Quantum Graph Pursuit: Analysis of the Advantages and Challenges of a Quantum Dynamic Combinatorial Optimization Model from a Software Developer Perspective

Simone Reale and Elisabetta Di Nitto
DEIB, Politecnico di Milano
Milano, Italy
{simone1.reale, elisabetta.dinitto}@polimi.it

***Abstract*—In this work, we experiment with quantum annealing and the D-Wave suite by developing a new application we call Quantum Graph Pursuit (QGP), which is a dynamic combinatorial optimization problem. We delve into two models' formulations and their implementation on the D-Wave system, relying both on the hybrid and fully quantum settings. The final objective of our work is twofold: to solve the initial problem, finding the right trade-off between expressiveness and complexity, and to obtain and share general indications on the formulation and implementation process themselves.**

***Index Terms*—Quantum Annealing, Combinatorial Optimization**

I. Introduction

Nowadays, the need for computing power is more prominent than ever; classical computing is approaching its physical limits, and without a paradigm shift, it seems impossible to face the new challenges proposed by business and the scientific community. Quantum computing promises to offer a dramatic speed-up that will likely revolutionize how we set about entire classes of problems in various fields. Quantum Annealing represents an exciting alternative to the predominant Quantum circuit model for solving specific classes of problems, in particular, optimization ones.

With the goal of demonstrating the current level of maturity of the Quantum Annealing implementation offered by D-Wave[1], in this paper we focus on a novel dynamic combinatorial optimization problem, which we call Quantum Graph Pursuit (QGP). We define two alternative mathematical models for this problem, we show how they can be implemented and executed on the D-Wave platform, and demonstrate that we can find feasible and, in some cases optimal, solutions for meaningful sizes of this problem. We describe our solutions as accurately as possible to ensure replicability and try to furnish practitioners with insightful and generalizable recommendations for developing effective implementations in the future.

Consistently with this objective, the paper is structured as follows: Section II presents the background on quantum annealing and D-Wave, Section III presents QGP, Section IV describes the two models we have defined to solve the problem, Section V provides details about their implementation on D-Wave, Section VI presents the results of our experiments, Section VII introduces our general findings, Section VIII presents the related work and Section IX concludes the paper.

[1]https://www.dwavesys.com/

II. Background

Adiabatic Quantum Computing (AQC) [1] is a form of universal quantum computation that leverages the principles of quantum mechanics to solve computational problems in a unique way, differently from both classical computation and the standard quantum gate model. It is based on the time-dependent Schrodinger equation:

$$i\hbar\frac{\partial}{\partial t}\Psi(\mathbf{r},t) = H(t)\Psi(\mathbf{r},t)$$

that regulates the evolution of the state of a quantum system. In order to use AQC, we need a problem Hamiltonian H_P, which encodes in its ground state the solution of our problem, and a starting Hamiltonian H_S with a known ground state that is easy to obtain. Moreover, we need to specify an adiabatic evolution path $p(t)$ as a function, dependent on time, having as constraints $p(t_0) = 1$ and $p(t_{end}) = 0$. Combining all the elements we obtain:

$$H(t) = p(t)H_S + (1 - p(t))H_P$$

We then let the system evolve according to our adiabatic evolution path, and the adiabatic quantum theorem [2] guarantees that the final quantum state will be the ground state of our problem Hamiltonian, giving us the solution.

The relationship between AQC and the standard quantum gate computation model is explored in [3] and in [4]. They show that it is possible to efficiently simulate any given quantum circuit as long as the adiabatic evolution theorem is respected, with the final result of a polynomial equivalence between the two computation models.

At the time of writing, the current quantum hardware cannot yet show a complete AQC behavior. A specific instantiation of AQC is, instead, offered by D-Wave: Quantum Annealing

(QA) is a specialized metaheuristic technique that loses the universality property of AQC and restricts its focus to optimization problems only. At the time of writing, D-Wave is the leading developer and manufacturer of quantum annealing technology, as well as the only company that offers end users a platform to gain access to a quantum annealer. Its current flagship machine is the Advantage Quantum computing system version 6.4; it is characterized by more than 5000 qubits with a number of couplers per qubit up to 15, reaching more than 35,000 couplers in total.

From the theoretical viewpoint, the paper by Kadowaki et al. [5] can be seen as the foundation of the Quantum Annealing technique. It has been expanded by the work by Farhi et al. [6] that established its theoretical basis within the broader scope of Quantum Computing.

We will look at two primary ways to describe QA optimization models according to D-Wave nomenclature: an unconstrained one through BQMs (Binary Quadratic Models) and a constrained one through CQMs (Constrained Quadratic Models).

A. BQM

In order to formulate a BQM, the user defines an objective function to be minimized; each solution of the problem is encoded in an energy level of the quantum system, and finding the lowest energy state corresponds to finding the optimal solution. BQM forks into the Ising model, originating from statistical physics, and the QUBO model, familiar to practitioners in the field of machine learning and quantum computing; these are the two possible alternatives used to encode a problem directly on a D-Wave processing unit.

These models are crafted to discover an optimal assignment for the binary variables $\mathbf{x} = (x_1...x_n)$ that minimizes the following objective function:

$$\sum_{i<j} Q_{ij} x_i x_j + \sum_i h_i x_i$$

given $Q_{ij}, h_i \in \mathbb{R}$, we call the resulting model QUBO if $x_i \in \{0, 1\}$ and Ising if $x_i \in \{-1, 1\}$, in the last case the variables are commonly called *spin variables*.

The equivalence between the QUBO model and the Ising model arises through a simple linear transformation, and the relationship is achieved by mapping binary variables in the QUBO model to spin variables in the Ising model. Consequently, the objective functions of the QUBO and Ising models become equivalent, and optimization problems formulated in one model can be translated into the other without altering the nature of the problem or its solutions.

In contrast to traditional optimization formulations, the defining characteristic of QUBO and Ising resides in their unconstrained nature; this implies that all the constraints necessary to describe the problem are incorporated within their objective function as penalties. The solution space can be seen as the set of all possible binary assignments that lead to optimal or near-optimal solutions according to the problem's defined objective, and therefore, the size of the solution space grows exponentially with the number of binary variables, making its NP-Hard nature explicit [7].

B. CQM

According to the definition given by D-Wave, Constrained Quadratic Models (CQM) are problems formulated as:

$$\min \sum_i a_i x_i + \sum_{i \leq j} b_{i,j} x_i x_j + c$$

subject to one or more constraints of the form:

$$\sum_i a_i x_i + \sum_{i \leq j} b_{i,j} x_i x_j + c \bigstar 0$$

where, in our case, the variables x_i for $i \in \{1...N\}$ are binary, a_i, $b_{i,j}$, c are real values, $\bigstar \in \{\leq, \geq, =\}$, and the range of the quadratic-term summation for binary variables is $i < j$ because for $\{0, 1\}$ binary values we have $x^2 = x$ and for $\{-1, 1\}$ spin variables $s^2 = 1$. The fact that CQMs present explicit constraints means that we can define a feasible solution space with definite boundaries more precisely than with the unconstrained models. This is very beneficial to complex optimization problems with hard constraints to be satisfied. CQMs can be transformed into an unconstrained model by translating constraints into penalties. The instantiation of this process to our problem is described in Section V.

C. D-Wave and the hybrid suite of instruments

Our interest concerns both hybrid and fully quantum solutions. In the first case, it is possible to use either the Dwave-hybrid framework [8] to develop a personalized hybrid workflow or use the preexisting solutions contained in the Leap hybrid solver service offered by D-Wave.

In the context of D-Wave, a personalized hybrid workflow is nothing but a computational process that integrates classical algorithms and quantum annealing in parallel branches to solve optimization problems; it is possible to use preexisting components from their portfolio or develop new ones.

The hybrid approach essentially uses quantum annealing as an acceleration tool on top of classical optimization algorithms; the input problem is, in fact, sent to one or more classical solvers that work in parallel, and during the execution, they can send to their quantum counterparts queries containing sub-problems small enough to be directly embedded on quantum architectures. The fully quantum approach, instead, relies solely on the annealer to solve the entire problem; this approach directly maps the problem onto the quantum hardware and uses just quantum annealing to find the solution.

There are some major differences between the two approaches. The most evident one regards scalability: fully quantum approaches are limited by the size and connectivity of the specific quantum annealer's architecture, while the hybrid approach utilizes classical computing to manage and decompose problems, making it possible to go far beyond what the annealer could tackle alone.

Another difference regards the class of inputs that can be managed by the two approaches in the context of the D-Wave platform; the hybrid one can, in fact, solve both constrained

and unconstrained optimization models natively, while the fully quantum can only deal with unconstrained ones.

III. The Quantum Graph Pursuit Problem: Overview

The Quantum Graph Pursuit (QGP) problem is a dynamic combinatorial optimization problem set on a square lattice graph; this setting serves as a stage for two fundamental types of entities: one or more *Preys* and a *Catcher*. Preys represent dynamic targets moving across the graph according to predetermined trajectories; the Catcher is tasked with navigating the square lattice graph to capture the Preys, of which the Catcher knows the trajectory.

An essential part of the problem is that the cost associated with moving along an edge depends not only on the specific edge but also on the specific time in which that edge is followed.

The goal of QGP is to compute the Catcher's path that ensures the interception of the Preys and minimizes the cost associated with the Catcher's movements on the graph. The cost metric can encapsulate factors such as distance traveled, time taken, or energy expended, reflecting multiple aspects of real-world problems. We explored different possibilities regarding the graph's topology; however, we resorted to framing the problem within the borders of a square lattice to abstract complex navigation and interception tasks into a manageable yet generalizable model. To generate prey paths, we used either a "Random" policy – that picks a random alternative from the set of valid outgoing edges for each time step – or an "Unvisited nodes" policy that prioritizes the discovery of nodes not yet visited to create less trivial paths.

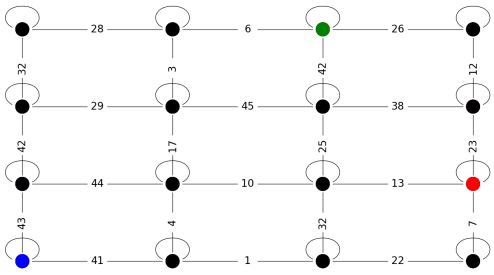


Fig. 1: An instance of the square lattice graph on which the QGP problem is set.

In Figure 1 we can see a frame of an animation produced to visualize the model and its setting. It represents an example of a 4x4 square lattice graph with the Catcher depicted in red and the two Preys, respectively, in blue and green. The labels on the edges represent the cost of following that edge in a particular discrete time instant; in this representation, the cost of self-loops is not depicted.

Possible practical applications of QGP concern planning dynamic paths in discrete contexts and navigating a dynamic environment where obstacles or goals move over time. Such problems are interesting especially in the robotics field; some examples of the algorithms being studied and developed in this area can be found in [9] and [10]. The solutions to QGP we propose in this paper constitute a promising alternative.

T	The set of discrete time instants indexed as natural numbers
V	The set of nodes in the square lattice graph
P	The set of Preys indexed as natural numbers starting from 0
E	The set of valid edges (i, j) with i,j ∈ V

TABLE I: Sets used in the formulation.

To solve QGP exploiting quantum annealing we went through the following steps: *i.* we modeled it as a Constrained Quadratic Model (in particular, we explored two different formulations showing different pros and cons); *ii.* we transformed the previously obtained models in QUBO formulations; and *iii.* we ran such formulations on D-Wave experimenting with both hybrid and fully quantum approaches.

IV. Mathematical Formulation

In this section we present two possible formulations for the QGP problem. The first one is the most expressive, and it leans toward hybrid solutions on D-Wave; the second one focuses on compactness at the cost of being less powerful. Both are based on binary-valued variables and rely on the following assumptions:

- No diagonal movements are possible;
- Catcher and Preys move at the same constant speed along graph edges, and this speed is precisely one action per discrete time instant;
- The Catcher must start in a node that is not occupied by a Prey (otherwise we would have a trivial capture).

Moreover, they rely on the definitions of sets summarized in Table I. As for the set of nodes V in the square lattice graph, we assume that the nodes are indexed starting from 0, which is assigned to the node on the first row and first column, and proceeding sequentially from left to right and top to bottom.

A. First Model: "Advanced"

We model the problem considering two aspects, the movement of the Catcher through the graph, and relative cost, and the movement and capture of Preys.

1) Catcher movement: The Catcher movement is represented through the binary decision variables $x_{i,j,t}$ with $(i,j) \in E$, which taken individually indicate whether the Catcher moves from node i to node j at time t. The initial movement is indicated as in Equation 1, where s_1 is the initial node and s_2 is one of its adjacent nodes. This grounds the model in a defined starting point.

$$x_{s_1,s_2,0} = 1 \qquad (1)$$
$$\text{s.t.} \quad (s_1, s_2) \in E$$

Equation 2 ensures that exactly one action (movement or stay) is chosen at each time step, encapsulating the limitation of performing a single action within a discrete time frame.

$$\sum_{i \in V} \sum_{j \in V} x_{i,j,t} = 1 \quad \forall t \in T \qquad (2)$$

$$\text{s.t.} \quad (i,j) \in E$$

Equation 3 guarantees logical progression in movements, with the current location directly reachable from the previous step, thereby embedding spatial coherence into the model.

$$x_{i,j,t} - \sum_{k \in V} x_{k,i,t-1} \leq 0 \quad \forall i,j \in V \quad \forall t \in T - \{0\} \tag{3}$$
$$\text{s.t.} \quad (i,j) \in E \quad (k,i) \in E$$

The cost for a Catcher to move from node i to node j at time t is represented as $\alpha(t)(c_{i,j,t} + \delta_{i,j,t})$, where:

- $\mathbf{c_{i,j,t}}$ represents the cost associated with moving from node i to node j at time t, it introduces an element of decision-making based on cost-efficiency, steering the model towards finding the least expensive path to achieve its objectives. Figure 5 is presented as a visual example.
- $\delta_{\mathbf{i,j,t}}$ is a noise value sampled from a probability distribution that models the variability of the edge cost and adds a layer to the cost computation that considers dynamic elements that could influence action costs.
- $\alpha(\mathbf{t})$: is a function with a discrete domain (t ∈ T) that serves as a Time-Dependent cost modifier. It is used to model temporal variations in costs that depend only on the specific time instant and not on the specific edge, so it can depict periods in which it is inherently costly to perform an action.

$\alpha(t)$ can be defined in various ways. It can be a trivial linearly increasing or decreasing function:

$$\alpha(t) = \alpha_0 \pm kt$$

where α_0 is the initial value and k is the rate of increase or decrease. In such manner, we could represent simple situations in which delaying actions would linearly increase or decrease costs as time goes on. Another alternative possible is to describe $\alpha(t)$ as a step function:

$$\alpha(t) = \begin{cases} \alpha_1 & if \quad t \leq T_1 \\ \alpha_2 & if \quad T_1 < t \leq T_2 \\ \dots & \\ \alpha_n & if \quad t > T_{n-1} \end{cases}$$

Consequently, we can explicitly model the inherent cost of specific intervals of time in our domain, which is useful when we know a priori that our operational context changes distinctly at known times.

2) Preys movement and capture: Preys are elements $p \in P$. Their movement is represented as a set of binary constant values of the form $\mathbf{y_{p,i,j,t}}$, taken individually they indicate that "Prey" p has moved from node i to node j at time t. We use the binary control variable $\mathbf{z_{p,t}}$ to indicate whether a Prey indexed by p has been intercepted by the Catcher at time t. Moreover, we use the binary variable $\mathbf{w_{p,t}}$ to model the permanence of p in the capture status over time, after it is captured at a time lower or equal than t. Based on these definitions, Equation 4 models the capture, that is, the interaction between the Catcher's path and the Preys' locations, with denoting the presence of Prey signifying an actual interception event. This is the heart of the model, capturing the core objective of interception.

$$\sum_{i_1 \in V} \sum_{i_2 \in V} \sum_{j \in V} x_{i_1,j,t} y_{p,i_2,j,t} = z_{p,t} \quad \forall p \in P \quad \forall t \in T \tag{4}$$
$$\text{s.t.} \quad (i_1,j) \in E \quad (i_2,j) \in E$$

The constraint, representing the main goal of the model, that all Preys must be intercepted at least once is modeled by Equation 5, while Equations 6 and 7 ensure that once a "Prey" is captured, it is considered captured for all subsequent times.

$$\sum_{t \in T} z_{p,t} \geq 1 \quad \forall p \in P \tag{5}$$

$$w_{p,t} \geq z_{p,t} \quad \forall p \in P \quad \forall t \in T - \{0\} \tag{6}$$

$$w_{p,t} \geq w_{p,t-1} \quad \forall p \in P \quad \forall t \in T - \{0\} \tag{7}$$

To enable the possibility to enforce a specific ordering in the capture process, we introduce $\beta_p(t)$ that represents a collection of functions (each Prey has one) with discrete domain (t ∈ T) used in conjunction with $w_{p,t}$ to reward the capture of a particular Prey in a predetermined period of time. The term in its entirety incentivizes the model to follow a user-defined order of captures. For what concerns the practical implementation, we have used a series of step functions for $\beta_p(t)$ to encode the value of the capture of a specific Prey in a determined time frame. Usually, in the simulations we cared about enforcing the order of only the first Prey p_1 to be catched and to give it precedence on the others we have set the values of $\beta_p(t)$ equal to 0 $\forall t \in T$ and $\forall p \in P - \{p_1\}$.

To improve the efficiency of the model, we introduce Equation 8 that links the individual capture statuses to a global control variable u_t, which ceases cost accumulation once all Preys have been intercepted by or before time t.

$$\sum_{p} w_{p,t} \geq |P| * (1 - u_t) \quad \forall t \in T \tag{8}$$

3) Objective Function: The core of our optimization model is its objective function, which is designed to capture the dynamics of the Catcher-Prey interaction within our square lattice graph.
The function is formulated as follows:

$$\min \sum_{t \in T} \sum_{i \in V} \sum_{j \in V} \alpha(t)(c_{i,j,t} + \delta_{i,j,t}) x_{i,j,t} u_t - \sum_{p \in P} \beta_p(t) w_{p,t} \tag{9}$$
$$\text{s.t.} \quad (i,j) \in E \quad \text{and} \quad (1) - (8)$$

This objective function is constructed to balance the operational costs of movements within the graph, given by the first set of summations against the strategic benefits of intercepting "Preys" over a discrete time horizon, given by the last summation. Of course, the objective function is a constrained one as all Equations presented before must hold.

B. Second Model: "Simple"

In the second model we are characterizing the Catcher and the Preys in terms of their successive positions on the graph instead of explicitly modeling their movements with variables $x_{i,j,t}$ and constants $y_{p,i,j,t}$.

For the sake of space, we introduce this formulation focusing mainly on the differences from the previous one. The definitions for sets $\mathbf{T}$, $\mathbf{V}$, $\mathbf{P}$, $\mathbf{E}$ and the time-dependent costs $\mathbf{c_{i,j,t}}$ associated to each arc hold here as well.

1) Catcher movement: $\mathbf{s_{i,t}}$ is a state variable that indicates the presence of the catcher at node i at time t. $\mathbf{s_1}$ defines the starting node for the Catcher and is given as a parameter to the model. Given this, we have that

$$s_{s_1,0} = 1 \tag{10}$$

This sets the starting position of the Catcher at node s_1.

Constraint 11 ensures that the Catcher occupies only one node at each time step.

$$\sum_{i \in V} s_{i,t} = 1 \quad \forall t \in T \tag{11}$$

Equation 12 enforces that if the Catcher is at node j at time t, it must have been at a node i at time $t-1$ such that $(j,i) \in E$. It effectively models the possibility of the Catcher moving only along valid edges on the graph.

$$s_{j,t} - \sum_{(i,j) \in E} s_{i,t-1} \leq 0 \quad \forall j \in V \quad \forall t \in T - \{0\} \tag{12}$$

2) Preys movement and capture: $\mathbf{y_{p,i,t}}$ is a parameter that represents the fact that the Prey p is at node i at time t. The capture process is modeled by checking that each Prey and the Catcher have been at least once in the same node at the same time instant (see Equation 13.

$$\sum_{t \in T} s_{i,t} y_{p,i,t} \geq 1 \quad \forall p \in P \tag{13}$$

3) Objective Function:

$$\min \sum_{t \in T - \{0\}} \sum_{i \in V} \sum_{j \in V} \alpha(t) c_{i,j,t} s_{i,t-1} s_{j,t} \tag{14}$$

$$\text{s.t.} \quad (i,j) \in E \quad \text{and} \quad (10) - (13)$$

The most evident difference between Objective Function 9 and 14 is, on the one side, the compactness of the second one (this point will be substantiated in Section IV-C) and, on the other side, the lack in Objective Function 14 of a mechanism to cease cost accumulation after capturing all the Preys, i.e., after reaching the goal of the model. This has a profound effect on the optimization process; in fact, the cost cessation mechanism in Objective Function 9 is implemented to prevent the model from continuing the optimization process after reaching the capture of all preys.

In Objective Function 14 this same mechanism could not be introduced because otherwise the model would have become cubic (we used the product $s_{i,t-1}s_{j,t}$ to implicitly model movement, so we had no room left to add an additional multiplicative factor u_t without increasing the degree of the formula). A possible solution to this issue would be to use state variables $s_{i,t}$ to infer additional movement variables $x_{i,j,t}$, but this would negate the gain in compactness given by using state variables in the first place. Despite this difference, which in any case can be leveled through a wise use of the T parameter, it is possible to use interchangeably the two models in the same contexts.

C. Numerical Interlude

In order to better contextualize the two models, we present a more in-depth analysis of some of their less-discussed details from a quantitative point of view.

1) On the number of variables of the two models: In both the two models, the action is set on a square lattice graph composed of $|V|$ nodes; in the case of the first one, the $x_{i,j,t}$ variables to be optimized are related to possible movements along valid edges, so are proportional to their number.

It is easy to demonstrate that in a square lattice graph, there are $3|V| - 2\sqrt{|V|}$ edges in the presence of self-loops and $2|V| - 2\sqrt{|V|}$ in their absence, but since we have a dynamic problem, we also have to account for the time dimension that adds a factor $|T|$. In addition to that, we also have to account for $w_{p,t}$ and $z_{p,t}$ variables that are combined $2 * |P| * |T|$ and for the u_t variables that are $|T|$.

As regards the second model, the state variables $s_{i,t}$ are simply related to the number of nodes $|V|$ with a factor $|T|$ given by the time dimension. The difference in the count of variables as a function of the number $|V|$ of nodes in the graph can be trivially calculated as $(|V| - 2\sqrt{|V|} + 2 * |P| + 1) * |T|$ without self-loops. It is evident how the second model outclasses the first in terms of overall compactness.

2) On the time parameter: One interesting problem to address is the definition of a heuristic to estimate T_{max}: the size of the time frame given to the Catcher to fulfill its mission of exploring the graph and catching all the Preys. This problem depends on the topology of the graph and the relative position of the Preys and the Catcher at the start. By increasing the time available it is possible to proportionally increase the dimension of the solution space and, possibly, find better results, but at the same, the complexity of the model proportionally scales up. Recalling the simplifying assumptions made in our models and described in IV, one simple and cautious approach would consider the maximum distance possible on the square lattice graph multiplied by the number of Preys:

$$T_{max} = 2(\sqrt{N} - 1) * |P| \tag{15}$$

where N is the total number of nodes in the graph, in this case we are sure to have a big enough solution space, but the complexity of the model is greatly affected. An alternative could be:

$$T_{max} = (\lceil D_{avg} + T_{adjust} \rceil) * |P| \tag{16}$$

Where D_{avg} is calculated as the mean of the Manhattan distances from the Catcher starting position to each Prey starting position and T_{adjust} is a time adjustment factor that

accounts for the dynamic aspects of Prey movement and the Catcher's need to recalibrate its path as Preys move. This factor can be derived empirically based on the specific initial configuration and graph topology. We have found that a good starting point is to use a fraction of D_{avg} adjusted following the outcomes of the simulations.

V. Implementing the QGP models on D-Wave

To implement our formulations, we used an open-source suite of Python tools offered directly by D-Wave and accessible through the Ocean software development kit [11]. for the sake of reproducibility we report below the necessary details of our implementation.

Our constrained models can be easily defined through the `ConstrainedQuadraticModel` class in the `dimod` package [12] of the SDK. After being constructed, the CQMs objects are ready to be executed with a hybrid approach.

As anticipated in Section II-B, the choice of CQMs is based on the fact that they allow for separating the objective function from the constraints it is subject to, simplifying the problem formulation and improving model clarity.

The drawback is that constrained models are not natively compatible with fully quantum setups, and they need to be transformed into either QUBO or Ising models to be embedded in quantum processing units.

This transformation involves converting all constraints into penalty terms added to the objective function. The difficulty of this conversion lies in assigning appropriate penalty weights that are large enough to enforce constraint satisfaction without overwhelming the original objective function.

Given a binary combinatorial optimization problem:

$$min\ x^T Cx \quad s.t. \quad Ax = b \quad \mathbf{x} \in \{0,1\}^n$$

where C is a real diagonal matrix, A is a matrix defining the constraints with real components, and b is a vector of real values. After choosing a scalar weight term P, it is possible to obtain an unconstrained formulation algorithmically by following these mathematical steps [13]:

$$\begin{aligned} & x^T Cx + P(Ax-b)^T(Ax-b) \\ =\ & x^T Cx + x' Dx + c \\ =\ & x^T Qx + c \end{aligned}$$

To perform such transformation, we relyed on the `cqm_to_bqm` function made available by D-Wave and contained in the `dimod` package. The function offers a Lagrange multiplier parameter that encodes the strength used when translating the corresponding constraint.

Before executing a BQM on a QPU, there is a further step to be executed, called *minor embedding* [14]. This technique is used to map a problem onto the physical hardware of a quantum annealer, even if the problem doesn't match the QPU architecture. The architecture of the QPU, in this case, simply refers to the topology of its lattice of interconnected qubits. In the case of the architecture behind the Advantage system 6.4 used in this paper, the connectivity of the qubits is not all-to-all; instead, they are interconnected in a specific pattern known as Pegasus topology. Whenever we formulate a computational problem for a quantum annealer, it can have an alternative representation as a graph in which nodes represent variables and edges represent interactions between variables. The problem graph often requires more connectivity than the hardware graph can provide; in this situation, a chain must be created, a set of coupled physical qubits used to represent a single logic qubit that can accommodate the connectivity needed by the corresponding problem variable. Long qubit chains are a symptom of bad implementation and can lead to weak performances; the Ocean SDK offers a convenient heuristic tool for minor embedding called `minorminer` [15] that tries to find embeddings while creating qubit chains of minimal length. In Figure 2, we can see the proportions of the

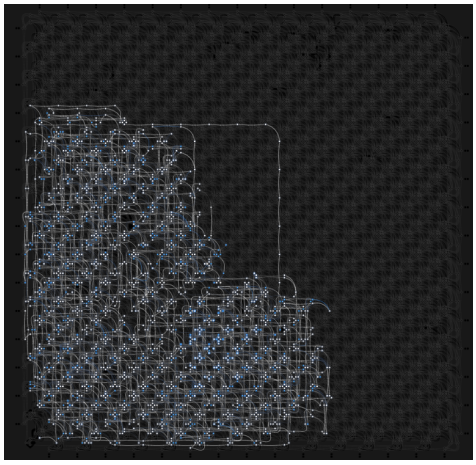


Fig. 2: An example of embedding of the second model proposed on an D-Wave Advantage 6.4 system.

occupied area on the QPU of the embedding of our second model on the Advantage 6.4 system, operated through the `minorminer` tool, in the case of an instance characterized by $|V| = 16$, $|T| = 10$, and $|P| = 2$. This representation was obtained with the help of the D-Wave problem inspector component [16].

VI. Experiments and analysis of results

A. Goals of the experiments

The main objective of the experiments is to get an understanding of D-Wave potential to implement a non-trivial problem as QGP. We focus both on the hybrid and the fully quantum setup.

In the first one we assess the feasibility and optimality properties of the samples obtained with graphs of 36, 49, 64, and 81 nodes, we contextualize the total run time needed for each problem size and observe the total QPU access time to quantify how efficiently the hybrid system can leverage quantum resources. We could treat even larger problems and tested up to 225 nodes, but we could not collect statistically relevant results due to time limits and limitations in the free access to the resources.

In the fully quantum setup we could run only the simple model and we focused on assessing the maximum model size in terms of nodes embeddable on our reference quantum system (Advantage system 6.4), as well as the feasibility and optimality properties of the samples obtained with the minimum non-trivial problem size (9 nodes).

B. Experimental setup

All experiments have been run on the D-Wave hardware. In the hybrid setup (Section VI-C), we used the `hybrid_constrained_quadratic_model` solver version 1.12.

The experiments in the fully quantum context (see Section VI-D) have been performed on the Advantage_system 6.4 QPU. In order to run our tests, we used the default parameters represented in table II; in addition, we used the default Lagrange multiplier parameter set to 10x, the biggest bias in the model for the conversion from CQM to BQM, the chain strength parameter set to 1000, and the number of reads parameter set to 100.

In all cases, the function used to determine the time parameter is always the most conservative one (Equation 15), and the number of Preys has been fixed to 2 unless otherwise specified; their path has been generated according to the "Unvisited nodes" policy that tries to create non-trivial paths by maximizing the total number of different nodes visited. In all the experiments, we considered self-loops in our formulations, and we used the $[5, 15]$ continuous range to assign the $c_{i,j,t}$ cost variables. Since our primary goals were to assess the feasibility and then the optimality of the solutions, we do not take into account the influence of the $\alpha(t)$ and $\beta_p(t)$ functions, which add expressiveness to the models but do not directly contribute to achieving valid solutions.

Parameter	Value
System Name	Advantage_system6.4
Number of Qubits	5760
Supported Problem Types	Ising, QUBO
Topology	Pegasus (16)
Annealing Time Range (μs)	0.5 to 2000
Default Annealing Time (μs)	20
Default Programming Thermalization (μs)	1000
Default Readout Thermalization (μs)	0
Max Anneal Schedule Points	12
Number of Reads Range	1 to 10000
Per Qubit Coupling Range	-18 to 15
Problem Run Duration Range (μs)	0 to 1000000
Programming Thermalization Range (μs)	0 to 10000
Readout Thermalization Range (μs)	0 to 10000
H Range	-4 to 4
H Gain Schedule Range	-4 to 4
J Range	-1 to 1
Extended J Range	-2 to 1

TABLE II: Specifications of the Advantage_system6.4 Quantum Annealer used in the context of our experiments.

C. Analysis in a hybrid setup

The feasibility of samples is a fundamental component of our study due to the hard nature of the constraints in the two models. In Figure 3, we have calculated the mean feasibility between the samples of a single iteration obtained as the ratio between the number of feasible samples read and the total number, and then we have taken the mean of means over 15 iterations. Predictably, our metric has a downward trajectory as the number of nodes increases, but for both models, it remains above 10% even in the worst case, meaning that in the context of our test frame, the two models are always able to provide a set of multiple feasible solutions.

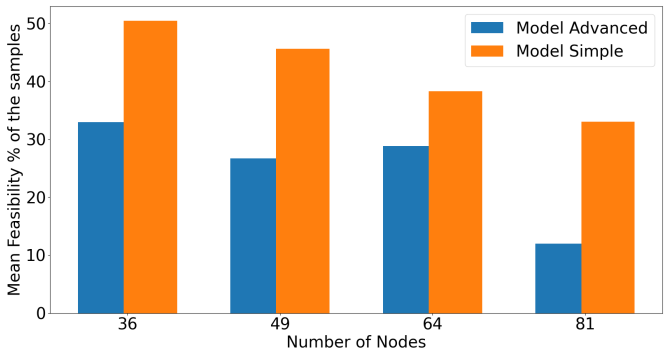


Fig. 3: Mean percentage of feasible samples of the two models, hybrid setup.

We then consider the quality of the results obtained with the maximum run time parameter of the sampling process set to the minimum required by the specific instance size of the problem, calculated with the help of the `min_time_limit` function offered by the hybrid sampler itself. In Figure 6, we can observe the percentage difference between the best solution found in a particular iteration and the global best, and we have a filled colored circle if the difference is zero, meaning that we have found the optimal solution. As it is clear from the figure, the model Simple has a maximum percentage error of 6% obtained in the second iteration with 81 nodes, while the model Advanced performed worse with a maximum difference of 17.5% in the sixth iteration with 36 nodes.

Furthermore, in Figure 4, we have the mean percentage difference between all feasible solutions and the global best solution for each of the different problem sizes considered. These results demonstrate the ability of the two models not only to return a single near-optimal solution but also to return a set composed of multiple solutions of comparable quality.

Figure 7 helps us to contextualize these results in terms of the total run time needed: the influence of a higher number of variables is evident in the trajectory of the average execution time of the Advanced Model that increases with the number of nodes even within our small test frame, on the other hand, the model Simple seems to be unbothered by this increase. The explanation is related to the `time_limit` parameter calculated with the `min_time_limit` function that accounts for the model's overall size. Since the Simple model grows in size much slower than the Advanced model, its `time_limit` parameter also grows slower, and so does its execution time. Figure 8, instead, focuses on the total QPU access time; it is easy to notice how the Advanced model finds it increasingly

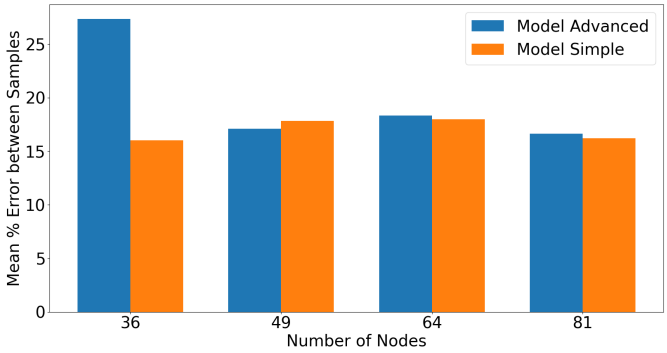


Fig. 4: Mean percentage difference between the feasible samples obtained and the ground state of the two models, hybrid setup.

difficult to make usage of the quantum processing unit as the number of nodes increases. This is due to the fact that as the model's size increases, it becomes more and more challenging to decompose it into bits that can be sent for execution to the QPU. Instead, as attested by the QPU access time, the Simple model continues to be able to leverage the quantum resources also in the largest case considered.

D. Analysis on a fully quantum approach

For what concerns the fully quantum approach, we focused our attention on the Simple model. We have been able to embed a problem instance characterized by $|V| = 16$, $|T| = 10$, and $|P| = 2$, but we obtained significant results in terms of feasibility and optimality only for instances up to $|V| = 9$, $|T| = 4$, and $|P| = 1$. In Figure 9b, we can observe the percentage of valid and invalid results obtained by the model over 30 iterations and also the percentage of optimal solutions found; as we can see, the model found a feasible solution 86% of the time and 33% of the time found an optimal solution. Furthermore, in Figure 9a, we can see how, not surprisingly, the percentage difference between the best solution at each iteration and the global best solution is higher on average than what we have obtained with the hybrid setup in Figure 6. The fully quantum approach had worse results than the hybrid while dealing with much smaller problem instances.

VII. Discussion

The Implementation process of our models provided us with several key insights. The most important one is related to the impact of a strategic model formulation on the efficiency and effectiveness of quantum computing solutions. By strategic model formulation, we mean the process of searching for the best abstractions and simplifications that can enhance the compatibility of a high-level model with quantum hardware without compromising the problem's core requirements and goals. In our case, for example, this meant searching for the best mathematical representation for each high-level constraint or finding the best graph topology to set the problem on. This search was driven by the practical goal of minimizing the overall number of variables and the number of interactions between them while keeping the expressiveness of the model unchanged. The comparison of the results of the Advanced and Simple models also highlights how there is a delicate balance between the expressiveness of a model and its computational efficiency on quantum hardware, both in terms of the feasibility of the solutions and their optimality, unconstrained formulations on fully quantum setup are just not ready for complex problems characterized by many hard constraints. On the other hand, hybrid setups by leveraging classical computing for problem decomposition, preprocessing, and integration of solutions seem to be a promising field of research that is entirely worth investigating. Furthermore, at the moment, finding optimal penalty weights to convert hard constraints thus enabling the conversion from CQM to BQM, is more of an art than a science; there are some studies like [17] or [18], but there is no general-purpose consolidated technique.

VIII. Related Work

The demonstration of QA potential in solving actual problems has been certified by Johnson et al. [19], giving way to the prolific research line related to the development of models and methods to solve increasingly complex problems with QA. In this context, despite its focus on the standard gate model, the work by Gemeinhardt et al. [20] represents a good starting point for describing the state of the art in our specific topic of interest, which is quantum combinatorial optimization. The progress in hardware technology made it possible to expand the scope of tractable problems; an example is Neukart et al. [21], who tried to solve the complex problem of traffic flow optimization, and Venturelli et al. [22] who approached portfolio optimization problems. From a theoretical standpoint, Glover et al. [13] detailed methods and best practices for dealing with the formulation of problems in the QUBO format, enabling further developments in the representation of complex problems on quantum hardware. Another notable work is that of Lucas [23], who provided the Ising formulation for many NP problems, contributing to extending the suite of valid examples available to practitioners.

It is evident that our specific field of interest, Quantum Annealing models and techniques applied to dynamic combinatorial optimization problems, is still in its infancy, while QA applications to static optimization problems are well-documented. Our work fits into this spot, trying to make the best of all the progress made in terms of quantum hardware to solve a novel dynamic combinatorial optimization problem and, by doing that, demonstrate the potential of the QA approach also in this context.

IX. Conclusion

The exploration of the possibilities given by Quantum Annealing in the context of our specific dynamic combinatorial optimization problem, as presented in our study, has led not only to the development of two distinct working models — Advanced and Simple — each showcasing different advantages in terms of expressiveness and efficiency, but also to some generalizable knowledge.

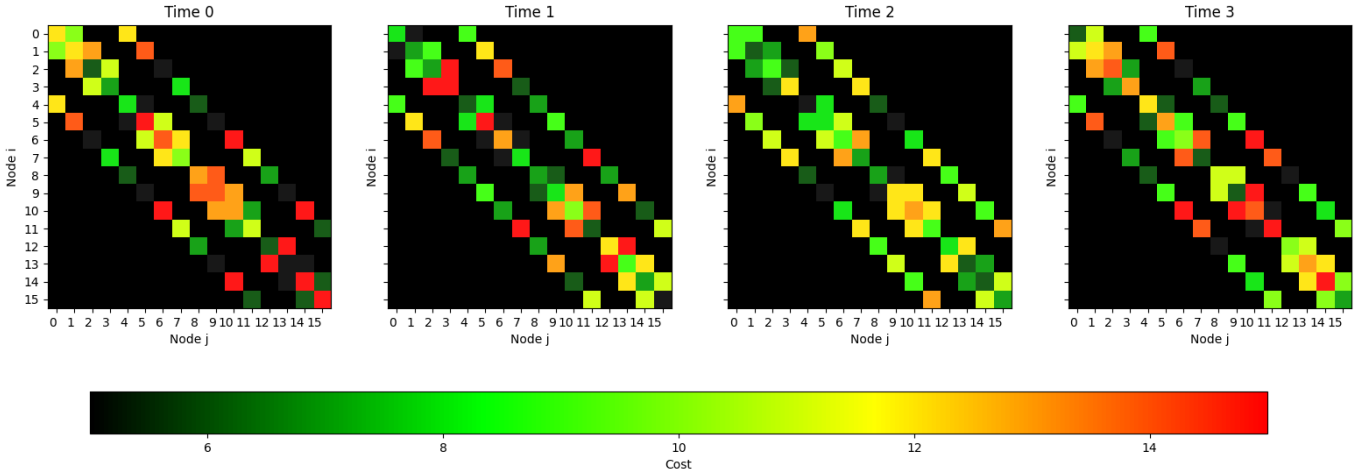


Fig. 5: A visual representation of the cost tensor in the case of a 4x4 square lattice graph with a $[5, 15]$ continuous cost range and a time frame of 4. In black, we have pairs of nodes that are not connected by an edge in the graph; as we can expect from the undirected nature of the graph, we obtain a symmetric matrix.

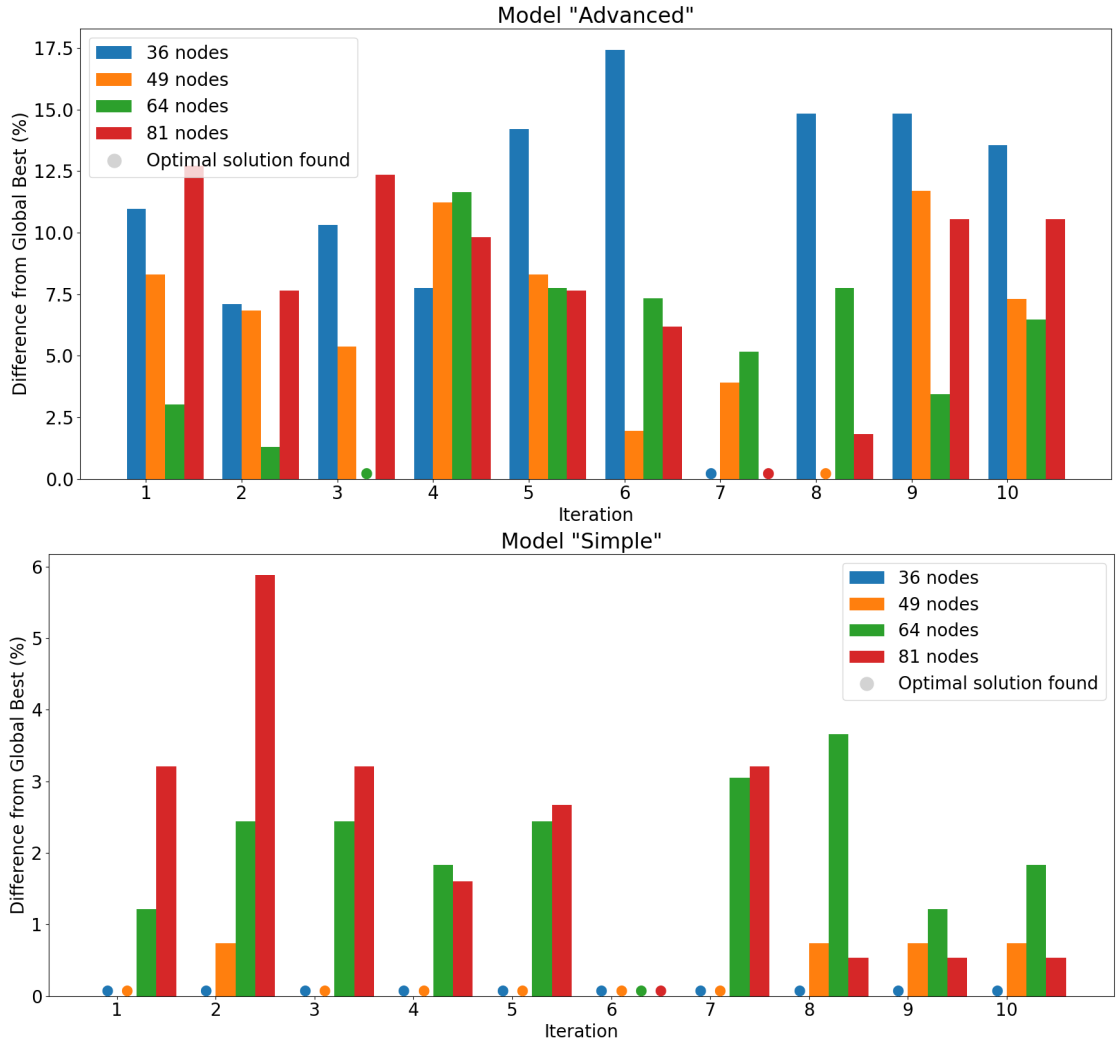
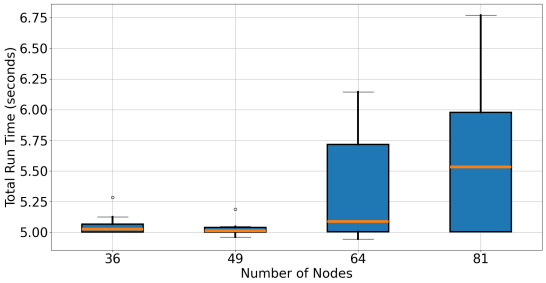
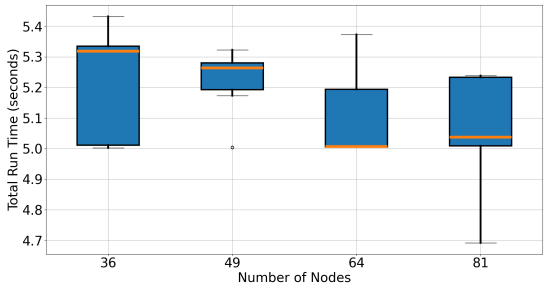


Fig. 6: The percentage difference between the best solution found in a particular iteration and the global best, a filled colored circle signals that an optimal solution was found, hybrid setup.

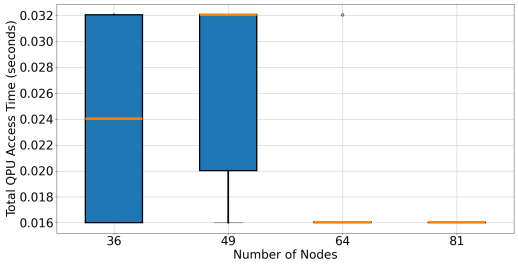


(a) Model Advanced

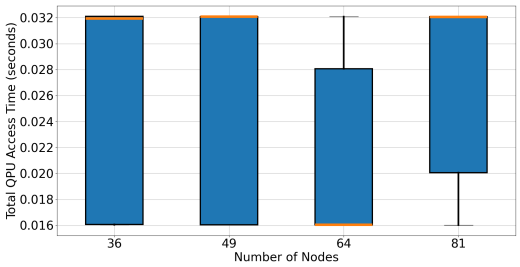


(b) Model Simple

Fig. 7: Total run time of the two models, hybrid setup.



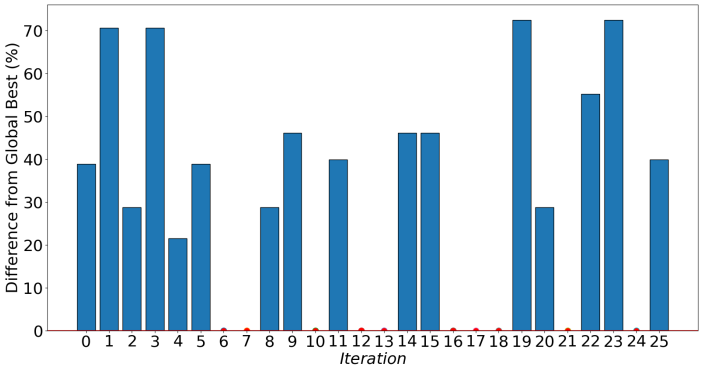
(a) Model Advanced



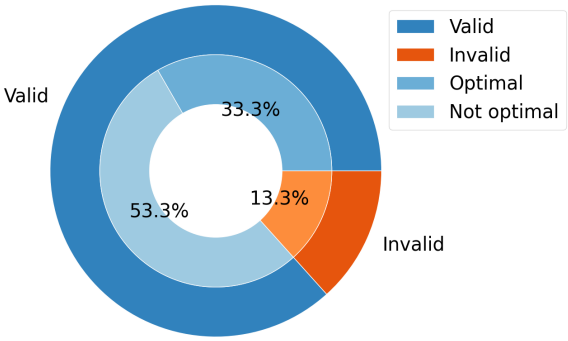
(b) Model Simple

Fig. 8: Total QPU access time of the two models, hybrid setup.

The idea of modeling problems in a way that indulges the physical architecture of the QPU has represented the background of most of our reasoning about the two different formulations, and this was a significant component in the positive goals we reached since at the time of writing it is still unfavorable to ignore completely the low-level details of the quantum hardware on which the model runs. An exciting topic for the future regards the transformation of the two models obtained into unconstrained representations by means of a manual translation of the constraints into



(a) The percentage difference between the best solution found in a particular iteration and the global best, a filled colored half-circle signals that an optimal solution was found, fully quantum setup, model Simple.



(b) Feasibility of the samples obtained.

Fig. 9: Quality of the samples and feasibility of the solutions, fully quantum setup, model Simple.

penalties to obtain a more tailored QUBO formulation that closely reflects the problem's characteristics and requirements. We could then change our workflow and start reasoning on natively unconstrained formulations, comparing the pros and cons of the two approaches. Another exciting path concerns the development of personalized presolving techniques and customized embedding algorithms, to reduce the size of the problem before execution in terms of the overall number of variables and interactions.

Acknowledgments

The authors would like to thank Riccardo Cavadini, Bruno Guindani, and Roberto Sala for their help and support. This work has been partially supported by the projects CN-HPC-S10 and "QUASAR: QUAntum software engineering for Secure, Affordable, and Reliable systems", grant 2022T2E39C, under the PRIN 2022 MUR program funded by the EU - NGEU".

References

[1] T. Albash and D. A. Lidar, "Adiabatic quantum computation," *Rev. Mod. Phys.*, vol. 90, p. 015002, Jan 2018. [Online]. Available: https://link.aps.org/doi/10.1103/RevModPhys.90.015002

[2] S. Teufel, *Quantum Adiabatic Theorem*. New York, NY: Springer US, 2022, pp. 419–431.

[3] D. Aharonov, W. van Dam, J. Kempe, Z. Landau, S. Lloyd, and O. Regev, "Adiabatic quantum computation is equivalent to standard quantum computation," *SIAM Review*, vol. 50, no. 4, pp. 755–787, 2008.

[4] A. Mizel, D. A. Lidar, and M. Mitchell, "Simple proof of equivalence between adiabatic quantum computation and the circuit model," *Physical Review Letters*, vol. 99, no. 7, 2007.

[5] T. Kadowaki and H. Nishimori, "Quantum annealing in the transverse ising model," *Phys. Rev. E*, vol. 58, pp. 5355–5363, Nov 1998. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevE.58.5355

[6] E. Farhi, J. Goldstone, S. Gutmann, J. Lapan, A. Lundgren, and D. Preda, "A quantum adiabatic evolution algorithm applied to random instances of an np-complete problem," *Science*, vol. 292, no. 5516, pp. 472–475, 2001.

[7] V. Mehta, F. Jin, K. Michielsen, and H. De Raedt, "On the hardness of quadratic unconstrained binary optimization problems," *Frontiers in Physics*, vol. 10, p. 956882, 2022. [Online]. Available: https://doi.org/10.3389/fphy.2022.956882

[8] D-Wave Systems, "D-Wave Hybrid," https://github.com/dwavesystems/dwave-hybrid, 2024, gitHub repository.

[9] T. Lei, T. Sellers, C. Luo, D. W. Carruth, and Z. Bi, "Graph-based robot optimal path planning with bio-inspired algorithms," *Biomimetic Intelligence and Robotics*, vol. 3, no. 3, p. 100119, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2667379723000335

[10] C. Li, F. Meng, H. Ma, J. Wang, and M. Q.-H. Meng, "Relevant region sampling strategy with adaptive heuristic for asymptotically optimal path planning," *Biomimetic Intelligence and Robotics*, vol. 3, no. 3, p. 100113, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S266737972300027X

[11] D-Wave Systems, "D-wave ocean software development kit," https://github.com/dwavesystems/dwave-ocean-sdk, 2024, gitHub repository.

[12] ——, "dimod," https://github.com/dwavesystems/dimod, 2024, gitHub repository.

[13] F. Glover, G. Kochenberger, R. Hennig, and Y. Du, "Quantum bridge analytics i: a tutorial on formulating and using qubo models," *Annals of Operations Research*, vol. 314, 07 2022.

[14] V. Choi, "Minor-embedding in adiabatic quantum computation: I. the parameter setting problem," *Quantum Information Processing*, vol. 7, no. 5, pp. 193–209, Oct 2008.

[15] D-Wave Systems, "minorminer," https://github.com/dwavesystems/minorminer, 2024, gitHub repository.

[16] ——, "dwave-inspector," https://github.com/dwavesystems/dwave-inspector, 2024, gitHub repository.

[17] M. D. García, M. Ayodele, and A. Moraglio, "Exact and sequential penalty weights in quadratic unconstrained binary optimisation with a digital annealer," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. Association for Computing Machinery, 2022, p. 184–187. [Online]. Available: https://doi.org/10.1145/3520304.3528925

[18] H. Djidjev, "Automaton-based methodology for implementing optimization constraints for quantum annealing," in *Proceedings of the 17th ACM International Conference on Computing Frontiers*. New York, NY, USA: Association for Computing Machinery, 2020, p. 118–125. [Online]. Available: https://doi.org/10.1145/3387902.3392619

[19] M. Johnson, M. H. S. Amin, S. Gildert, T. Lanting, F. Hamze, N. Dickson, R. Harris, A. Berkley, J. Johansson, P. Bunyk, E. Chapple, C. Enderud, J. Hilton, K. Karimi, E. Ladizinsky, N. Ladizinsky, T. Oh, I. Perminov, C. Rich, M. Thom, E. Tolkacheva, C. J. S. Truncik, S. Uchaikin, J. Wang, B. Wilson, and G. Rose, "Quantum annealing with manufactured spins," *Nature*, vol. 473, no. 7346, p. 194—198, May 2011.

[20] F. Gemeinhardt, A. Garmendia, M. Wimmer, B. Weder, and F. Leymann, "Quantum combinatorial optimization in the nisq era: A systematic mapping study," *ACM Comput. Surv.*, vol. 56, no. 3, oct 2023.

[21] F. Neukart, G. Compostella, C. Seidel, D. von Dollen, S. Yarkoni, and B. Parney, "Traffic flow optimization using a quantum annealer," *Frontiers in ICT*, vol. 4, 2017. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fict.2017.00029

[22] D. Venturelli and A. Kondratyev, "Reverse quantum annealing approach to portfolio optimization problems," *Quantum Machine Intelligence*, vol. 1, no. 1, pp. 17–30, May 2019.

[23] A. Lucas, "Ising formulations of many np problems," *Frontiers in Physics*, vol. 2, 2014.