0, 0, 1, 1, 1, 3, ..., 3],
               [1, 1, 1, 0, 0, 0, 0, 3, ..., 3],
               [0, 0, 1, 0, 0, 1, 0, 3, ..., 3] ]}
```

Figure 3: QUBO Example: Maximum Independent Set.

8 The network broadcast problem

Broadcasting concerns the dissemination of a message originating at one node of a network to all other nodes [26, 34]. This task is accomplished by placing a series of calls over the communication lines of the network between neighboring nodes. Each call requires a unit of time, a call can involve only two nodes and a node can participate in only one call per time step.

A *broadcast tree/protocol* for a vertex v (called the originator) of an undirected graph $G = (V, E)$ is a sequence $V_0 = \{v\}, E_1, V_1, E_2, \ldots, E_t, V_t = V$ (*of broadcast height* t) such that each $V_i \subseteq V$, each $E_i \subseteq E$, and for every $1 \leq i \leq t$: (1) each edge in E_i has exactly one endpoint in V_{i-1}, (2) no two edges in E_i share a common endpoint, and (3) $V_i = V_{i-1} \cup \{w \mid \{u, w\} \in E_i\}$.

The *Broadcast Problem* is the following: *Given a connected graph* $G = (V, E)$, *originator* $v \in V$ *and integer* t, *find whether there exists a broadcast tree* T_v *rooted at* v *with the height of* T_v *at most* t*?* This is a well-known NP-complete problem (see [ND49] of [28]), even for graphs of maximum vertex degree 3 (see [22]). The optimization version of this problem is approximable within $O(\log^2 |V|/\log\log|V|)$, but is not expected to have a polynomial-time approximation scheme [51].

In the example shown in Figure 4 (hypercube Q_3) an optimal broadcast tree is illustrated for the originator vertex 0.

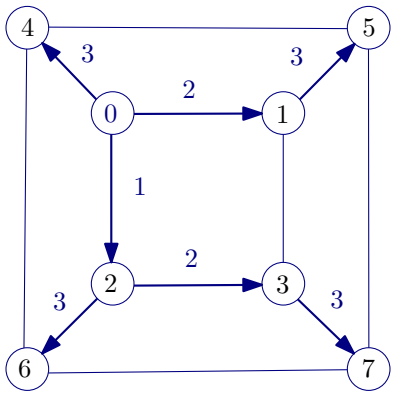


Figure 4: The graph Q_3 with broadcast time 3.

The development of the quantum solution will be presented in a sequence of four phases.

8.1 Integer programming formulation

In the first phase we present a simple formulation (i.e. polynomial-time reduction) of the Broadcast Problem with the originator fixed[13] at $v = 0$ as an *Integer Programming (IP) Optimization Problem* (see [60]). The input is a connected graph $G = (V = \{0, 1, \ldots, n-1\}, E)$ representing a network with $n = |V|$ vertices and $m = |E|$ edges. For the graph G, we use the following $n + 2m + 1$ variables:

- t is the required time to complete a broadcast,

[13]Solving the problem for other originators can be easily done by relabelling the vertices of the graph or doing obvious modifications in the formulation below.

- v_i is the time in $\{0, \ldots, t\}$ in which the vertex $i \in V$ receives the message, $0 \leq i < n$,
- $b_{i,j}$ is a binary variable which is 1 if and only if the vertex i broadcasts to the vertex j (for each $\{i, j\} \in E$).

The objective function for our optimization problem is $\min t$ (or equivalently, $\max(n - t)$). First, the time t must be at most $n - 1$:

$$0 \leq t \leq n - 1. \tag{3}$$

Every vertex receives the message at a time step at most t:

$$0 \leq v_i \leq t, \text{ for all } i \in V. \tag{4}$$

The originator vertex has no parent and every other vertex must have exactly one parent in the broadcast tree:

$$\sum_{j \neq 0} b_{j,0} = 0, \tag{5}$$

$$\sum_{j \neq i} b_{j,i} = 1, \quad \text{for all } i \in V \setminus \{0\}. \tag{6}$$

There are no broadcast cycles, that is for a child vertex, the informed time of the parent must be strictly less than its message received time:

$$b_{i,j}(1 + v_i - v_j) \leq 0, \quad \text{for all } \{i, j\} \in E. \tag{7}$$

Finally, every two child vertices (j, k) informed by the same parent i must occur at different times:

$$b_{i,j} + b_{i,k} - (v_j - v_k)^2 \leq 1, \quad \text{for all } \{i, j\} \in E, \{i, k\} \in E \text{ with } j \neq k. \tag{8}$$

8.2 Binary integer programming formulation

Next we convert all non-binary variables (in IP formulation) into binary variables. The following simple procedure converts an integer constrained variable $0 \leq x \leq D$ into a set of $O(\log D)$ binary variables $x_0, x_1, \ldots, x_c$ representing its binary representation:

$$x = x_0 + 2x_1 + 4x_2 + \cdots + 2^c x_c = \sum_{i=0}^{c} 2^i x_i,$$

where $x_i \in \{0, 1\}$ and $2^c \leq D < 2^{c+1}$. Each constraint of the form $x \leq D$ is replaced by the following equivalent constraint:

$$\sum_{i=0}^{c} 2^i x_i \leq D. \tag{9}$$

8.3 Linear binary integer programming formulation

Using standard techniques (e.g., see [35]) we convert the above quadratic binary IP formulation into a linear formulation. Each occurrence of a product of two binary variables xy is replaced by a new variable z_{xy} and the following two linear constraints:

$$0 \leq x + y - z_{xy} \leq 1, \tag{10}$$

$$-1 \leq 2z_{xy} - (x + y) \leq 0, \tag{11}$$

enforce $z_{xy} = xy$.

We can reduce the number of "product" binary variables by observing that for equation (7) we do not need to consider $j = 0$ and for equation (8) we can only consider vertices $j > 0$ and $k > 0$ with a common neighbor.

Note that the above reduction was automated and the Sage Mixed Integer Program Solver [57] was used to verify correctness of many small graphs [13].

8.4 QUBO formulation

The first step to converting the current binary linear IP formulation to QUBO is to use a "standard form," where all inequalities are replaced with equalities by introducing slack variables [60].

The next step is to build an equivalent QUBO of the IP formulation and add rules to force all linear equation constraints to be satisfied when assigning 0/1 to the binary variables. Consider a linear equality constraint C_k of the form $\sum_{i=1}^{n} c_{(k,i)}x_i = d_k$ for $x_i \in \{0, 1\}$ with fixed integer constants $c_{(k,i)}$ and d_k. This equation is satisfied if and only if $\sum_{i=1}^{n} c_{(k,i)}x_i - d_k = 0$, or equivalently, if $\langle \mathbf{c}_k, \mathbf{x} \rangle - d_k = 0$, where $\mathbf{c}_k = (c_{(k,i)}, c_{(k,2)}, \ldots, c_{(k,n)})$ and $\langle \mathbf{c}_k, \mathbf{x} \rangle$ is the product of the vectors $\mathbf{c}_k$ and $\mathbf{x}$. If $\langle \mathbf{c}_k, \mathbf{x} \rangle - d_k$ is not zero we need to have a penalty greater than the maximum feasible value of t, which is n. Thus, we can construct the following QUBO that is equivalent to the IP formulation of the Broadcast Problem:

$$x^* = \min_{\mathbf{x}} \left(t + n \cdot \sum_{k} \left(\langle \mathbf{c}_k, \mathbf{x} \rangle - d_k \right)^2 \right), \text{ where } x_i \in \{0, 1\}.$$

Note first that the variable t is obtained from the variables used in equation (3) of Section 8.1 and is added to the other QUBO entries in Q from the set of linear constraints C_k. The QUBO constants for the binary variables representing t will be powers of 2, as given by equation (9). Second, any term d_k^2 in the square terms of $(\langle \mathbf{c}_k, \mathbf{x} \rangle - d_k)^2$ which does not involve a variable x_i can be ignored since those additive terms are independent of any assignment of variables (i.e. we have a fixed additive QUBO *offset* to the objective solution). Third, since variables x_i are binary, we have $x_i^2 = x_i$ and the constants for those terms are included in the main diagonal entries of Q. Finally, the conversion from an arbitrary Broadcast Problem instance to QUBO was automated [13].

8.5 Q_3 example

In this section we illustrate the quantum solution phases for Q_3. In the first phase (IP formulation) we get 33 integer variables (the variable t, eight variables of the type v_i, and 24 variables of the type $b_{i,j}$) and 65 quadratic constraints. The conversion to the binary formulation results in 51 binary variables as we need three binary variables for each of the previous integer variables t, $v_0, \ldots, v_7$;

e.g., $51 = 33 + 2 \cdot 9$. The next conversion (Section 8.3) produces 447 binary variables and 851 linear constraints. Finally, the conversion to QUBO generated 999 slack variables, so in total 1446 binary variables: they represent the number of logical qubits for our QUBO formulation. See [13] for all details.

To be able to solve this QUBO problem on D-Wave we need one more step (see [20]) which will be illustrated in the next section with a feasible example for the D-Wave Two.

8.6 K_2 example

We present both the final IP formulation (see Table 1) and QUBO matrix Q (see Table 2) for the Broadcast Problem for the (toy) graph K_2 of one edge. The total number of binary variables is 22 (13 of them are slack variables) and the QUBO offset is 12. When run on the D-Wave simulator [20] (without embedding onto the hardware, which has limited qubit connections) we get the following expected result:

```
answer={
'energies': [-11.0],
'solutions': [[1,0,0,1,1,0,0,1,0,0,1,1,0,0,0,0,1,0,1,0,0,0]]
}
```

When we add the offset 12 to the minimum energy state we get our expected broadcast time of 1. We can also see that $t = x_1 = 1$, $b_{0,1} = x_7 = 1$ and $b_{1,0} = x_8 = 0$, which indicates a valid broadcast tree from the obtained solution x^*.

To actually run this QUBO instance on the D-Wave machine we need to find a minor-containment embedding on the actual physical qubit hardware (the Chimera graph). One valid heuristic is to map each logical qubit to a path of physical qubits. One such example is given below where our 22 logical qubits, labeled 0 to 21, become 50 active hardware qubits.

```
'embedding=': [ 0=[224, 226, 228], 1=[230], 2=[276, 283, 284, 288, 292],
3=[290], 4=[227, 291, 348, 355, 356, 357], 5=[229], 6=[336, 338, 341, 347, 349],
7=[293, 297, 301, 361], 8=[294, 296, 302], 9=[289], 10=[300], 11=[298, 362, 365],
12=[359, 367], 13=[364], 14=[345, 351], 15=[344], 16=[346],
17=[275, 277, 281, 285, 339], 18=[272], 19=[274], 20=[303], 21=[343] ]
```

This best energy solution of -11 is also obtained when we run it on an actual D-Wave Two machine. This optimal answer occurs about 33% of the time for our trials of about 1000 runs. In the other cases, the machine did not converge to the optimal ground-state energy.

8.7 Experimental results

We have produced QUBO representations of the Broadcast Problem for several small common graphs using the above IP formulation procedure. Tables 3 and 4 summarize them for some small common graph families and known special graphs (all graphs can be obtained from Sage [57, 13]). Recall that for non-symmetric graphs we initiate the broadcast at vertex labeled 0, using the vertex labels given by Sage's adjacency lists. In these tables, columns 2 and 3 (Integer Variables and Quadratic Constraints) indicate the size of the IP formulation presented in Section 8.1. Next, columns 4 and 5 (Binary Variables and Binary Constraints) indicate the size of the IP formulation given in Section 8.3. Finally, columns 6–8 (Slack Variables, Logical Qubits and Chimera/Physical

Table 1: Final Binary Integer Program for broadcasting in K_2.

Integer Program Constraints	Comments
$x_0 + x_1 = 1$	x_0 is objective variable t and x_1 is a slack variable
$-x_0 + x_2 + x_3 = 0$	x_2 is vertex variable v_0; equation (4)
$-x_0 + x_4 + x_5 = 0$	x_4 is vertex variable v_1; equation (4)
$x_6 = 0$	x_6 is broadcast variable $b_{1,0}$; equation (5)
$x_7 = 1$	x_7 is broadcast variable $b_{0,1}$; equation (6)
$x_2 + x_7 - x_8 + x_9 = 1$	x_8 is for product $b_{0,1}v_0$ with equation (10)
$-x_2 - x_7 + 2x_8 + x_{10} = 0$	equation (7) with equation (11)
$x_4 + x_7 - x_{11} + x_{12} = 1$	x_{11} is for product $b_{0,1}v_1$
$-x_4 - x_7 + 2x_{11} + x_{13} = 0$	equation (7)
$x_4 + x_6 - x_{14} + x_{15} = 1$	x_{14} is for product $b_{1,0}v_1$
$-x_4 - x_6 + 2x_{14} + x_{16} = 0$	equation (7)
$x_2 + x_6 - x_{17} + x_{18} = 1$	x_{17} is for product $b_{1,0}v_0$
$-x_2 - x_6 + 2x_{17} + x_{19} = 0$	equation (7)
$x_7 + x_8 - x_{11} + x_{20} = 0$	equation (8)
$x_6 + x_{14} - x_{17} + x_{21} = 0$	equation (8)

Table 2: Final QUBO matrix Q for broadcasting in K_2.

3	4	-4	-4	-4	-4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	**-2**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-4	0	**2**	4	0	0	8	8	-12	4	-4	0	0	0	0	0	0	-12	4	-4	0	0
-4	0	4	**2**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-4	0	0	0	**2**	4	8	8	0	0	0	-12	4	-4	-12	4	-4	0	0	0	0	0
-4	0	0	0	4	**2**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	8	0	8	0	**4**	0	0	0	0	0	0	0	-8	4	-4	-16	4	-4	0	4
0	0	8	0	8	0	0	**0**	-8	4	-4	-16	4	-4	0	0	0	0	0	0	4	0
0	0	-12	0	0	0	0	-8	**16**	-4	8	-4	0	0	0	0	0	0	0	0	4	0
0	0	4	0	0	0	0	4	-4	**-2**	0	0	0	0	0	0	0	0	0	0	0	0
0	0	-4	0	0	0	0	-4	8	0	**2**	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	-12	0	0	-16	-4	0	0	**16**	-4	8	0	0	0	0	0	0	-4	0
0	0	0	0	4	0	0	4	0	0	0	-4	**-2**	0	0	0	0	0	0	0	0	0
0	0	0	0	-4	0	0	-4	0	0	0	8	0	**2**	0	0	0	0	0	0	0	0
0	0	0	0	-12	0	-8	0	0	0	0	0	0	0	**16**	-4	8	-4	0	0	0	4
0	0	0	0	4	0	4	0	0	0	0	0	0	0	-4	**-2**	0	0	0	0	0	0
0	0	0	0	-4	0	-4	0	0	0	0	0	0	0	8	0	**2**	0	0	0	0	0
0	0	-12	0	0	0	-16	0	0	0	0	0	0	0	-4	0	0	**16**	-4	8	0	-4
0	0	4	0	0	0	4	0	0	0	0	0	0	0	0	0	0	-4	**-2**	0	0	0
0	0	-4	0	0	0	-4	0	0	0	0	0	0	0	0	0	0	8	0	**2**	0	0
0	0	0	0	0	0	0	4	4	0	0	-4	0	0	0	0	0	0	0	0	**2**	0
0	0	0	0	0	0	4	0	0	0	0	0	0	0	4	0	0	-4	0	0	0	**2**

Qubits) indicate the size of the final QUBO representation described in Section 8.4. Using this approach, the number of logical qubits equals the number of binary variables plus the number of slack variables.

9 Quassical computing

A handful of quantum algorithms are provably faster than the best competing classical algorithms. However, this does not imply that quantum computing is generally superior to classical computing. Quite the contrary, the computational complexity of many classical algorithms provably cannot be improved through the use of quantum computing; the Fourier Transform discussed in Section 3 is such an example.

Classical computing and quantum computing have obvious complementary strengths, so instead of opposing them it might be better to combine them into a new type of computing. Such a proposal, called "hybrid quantum-classical computing,"[14] was discussed in [37].

Why quassical computing? Here are two examples. Grover's black box oracle quantum algorithm searches an unsorted database with N entries[15] in $O(\sqrt{N})$ time with $O(\log N)$ storage space [31], while classical algorithms cannot search an unsorted database in less than linear time. As Grover's algorithm is optimal within the quantum model for black box oracles [7], we may seem to have an example where quantum computing *is provably superior* to classical computing. A more careful analysis shows that there is a significant difference between the complexity-theoretical and the applicability of the Grover's algorithm. To search with Grover's algorithm we need a preprocessing of the database, i.e. we need to compute a quantum description of the original (classical) database, which requires $O(N)$ storage space.[16] A fairer comparison between Grover's quantum algorithm and the best classical counterpart algorithms presented in [37] concludes that classical computing offers a better solution than the quantum one. More interestingly, the paper [37] shows how an enhanced Grover's quantum search method can be incorporated into classical procedural algorithms to achieve performance that is in some cases *provably more efficient* than any classical computing solution alone. The main finding is a solution which mitigates (in this instance) the negative effect of the no-cloning theorem—the impossibility of making a perfect copy (or clone) of an arbitrary (unknown) quantum state [62].

Another example is the development of a quassical computing algorithm for the Sudoku problem; an experimental analysis of its feasibility is presented in [52].

10 Final comments and open problems

Quantum computing will continue to be intensively researched and experimentally tested in both academia and the private sector. How the field is going to evolve even in the near future is anybody's guess.

[14] The short name "quassical computing" was coined by N. Allen who first noticed the natural emergence of the quassicality as a result of integrating the D-Wave machines into the CAE network at Lockheed Martin. He first used the term in a January 2014 letter discussing the phenomenon with M. Bremner [6].

[15] A rather contrived problem.

[16] The cost of the "quantum embedding" (see also Sections 2.3 and 3) necessary for the use of a quantum algorithm is not easy to evaluate, so frequently omitted.

Table 3: Number of qubits required for some small graphs families.

Graph	Order	Size	Integer Variables	Quadratic Constraints	Binary Variables	Binary Constraints	Slack Variables	Logical Qubits	Chimera Qubits
C3	3	3	10	16	50	86	96	146	394
C4	4	4	13	21	74	131	146	220	662
C5	5	5	16	26	178	324	366	544	3258
C6	6	6	19	31	240	443	495	735	4164
C7	7	7	22	36	311	580	642	953	
C8	8	8	25	41	391	735	807	1198	
C9	9	9	28	46	778	1484	1608	2386	
C10	10	10	31	51	944	1809	1948	2892	
C11	11	11	34	56	1126	2166	2320	3446	
C12	12	12	37	61	1324	2555	2724	4048	
Grid2x3	6	7	21	37	254	472	543	797	4306
Grid3x3	9	12	34	65	832	1597	1816	2648	
Grid3x4	12	17	47	93	1414	2745	3084	4498	
Grid4x4	16	24	65	133	2420	4737	5252	7672	
Grid4x5	20	31	83	173	5537	10909	11815	17352	
K2	2	1	5	7	9	15	13	22	47
K3	3	3	10	16	50	86	96	146	394
K4	4	6	17	33	94	171	202	296	1378
K5	5	10	26	61	248	469	606	854	7973
K6	6	15	37	103	366	713	981	1347	
K7	7	21	50	162	507	1014	1482	1989	
K8	8	28	65	241	671	1375	2127	2798	
K9	9	36	82	343	1264	2591	4200	5464	
K10	10	45	101	471	1574	3279	5588	7162	
K2x1=P2	3	2	8	12	36	59	64	100	170
K1x2=S2	3	2	8	12	40	68	76	116	238
K2x2=C4	4	4	13	21	74	131	146	220	662
K2x3	5	6	18	32	192	353	414	606	4823
K3x3	6	9	25	49	282	529	633	915	
K3x4	7	12	32	69	381	727	894	1275	
K4x4	8	16	41	97	503	973	1227	1730	
K4x5	9	20	50	129	976	1906	2432	3408	
K5x5	10	25	61	171	1214	2391	3124	4338	
K5x6	11	30	72	118	1468	2914	3896	5364	
K6x6	12	36	85	277	1756	3511	4804	6560	
S2=K1x2	3	2	8	12	40	68	76	116	238
S3	4	3	11	18	64	114	130	194	505
S4	5	4	14	25	164	301	354	518	3711
S5	6	5	17	33	226	423	501	727	5120
S6	7	6	20	42	297	564	672	969	
S7	8	7	23	52	377	724	867	1244	
S8	9	8	26	63	760	1471	1736	2496	
S9	10	9	29	75	926	1803	2132	3058	
S10	11	10	32	88	1108	2168	2568	3676	

Table 4: Number of qubits required for hypercubes and some other small known graphs.

Graph	Order	Size	Integer Variables	Quadratic Constraints	Binary Variables	Binary Constraints	Slack Variables	Logical Qubits	Chimera Qubits
Q1=K2	2	1	5	7	9	15	13	22	47
Q2=C4	4	4	13	21	74	131	146	220	662
Q3	8	12	33	65	447	851	999	1446	
Q4	16	32	81	193	2564	5045	5860	8424	
BidiakisCube	12	18	49	97	1432	2779	3124	4556	
Bull	5	5	16	28	178	324	366	544	3523
Butterfly	5	6	18	33	192	353	414	606	5927
Chvatal	12	24	61	145	1540	3013	3604	5144	
Clebsch	16	40	97	273	2708	5373	6628	9336	
Diamond	4	5	15	27	84	151	174	258	742
Dinneen	9	21	52	142	994	1950	2552	3546	
Dodecahedral	20	30	81	161	5515	10855	11645	17160	
Durer	12	18	49	97	1432	2779	3124	4556	
Errera	17	45	108	320	4480	8900	10890	15370	
Frucht	12	18	49	97	1432	2779	3124	4556	
GoldnerHarary	11	27	66	209	1414	2814	3792	5206	
Grotzsch	11	20	52	118	1288	2508	2968	4256	
Heawood	14	21	57	113	1894	3691	4100	5994	
Herschel	11	18	48	101	1252	2429	2800	4052	
Hexahedral	8	12	33	65	447	851	999	1446	
Hoffman	16	32	81	193	2564	5045	5860	8424	
House	5	6	18	32	192	353	414	606	4176
Icosahedral	12	30	73	205	1648	3257	4164	5812	
Krackhardt	10	18	47	114	1088	2116	2548	3636	
Octahedral	6	12	31	73	324	619	795	1119	
Pappus	18	27	73	145	4514	8869	9575	14089	
Petersen	10	15	41	81	1034	1995	2276	3310	
Poussin	15	39	94	276	2446	4863	6152	8598	
Robertson	19	38	96	229	5211	10287	11570	16781	
Shrikhande	16	48	113	369	2852	5715	7508	10360	
Sousselier	16	27	71	154	2474	4849	5452	7926	
Tietze	12	18	49	97	1432	2779	3124	4556	
Wagner	8	12	33	65	447	851	999	1446	

There are many open problems. Developing efficient graph minor embeddings into Chimera graphs and reducing the number of variables in the linear binary programming formulation of various algorithms (both to reduce number of physical qubits) and developing direct conversions of hard combinatorial problems to QUBO format (to reduce number of logical qubits) are just a few.

Error correcting methods are announced for future D-Wave machines [49, 50], a significant change of the current approach from dealing with noisy qubits; Google is also developing quantum annealing hardware with an emphasis on reliable qubits [46]. Can adiabatic quantum computing continue to be scaled up while maintaining a reasonable probability of correctness?

Finally, a more challenging open problem is to develop a methodology combining quantum and classical components in efficient quassical computing algorithms.

Acknowledgements

We thank A. A. Abbott, N. Allen, L. Hemaspaandra, M. Triplett, K. L. Pudenz, M. Stay and G. J. Tee for discussions and comments leading to a better presentation.

Dedication

The first two authors' contribution is dedicated to one of the co-founders of pseudo-boolean optimization [33], Prof. S. Rudeanu, on the occasion of his 80th Birthday.

References

[1] A. A. Abbott. De-quantisation of the quantum Fourier transform. *Applied Mathematics and Computation*, 291(1):3–13, 2012.

[2] A. A. Abbott and C. S. Calude. Understanding the quantum computational speed-up via de-quantisation. *EPTCS*, 26:1–12, 2010.

[3] A. A. Abbott, C. S. Calude and K. Svozil. Value indefiniteness is almost everywhere. *Physical Review A*, 89(3):032109–032116, 2014.

[4] D. Aharonov, W. v. Dam, J. Kempe, Z. Landau, S. Lloyd and O. Regev. Adiabatic quantum computation is equivalent to standard quantum computation. arXiv:quant-ph/0405098, March 2005.

[5] D. Aharonov, Z. Landau and J. Makowsky. The quantum FFT can be classically simulated. arXiv:quant-ph/0611156v2, 2007.

[6] N. Allen. Personal communication to C. S. Calude, November 7, 2014.

[7] C. H. Bennett, E. Bernstein, G. Brassard and U. Vazirani. Strengths and weaknesses of quantum computing. *SIAM Journal on Computing*, 26:1510–1523, 1997.

[8] S. Boixo, T. F. Rønnow, S. V. Isakov, Z. Wang, D. Wecker, D. A. Lidar, J. M. Martinis and M. Troyer. Quantum annealing with more than one hundred qubits. arXiv:1304.4595v2, July 2013.

[9] M. Born and V. Fock. Beweis des adiabatensatzes. *Zeitschrift für Physik*, 51(3-4):165–180, 1928.

[10] E. Boros and P. L. Hammer. Pseudo-boolean optimization. *Discrete Applied Mathematics*, 123(1–3):155–225, 2002.

[11] D. E. Browne. Efficient classical simulation of the quantum Fourier transform. *New Journal of Physics*, 9(5):146, May 2007.

[12] C. S. Calude. De-quantizing the solution of Deutsch's problem. *International Journal of Quantum Information*, 5(3):409–415, 2007.

[13] C. S. Calude and M. J. Dinneen. Solving the broadcast time problem using a D-Wave quantum computer. Report CDMTCS-473, Centre for Discrete Mathematics and Theoretical Computer Science, University of Auckland, Auckland, New Zealand, November 2014.

[14] C. S. Calude, M. J. Dinneen and K. Svozil. Reflections on quantum computing. *Complexity*, 6:35–37, 2000.

[15] C. S. Calude and G. Păun. *Computing with Cells and Atoms*. Taylor & Francis Group, London and New York, 2001.

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

[17] R. Cleve, A. Ekert, C. Macchiavello and M. Mosca. Rapid solution of problems by quantum computation. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 454(1969):339–354, 1998.

[18] E. Cohen and B. Tamir. D-Wave and predecessors: From simulated to quantum annealing. *International Journal of Quantum Information*, 12(03):1430002, 2014.

[19] D-Wave. D-Wave overview: A brief introduction to D-Wave and quantum computing, 2013. http://www.dwavesys.com/sites/default/files/D-Wave-brochure-102013F-CA.pdf.

[20] D-Wave. Programming with QUBOs. Technical report, D-Wave Systems, Inc., 2013. Python Release 1.5.1-beta4 (for Mac/Linux), 09-1002A-B.

[21] D. Deutsch. Quantum theory, the Church-Turing principle and the universal quantum computer. *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences (1934-1990)*, 400(1818):97–117, 1985.

[22] M. J. Dinneen. The complexity of broadcasting in bounded-degree networks. Technical Report Combinatorics report LACES-[05C-94-31], Los Alamos National Laboratory, 1994. http://arxiv.org/abs/math/9411222.

[23] D. P. Divincenzo. Topics in quantum computers. In G. S. L. Sohn, L. Kouwenhoven, editor, *Mesoscopic Electron Transport*, pages 657–677, Dordrecht, 1996. Kluwer Academic Publishers.

[24] D. P. Divincenzo. The physical implementation of quantum computation. *Fortsch. Phys.*, 48:771–783, 2000.

[25] E. Farhi, J. Goldstone, S. Gutmann and M. Sipser. Quantum computation by adiabatic evolution. arXiv:quant-ph/0001106, January 2000.

[26] A. Farley, S. Hedetniemi, S. Mitchell and A. Proskurowski. Minimum broadcast graphs. *Discrete Mathematics*, 25:189–193, 1979.

[27] R. P. Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, 21:467–488, 1982.

[28] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.

[29] N. Gershenfeld and I. L. Chuang. Quantum computing with molecules. *Scientific American*, 6:66–71, 1998.

[30] R. Griffiths and C. Niu. Semiclassical Fourier transform for quantum computation. *Physical Review Letters*, 76(17):3228–3231, January 1996.

[31] L. K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing*, pages 212–219. ACM Press, 1996.

[32] J. Gruska. *Quantum Computing*. McGraw-Hill, London, 1999.

[33] P. L. Hammer and S. Rudeanu. *Boolean Methods in Operations Research and Related Areas*. Springer-Verlag, Berlin, Heidelberg, New York, 1968.

[34] S. M. Hedetniemi, S. T. Hedetniemi and A. L. Liestman. A survey of gossiping and broadcasting in communication networks. *Networks*, 18:319–349, 1998.

[35] M. Khosravani. *Searching for Optimal Caterpillars in General and Bounded Treewidth Graphs*. PhD dissertation, University of Auckland, Auckland, New Zealand, 2011.

[36] S. Kirkpatrick, C. D. Gelatt and M. P. Vecchi. Optimization by simulated annealing. *SCIENCE*, 220(4598):671–680, 1983.

[37] M. Lanzagorta and J. K. Uhlmann. Hybrid quantum-classical computing with applications to computer graphics. In *ACM SIGGRAPH 2005 Courses*, SIGGRAPH '05, New York, NY, USA, 2005. ACM.

[38] A. Lucas. Ising formulations of many NP problems. *Frontiers in Physics*, 2(5), 2014.

[39] Y. I. Manin. Vychislimoe i nevychislimoe [Computable and Noncomputable] (in Russian). Sov. Radio. pp. 13–15. (Checked 30 October 2014). http://www.worldcat.org/title/vychislimoe-i-nevychislimoe/oclc/11674220, 1980.

[40] Y. I. Manin. Classical computing, quantum computing, and Shor's factoring algorithm. arXiv:quant-ph/9903008v1, March 1999.

[41] C. C. McGeoch and C. Wang. Experimental evaluation of an adiabatic quantum system for combinatorial optimization. In *Proceedings of the ACM International Conference on Computing Frontiers*, CF '13, pages 23:1–23:11, New York, NY, USA, 2013. ACM.

[42] D. N. Mermin. What's wrong with these elements of reality? *Physics Today*, 43(6):9–10, June 1990.

[43] D. N. Mermin. *Quantum Computer Science*. Cambridge University Press, Cambridge, 2007.

[44] N. C. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller and E. Teller. Equations of state calculations by fast computing machines. *Journal of Chemical Physics*, 21(6):1087–1092, 1953.

[45] L. Mirani and G. Lichfield. Why nobody can tell whether the worlds biggest quantum computer is a quantum computer, April 2014. http://qz.com/194738/why-nobody-can-tell-whether-the-worlds-biggest-quantum-computer-is-a-quantum-computer/.

[46] H. Neven. Hardware initiative at quantum artificial intelligence lab. URL: https://plus.google.com/+QuantumAILab/posts/UcWGvc9Y6dU [Accessed 28 October 2014], 2014.

[47] M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, Cambridge, 2000.

[48] C. A. Perez-Delgado and P. Kok. What is a quantum computer, and how do we build one? arXiv:0906.4344v2, 2010.

[49] K. L. Pudenz, T. Albash and D. A. Lidar. Error-corrected quantum annealing with hundreds of qubits. *Nat. Commun.*, 5, 02 2014.

[50] K. L. Pudenz, T. Albash and D. A. Lidar. Quantum annealing correction for random Ising problems. arXiv:1408.4382v1, August 2014.

[51] R. Ravi. Rapid rumor ramification: Approximating the minimum broadcast time (extended abstract). In *Proceedings of the 35th Annual Symposium on Foundations of Computer Science, FOCS'94*, pages 202–213. IEEE Computer Society Press, 1994.

[52] T. Resnick. Sudoku at the Intersection of Classical and Quantum Computing. Report CDMTCS-475, Centre for Discrete Mathematics and Theoretical Computer Science, University of Auckland, Auckland, New Zealand, Dec. 2014.

[53] T. F. Rønnow, Z. Wang, J. Job, S. Boixo, S. V. Isakov, D. Wecker, J. M. Martinis, D. A. Lidar and M. Troyer. Defining and detecting quantum speedup. *Science*, 345(6195):420–424, 2014.

[54] G. Rose and W. Macready. An introduction to quantum annealing. Technical Report Document 0712, DWave Systems, Inc., 2007.

[55] S. W. Shin, G. Smith, J. A. Smolin and U. Vazirani. How "quantum" is the D-Wave machine? arXiv:1401.7087, May 2014.

[56] P. W. Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings of the 35th Annual Symposium of on Foundations of Computer Science, Santa Fe, NM, Nov. 20-22, 1994*. IEEE Computer Society Press, November 1994. arXiv:quant-ph/9508027.

[57] W. Stein et al. *Sage Mathematics Software (Version 6.3)*. The Sage Development Team, 2014. http://www.sagemath.org.

[58] G. J. Tee. *The Monte Carlo Method*. Pergamon Press, Oxford and New York, 1966.

[59] Webster. Quantum computer (Accessed: 30 October 2014). http://www.webster-dictionary.org/definition/quantum%20computer.

[60] Wikipedia. Integer programming (Accessed 30 October 2014). http://en.wikipedia.org/wiki/Integer_programming.

[61] Wikipedia. Quantum computer (Accessed: 30 October 2014). http://en.wikipedia.org/wiki/Quantum_computer.

[62] W. K. Wooters and W. H. Zurek. A single quantum cannot be cloned. *Nature*, 299:802–803, 1982.