Navigating a Maze using a Quantum Annealer

Scott Pakin

Computer, Computational, and Statistical Sciences Division
Los Alamos National Laboratory
Los Alamos, New Mexico
pakin@lanl.gov

ABSTRACT

Quantum annealers exploit quantum effects in an attempt to minimize a classical Hamiltonian function with a higher likelihood of reaching optimality than can be expected from simulated annealing. The key programming challenge is how to express algorithms in terms of the specific Hamiltonian supported by the quantum-annealing hardware. In this paper, we present an algorithm for finding the shortest path through a maze not via a traditional backtracking mechanism but rather by expressing the shortest path as the globally optimal value of a Hamiltonian.

CCS CONCEPTS

• **Computer systems organization** → **Quantum computing**; • **Hardware** → **Quantum computation**; • **Theory of computation** → *Discrete optimization*; *Shortest paths*;

KEYWORDS

quantum computing, quantum annealing, D-Wave, mazes, puzzles

ACM Reference Format:

Scott Pakin. 2017. Navigating a Maze using a Quantum Annealer. In *PMES'17: PMES'17: Second International Workshop on Post Moore's Era Supercomputing, November 12–17, 2017, Denver, CO, USA*. ACM, New York, NY, USA, 7 pages. <https://doi.org/10.1145/3149526.3149532>

1 INTRODUCTION

A *quantum computer* exploits quantum effects to perform a computation, in some cases asymptotically faster than can be achieved classically [15]. Two very different types of quantum computers are already currently commercially available: *gate-model* (also known as *circuit-model*) quantum computers [8] and *quantum annealers* [7, 10].

Both models involve manipulating quantum bits (*qubits*), which, unlike classical bits, are not limited to being either 0 or 1 but can additionally lie in a *superposition* of 0 and 1, exhibiting properties of both values; and qubits can be *entangled* with each other so that changes to one qubit implicitly affect other qubits. In both models, a program begins and ends in a classical state: an n -bit value. However, in gate-model quantum computers, a program transforms the current state using a set of unitary matrices (“gates”) applied

in sequence to explicitly introduce superpositioning and entanglement en route to a solution. One may therefore say that gate-model quantum computers are *explicitly* quantum. In contrast, one programs a quantum annealer by specifying real-valued coefficients to an Ising-model Hamiltonian function. The quantum annealer then uses quantum effects (superposition and entanglement) to find the n classical bits that minimize the value of the Hamiltonian. One may therefore say that quantum annealers are *implicitly* quantum.

In this paper we present an approach to programming quantum annealers. As with most post-Moore’s era architectures, a significant programming challenge is how to map an arbitrary problem into the form accepted by the device. In the case of a quantum annealer, this means expressing problems in the form of an Ising-model Hamiltonian,

$$\mathcal{H} = \sum_{i=0}^{N-1} h_i \sigma_i + \sum_{i=0}^{N-2} \sum_{j=i+1}^{N-1} J_{i,j} \sigma_i \sigma_j, \quad (1)$$

in which a solution corresponds to a set of σ s that minimize $\mathcal{H}$. (Note that there is not necessarily a single, unique solution.) These σ variables are Booleans, typically expressed as $\{-1, +1\}$. A quantum-annealing *program* is merely the corresponding h and J coefficients, which are real-valued.

As an entertaining sample problem we will program a quantum annealer to find the shortest path through a maze. Figure 1 illustrates the particular (but randomly generated) 6×6 maze that we will use. The question we answer in this work is,

How can one express the maze shown in Figure 1 in terms of Equation 1’s h and J coefficients such that the σ values that minimize $\mathcal{H}$ correspond to the shortest path through the maze?

Figure 1: A maze to solve

The two main contributions of this work are

Publication rights licensed to ACM. ACM acknowledges that this contribution was authored or co-authored by an employee, contractor or affiliate of the United States government. As such, the Government retains a nonexclusive, royalty-free right to publish or reproduce this article, or to allow others to do so, for Government purposes only.

PMES'17, November 12–17, 2017, Denver, CO, USA

© 2017 Copyright held by the owner/author(s). Publication rights licensed to Association for Computing Machinery.

ACM ISBN 978-1-4503-5126-3/17/11...\$15.00

<https://doi.org/10.1145/3149526.3149532>

- (1) a presentation of a methodology that can be used to express certain types of problems in a form amenable to execution on a quantum annealer, and
- (2) empirical results from an application of this methodology to a quantum annealer containing over 1000 qubits.

The rest of the paper is structured as follows. We begin by discussing related efforts in Section 2. Section 3 presents our approach to mapping problems to a Hamiltonian. In Section 4 we explain how our approach can be implemented and quantify how well it performs on a D-Wave 2X quantum annealer. Finally, we draw some conclusions in Section 5.

2 RELATED WORK

There have been various attempts to map specific problems into the form of Equation 1. Lucas [12] and Kochenberger et al. [11] each summarize a number of these in respective 2014 survey papers.

With the advent of commercial quantum annealers from D-Wave Systems, Inc. and more users gaining access to such systems, there is an increasing interest in mapping problems to Ising-model Hamiltonians. There is a particular desire to express problems with practical importance in that form for execution on a D-Wave system. For example, Volkswagen used a D-Wave system to reduce traffic congestion by optimizing route selection [17]. The German Aerospace Center used a D-Wave system to record as much data as possible from a satellite subject to constraints in battery charge, downlink availability, and storage capacity [16]. And NASA used a D-Wave system to plan a rover's movements to minimize motion when collecting samples from multiple locations [14].

The work described in this paper is more focused on methodology than on a particular problem, although we do use the maze problem presented in the previous section as a motivating example. In this sense, our work is more along the lines of Dahl's map-coloring white paper [6] than the efforts described above.

3 APPROACH

Classically solving a maze is straightforward: Begin at the ingress, take a step that does not go through a wall or re-enter a visited room, and recursively solve the maze from that point, stopping when the egress is found or backtracking if a dead end is encountered.

It may at first seem like a daunting task to express even a simple problem like solving a maze as a minimization of an Ising-model Hamiltonian. However, we demonstrate that we can break down the work into the following steps to simplify algorithm development:

- (1) Characterize solutions to the problem (Section 3.1).
- (2) Identify simple, repeated subproblems (Section 3.2).
- (3) Manually solve the simple subproblems (Section 3.3).
- (4) Combine the simple subproblems into a complex full problem for quantum-annealing solution (Section 3.4).

3.1 Characterize solutions

Solving a maze is a shortest-path problem. We will therefore need the σ variables in Equation 1 somehow to represent paths. Longer paths should result in larger values of $\mathcal{H}$ than shorter paths, and invalid paths—cutting through a wall, not entering/exiting the maze, taking more than one path at once, etc.—should result in extremely large values of $\mathcal{H}$.

The first challenge is to represent a variable-length path given a fixed-length σ vector. This is the first point at which a modicum of creativity is involved:

Insight #1: Because the σ s are Boolean values, we can use four per room to indicate whether the path continues from that room to the north, east, south, and/or west, respectively. Then, given a solution σ , it is simple to read off the path by starting from the ingress and repeatedly taking whichever step does not re-enter the previous room. (Only one such step can exist at each point.)

3.2 Identify subproblems

A maze must have an ingress and an egress. (Technically, there is no difference between the two, as the shortest path is not directional.) Our first subproblem is to enforce that the ingress and egress are both used. This corresponds to one of the base cases for the recursion in the classical implementation. Our second subproblem is to prevent the path from passing through a wall. This corresponds to the other base case for the recursion in the classical implementation. As a maze consists of rooms, our third subproblem is to define a valid room. That is, a path through a room must obey certain constraints, which are described below. This subproblem corresponds to the recursive case in the classical implementation.

Figure 2 zooms in on the upper-right corner of Figure 1 (rooms E1 and F1), showing the rooms' north (σ_N), east (σ_E), south (σ_S), and west (σ_W) path variables. (σ_i is introduced in Section 3.3.) Dark lines represent walls, and light lines represent open passageways.

Figure 2: Variables associated with rooms E1 and F1

Our goal for the first subproblem is to force room F1's σ_N variable to be TRUE because the ingress to the maze lies to the north of room F1. Similarly, we want to force room B6's σ_S variable to be TRUE because the egress to the maze lies to the south of room B6.

Our goal for the second subproblem is to force room E1's σ_N and σ_W variables and room F1's σ_E variable to be FALSE because the path through the maze should not go north or west from room E1 or east from room F1. Similarly, we want to force the paths through all of the other walls in the maze to be FALSE.

Our goal for the third subproblem is to ensure that σ_N , σ_E , σ_S , and σ_W collectively represent a valid path through each room. But what is a valid path? A bit of thought leads to the following observation:

Insight #2: For a room in a maze to be valid, the shortest path must pass through either no passageways or exactly two passageways.

Figure 3: Examples of valid and invalid rooms

 Table 1: Effect of different values of h_0 on $\mathcal{H}$ ($= h_0\sigma_0$)

σ_0	$h_0\sigma_0$	Min?	σ_0	$h_0\sigma_0$	Min?	σ_0	$h_0\sigma_0$	Min?
-1	> 0		-1	0	✓	-1	< 0	✓
+1	< 0	✓	+1	0	✓	+1	> 0	
(a) $h_0 < 0$			(b) $h_0 = 0$			(c) $h_0 > 0$		

Figure 3 illustrates this point. With none of σ_N , σ_E , σ_S , or σ_W set to TRUE, Figure 3(a) represents the valid case of a room that does not lie on the shortest path from ingress to egress. With only σ_N set to TRUE, Figure 3(b) represents the invalid case of a room that the shortest path enters but does not exit (or vice versa). Figures 3(c) and 3(d) represent rooms that lie on the shortest path, which both enters and exits the room exactly once. Finally, Figures 3(e) and 3(f) represent rooms that are invalid because the shortest path enters or exits them too many times.

3.3 Solve subproblems

As stated in Section 1, solving a problem on a quantum annealer amounts to providing a set of h and J coefficients that constrain the hardware’s search for σ variables that minimize $\mathcal{H}$ in Equation 1. We perform the *opposite* operation for each subproblem. That is, given one or more sets of σ variables, we want to find the h and J coefficients that minimize $\mathcal{H}$.

The ingress/egress subproblem. The first subproblem described in Section 3.2 is to ensure that both the ingress and egress belong to the shortest path through the maze. Because room F1 opens to the north, this means we want to encourage room F1’s σ_N to be TRUE (+1) to make it part of the shortest path.

To indicate that a subproblem should favor TRUE for a variable, consider an instantiation of Equation 1 with $N = 1$, i.e., $\mathcal{H} = h_0\sigma_0$. Three distinct cases exist: $h_0 < 0$, $h_0 = 0$, and $h_0 > 0$. Table 1 fully evaluates these three cases, showing which of $\sigma_0 = -1$ and $\sigma_0 = +1$ minimizes $\mathcal{H}$ in each case.

As one can see from Table 1, any negative h_0 expresses a preference for the corresponding σ_0 being TRUE (+1). In practice, however, a more robust Hamiltonian exists. Because D-Wave quantum annealers provide limited precision and a limited range for both h and J coefficients, we have found that it is better to introduce a

helper variable, assign it a negative h coefficient, then indicate a preference for both variables having the same value.

Table 2 shows how this can be achieved. For an instantiation of Equation 1 with $N = 2$ and $h_0 = h_1 = 0$, the table illustrates the impact of negative, zero, and positive $J_{i,j}$ values in terms of what $\{\sigma_0, \sigma_1\}$ pairs minimize $\mathcal{H}$. According to the table, any negative $J_{0,1}$ favors the two variables having the same value while any positive $J_{0,1}$ favors the two variables having different values.

Consequently, to solve the ingress subproblem, we introduce an “ingress” variable, labeled σ_i in Figure 2, assign the corresponding h_i a negative value (e.g., -1), and couple σ_i to room F1’s σ_N with a negative $J_{N,i}$ (e.g., also -1):

$$\begin{aligned} \mathcal{H}_{\text{ingress}} &= h_N\sigma_N + h_i\sigma_i + J_{N,i}\sigma_N\sigma_i \\ &= -\sigma_i - \sigma_N\sigma_i \end{aligned} \quad (2)$$

The egress subproblem, with room B6 exiting to the south, has an analogous solution:

$$\begin{aligned} \mathcal{H}_{\text{egress}} &= h_S\sigma_S + h_e\sigma_e + J_{S,e}\sigma_S\sigma_e \\ &= -\sigma_e - \sigma_S\sigma_e \end{aligned} \quad (3)$$

The solid-walls subproblem. Blocking the shortest path from passing through a wall represents the opposite of the ingress/egress subproblem. Instead of encouraging the path from reaching the ingress and egress, we discourage it from going through a wall. While encouraging a direction is based on coupling to a negative h coefficient (as per Table 1), discouraging a direction is based on coupling to a positive h coefficient. For example, to prevent the path from going west from room E1 in Figure 2, we can define

$$\begin{aligned} \mathcal{H}_{\text{wall}} &= h_W\sigma_W + h_{\text{wall}}\sigma_{\text{wall}} + J_{W,\text{wall}}\sigma_W\sigma_{\text{wall}} \\ &= \sigma_{\text{wall}} - \sigma_W\sigma_{\text{wall}} \end{aligned} \quad (4)$$

The path-through-a-room subproblem. As discussed in Section 3.2, the global shortest path must pass through each room either exactly once (represented by exactly two of σ_N , σ_E , σ_S , and σ_W being TRUE) or not at all (represented by none of σ_N , σ_E , σ_S , and σ_W being TRUE). Expanding Equation 1 to $N = 4$ (for the four cardinal directions) produces a template for our room Hamiltonian,

$$\begin{aligned} \mathcal{H}_{\text{room}} &= h_N\sigma_N + h_E\sigma_E + h_S\sigma_S + h_W\sigma_W \\ &\quad + J_{N,E}\sigma_N\sigma_E + J_{N,S}\sigma_N\sigma_S + J_{N,W}\sigma_N\sigma_W \\ &\quad + J_{E,S}\sigma_E\sigma_S + J_{E,W}\sigma_E\sigma_W \\ &\quad + J_{S,W}\sigma_S\sigma_W. \end{aligned} \quad (5)$$

Table 2: Effect of different values of $J_{0,1}$ on $\mathcal{H}$ ($= J_{0,1}\sigma_0\sigma_1$)

σ_0	σ_1	$J_{0,1}\sigma_0\sigma_1$	Min?
-1	-1	< 0	✓
-1	+1	> 0	
+1	-1	> 0	
+1	+1	< 0	✓

(a) $J_{0,1} < 0$

σ_0	σ_1	$J_{0,1}\sigma_0\sigma_1$	Min?
-1	-1	0	✓
-1	+1	0	✓
+1	-1	0	✓
+1	+1	0	✓

(b) $J_{0,1} = 0$

σ_0	σ_1	$J_{0,1}\sigma_0\sigma_1$	Min?
-1	-1	> 0	
-1	+1	< 0	✓
+1	-1	< 0	✓
+1	+1	> 0	

(c) $J_{0,1} > 0$

Table 3: Preparing a system of inequalities to represent a room

σ_N	σ_E	σ_S	σ_W	Valid?	$\mathcal{H}_{\text{room}}$
-1	-1	-1	-1	✓	$= k$
-1	-1	-1	+1		$> k$
-1	-1	+1	-1		$> k$
-1	-1	+1	+1	✓	$= k$
-1	+1	-1	-1		$> k$
-1	+1	-1	+1	✓	$= k$
-1	+1	+1	-1	✓	$= k$
-1	+1	+1	+1		$> k$
+1	-1	-1	-1		$> k$
+1	-1	-1	+1	✓	$= k$
+1	-1	+1	-1	✓	$= k$
+1	-1	+1	+1		$> k$
+1	+1	-1	-1	✓	$= k$
+1	+1	-1	+1		$> k$
+1	+1	+1	-1		$> k$
+1	+1	+1	+1		$> k$

As before, our goal is to solve the subproblem for the h and J coefficients given the variables. To do so for this more complex subproblem we set up then solve a system of inequalities. All values of variables σ_N , σ_E , σ_S , and σ_W that represent valid rooms should cause $\mathcal{H}_{\text{room}}$ to evaluate to the same, minimal value; call it k . All values of the variables that represent invalid rooms should cause $\mathcal{H}_{\text{room}}$ to evaluate to any value greater than k (not necessarily the same for all sets of values).

Table 3 shows how to set up the system of inequalities. It fully enumerates all 16 sets of possible values for σ_N , σ_E , σ_S , and σ_W , labels each set as valid or invalid, and asserts that Equation 5 is respectively either equal to or greater than k .

Unfortunately, in this particular case, the system of inequalities is unsolvable. However, it can be made solvable by introducing an ancilla variable, which we call σ_a . We maintain the same number of valid rows (7) while more than doubling the number of invalid rows (from 9 to 25).

Table 4 augments Table 3 with a σ_a variable. (For brevity, only the valid rows are shown.) For example, the unused-room case ($\sigma_N = \sigma_E = \sigma_S = \sigma_W = -1$) is considered valid for $\sigma_a = +1$ but invalid for $\sigma_a = -1$. The case of the shortest path leading to the

Table 4: A solvable variation of Table 3 (valid rows only)

σ_N	σ_E	σ_S	σ_W	σ_a	Valid?	$\mathcal{H}_{\text{room}}$
-1	-1	-1	-1	+1	✓	$= k$
-1	-1	+1	+1	-1	✓	$= k$
-1	+1	-1	+1	-1	✓	$= k$
-1	+1	+1	-1	-1	✓	$= k$
+1	-1	-1	+1	-1	✓	$= k$
+1	-1	+1	-1	-1	✓	$= k$
+1	+1	-1	-1	-1	✓	$= k$

north and east ($\sigma_N = \sigma_E = +1$; $\sigma_S = \sigma_W = -1$) is considered valid for $\sigma_a = -1$ but invalid for $\sigma_a = +1$.

The additional degrees of freedom introduced by the ancilla variable and the corresponding selection of valid rows make the room subproblem solvable. In one possible solution,

$$\begin{aligned}
\mathcal{H}_{\text{room}} = & \frac{1}{2}\sigma_N + \frac{1}{2}\sigma_E + \frac{1}{2}\sigma_S + \frac{1}{2}\sigma_W + \sigma_a \\
& + \frac{1}{4}\sigma_N\sigma_E + \frac{1}{4}\sigma_N\sigma_S + \frac{1}{4}\sigma_N\sigma_W + \frac{1}{2}\sigma_N\sigma_a \\
& + \frac{1}{4}\sigma_E\sigma_S + \frac{1}{4}\sigma_E\sigma_W + \frac{1}{2}\sigma_E\sigma_a \\
& + \frac{1}{4}\sigma_S\sigma_W + \frac{1}{2}\sigma_S\sigma_a \\
& + \frac{1}{2}\sigma_W\sigma_a
\end{aligned} \tag{6}$$

Determining the minimum number of ancilla variables needed and an appropriate set of ancilla values to associate with valid rows is an open problem. For Hamiltonians involving only a few variables, brute-force techniques suffice. For larger Hamiltonians, the search space can be reduced using techniques such as those developed by Bian et al. [1].

3.4 Combine subproblems

The final step is to combine the simple—and now solved—subproblems into a complex problem suitable for solution by a quantum annealer. A convenient feature of Hamiltonians is that they can simply be added to produce a Hamiltonian that represents the union of the constraints imposed by each sub-Hamiltonian's coefficients. In the case of the maze problem, the necessary “glue” is a set of assertions that passageways be symmetric. For example, if one can get from room X to room Y by going north, then one can get from room Y to room X by going south.

First, however, we need to specialize each room based on its pattern of walls and the ingress/egress. For example, referring to Figure 2 and the definitions of $\mathcal{H}_{\text{ingress}}$, $\mathcal{H}_{\text{egress}}$, $\mathcal{H}_{\text{wall}}$, and $\mathcal{H}_{\text{room}}$

from Section 3.3, parameterized by cardinal direction,

$$\mathcal{H}_{E1} = \mathcal{H}_{\text{room}} + \mathcal{H}_{\text{wall}}(N) + \mathcal{H}_{\text{wall}}(W) \quad (7)$$

and

$$\mathcal{H}_{F1} = \mathcal{H}_{\text{room}} + \mathcal{H}_{\text{wall}}(E) + \mathcal{H}_{\text{ingress}}(N). \quad (8)$$

Given similar definitions for all 36 rooms in the maze, we can now combine all of the rooms into a complete specification of a maze. As indicated by Table 2, a negative J coefficient expresses a preference for two variables having equal values. Hence, one can ensure symmetric passageways by coupling the relevant variables with a negative J . For example, in Figure 2, the Hamiltonians for rooms E1 and F1 can be combined by coupling room E1's east variable with room F1's west variable:

$$\mathcal{H}_{E1,F1} = \mathcal{H}_{E1} + \mathcal{H}_{F1} - \sigma_E^{(E1)} \sigma_W^{(F1)}. \quad (9)$$

The Hamiltonian for the complete maze problem contains 628 terms:

1 ingress	× 2 terms per $\mathcal{H}_{\text{ingress}}$	(Equation 2)
1 egress	× 2 terms per $\mathcal{H}_{\text{egress}}$	(Equation 3)
42 walls	× 2 terms per $\mathcal{H}_{\text{wall}}$	(Equation 4)
+	36 rooms × 15 terms per $\mathcal{H}_{\text{room}}$	(Equation 6)
628 total terms for the complete maze		

While this may seem like a large number, the problem Hamiltonian comprises only four types of sub-Hamiltonians ($\mathcal{H}_{\text{ingress}}$, $\mathcal{H}_{\text{egress}}$, $\mathcal{H}_{\text{wall}}$, and $\mathcal{H}_{\text{room}}$) and can easily be generated mechanically.

4 EVALUATION

In this section, we perform a quantitative evaluation of our maze problem, analyzing its performance. Section 4.1 discusses how we mapped the logical problem onto the physical topology of a D-Wave 2X quantum annealer; Section 4.2 presents end-to-end performance figures from program execution on a D-Wave 2X; and Section 4.3 quantifies how successful the D-Wave 2X was at finding correct solutions.

4.1 Implementation

The D-Wave 2X is a quantum annealer sold commercially by D-Wave Systems, Inc. In a D-Wave 2X, each σ variable is implemented as a superconducting flux qubit; each h coefficient is implemented by introducing an external flux bias; and each J coefficient is implemented by introducing a pairwise coupling energy [9].

Mapping our problem Hamiltonian onto a D-Wave 2X requires some preparation. First, as in Equation 1, σ variables are numbered ($\sigma_0, \sigma_1, \sigma_2, \dots$) while Section 3 worked exclusively with *named* variables ($\sigma_N, \sigma_E, \sigma_S$, etc.). Second, a D-Wave 2X does not provide $J_{i,j}$ couplers for arbitrary $\{i, j\}$ pairs. Rather, it implements a particular topology called a *Chimera graph* [3], in which at most six $J_{i,j}$ coefficients are associated with any given σ_i . Furthermore, due to manufacturing and calibration imperfections, not all qubits and couplers will be present in any given installation. As a specific example, Figure 4 illustrates the topology of Ising, the D-Wave 2X system installed at Los Alamos National Laboratory that we use for our experiments. Ising is arranged in a C_{12} Chimera graph (i.e., a 12×12 array of 8-qubit bipartite graphs, called *unit cells*) with 1095 active qubits; this represents a 5% qubit dropout.

Figure 4: Topology of LANL's 1095-qubit D-Wave 2X system, "Ising"

Logical, problem Hamiltonians need to be mapped onto the physical Hamiltonian via a *minor embedding*—splitting certain variables into two or more variables made equal using $J < 0$ values—to compensate for the limited graph degree. As an example of a physical Hamiltonian,

$$\begin{aligned} \mathcal{H}_{\text{room}^*} = & \frac{1}{6}\sigma_0 + \frac{1}{2}\sigma_2 + \frac{1}{6}\sigma_4 + \frac{1}{6}\sigma_6 + \frac{1}{4}\sigma_7 + \frac{1}{2}\sigma_8 + \frac{1}{4}\sigma_9 + \frac{1}{2}\sigma_{10} \\ & + \frac{1}{4}\sigma_{101} + \frac{1}{4}\sigma_{102} - \sigma_0\sigma_4 - \sigma_0\sigma_6 + \frac{1}{2}\sigma_2\sigma_4 - \sigma_2\sigma_8 \\ & + \frac{1}{4}\sigma_6\sigma_{100} + \frac{1}{4}\sigma_6\sigma_{101} + \frac{1}{4}\sigma_6\sigma_{102} + \frac{1}{4}\sigma_7\sigma_{100} \\ & + \frac{1}{8}\sigma_7\sigma_{101} - \sigma_7\sigma_{102} + \frac{1}{2}\sigma_8\sigma_{100} + \frac{1}{2}\sigma_8\sigma_{101} \\ & + \frac{1}{2}\sigma_8\sigma_{102} + \frac{1}{4}\sigma_9\sigma_{100} - \sigma_9\sigma_{101} + \frac{1}{8}\sigma_9\sigma_{102} \end{aligned} \quad (10)$$

represents Equation 6 after being minor-embedded into the upper-left unit cell in Figure 4 (qubits σ_0 – σ_7 , with σ_7 missing) and the unit cell directly below that (qubits σ_6 – σ_{103} , with σ_{103} missing).

Naturally, one would want to automate the logical-to-physical Hamiltonian conversion, including both mapping symbolic to numerical variables and performing the minor embedding. This is indeed possible. We have developed a *quantum macro assembler*, QMASM,¹ which performs both tasks [13]. QMASM lets a programmer define a subproblem as a macro then instantiate that macro as many times as needed.

Listing 1 shows how $\mathcal{H}_{\text{room}}$ (Equation 6) can be expressed directly as a QMASM macro, `room`. Hamiltonian variables $\sigma_N, \sigma_E, \sigma_S, \sigma_W$, and σ_a are expressed as QMASM variables `N`, `E`, `S`, `W`, and `a`, respectively. (The “\$” indicates that a variable is “uninteresting” and that its value should not be reported except at high verbosity levels.) Listing 2 shows an excerpt of the complete maze-solving program that corresponds to Figure 2. For readability, the program defines `egress` and `wall` as `TRUE` and `FALSE`, respectively. It then instantiates the `room` macro from Listing 1, naming the result `E1`. It “pins” `E1.N` to `FALSE` (−1) using the approach shown in Equation 4. It

¹QMASM is open-source software, available from <https://github.com/lanl/qmasm>.

specifies that E2.N should be equal to E1.S using a highly negative J coefficient (computed after all other J coefficients are known). The code then describes similar constraints for room F1.

Listing 1: QMASM version of $\mathcal{H}_{\text{room}}$ (Equation 6)

```

1 # A room in a maze.
2 # Either 0 or 2 of N/E/S/W
3 # should be True.
4 !begin_macro room
5   N 0.50
6   E 0.50
7   S 0.50
8   W 0.50
9   $a 1.00
10
11   N E 0.25
12   N S 0.25
13   N W 0.25
14   N $a 0.50
15   E S 0.25
16   E W 0.25
17   E $a 0.50
18   S W 0.25
19   S $a 0.50
20   W $a 0.50
21 !end_macro room

```

Listing 2: Excerpt from the QMASM maze-solving program

```

1 !alias egress TRUE
2 !alias wall FALSE
3
4 !use_macro room E1
5 E1.N := wall
6 E1.W := wall
7 E2.N = E1.S
8
9 !use_macro room F1
10 F1.N := egress
11 F1.E := wall
12 F1.W = E1.E
13 F2.N = F1.S

```

QMASM maps variable names to logical qubit numbers then uses a D-Wave-provided, heuristic, randomized minor-embedding algorithm [4] to convert logical qubit numbers to physical qubit numbers. (Measured over 60 trials, the maze program’s 219 logical qubits are mapped to 580.25 ± 39.71 physical qubits.) QMASM uses D-Wave’s SAPI library [5] to execute the program on a D-Wave system. Finally, QMASM reports the value of each program variable in a human-readable format.

4.2 Performance

Reporting the time of a program running on a quantum annealer is not a straightforward task. All quantum computers are fundamentally stochastic devices. Hence, one must run each program repeatedly to acquire a distribution of solutions.

Quantum annealers in particular increase their likelihood of returning a correct answer with increased annealing time—assuming ideal conditions (zero temperature and no noise) [2]. However, (a) real-world conditions violate that property, introducing nonlinear effects, (b) a D-Wave 2X offers a limited range of annealing times (5–2000 μ s for Ising), and (c) it is computationally expensive to determine the minimum annealing time, even in the ideal case, although this *can* be worked out analytically in a few special cases [7].

We therefore take an empirical approach to reporting performance. Table 5 reports our measurements of the maze program, stating the mean, standard deviation, and coefficient of variation (CV) over 30 runs. Each run comprised 10,000 anneals at 20 μ s/anneal. To present a highly accurate picture of performance, *all* time components are reported, including compilation time, job-scheduling time, and file I/O time in addition to the time spent actually finding the

Table 5: End-to-end timing breakdown for 10,000 anneals at 20 μ s/anneal

Operation	Time (μ s)	CV
I/O, lexing, and parsing	$81,000 \pm 3,051$	(4%)
Minor embedding	$86,253,000 \pm 55,760,716$	(65%)
QMASM optimizations	$-71,824,667 \pm 57,626,986$	(80%)
Network and job queue	$9,029,645 \pm 22,822,557$	(253%)
QPU programming	$16,011 \pm 28$	(0%)
QPU annealing	$200,000 \pm 0$	(0%)
QPU readout	$1,410,000 \pm 0$	(0%)
QPU delay	$210,000 \pm 0$	(0%)
QPU postprocessing	$49,677 \pm 142,935$	(288%)
SAPI postprocessing	$-5,117,000 \pm 22,964,657$	(449%)
<i>Total</i>	$20,307,667 \pm 14,769,523$	(73%)

shortest path through the maze. Note that the QPU (quantum processing unit) programming/annealing/readout/delay times shown in Table 5 scale proportionally with the number of anneals, and the QPU annealing time further scales proportionally with the annealing time.

The timings are extremely noisy, and the end-to-end time is a somewhat disappointing 20.3s, which is slow for a mere 6 $\times$ 6 maze. However, the time spent in the QPU proper is only a fraction of the total: 1.9s, of which only 0.2s are spent annealing.

One surprise is that QMASM optimizations, which remove redundant variables from the Hamiltonian, so substantially improve performance. This is because they simplify the task of D-Wave’s minor embedder, which is the main performance bottleneck. Classical postprocessing of results (*SAPI postprocessing* in the table) also appears as a negative cost. This may be because postprocessing affects the number of unique solutions found (e.g., by mapping previously different solutions to the same, improved solution), which in turn affects the time needed to bin, sort, and return the histogram of solutions to the caller. The high CV indicates that little can be said with confidence about the precise cost of this postprocessing step, however.

4.3 Correctness

In the case of our maze example, it is simple to check each proposed solution for correctness. As indicated in Insight #1, a program can verify that there exists a path from ingress to egress that does not retrace its steps. For completeness, it additionally needs to verify that each room is valid as per Insight #2 and that all walls are honored.

As implied by the adiabatic theorem [2], longer annealing times lead to a higher likelihood of observing a correct solution (a global minimum of Equation 1). However, the adiabatic theorem applies in an idealized environment, meaning zero temperature and no environmental noise. In a more realistic setting, longer annealing times are not necessarily better because they represent a larger window of vulnerability to thermal effects and noise.

We measured correctness empirically by performing 1,000,000 anneals using different annealing times and both with and without

Table 6: Correct solutions from 1,000,000 anneals

Annealing time (μ s)	Correct solutions, no postprocessing	Correct solutions with postprocessing
5	7 (1.4/s)	4,730 (946.0/s)
20	241 (12.1/s)	4,561 (228.1/s)
2000	394 (0.2/s)	505,905 (253.0/s)

classical postprocessing enabled. Table 6 presents the corresponding measurements. The table indicates that correct solutions are found relatively rarely by quantum annealing, at a rate of at most 12.1 per annealing second. The 2000 μ s anneals observe the greatest number of correct solutions, but the 20 μ s anneals observe the greatest number of correct solutions per unit time.

Classical postprocessing helps substantially in all three cases, increasing the rate to at most 946.0 per annealing second. This indicates that in many cases in which the D-Wave 2X fails to find the correct solution—the global minimum—it at least gets stuck in a nearby local minimum from which classical postprocessing can help it escape.

Although Table 6 presents data for only single batches of 1,000,000 samples we suspect that multiple runs would observe large standard deviations. The table data should therefore be considered only crude estimates of solution robustness. We further remind the reader that all the data presented here are extremely problem-specific (and machine-specific) and cannot be generalized to other problems (or D-Wave instances).

5 CONCLUSIONS

We presented an approach for solving problems using a quantum annealer. The key idea is to split the problem into subproblems, solve these manually in the *reverse* direction—solving for the Hamiltonian function’s coefficients given values for the variables—combine the subproblems, and run the difficult complete problem through a quantum annealer, this time solving for the variables.

As a demonstration of this approach, we showed how to find the shortest path through a maze. However, the same approach is in fact widely applicable. For example, circuit satisfiability problems can be broken down into subproblems representing binary operators (AND, OR, NOT, etc.). And a four-coloring of a planar map can be broken down into subproblems representing the individual regions.

We reported empirical performance data from solving a 6 $\times$ 6 maze on a 1095-qubit D-Wave 2X quantum annealer. For completeness, we included not only the annealing time (a total of 0.2s) but all aspects of compilation, job submission, and reporting of results. Although the total time is slow for such a small problem, we expect that future architectures will offer increased qubit counts, enabling larger problems to be expressed; new, faster minor-embedding algorithms will be developed; and engineering improvements will reduce the various overheads involved. The programming approach presented in this paper will facilitate exploiting whatever performance is provided by the higher-performance quantum annealers of the future.

ACKNOWLEDGMENTS

Research presented in this paper was supported by the Laboratory Directed Research and Development program of Los Alamos National Laboratory under project number 20160069DR. Los Alamos National Laboratory is operated by Los Alamos National Security LLC for the US Department of Energy under contract DE-AC52-06NA25396.

REFERENCES

- [1] Zhengbing Bian, Fabian Chudak, Robert Israel, Brad Lackey, William G. Macready, and Aidan Roy. 2014. Discrete Optimization using Quantum Annealing on Sparse Ising Models. *Frontiers in Physics* 2 (Sept. 2014), 56. <https://doi.org/10.3389/fphy.2014.00056>
- [2] M[ax] Born and V[ladimir] Fock. 1928. Beweis des Adiabatsatzes. *Zeitschrift für Physik* 51, 3–4 (March 1, 1928), 165–180. <https://doi.org/10.1007/BF01343193>
- [3] Paul I. Bunyk, Emile M. Hoskinson, Mark W. Johnson, Elena Tolkacheva, Fabio Altomare, Andrew J. Berkley, Richard Harris, Jeremy P. Hilton, Trevor Lanting, Anthony J. Przybysz, and Jed Whittaker. 2014. Architectural Considerations in the Design of a Superconducting Quantum Annealing Processor. *IEEE Transactions on Applied Superconductivity* 24, 4 (Aug. 2014), 1–10. <https://doi.org/10.1109/TASC.2014.2318294>
- [4] Jun Cai, Bill Macready, and Aidan Roy. 2014. *A Practical Heuristic for Finding Graph Minors*. Technical Report. D-Wave Systems Inc., Burnaby, British Columbia, Canada. arXiv:quant-ph/1406.2741
- [5] D-Wave Systems, Inc. 2017. *Developer Guide for Python*. D-Wave Systems, Inc., Burnaby, British Columbia, Canada.
- [6] E. D. Dahl. 2013. *Programming with D-Wave: Map Coloring Problem*. White Paper. D-Wave Systems, Burnaby, British Columbia, Canada. <https://www.dwavesys.com/sites/default/files/Map%20Coloring%20WP2.pdf>
- [7] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, and Michael Sipser. 2000. *Quantum Computation by Adiabatic Evolution*. Technical Report MIT-CTP-2936. Massachusetts Institute of Technology. arXiv:quant-ph/0001106
- [8] Richard P. Feynman. 1986. Quantum Mechanical Computers. *Foundations of Physics* 16, 6 (June 1986), 507–531. <https://doi.org/10.1007/BF01886518>
- [9] M. W. Johnson, M. H. S. Amin, S. Gildert, T. Lanting, F. Hamze, N. Dickson, R. Harris, A. J. Berkley, J. Johansson, P. Bunyk, E. M. Chapple, C. Enderud, J. P. Hilton, K. Karimi, E. Ladizinsky, N. Ladizinsky, T. Oh, I. Perminov, C. Rich, M. C. Thom, E. Tolkacheva, C. J. S. Truncik, S. Uchaikin, J. Wang, B. Wilson, and G. Rose. 2011. Quantum Annealing with Manufactured Spins. *Nature* 473, 7346 (May 2011), 194–198. <https://doi.org/10.1038/nature10012>
- [10] Tadashi Kadowaki and Hidetoshi Nishimori. 1998. Quantum Annealing in the Transverse Ising Model. *Physical Review E* 58 (Nov. 1, 1998), 5355–5363. Issue 5. <https://doi.org/10.1103/PhysRevE.58.5355>
- [11] Gary Kochenberger, Jin-Kao Hao, Fred Glover, Mark Lewis, Zhipeng Lü, Haibo Wang, and Yang Wang. 2014. The Unconstrained Binary Quadratic Programming Problem: A Survey. *Journal of Combinatorial Optimization* 28, 1 (July 2014), 58–81. <https://doi.org/10.1007/s10878-014-9734-0>
- [12] Andrew Lucas. 2014. Ising Formulations of Many NP Problems. *Frontiers in Physics* 2 (Feb. 12, 2014), 5. <https://doi.org/10.3389/fphy.2014.00005>
- [13] Scott Pakin. 2016. A Quantum Macro Assembler. In *Proceedings of the 20th Annual IEEE High Performance Extreme Computing Conference (HPEC 2016)*. IEEE, Waltham, Massachusetts, USA, 8. <https://doi.org/10.1109/HPEC.2016.7761637>
- [14] Eleanor G. Rieffel, Davide Venturelli, Bryan O’Gorman, Minh B. Do, Elicia M. Prystay, and Vadim N. Smelyanskiy. 2015. A Case Study in Programming a Quantum Annealer for Hard Operational Planning Problems. *Quantum Information Processing* 14, 1 (Jan. 2015), 1–36. <https://doi.org/10.1007/s11128-014-0892-x>
- [15] Peter W. Shor. 1999. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM Rev.* 41, 2 (1999), 303–332. <https://doi.org/10.1137/S0036144598347011>
- [16] Tobias Stollenwerk and Achim Basermann. 2016. Experiences with Scheduling Problems on Adiabatic Quantum Computers. In *Proceedings of the First International Workshop on Post-Moore’s Era Supercomputing (PMES 2016)*, Jeffrey S. Vetter and Satoshi Matsuoka (Eds.). IEEE/ACM, Salt Lake City, Utah, USA, 45–46. <https://sites.google.com/site/2016pmes/program/PMES16-position-papers.pdf> Oak Ridge National Laboratory, Future Technologies Group Technical Report FTGTR-2016-11.
- [17] Volkswagen. 2017. *Research Project Successful: Volkswagen IT Experts use Quantum Computing for Traffic Flow Optimization*. Press Release. Volkswagen, Wolfsburg/Hanover, Germany. <https://www.volkswagen-media-services.com/en/detailpage/-/detail/Research-project-successful-Volkswagen-IT-experts-use-quantum-computing-for-traffic-flow-optimization/view/4753722/6e1e015af7bda8f2a4b42b43d2dc9b5>