IEEE Transactions on Quantum Engineering

Received July 27, 2020; accepted August 21, 2020; date of publication September 4, 2020; date of current version October 28, 2020.

Digital Object Identifier 10.1109/TQE.2020.3021921

Solving the Network Shortest Path Problem on a Quantum Annealer

THOMAS KRAUSS[1] AND JOEY MCCOLLUM[1]

Ted and Karyn Hume Center for National Security and Technology, Virginia Polytechnic Institute and State University, Blacksburg, VA 24060 USA

Corresponding author: Thomas Krauss. (tkrauss@vt.edu)

ABSTRACT This article addresses the formulation for implementing a single source, single-destination shortest path algorithm on a quantum annealing computer. Three distinct approaches are presented. In all the three cases, the shortest path problem is formulated as a quadratic unconstrained binary optimization problem amenable to quantum annealing. The first implementation builds on existing quantum annealing solutions to the traveling salesman problem, and requires the anticipated maximum number of vertices on the solution path $|P|$ to be provided as an input. For a graph with $|V|$ vertices, $|E|$ edges, and no self-loops, it encodes the problem instance using $|V||P|$ qubits. The second implementation adapts the linear programming formulation of the shortest path problem, and encodes the problem instance using $|E|$ qubits for directed graphs or $2|E|$ qubits for undirected graphs. The third implementation, designed exclusively for undirected graphs, encodes the problem in $|E| + |V|$ qubits. Scaling factors for penalty terms, complexity of coupling matrix construction, and numerical estimates of the annealing time required to find the shortest path are made explicit in the article.

INDEX TERMS Quantum annealing, quantum computing, shortest path problem, simulated annealing.

I. INTRODUCTION

The shortest path problem is a well-studied primitive in graph theory. The single source, single-destination variant of the problem is the simplest to describe: Given a graph with costs assigned to its edges and given a source and terminal vertex in the graph, find a continuous path from the source vertex to the terminal vertex whose constituent edges have minimum total cost. The problem is central to a wide variety of real-world applications ranging from navigation, autonomous vehicle route planning, optimizing trace layout on printed circuit boards, and social network analysis.

Quantum annealing is an optimization technique that exploits a time-dependent Hamiltonian for a set of N qubits by gradually decreasing quantum fluctuations [1]. The Hamiltonian, or energy function, begins in a fully entangled, random state and is slowly transformed to the Hamiltonian of the optimization problem. The slow progression of the quantum energy state allows the system of qubits to traverse energy barriers and find the global minimum of the high-dimensional energy function. The annealing process begins in a state with all qubits in superposition. At the completion of the annealing, the qubits remain in the ground state of the desired problem Hamiltonian and, therefore, represent the global optimum [2]–[4]. The solution vector of qubit values is then read and returned. The energy function is generally represented as an Ising model, or a related quadratic unconstrained binary optimization (QUBO) formulation.

This article will detail a QUBO formulation of this shortest path (lowest cost) problem suitable for execution on a D-Wave quantum annealing machine or via a simulated annealing package. We begin by providing background on the QUBO formulation and notation in Section II, followed by the related work in Section III. The following three sections develop three separate but related approaches to solving the shortest path problem. Section IV describes a vertex- or hop-based approach, while Sections V and VI develop edge-based approaches for directed and undirected graphs. For each algorithm, a brief analysis of complexity and qubit resource requirements is offered. Section VII discusses the subject of the annealing time required to find the minimum-energy solution with high probability. Section VIII presents a concise comparison of the complexity of various classical and quantum gate algorithms with the proposed approaches. In Section IX, all of the proposed formulations are demonstrated using a toy problem. Finally, we offer some conclusions in Section X.

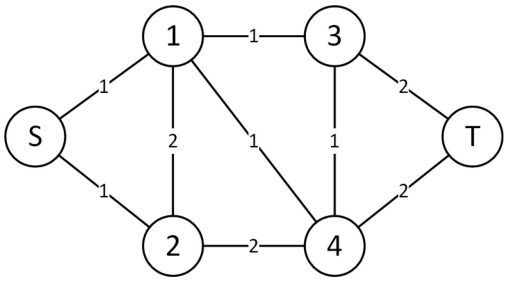

FIGURE 1. Example undirected graph with specified edge costs.

II. PRELIMINARIES

Consider the example undirected graph shown in Fig. 1. Here, the source vertex is identified as vertex s and the terminal vertex as t. Each edge has a transition cost as marked in the graphic (e.g., the edge from vertex 2 to vertex 4 has a cost of 2 units).

Clearly, for the simple graph in Fig. 1 the lowest cost route is $(s, 1, 3, t)$ with a total cost of 4. Note that this lowest cost path is not unique; $(s, 1, 4, t)$ has the same cost.

A *graph* $G = (V, E, C)$ consists of a set of vertices $V = \{1, 2, \ldots\}$, a set of edges $E = \{(1, 2), (1, 3), \ldots\}$, and a set of edge costs $C = \{c_{1,2}, c_{1,3}, \ldots\} \subset \mathbb{R}$. Edge costs are assumed to be real-valued and may be negative, subject to certain restrictions (see next paragraph). In a *directed* graph, edges connecting the same two vertices may have opposite orientations (so $(i, j) \neq (j, i)$), while in an *undirected* graph, all edges lack orientation (so $(i, j) = (j, i)$). By convention, the vertex indices of edges in an undirected graph will be listed in ascending order. The total number of vertices and edges in the graph are denoted by $|V|$ and $|E|$, respectively. The *degree* of vertex i, denoted $\deg(i)$, is defined as the number of edges (both incoming and outgoing, in the case of directed graphs) with one end at i. The *minimum degree* and *maximum degree* of a graph, denoted by δ and Δ, respectively, are the minimum and maximum degree taken over all of the graph's vertices.

Throughout this article, the class of graphs considered to be "general" for the purpose of the shortest path problem excludes undirected graphs with negative edge costs and directed graphs with cycles of negative total cost. This is because for such graphs, the set of shortest path solutions is unbounded; one can simply traverse a negative-cost cycle to make the total cost of the path arbitrarily low. Once these cases are ruled out, we can guarantee that in any shortest path, the multiplicity of every edge in the path is at most one.

A QUBO problem in N variables is formulated as a computation to find the solution vector $\boldsymbol{x}$ that produces the minimum "energy" where the energy is described as a homogeneous quadratic polynomial

$$H = \arg\min_{\boldsymbol{x}} \{\boldsymbol{x}^\top \boldsymbol{Q} \boldsymbol{x}\} \tag{1}$$

where $\boldsymbol{x} \in \{0, 1\}^N$ and $\boldsymbol{Q} \in \mathbb{R}^{N \times N}$. The objective function to be minimized is a homogeneous quadratic polynomial (i.e., a degree-2 polynomial with no linear or constant terms) with real coefficients in N variables. Since the products between variables in the objective function are commutative, the matrix $\boldsymbol{Q}$ containing their coefficients must be symmetric; for succinctness, it is conventionally represented as upper triangular. As its name suggests, the QUBO formulation does not allow the inclusion of constraints, so to convert a constrained optimization problem to a QUBO problem, the constraints must be converted into penalty terms that will prevent invalid solutions from achieving the minimum value of the objective function. Problems in the QUBO formulation can be converted to problems in the Ising formulation in a straightforward way [5].

All approaches to the shortest path problem outlined in this article will define the problem using the QUBO formulation. For the purposes of runtime analysis, it is assumed that the $\boldsymbol{Q}$ matrix containing the coefficients of the objective function is represented in sparse format, so that the time required to populate it scales with the number of writes to its cells rather than the N^2 size of its explicit representation.

III. RELATED WORK

The shortest path algorithm has a rich history of classical approaches composed of increasingly inventive algorithms for general graphs and for specific classes of graphs. For the most general case, where negative-cost edges are permitted as long as there are no negative-cost cycles, the algorithm of Bellman and Ford based on dynamic programming can solve the problem in $\mathcal{O}(|V||E|)$ time using $\mathcal{O}(|V|)$ space [6], [7]. Virtually all other general-case algorithms require the additional restriction that the graph contain no negative edge costs. All of these algorithms are essentially implementations of the same algorithm with progressively more efficient data structures, and they all share a space bound of $\mathcal{O}(|V|)$. For graphs with nonnegative edge costs, the original formulation of Dijkstra's algorithm using simple arrays can solve the problem in $\mathcal{O}(|V|^2)$ steps [8]. For sparse graphs, a min-priority queue can improve the performance to $\mathcal{O}((|V| + |E|) \log |V|)$ [9]. Replacing the queue with a Fibonacci heap improves this further to $\mathcal{O}(|E| + |V| \log |V|)$, which is the current best-known runtime for general graphs [10].

For special classes of graphs, optimal or near-optimal algorithms for finding shortest paths have been developed. For unweighted graphs, breadth-first search can trivially solve the problem in linear time and space relative to the size of the graph (i.e., $\mathcal{O}(|V| + |E|)$. Even for the more general class of directed acyclic graphs, a shortest path can be found within the same time and memory bounds by topologically sorting the graph [11, pp. 655–657]. For planar graphs, which satisfy the property $|E| \leq 3|V| - 6$ (and, hence, $|E| = \mathcal{O}(|V|)$), the fastest implementation of Dijkstra's algorithm would have a runtime of $\mathcal{O}(|V| \log |V|)$, but this has been successively improved to $\mathcal{O}(|V| \sqrt{\log |V|})$ [12], then to an optimal $\mathcal{O}(|V|)$ [13]; both algorithms assume nonnegative edge costs and have a space requirement linear in the size of the graph. For general directed graphs whose

IEEE Transactions on Quantum Engineering

costs are restricted to integers in $\{0, 1, \ldots, C\}$, implementing Dijkstra's algorithm with a van Emde Boas tree yields an $\mathcal{O}(|E| \log\log C)$-time, $\mathcal{O}(|V| + C)$-space algorithm [14]. In combination with other data structures, this can be improved to an $\mathcal{O}(|E| + |V|\sqrt{\log C})$-time, $\mathcal{O}(|E| + \log^2 C)$-space algorithm [15]. Recent advancements in data structure design have improved the time bound to $\mathcal{O}(|E| + |V| \log\log |V|)$ or $\mathcal{O}(|E| + |V| \log\log C)$ and the space bound to linear in the size of the graph [16]. For undirected graphs with the same restriction on edge costs, a remarkable algorithm that runs in linear time and space complexity has been demonstrated [17].

Quantum algorithms for the shortest path problem are much less abundant. For complete graphs, where $|E| = \mathcal{O}(|V|^2)$, a quantum algorithm that leverages Grover search can solve the shortest path problem in $\mathcal{O}(|V|^{7/4})$ time and $\mathcal{O}(|V|)$ qubits compared to the $\mathcal{O}(|V|^2)$-time classical solution using Dijkstra's algorithm [18]. More recently, a gate-based quantum algorithm was developed to solve the single-source shortest path problem in directed acyclic graphs in $\mathcal{O}(\sqrt{|V||E|} \log |V|)$ steps using $\mathcal{O}(|V|)$ qubits [19]. Notably, both algorithms limit the quantum advantage to cases where the input graph is dense, and they both appear to require the nonnegative edge cost restriction of Dijkstra's algorithm. At present, no quantum gate algorithm has been developed that improves on the classical benchmarks of $\mathcal{O}(|V||E|)$ for general graphs and $\mathcal{O}(|E| + |V| \log |V|)$ for general nonnegatively weighted graphs. And unfortunately, since quantum gate computers have yet not been developed at a scale large enough to handle problems of interest, the algorithms developed for them remain, at least for the time being, a theoretical novelty.

For adiabatic quantum computing, on the other hand, progress has been relatively fast. Last year, D-Wave announced that they will be releasing a 5000-qubit machine [20], so scalability for quantum annealing is already being realized. In terms of algorithms, Lucas laid the groundwork for adiabatic approaches to the shortest path problem by working out Ising formulations for even more complex problems on graphs [21]. Bauckhage *et al.* [22] later incorporated several elements of Lucas's Ising formulation of the traveling salesman problem to sketch an adiabatic approach to the shortest path problem, but they left further details of their approach as a subject for future work. As their intent seems to have been to augment a classical algorithm for finding the length of a shortest path with an adiabatic procedure to find a representation of the path itself, their sketch of the approach includes some redundant elements. Most significantly, it proposes the use of Dijkstra's algorithm as a preprocessing step to find the lengths of the shortest path from a given source. Pakin [23] developed a shortest path algorithm for solving mazes. His algorithm has both space and time complexity of $\mathcal{O}(|V|)$ for the restricted set of unweighted grid graphs.

The main motivation for our research was to develop adiabatic quantum formulations of the shortest path problem that require a minimal amount of classical preconditioning, and, if possible, to solve the problem faster than the current best classical algorithms for the most general classes of input graphs. The first two formulations proposed in this article are robust to directed graphs containing negative edge costs, but for the reasons stated in Section II, their output is only guaranteed to be valid if the input graph is free of cycles with negative total cost. In terms of their classical construction times, our last two proposed approaches are asymptotically faster than the $\mathcal{O}(|V||E|)$ Bellman–Ford algorithm in the general case, and for graphs with nonnegative edge costs and fixed degree, the same two approaches are asymptotically faster than the $\mathcal{O}(|E| + |V| \log |V|)$ implementation of Dijkstra's algorithm.

IV. HOP-BASED APPROACH

We will now discuss a detailed implementation of the formulation proposed by Bauckhage *et al.* To make this implementation more practical, we have introduced a number of changes. The most significant of these is that our implementation dispenses with the need to invoke Dijkstra's algorithm as a classical preprocessing step; i.e., it can find the shortest path without knowing the cost of the path beforehand. It also drops an unnecessary constraint on revisiting vertices in order to make the approach more robust (for details, see Section IV-B).

A. FORMULATION

The first approach begins with the realization that traversing the graph involves passing through a sequence of vertices of the graph from source vertex to terminal vertex. Starting at vertex s, the path proceeds along graph edges, vertex-to-vertex, until it reaches the vertex t. Implicit in this are restrictions and constraints on valid traversals of the graph. Each of these constraints will be captured in the QUBO energy equation. Following the Lucas formulation, we will describe the space of possible solutions as a space of (vertex, position) pairs. That is, we will define a solution path as having vertex i at position p through the path, vertex j at position $p + 1$, and so on. This leads to a few simple constraints. First, we require that any vertex only appear once in a path. Second, there must be a vertex in each position. That is, if the path is three steps long, then positions 1, 2, and 3 must have a vertex each, with no position being skipped. Third, if vertex i is at position p and vertex j is at position $p + 1$, then the traversal incurs a cost equal to the edge cost $c_{i,j}$. If there is no edge (i, j), then traversals between i and j should be penalized.

Let us develop these constraints. First, we will represent a solution as a set of binary variables, each denoted $x_{i,p}$, where the variable $x_{i,p}$ indicates whether vertex i is at position p. Put another way, these variables represent the sequencing of the vertices within the path we want to find. So $x_{1,2} = 1$ means that vertex 1 occurs at position 2 in the path. Writing these binary variables as a matrix $\boldsymbol{X}$ as in (2) enables us to better

understand the constraints

$$\boldsymbol{X} = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,|P|} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,|P|} \\ \vdots & \vdots & \ddots & \vdots \\ x_{|V|,1} & x_{|V|,2} & \cdots & x_{|V|,|P|} \end{bmatrix}. \tag{2}$$

Here, we have identified the number of vertices as $|V|$ and the desired number of hops as $|P|$. Now, we can enumerate the constraints defined previously and specify their contribution to the complete energy equation that will be described as a weighted sum of individual constraint contributions ($H = \sum_k \alpha_k H_k$).

1) Each position must be occupied exactly once in a path. Assume the path visits $|P| \leq |V|$ vertices. Then, we can reformulate the first constraint as the following quadratic penalty function:

$$H_P = \sum_{p=1}^{|P|} \left(1 - \sum_{i=1}^{|V|} x_{i,p} \right)^2. \tag{3}$$

We can identify the innermost sum $\sum_{i=1}^{|V|} x_{i,p}$ as the sum down column p in the matrix $\boldsymbol{X}$ above-mentioned. Any valid solution should have exactly one row in column p equal to 1, to correspond to one vertex at position p. Accordingly, the squared difference of 1 minus this sum will be minimized at 0 when the constraint is met at a single vertex position, and the entire penalty term will be minimized at 0 when the constraint is met at all positions.

2) Each vertex can appear at most one time in a path. This constraint is captured by the following quadratic penalty function:

$$H_V = \sum_{i=1}^{|V|} \sum_{1 \leq p < q \leq |P|} x_{i,p} x_{i,q}. \tag{4}$$

This term is minimized at 0 precisely when each row of $\boldsymbol{X}$ contains at most one 1. This term and the previous one will be minimized together if and only if no 1 in $\boldsymbol{X}$ occurs more than once in the same row or column. The rows that contain a 1 correspond to the vertices included at unique positions in the path, and the rows without a 1 correspond to the vertices excluded from the path.

3) Traversing an edge (i, j) imposes a cost penalty of $c_{i,j}$. This can be easily encoded as the following quadratic function:

$$H_C = \sum_{p=1}^{|P|-1} \sum_{i=1}^{|V|} \sum_{j=1}^{|V|} c_{i,j} x_{i,p} x_{j,p+1}. \tag{5}$$

We can forbid transitions between vertices i and j that have no connecting edge by setting $c_{i,j}$ to a large value (the value of which will be made explicit in Section IV-C).

4) Two additional constraints are implied—that the flow starts at the source vertex and ends at the terminal vertex. These constraints are captured in the following penalty terms:

$$H_s = (1 - x_{s,1})^2 \tag{6}$$

$$H_t = (1 - x_{t,|P|})^2. \tag{7}$$

Valid solutions will satisfy all constraints; therefore, the complete, final energy equation will be a (weighted) sum of the individual constraints energy contributions

$$H = \alpha_P H_P + \alpha_V H_V + \alpha_C H_C + \alpha_s H_s + \alpha_t H_t. \tag{8}$$

Here, we have included scaling terms $\alpha_P, \alpha_V, \ldots$ that allow us to weigh the relative importance of the constraints. Optimal values for the scaling terms are likely problem dependent and are a subject of ongoing research, but we can at least extrapolate explicit choices that are feasible. We want to ensure that any violated constraint has a higher cost than the total cost c_P of the shortest path. Since we do not know c_P *a priori*, a reasonable substitute is any value greater than $c_E = \sum_{(i,j) \in E} |c_{i,j}|$, the sum of all edge costs (taking the absolute value in the case of negative edge costs). This yields the scaling factors $\alpha_C = 1, \alpha_P = \alpha_V = \alpha_s = \alpha_t > c_E$. For succinctness, we will use α to represent the equivalent scaling factors in the latter group.

We note that this is not a true shortest path algorithm. This implementation, as will be demonstrated in Section IX, will produce the lowest-cost path *of the specified length*. Namely, a path that visits exactly $|P|$ vertices will be returned, if one exists, regardless of the existence of lower-cost paths that visit more or fewer vertices.

B. MAXIMUM HOP EXTENSION

Section IV-A described a formulation to identify a shortest path that visits a specific number of vertices. While such a construct may be of interest in some applications, there are other more common instances in which the true lowest-cost path is desired, regardless of how many vertices lie along it. There is a simple extension to the formulation of Section IV-A that overcomes the deficiency of that formulation.

First, recognize that the second constraint captured in (4) is a carry-over from the formulations for the Hamiltonian path problem. In that problem, the requirement is to visit every vertex in the graph once and only once, thus requiring the constraint. For our problem, no such constraint is strictly necessary; in practice, revisiting the same vertex in a path will induce an unnecessary additional cost in the objective function and will, therefore, be rejected. But if we augment the graph with zero-cost "ghost edges" in the form of self-loops (i, i) with $c_{i,i} = 0$ for each vertex $i \in V$, then we can exploit this relaxation to allow the algorithm to find the lowest cost path through the graph that visits *at most* $|P|$ distinct vertices. The returned solution will still be exactly $|P|$ "hops," but multiple hops to the same vertex can be collapsed in the final result.

IEEE Transactions on Quantum Engineering

For example, one solution found by this relaxed procedure may be $(s, 3, 3, 3, 8, t)$, which, when collapsed to a sequence of distinct vertices, coincides with the shortest path solution $(s, 3, 8, t)$; in this case, the number of hops requested was $|P| = 6$ but the lowest-cost path found happened to be one with $|P| = 4$. One consequence of the ghost edges is that the returned solution space can be greatly increased since there are now multiple valid solutions. For instance, $(s, 3, 3, 8, 8, t)$ and $(s, s, s, 3, 8, t)$ represent the same valid shortest path. Postprocessing of the solutions to collapse the path is required, but the cost of such postprocessing is linear in $|P|$. Since all the "ghost edge" solutions will collapse to the same true solution, choosing and collapsing the first is often sufficient. There may be a significant penalty to pay, however, if there is a desire to find *all* shortest paths from s to t in the graph. In that case, all the returned ghost edge solutions would need to be collapsed in order to identify the complete set of true shortest paths.

C. EVALUATION

When evaluating the performance of the quantum annealing algorithms, there are three parameters of interest: the time needed to populate the matrix $\boldsymbol{Q}$, the required number of qubits to represent the problem, and the annealing time required to solve the problem in (1). The annealing time required is dependent on the problem specifics, the scaling factors used, and annealing parameters used by the specific quantum annealing hardware. It is the subject of ongoing research, but we will provide a rudimentary analysis in Section VII. The qubit requirements and fill time estimates will be addressed explicitly in the following sections.

For any graph, directed or undirected, this formulation encodes the shortest path problem using one qubit per vertex per possible position. Since the presumed length of the shortest path is predetermined to be $|P|$, we have a required qubit usage of $|P||V|$. A shortest path cannot be longer than the number of vertices, so this is upper bounded by $|V|^2$ qubits. Note that this bound does not account for requirements for minor embedding and the inefficiencies of specific qubit hardware connectivity which may impose additional, not-insignificant qubit overhead.

Fill time for the construction of the $\boldsymbol{Q}$ matrix includes penalty term contributions from each constraint. First is the calculation of the scaling factor for the constraint terms. This requires one loop through the edges to add up their costs and, therefore, takes $|E|$ computations.

Each of H_s and H_t consists of just one square term $-x_{s,1}^2$ and $-x_{t,|P|}^2$, respectively, after constant terms have been dropped and linear terms have been added into the quadratic terms, so they take one step to write. The quadratic penalty term for H_P consists of $|P||V|(|V|+1)/2$ terms, so adding its terms to the matrix will take $\mathcal{O}(|P||V|^2)$ steps. The constraint corresponding to H_V has been dropped in the maximum hop extension, so we can ignore its contribution. Finally, the objective function term H_C consists of $|P||V|(|V|-1)/2$ terms, as all distinct pairs of vertices (whether an edge connects them or not) induce a traversal cost, so its contribution will be $\mathcal{O}(|P||V|^2)$. Combining the constraint contributions yields a total complexity of $\mathcal{O}(|E| + |P||V|^2)$. The classical postprocessing required to collapse the ghost edges is $\mathcal{O}(|P|)$, so it does not change the overall complexity.

V. DIRECTED EDGE-BASED APPROACH

A. FORMULATION

An alternative approach would be to adapt the objective function and constraints of the shortest path problem's linear programming formulation for directed graphs

$$\begin{aligned} &\text{minimize} \sum_{(i,j)\in E} c_{i,j}x_{i,j} \\ &\text{subject to } 0 \le x_{i,j} \le 1,\ \forall (i,j) \in E \\ &\text{and} \sum_j x_{i,j} - \sum_k x_{k,i} = \begin{cases} 1, & \text{if } i = s \\ -1, & \text{if } i = t \\ 0, & \text{otherwise} \end{cases} \\ &\forall i \in V. \end{aligned} \tag{9}$$

Here, $x_{i,j}$ indicates whether the edge (i, j) is included in the path. The first constraint technically allows these variables to take fractional values between 0 and 1, but because the shortest path linear program happens to be integral, any optimal solution will have all $x_{i,j} \in \{0, 1\}$. The second constraint ensures three things that: 1) the source vertex s has one more edge leaving it than entering it; 2) the terminal vertex t has one more edge entering it than leaving it; 3) every other vertex has as many edges entering it as it has leaving it.

In fact, these constraints define a shortest trail problem rather than a strict shortest path problem, because they admit solutions that visit vertices more than once, but the assumption that the graph contains no negative-cost cycles implies that any shortest trail will in fact be a shortest path. For the same reason, while the above-mentioned constraints on their own do not rule out solutions that consist of a proper path and any number of disjoint cycles, the assertion that no cycles will improve the objective function value should prevent the inclusion of disjoint cycles in solutions. (Of course, zero-cost cycles may occur in otherwise valid solutions, but these can be removed efficiently in classical postprocessing via a depth-first search on the edges in the solution starting at s and ending at t.)

In an undirected graph, where no distinction is made between a given vertex's incoming edges and outgoing edges, enforcing the second constraint is impossible. A trivial solution is to transform the undirected graph into a directed one by turning each of its edges into two directed edges with opposite orientations. The above-mentioned formulation will, then, be applicable to the directed version of the graph, but the directed version of the graph will have twice as many edges as the undirected version, and the summations in the

second constraint will have twice as many terms. A nontrivial approach for undirected graphs is discussed in Section VI.

The objective function and constraints in (9) lend themselves easily to reformulation as a QUBO objective function. The first constraint can be dropped entirely, as the variables of a QUBO function are already restricted to be binary. The separate versions of the second constraint can be rewritten as the following quadratic penalty functions:

$$H_s = \left(\sum_j x_{s,j} - \sum_k x_{k,s} - 1 \right)^2 \tag{10}$$

$$H_t = \left(\sum_j x_{t,j} - \sum_k x_{k,t} + 1 \right)^2 \tag{11}$$

$$H_i = \left(\sum_j x_{i,j} - \sum_k x_{k,i} \right)^2. \tag{12}$$

In total, there will be $|V|$ such penalty functions, $|V| - 2$ of which will take the third form. Both H_s and H_t, when expanded, will include a constant coefficient of 1 and linear coefficients of 2 and -2. Because all $x_{i,j} \in \{0, 1\}$, it follows that $x_{i,j}^2 = x_{i,j}$, and thus, all linear terms can be converted to square terms without changing the objective function's values. After the constant coefficients are dropped, both H_s and H_t have minimum values of -1. The remaining penalty functions for all other vertices will consist entirely of quadratic terms and will have minimum values of 0. Finally, using the same linear-to-quadratic conversion mentioned previously, the original objective function can be rewritten quadratically as

$$H_C = \sum_{(i,j)\in E} c_{i,j} x_{i,j}^2. \tag{13}$$

The objective function is 0 when all $x_{i,j} = 0$ (i.e., when all edges are excluded), and this should only be the minimum value when no path from s to t exists. We, therefore, want to scale the terms in the total energy function

$$H = \alpha_s H_s + \alpha_t H_t + \sum_{i \notin \{s,t\}} \alpha_i H_i + \alpha_C H_C \tag{14}$$

so that any valid assignment of the variables achieves a value less than 0 and any invalid assignment achieves a value greater than 0. If we knew the cost c_P of the shortest path up front, then we could scale H_C by $\alpha_C = 1$ and all of the penalty terms by any $\alpha > c_P$. This would ensure that the energy corresponding to the shortest path would be $-c_P < 0$, while any infeasible solution would have a higher energy regardless of its total edge cost. Of course, since we do not know the shortest path length ahead of time, we will need to use an appropriate surrogate. The simplest choice is $\alpha > c_E$ as defined in the previous section, since $c_E \geq c_P$, with equality holding when the graph is itself a path.

B. EVALUATION

For sparse graphs, this formulation is considerably more economical with respect to qubit usage, as it encodes the shortest path problem in $|E|$ qubits. If, however, the input graph is undirected, then the encoded problem will need $2|E|$ qubits to represent a directed version of the graph.

The first step in the construction of the $\boldsymbol{Q}$ matrix is the calculation of the scaling factor for the constraint terms. This requires one loop through the edges to add up their costs and, therefore, takes $|E|$ steps. The constraint term associated with vertex i is a quadratic function of $\deg(i)$ variables, so adding its terms to the matrix takes $\deg(i)(\deg(i) + 1)/2$ steps. Finally, updating the matrix with the coefficients of the H_C term takes $|E|$ steps. The total number of steps needed to construct the $\boldsymbol{Q}$ matrix is, therefore

$$\begin{aligned} &2|E| + \sum_{i=1}^{|V|} \frac{\deg(i)(\deg(i)+1)}{2} \\ &\quad = 2|E| + \frac{1}{2}\left(\sum_{i=1}^{|V|} \deg(i)^2 + \sum_{i=1}^{|V|} \deg(i) \right) \\ &\quad = 3|E| + \frac{1}{2}\left(\sum_{i=1}^{|V|} \deg(i)^2 \right) \end{aligned} \tag{15}$$

where the last line follows from the degree sum formula. This quantity is lower and upper bounded by $(3 + \delta)|E|$ and $(3 + \Delta)|E|$, respectively. When the original graph is undirected and has $|E|$ edges, minimum degree δ, and maximum degree Δ, then the upper and lower bounds become $(6 + 2\delta)|E|$ and $(6 + 2\Delta)|E|$, respectively.

VI. UNDIRECTED EDGE-BASED APPROACH

A. FORMULATION

One way to avoid doubling the number of qubits required when the previous approach is applied to an undirected graph is to modify the constraints in a way that is agnostic to edge orientation. Consider the following constraint satisfaction problem:

$$\begin{aligned} &\text{minimize} \sum_{(i,j)\in E} c_{i,j} x_{i,j} \\ &\text{subject to } x_{i,j} \in \{0,1\},\ \forall (i,j) \in E \\ &\text{and} \sum_j x_{i,j} = \begin{cases} 1, & \text{if } i \in \{s,t\} \\ 2, & \text{if } i \notin \{s,t\} \text{ and } i \in P \\ 0, & \text{if } i \notin \{s,t\} \text{ and } i \notin P \end{cases} \\ &\quad \forall i \in V. \end{aligned} \tag{16}$$

This formulation replaces flow-based constraints with cardinality-based constraints. In a valid path, each of vertices s and t will be incident to one edge in the path, every other vertex on the path will be incident to two edges in the path,

IEEE Transactions on Quantum Engineering

and every vertex not on the path will be incident to zero edges in the path.

These constraints can be rewritten as quadratic penalty terms even simpler than their counterparts in the directed formulation. All we have to do is include $|V|$ additional variables $x_1, x_2, \ldots$, where x_i indicates whether or not vertex i is included in the path. So to start with s and t, we have

$$H_s = -x_s^2 + \left(x_s - \sum_j x_{s,j}\right)^2 \tag{17}$$

$$H_t = -x_t^2 + \left(x_t - \sum_i x_{i,t}\right)^2. \tag{18}$$

The square term at the beginning of each function enforces the constraint that vertices s and t must be included in the solution. The sum on the right-hand side ensures that when the vertex in question is included in the solution, it is incident to exactly one edge. Each function is minimized at -1 when both constraints are met.

The penalty functions for the remaining vertices are even simpler

$$H_i = \left(2x_i - \sum_j x_{i,j}\right)^2. \tag{19}$$

If i is not in the path, then $x_i = 0$, and this function is minimized at 0 when no edges incident to i are included in the path. If i is in the path, then the function is minimized at 0 when exactly two edges incident to i are included.

The objective function can be written as (13). Since all of the penalty functions have the same minimum values as they do in the directed formulation, the same scaling factors for the penalty terms and the objective function term can be used in this formulation.

B. EVALUATION

For an undirected graph, this formulation encodes the shortest path problem using one qubit for each edge and one qubit for each vertex, for a total of $|V| + |E|$ qubits. For any undirected graph with $|E| > |V|$ (a class encompassing most graphs of interest), this is a more economical use of qubits than the approach of converting the undirected graph to a directed one and using the previous approach.

As in the directed formulation, the first step in constructing the $\boldsymbol{Q}$ matrix is to calculate the scaling factor for the constraint penalty terms, which is a sum involving $|E|$ terms. The constraint penalty term for each i is a quadratic function in $\deg(i) + 1$ variables, so it will have $(\deg(i) + 1)(\deg(i) + 2)/2$ terms to add to the matrix. As before, updating the matrix with the coefficients of the H_C term takes $|E|$ steps. The total number of steps needed to construct the $\boldsymbol{Q}$ matrix is, therefore, bounded above by

$$\begin{aligned} &2|E| + \sum_{i=1}^{|V|} \frac{(\deg(i)+1)(\deg(i)+2)}{2} \\ &= 2|E| + \sum_{i=1}^{|V|} \frac{\deg(i)^2 + 3\deg(i) + 2}{2} \\ &= 5|E| + |V| + \frac{1}{2}\sum_{i=1}^{|V|} \deg(i)^2 \end{aligned} \tag{20}$$

which, in turn, is upper bounded by

$$(5 + \Delta)|E| + |V|. \tag{21}$$

If $|E| > |V|$, then this is slightly faster than the construction of the $\boldsymbol{Q}$ matrix for the directed formulation using the directed version of the graph.

VII. ANNEALING TIME

Now that we have established bounds on the time required to prepare a QUBO formulation of the shortest path problem, we will turn to the question of the complexity of the quantum annealing process. According to the adiabatic theorem [24], by slowly evolving a system from an initial state to a final state corresponding to the problem we want to solve, we can track the ground state of the system, even as it changes to the minimum-energy eigenstate of the Hamiltonian describing our problem. For the purpose of this analysis, we will assume that for an n-qubit system, the initial Hamiltonian H_0 describing the energy at the start of the evolution is the Hamiltonian whose minimum-energy eigenstate is $|+\rangle^{\otimes n}$. The final Hamiltonian H_1 consists of the one- and two-qubit interactions encoded in our QUBO matrix $\boldsymbol{Q}$, and its ground state corresponds to the solution we want to find. Starting at time $s = 0$ and proceeding to $s = 1$, the energy of the evolving system will be described by a time-dependent Hamiltonian $H(s)$ that interpolates between the initial and final Hamiltonians

$$H(s) = (1 - s)H_0 + sH_1. \tag{22}$$

A generally accepted rule of thumb estimates the minimum time τ required to track the ground state as

$$\tau \gg \int_0^1 \frac{||\frac{d}{ds}H(s)||}{\gamma(s)^2} ds \tag{23}$$

where $\gamma = \min_{s\in[0,1]} E_1 - E_0$ describes the minimum spectral gap (i.e., the difference between the eigenvalues or energies of the ground state and first excited state) of $H(s)$ over the course of the evolution [25], [26]. The latter parameter γ is a major problem-dependent factor in the runtime of quantum annealing. If both $||\frac{d}{ds}H(s)||$ and γ are measured in Joules, then τ will be have units in inverse Joules; this can be converted to seconds by multiplying it by the reduced Planck constant $\hbar \approx 1.054571817\text{x}10^{-34}$Js (which is commonly divided out from the derivation of τ for simplicity).

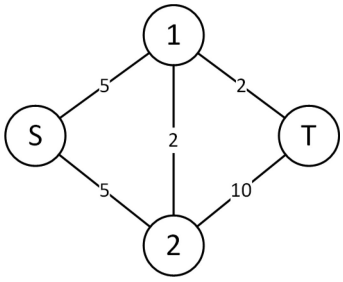

FIGURE 2. Example undirected graph with specified edge costs for numerical example.

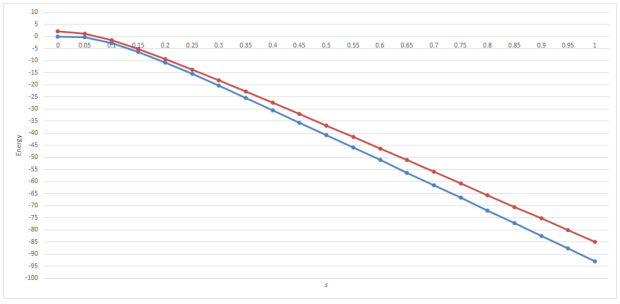

FIGURE 3. Plot of the two smallest eigenvalues of $H(s)$ for the hop-based formulation of the graph in Fig. 2.

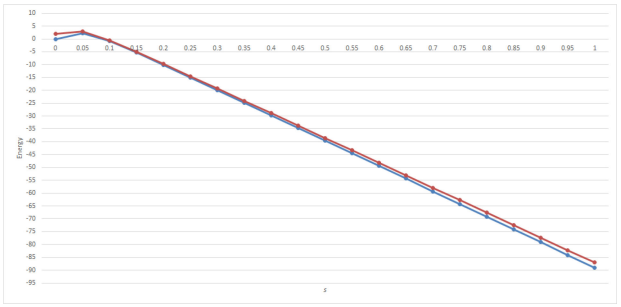

FIGURE 4. Plot of the two smallest eigenvalues of $H(s)$ for the directed edge-based formulation of the graph in Fig. 2.

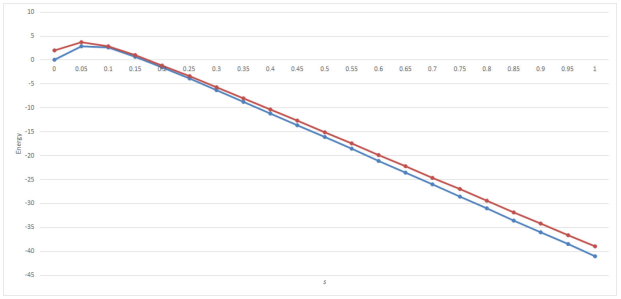

FIGURE 5. Plot of the two smallest eigenvalues of $H(s)$ for the undirected edge-based formulation of the graph in Fig. 2.

We can estimate the value of γ for each of the formulations previously detailed by calculating the eigenvalues of the corresponding time-dependent Hamiltonian $H(s)$ over a discrete set of values for s. Using the trapezoid rule to approximate the integral in (23), we can estimate reasonable lower bounds for τ. Because $H(s)$ is a linear interpolation between H_0 and H_1, we have $||\frac{d}{ds}H(s)|| = ||H_1 - H_0||$, and since this is a constant term, it can be factored out of the integral to simplify things even further.

Because a $2^n \times 2^n$ matrix is needed to represent an n-qubit Hamiltonian explicitly, it is computationally infeasible to approximate these values for problems even of moderate size. Thus, for our purposes, a small example problem will have to suffice. We have chosen the four-vertex, five-edge undirected graph in Fig. 2, as it is amenable to all three quantum annealing formulations we have described and does not require too many qubits to be represented in any of them.

For the hop-based formulation, we have $|P| = 3$ and a total number of required qubits $|P||V| = 12$. Calculating and sorting the $2^{12} = 4096$ eigenvalues of $H(s)$ at 21 discrete points in time gives us the plot of eigenvalues shown in Fig. 3. The minimum spectral gap observed among these eigenvalues is $\gamma = 1.225$ J, and the corresponding lower bound for the annealing time is $\tau = 1229.694\ \text{J}^{-1}$.

For the directed edge-based formulation, the total number of qubits is $|E| = 10$ (after transforming each undirected edge into two directed edges). This savings in qubit usage seems to incur a cost in performance: the eigenvalues become significantly closer, as shown in Fig. 4. We observe a much smaller minimum spectral gap $\gamma = 0.283$ J, and a much larger lower bound for the annealing time at $\tau = 25810.324\ \text{J}^{-1}$.

Finally, for the undirected edge-based formulation, the total number of qubits is reduced to $|V| + |E| = 9$, but the spectral gap appears to shrink further, as shown in Fig. 5. The minimum spectral gap in this case is $\gamma = 0.243$ J; however, the lower bound for the annealing decreases to $\tau = 9904.343\ \text{J}^{-1}$.

The increase in the expected annealing time of the edge-based formulations relative to the hop-based formulation seems to be due in part to the size of the penalty term scaling factor α. For the hop-based formulation, if the anticipated number of vertices visited (and, therefore, the number of edges in the path) is small, then the penalty scaling factor will likely be significantly less than the sum of all edge costs. If we use the scaling factor for the hop-based formulation in the edge-based formulations, the values of γ and τ parameters for these formulations do improve, but they still remain significantly worse than the parameters for the hop-based formulation. For this reason, we suspect that some factor inherent to the structure of a QUBO formulation gives rise to a tradeoff between qubit complexity and annealing complexity.

Unfortunately, because it takes exponential space to construct a Hamiltonian matrix corresponding to a shortest path formulation, we cannot provide the estimates of γ and τ for larger problem instances needed to extrapolate how these parameters change with increasing problem size. Theoretically, applying the annealing process on a D-Wave machine for

IEEE Transactions on Quantum Engineering

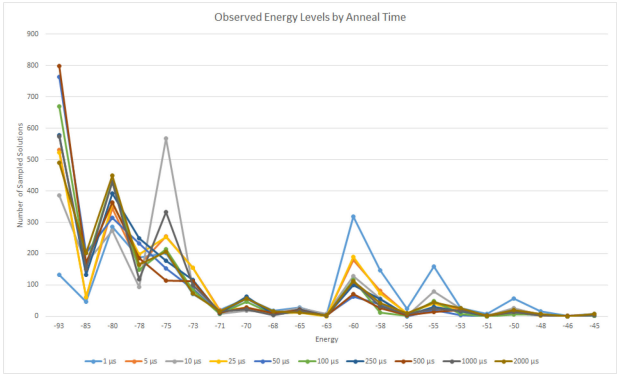

FIGURE 6. Histograms of measured energies (rescaled to match the corresponding eigenvalues of the unnormalized Q matrix) sampled over 2000 reads for annealing durations between 1 μs and 2000 μs. D-Wave postprocessing (which, by default, involves a classical search for local improvements to the best-found solutions) was disabled. The tail of the histogram has been cut off at −45 for the sake of space. Peaks at suboptimal energy levels are the result of a multiplicity of eigenstates corresponding to those eigenvalues.

increasing durations and then sampling solutions repeatedly over many reads (i.e., trials) should produce different histograms of measured energies, with longer annealing times yielding distributions that favor the ground-state energy. We ran this test on the hop-based formulation (which has the shortest annealing time requirement of all three formulations) of the problem in Fig. 2 on a D-Wave 2000Q. We generated the $\boldsymbol{Q}$ matrix for this formulation, normalized it so that its maximum entry had an absolute value of 1 (as required by the D-Wave), and produced histograms of the energies of sampled solutions from 2000 reads for annealing durations ranging from 1 μs to 2000 μs. The results are displayed in Fig. 6. The more uniform distribution of energies among sampled solutions for the 1 μs anneal time is to be expected, but it is surprising that the distribution does not consistently improve as the anneal time increases. For instance, the distribution after 500 μs of annealing favors the minimum-energy solution more than the corresponding distributions associated with 1000 μs and 2000 μs of annealing. The success of the annealing process even with relatively short anneal times is likely due to the D-Wave's strategy of starting in a random state rather than in the same $|+\rangle^{\otimes n}$ state for each read. An additional possibility is that the distribution of states improves more as a step function of the anneal time, with the distribution improving dramatically only after a certain threshold; however, because the D-Wave 2000Q caps annealing time at 2000 μs, we were unable to evaluate this hypothesis.

VIII. COMPARISON

At this point, let us turn to a comparison of the algorithms. As we have discussed in Section III, the best classical algorithm exhibits a worst-case complexity of $\mathcal{O}(|E| + |V| \log |V|)$ on general directed or undirected graphs. At present, known quantum universal gate algorithms only offer an improvement for specific classes of graphs. For the adiabatic algorithms described herein, the number of classical steps required to prepare the $\boldsymbol{Q}$ matrix is $\mathcal{O}(|E|)$ on general, directed graphs. For the purposes of this comparison, we have assumed the annealing time is constant or $\mathcal{O}(1)$. For simplicity in representation, Table 1 replaces the size syntax $|V|$, $|E|$, and $|P|$ (the vertex count, edge count, and maximum number of vertices in the desired path, respectively) with V, E, and P to reduce the noise in the table.

Beyond the computational complexity, the defining limitation of the usefulness of the quantum algorithms is the number of qubits required. Given that the quantum computers of today are limited to very low qubit counts, it is illustrative to quantify the requirements. The three algorithms described in this article have minimum qubit requirements, as shown in Table 2. Note that, this is a direct evaluation of the minimum number of logical qubits required to represent the problem. Depending on the specifics of the quantum annealing computer used, the "embedding" of the logical qubits (i.e., the mapping of logical qubits to physical qubits) may result in dramatically more required qubits. This is especially prevalent for dense, highly-coupled problems wherein the entanglement between physical qubits requires chaining multiple qubits in order to entangle two logical qubits.

Based on Table 2, we can get a rough approximation of the problem size solvable on today's quantum annealing computer hardware. At present the largest machine available has 2000 qubits with 5000 qubits soon to be available [20]. Assuming no embedding overhead, that would mean the edge-based, directed graph algorithm could support finding the shortest path through a graph with hundreds or even thousands of edges, depending on the average degree of the vertices. As an example of real-world sizing, if we assume each vertex is an intersection and each edge is a road segment, then we could compute the shortest path route through a small city. By way of example, the town of Blacksburg, Virginia, has roughly 2000 road segments.

IX. EXAMPLES

In order to simplify the verification and implementation of the formulations presented in this article, examples of the $\boldsymbol{Q}$ matrices for these formulations will be instructive. We will use the simple "toy" problem instance presented in the graph of Fig. 2. Here, we have four vertices with a desired path from s to t. This example is small enough that it is trivial to identify the solution $(s, 1, t)$ with a cost of 7).

Each of the algorithms will be demonstrated in the following sections.

A. HOP-BASED EXAMPLE

To demonstrate the algorithm as described in Section IV, we will compute the QUBO formulation for the graph of Fig. 2. If we assume or know that the path is $|P| = 3$ hops long, then we can generate a 12×12 matrix whose rows and columns correspond to each vertex at each possible position.

While not strictly necessary, setting the scaling factors for the penalty terms added into the $\boldsymbol{Q}$ matrix relies on the

TABLE 1. Comparison of Algorithm Complexities. for the Annealing Approaches, the (Classical) Time and Space Complexities are for the Construction of the Q Matrix

	Algorithm	(Classical) Time Complexity	(Classical) Space Complexity	Graph Class
Classical				
	Bellman-Ford	$\mathcal{O}(VE)$	$\mathcal{O}(V)$	general, $C \subset \mathbb{R}$
	Dijkstra	$\mathcal{O}(V^2)$	$\mathcal{O}(V)$	general, $C \subset \mathbb{R}_{\geq 0}$
	Dijkstra with min-priority queue	$\mathcal{O}((V+E)\log V)$	$\mathcal{O}(V)$	general, $C \subset \mathbb{R}_{\geq 0}$
	Dijkstra with Fibonacci heap	$\mathcal{O}(E+V\log V)$	$\mathcal{O}(V)$	general, $C \subset \mathbb{R}_{\geq 0}$
Quantum gate				
	Khadiev & Safina	$\mathcal{O}(\sqrt{VE}\log V)$	$\mathcal{O}(V)$	directed acyclic, $C \subset \mathbb{R}_{\geq 0}$
	Heiligman	$\mathcal{O}(V^{7/4}))$	$\mathcal{O}(V)$	complete, $C \subset \mathbb{R}_{\geq 0}$
Quantum annealing				
	Hop-based	$\mathcal{O}(E+PV^2)$	$\mathcal{O}(E+PV^2)$	general, $C \subset \mathbb{R}$
	Directed edge-based	$\mathcal{O}(\Delta E)$	$\mathcal{O}(\Delta E)$	directed, $C \subset \mathbb{R}$
	Undirected edge-based	$\mathcal{O}(\Delta E+V)$	$\mathcal{O}(\Delta E+V)$	undirected, $C \subset \mathbb{R}_{\geq 0}$

TABLE 2. Comparison of Qubit Requirements and Estimated Performance Parameters for Quantum Annealing Formulations in This Article

Algorithm	Qubits	Experimental γ (J)	Experimental τ (J^{-1})	Graph Class
Hop-based	PV	1.225	1229.694	general, $C \subset \mathbb{R}$
Directed edge-based	E	0.283	25810.324	directed, $C \subset \mathbb{R}$
Undirected edge-based	$V+E$	0.243	9904.343	undirected, $C \subset \mathbb{R}_{\geq 0}$

TABLE 3. Q Matrix for the Hop-Based Shortest Path Formulation of the Graph in Fig. 2

		vertex s			vertex 1			vertex 2			vertex t		
		1	2	3	1	2	3	1	2	3	1	2	3
vertex s	1	-2α	α	0	α	5	0	α	5	0	α	α	0
	2	0	$-\alpha$	α	5	α	5	5	α	5	α	α	α
	3	0	0	$-\alpha$	0	5	α	0	5	α	0	α	α
vertex 1	1	0	0	0	$-\alpha$	α	0	α	2	0	α	2	0
	2	0	0	0	0	$-\alpha$	α	2	α	2	2	α	2
	3	0	0	0	0	0	$-\alpha$	0	2	α	0	2	α
vertex 2	1	0	0	0	0	0	0	$-\alpha$	α	0	α	10	0
	2	0	0	0	0	0	0	0	$-\alpha$	α	10	α	10
	3	0	0	0	0	0	0	0	0	$-\alpha$	0	10	α
vertex t	1	0	0	0	0	0	0	0	0	0	$-\alpha$	α	0
	2	0	0	0	0	0	0	0	0	0	0	$-\alpha$	α
	3	0	0	0	0	0	0	0	0	0	0	0	-2α

TABLE 4. Q Matrix for the Directed Edge-Based Shortest Path Formulation of the Graph in Fig. 2

	$x_{s,1}$	$x_{s,2}$	$x_{1,s}$	$x_{1,2}$	$x_{1,t}$	$x_{2,s}$	$x_{2,1}$	$x_{2,t}$	$x_{t,1}$	$x_{t,2}$
$x_{s,1}$	5	2α	-4α	-2α	-2α	-2α	2α	0	2α	0
$x_{s,2}$	0	5	-2α	2α	0	-4α	-2α	-2α	0	2α
$x_{1,s}$	0	0	$4\alpha+5$	2α	2α	2α	-2α	0	-2α	0
$x_{1,2}$	0	0	0	$2\alpha+2$	2α	-2α	-4α	-2α	-2α	2α
$x_{1,t}$	0	0	0	0	2	0	-2α	2α	-4α	-2α
$x_{2,s}$	0	0	0	0	0	$4\alpha+5$	2α	2α	0	-2α
$x_{2,1}$	0	0	0	0	0	0	$2\alpha+2$	2α	2α	-2α
$x_{2,t}$	0	0	0	0	0	0	0	10	-2α	-4α
$x_{t,1}$	0	0	0	0	0	0	0	0	$4\alpha+2$	2α
$x_{t,2}$	0	0	0	0	0	0	0	0	0	$4\alpha+10$

cumulative sum of the edge costs. For this problem, $c_E = \sum_{(i,j)\in E} |c_{i,j}| = 24$. Furthermore, the penalty for no connections between vertices needs to be "large." We will use the same value c_E for this numerical example. Using the quadratic coefficients from (8), we get the $\boldsymbol{Q}$ matrix shown in Table 3.

B. DIRECTED EDGE-BASED EXAMPLE

For the directed edge-based formulation described in Section V, we must convert the undirected graph in Fig. 2 to a directed graph with $|E| = 10$ edges. The $\boldsymbol{Q}$ matrix for this formulation will, therefore, be a 10×10 matrix where each row and column corresponds to a directed edge. Using a scaling factor of 1 for the objective function term H_C and a scaling factor of $c_E = \sum_{(i,j)\in E} |c_{i,j}| = 48$ for the $|V| = 4$ penalty terms corresponding to flow constraints, we can populate the $\boldsymbol{Q}$ matrix so that it represents the example graph. The resulting matrix is shown in Table 4.

C. UNDIRECTED EDGE-BASED EXAMPLE

For the undirected edge-based formulation described in Section VI, the $\boldsymbol{Q}$ matrix is a 7×7 matrix with a row and

IEEE Transactions on Quantum Engineering

TABLE 5. Q **Matrix for the Undirected Edge-Based Shortest Path Formulation of the Graph in Fig. 2**

	x_s	x_1	x_2	x_t	$x_{s,1}$	$x_{s,2}$	$x_{1,2}$	$x_{1,t}$	$x_{2,t}$
x_s	0	0	0	0	-2α	-2α	0	0	0
x_1	0	4α	0	0	-2α	0	-2α	-2α	0
x_2	0	0	4α	0	0	-2α	-2α	0	-2α
x_t	0	0	0	0	0	0	0	-2α	-2α
$x_{s,1}$	0	0	0	0	$2\alpha+5$	2α	2α	2α	0
$x_{s,2}$	0	0	0	0	0	$2\alpha+5$	2α	0	2α
$x_{1,2}$	0	0	0	0	0	0	$2\alpha+2$	2α	2α
$x_{1,t}$	0	0	0	0	0	0	0	$2\alpha+2$	2α
$x_{2,t}$	0	0	0	0	0	0	0	0	$2\alpha+10$

column for each of the $|E| = 5$ undirected edges and an auxiliary row and column for each vertex $i \notin \{s, t\}$. Using a scaling factor of 1 for the objective function term H_C and a scaling factor of $c_E = \sum_{(i,j)\in E} |c_{i,j}| = 24$ for the penalty terms corresponding to the $|V| = 4$ cardinality constraints, we can populate the $\boldsymbol{Q}$ matrix so that it represents the example graph. The resulting matrix is shown in Table 5.

X. CONCLUSION

We presented three different approaches for solving the shortest path problem on general directed and undirected graphs using a QUBO formulation suitable to exploitation on a quantum annealing machine. This formulation lets the algorithm run on currently available quantum annealing hardware, including that of D-Wave systems. Notional resource requirements and complexity orders were developed for each algorithm, allowing for direct comparison with existing classical algorithms. We also provided estimates for the minimal spectral gap and annealing time required to track the ground state of the problem Hamiltonian with high probability.

All three of the approaches outlined in this article accept as input any simple graph (i.e., one that does not contain self-loops or multiple edges between the same source and destination vertices). The two edge-based approaches could easily be adapted to work with multigraphs, but for the purposes of finding a shortest path, it is more efficient to remove self-loops and long redundant edges in preprocessing, since such edges will never appear in the shortest path. The only other restrictions are on the orientation of the graph: the hop-based formulation works with both directed and undirected graphs; the directed edge-based formulation is designed for directed graphs, but can work with undirected graphs, as long as they are first converted to directed ones; and the undirected edge-based formulation works exclusively for undirected graphs. The first two approaches will work with directed graphs that contain negative edge costs, but the results will only be valid if the input graph contains no negative-cost cycles.

After filling in the gaps left in the proposal of Bauckhage *et al.* [22], we have shown that a practical adiabatic implementation of their hop-based formulation offers little improvement over the classical approach. The time required to construct the $\boldsymbol{Q}$ matrix for the problem, which also serves as an upper bound for the size of $\boldsymbol{Q}$ in terms of its nonzero entries, scales as $\mathcal{O}(|E| + |P||V|^2)$, which is potentially competitive with the Bellman–Ford algorithm, but asymptotically worse than the slowest implementation of Dijkstra's algorithm.

To remedy this deficiency, we have proposed two other QUBO formulations of the shortest path problem that adapt flow and cardinality constraints from linear programming and constraint satisfaction formulations of the shortest path problem in a straightforward manner. Both formulations use qubits to represent vertices and edges rather than (vertex, position) pairs and are, therefore, more efficient than the hop-based formulation for sparse graphs. For the flow-based formulation for directed graphs, the construction time for the $\boldsymbol{Q}$ matrix scales as $\mathcal{O}(\Delta|E|)$, and for the cardinality-based formulation for undirected graphs, it scales as $\mathcal{O}(\Delta|E| + |V|)$. Notably, if the graph degree remains fixed, then both of these approaches are asymptotically faster than the fastest-known implementation of Dijkstra's algorithm. We expect the proposed approaches to demonstrate the greatest advantage overknown classical algorithms in directed graphs with negative edge costs and nonplanar graphs of low degree. Real-world inputs like road intersection networks with overpasses and tunnels fit nicely into the latter paradigm.

There are a number of directions for future work. The main bottleneck in the matrix construction for all of the proposed algorithms is the step of adding the quadratically many terms of the constraint penalty functions to the $\boldsymbol{Q}$ matrix. It is not clear to us how the number of terms that need to be written might be intelligently reduced, but we cannot rule out the possibility, either. On the same note, we welcome any improvements to the scaling factors used for the constraint penalty terms in our formulations, as $c_E = \sum_{(i,j)\in E} |c_{i,j}|$ seems unnecessarily large, and better scaling factors might yield a larger minimum spectral gap and an improved annealing time.

REFERENCES

[1] T. Kadowaki and H. Nishimori, "Quantum annealing in the transverse Ising model," *Phys. Rev. E*, vol. 58, no. 5, 1998, Art. no. 5355, doi: 10.1103/PhysRevE.58.5355.

[2] J. Brooke, D. Bitko, T. F. Rosenbaum, and G. Aeppli, "Quantum annealing of a disordered magnet," *Sci.*, vol. 284, no. 5415, pp. 779–781, 1999, doi: 10.1126/science.284.5415.779.

[3] A. B. Finnila, M. Gomez, C. Sebenik, C. Stenson, and J. D. Doll, "Quantum annealing: A new method for minimizing multidimensional functions," *Chem. Phys. Lett.*, vol. 219, no. 5–6, pp. 343–348, 1994, doi: 10.1016/0009-2614(94)00117-0.

[4] Z. Bian, F. Chudak, W. G. Macready, and G. Rose, "The Ising model: Teaching an old problem new tricks," 2010. [Online]. Available: https://www.dwavesys.com/sites/default/files/weightedmaxsat_v2.pdf, Accessed: Apr. 14, 2020.

[5] "Reformulating a problem." [Online]. Available: https://docs.dwavesys.com/docs/latest/c_handbook_3.html, Accessed: Apr. 6, 2020.
[6] L. R. Ford, "Network flow theory," RAND Corp., Santa Monica, CA, USA, Tech. Rep. P-923, 1956.
[7] R. Bellman, "On a routing problem," *Quart. Appl. Math.*, vol. 16, no. 1, pp. 87–90, 1958, doi: 10.1090/qam/102435.
[8] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numer. Math.*, vol. 1, no. 1, pp. 269–271, 1959, doi: 10.1007/BF01386390.
[9] D. B. Johnson, "Efficient algorithms for shortest paths in sparse networks," *J. ACM*, vol. 24, no. 1, pp. 1–13, 1977, doi: 10.1145/321992.321993.
[10] M. L. Fredman and R. E. Tarjan, "Fibonacci heaps and their uses in improved network optimization algorithms," *J. ACM*, vol. 34, no. 3, pp. 596–615, 1987, doi: 10.1145/28869.28874.
[11] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. Cambridge, MA, USA: MIT Press, 2009.
[12] G. N. Federickson, "Fast algorithms for shortest paths in planar graphs, with applications," *SIAM J. Comput.*, vol. 16, no. 6, pp. 1004–1022, 1987, doi: 10.1137/0216064.
[13] M. R. Henzinger, P. Klein, S. Rao, and S. Subramanian, "Faster shortest-path algorithms for planar graphs," *J. Comput. Syst. Sci.*, vol. 55, no. 1, pp. 3–23, 1997, doi: 10.1006/jcss.1997.1493.
[14] P. van Emde Boas, R. Kaas, and E. Zijlstra, "Design and implementation of an efficient priority queue," *Math. Syst. Theory*, vol. 10, no. 1, pp. 99–127, 1976, doi: 10.1007/BF01683268.
[15] R. K. Ahuja, K. Mehlhorn, J. Orlin, and R. E. Tarjan, "Faster algorithms for the shortest path problem," *J. ACM*, vol. 37, no. 2, pp. 213–223, 1990, doi: 10.1145/77600.77615.
[16] M. Thorup, "Integer priority queues with decrease key in constant time and the single source shortest paths problem," *J. Comput. Syst. Sci.*, vol. 69, no. 3, pp. 330–353, 2004, doi: 10.1016/j.jcss.2004.04.003.
[17] M. Thorup, "Undirected single-source shortest paths with positive integer weights in linear time," *J. ACM*, vol. 46, no. 3, pp. 362–394, 1999, doi: 10.1145/316542.316548.
[18] M. Heiligman, "Quantum algorithms for lowest weight paths and spanning trees in complete graphs," 2003, *arXiv:quant-ph/0303131*.
[19] K. Khadiev and L. Safina, "Quantum algorithm for shortest path search in directed acyclic graph," *Moscow Univ. Comput. Math. Cybern.*, vol. 43, no. 1, pp. 47–51, 2019, doi: 10.3103/S0278641919010023.
[20] "D-Wave previews next-generation quantum computing platform," [Online]. Available: https://www.dwavesys.com/press-releases/d-wave-previews-next-generation%-quantum-computing-platform, Accessed: Apr. 6, 2020.
[21] A. Lucas, "Ising formulations of many NP problems," *Front. Phys.*, vol. 2, 2014, doi: 10.3389/fphy.2014.00005.
[22] C. Bauckhage, E. Brito, K. Cvejoski, and C. Ojeda, "Towards shortest paths via adiabatic quantum computing," in *Proc. Mining Learn. Graphs*, London, U.K., 2018.
[23] S. Pakin, "Navigating a maze using a quantum annealer," in *Proc. 2nd Int. Workshop Post Moores Era Supercomput.*, 2017, pp. 30–36, doi: 10.1145/3149526.3149532.
[24] M. Born and V. Fock, "Beweis des adiabatensatzes," *Zeitschrift Physik*, vol. 51, pp. 165–180, 1928, doi: 10.1007/BF01343193.
[25] A. M. Childs, E. Farhi, and J. Preskill, "Robustness of adiabatic quantum computation," *Phys. Rev. A*, vol. 65, doi: 10.1103/PhysRevA.65.012322.
[26] B. W. Reichardt, "The quantum adiabatic optimization algorithm and local minima," in *Proc. 36th Annu. ACM Symp. Theory Comput.*, 2004, pp. 502–510, doi: 10.1145/1007352.1007428.

Thomas Krauss received the B.S. and M.S. degrees from Michigan State University, East Lansing, MI, USA, in 1987 and 1989, respectively, both in physics.

He is currently a Senior Research Associate with the Ted and Karyn Hume Center for National Security and Technology, Virginia Polytechnic Institute and State University, Blacksburg, VA, USA. He joined the Hume Center in August 2015, bringing a wealth of experience with electromagnetic modeling, signal propagation, waveform analysis, software development, and high-performance computing. He has extensive experience with high-reliability, high-performance, and quantum computing, as well as machine learning and network modeling and simulation.

Joey McCollum received the bachelor's degree in mathematics and creative writing (double major) from the University of Maine at Farmington, Farmington, ME, USA, in 2011.

He is currently a Research Associate with the Electronic Systems Lab (ESL), Ted and Karyn Hume Center for National Security and Technology, Virginia Polytechnic Institute and State University, Blacksburg, VA, USA. He joined ESL in 2018, and since then, he has worked primarily on projects related to signals analysis and wireless networks. Prior to his employment at the Hume Center, he worked for seven years in industry and government settings.