



Universiteit  
Leiden  
The Netherlands

## Applications of quantum annealing in combinatorial optimization

Yarkoni, S.

### Citation

Yarkoni, S. (2022, December 20). *Applications of quantum annealing in combinatorial optimization*. Retrieved from <https://hdl.handle.net/1887/3503567>

Version: Publisher's Version

License: [Licence agreement concerning inclusion of doctoral thesis in the Institutional Repository of the University of Leiden](#)

Downloaded from: <https://hdl.handle.net/1887/3503567>

**Note:** To cite this publication please use the final published version (if applicable).

---

# Combinatorial optimization and quantum annealing

Metaheuristic algorithms are often used in practice to solve a variety of traditionally difficult optimization problems. Their power derives from the tunability of their parameter sets, often providing practitioners with trade-offs between global search space exploration and local optimization. Annealing algorithms in particular are popular due to their strong theoretical motivation from physical processes and relative ease of implementation [33]. In this chapter we introduce the basic concepts of the quantum annealing algorithm, from both the theoretical and practical aspects, and present the methodology with which optimization problems are solved using quantum annealing hardware.

## 2.1 Adiabatic quantum computing and the adiabatic theorem

The Adiabatic Quantum Computing (AQC) model<sup>1</sup> is a computational model for quantum computing which exploits the *adiabatic theorem*: given a quantum system in its ground (minimum) energy state, if the governing Hamiltonian is changed “sufficiently slowly”, then the quantum system remains in its instantaneous ground state. It has been shown that, given a register of qubits, this adiabatic theorem can be used in order to simulate quantum Hamiltonians and perform algorithmic computation [40]. Furthermore, it has been shown that this AQC

---

<sup>1</sup>This model of computation is also known as the Adiabatic Quantum Optimization model, abbreviated AQO.

## 2. COMBINATORIAL OPTIMIZATION AND QUANTUM ANNEALING

---

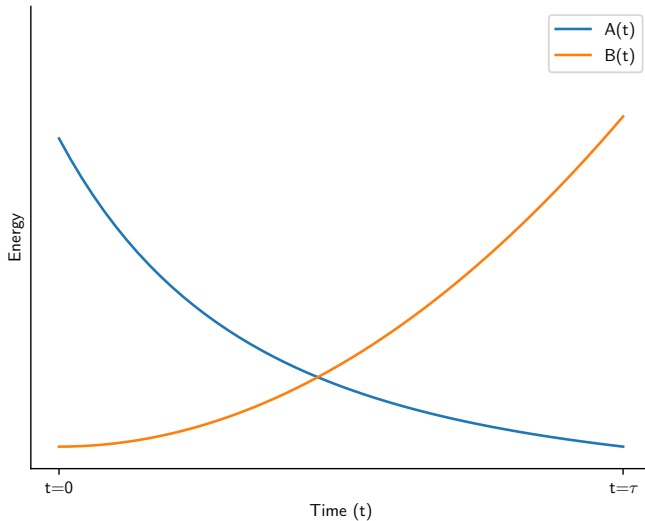
model is polynomially equivalent to the gate-model of quantum computing, by showing how quantum Hamiltonians can be constructed to simulate arbitrary quantum circuits [41]. However, the ability of AQC to solve (or simulate) arbitrary problems is not perfectly understood. The notion of “sufficiently slowly” depends on the both the specific Hamiltonian and the conditions of evolution, and can be difficult to compute. A generic time-dependent Hamiltonian for adiabatic evolution is given by:

$$\mathcal{H}(t) = A(t)\mathcal{H}_i + B(t)\mathcal{H}_f. \quad (2.1)$$

Here,  $\mathcal{H}_i$  is the Hamiltonian in which the system is initialized (referred to as the *initial* or *driver Hamiltonian*), and  $\mathcal{H}_f$  is the Hamiltonian at the end of the evolution (referred to as the *final* or *target Hamiltonian*). The boundary conditions on the relative magnitudes of each Hamiltonian are given by  $B(t = 0) = 0$  and  $A(t = \tau) = 0$ , for an evolution on timescale  $\tau$ . Thus, one could construct  $\mathcal{H}_f$  to represent a quantum circuit and use Eq. (2.1) to perform AQC. A visualization of the transition in magnitudes is shown in Fig. 2.1.

The theoretical motivations for AQC can also be used in the context of optimization. Instead of setting  $\mathcal{H}_f$  to represent a quantum circuit, it can be used as a Hamiltonian representation of an optimization problem. Thus, successfully evolving to a ground state of  $\mathcal{H}_f$  can solve complex combinatorial optimization problem with Hamiltonian representations. In realizable quantum hardware it is not always possible to evolve adiabatically (as any physical system is always coupled to its environment), but it is known that even in the absence of perfect adiabaticity, a quantum system that is evolved “sufficiently slowly” maintains proximity to its ground state [42]. Here we consider only models in which time evolution dependence can be parameterized by a single variable  $t \in [0, \tau]$ . This parameter is also sometimes normalized by  $\tau$ , and referred to as *normalized time*,  $s = t/\tau \in [0, 1]$ . Generally, it is known that the magnitude of  $\tau$  required to remain near the ground state is related to the difference between the two lowest energy states,  $E_0$  and  $E_1$ . This point at which this energy difference occurs is called the *avoided crossing* or *minimum gap*, and is visualized in Fig. 2.2. The timescale of evolution is typically expressed via the approximation:

$$\max_{t_i \leq t \leq t_f} \frac{\langle \mathcal{H}_f(t) | \frac{d\mathcal{H}(t)}{dt} | \mathcal{H}_i(t) \rangle}{|E_1(t) - E_0(t)|^2} \ll \tau. \quad (2.2)$$



**Figure 2.1:** Generic evolution of a Hamiltonian as dictated by  $A(t)$  and  $B(t)$  in the AQC model. In software (meaning, when simulating quantum annealing), the shape of these functions can be programmed and controlled. In hardware, the shapes are hard-coded via the control electronics of the quantum processor and its calibration. To allow for full manipulation of these curves would require controls which are beyond the current capabilities of the technology.

Thus, if the final Hamiltonian  $\mathcal{H}_f$  represents a target optimization problem, then an evolution on timescale  $\tau$  can be used to solve it. However, in practice, calculating  $|E_0(t) - E_1(t)|$  requires knowledge of the entire eigenspectrum of the Hamiltonian  $\mathcal{H}$  (in the worst case), and is therefore NP-hard in itself.

## 2.2 Theory of quantum annealing

Originally described as a metaheuristic in classical software used to solve combinatorial optimization problems, quantum annealing (QA) is closely related to AQC [35]. Analogous to classical simulated thermal annealing, quantum fluctuations are used in order to tunnel through energy barriers in a combinatorial landscape, as opposed to thermally-assisted hops in simulated annealing. In quantum annealing, these fluctuations induce *quantum tunneling* effects, where the wavefunction is split

## 2. COMBINATORIAL OPTIMIZATION AND QUANTUM ANNEALING

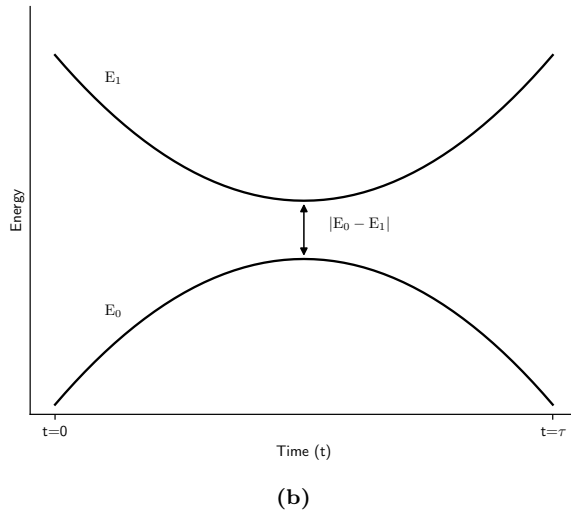
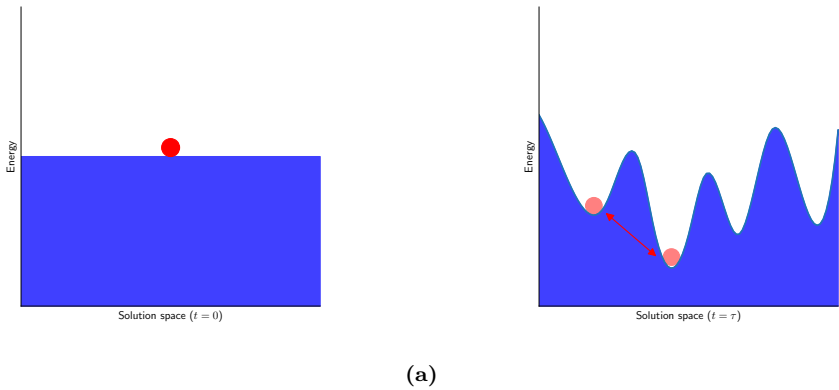
---

across energy barriers to different energy minima. At the end of the evolution (ideally) the wavefunction is concentrated at global optima of the energy landscape. However, instead of initializing the algorithm with random initial configurations (like in thermal annealing) the QA algorithm is initialized by setting qubits' states to a simple ground state (generally a superposition over all states), and is evolved over time to target a final Hamiltonian  $\mathcal{H}_f$ . Quantum annealing can be viewed as a subclass of AQC, where the Hamiltonian type used for  $\mathcal{H}_f$  is fixed, and adiabaticity is not guaranteed. Thus, the result is a heuristic quantum optimization algorithm that has a non-zero probability of returning a candidate solution to an optimization problem, rather than a deterministic quantum simulation. In its simplest form, quantum annealing is implemented using the *transverse-field Ising Hamiltonian*:

$$\mathcal{H}(t) = A(t)\mathcal{H}_i + B(t)\mathcal{H}_f = A(t) \left[ \sum_i \sigma_i^x \right] + B(t) \left[ \sum_i h_i \sigma_i^z + \sum_{i<j} J_{ij} \sigma_i^z \sigma_j^z \right], \quad (2.3)$$

where  $\sigma_i^x$  and  $\sigma_i^z$  are the Pauli-X and -Z spin matrices applied to the  $i$ th qubit, and  $h_i$  and  $J_{ij}$  are used to define the spin-glass representation of the optimization problem. Here, as in Eq. (2.1),  $A(t)$  and  $B(t)$  are time-dependent functions that dictate the magnitudes of each term in the Hamiltonian. Since it is known that finding the ground state of the 2D Ising spin-glass problem is NP-hard [43], this Hamiltonian definition is sufficient to address many interesting combinatorial optimization problems. However, as with AQC, the probability of obtaining (or remaining close to) a ground state at the end of the computation is bound by the evolution timescale  $\tau$  and the smallest energy difference between the ground state and first excited state in Eq. (2.2). As this evolution time is a parameter to the quantum annealing algorithm, it is typically referred to as the *annealing time*,  $t_a$ . The dependence of QA's performance on time is a known problem in quantum optimization. Particularly, how optimal (minimal)  $t_a$  scales as a function of problem size for different problem classes is generally regarded as the true scaling of the algorithm, as the goal is to relate the evolution time of QA to observing the ground state of the system.

Many interesting experiments have been performed to assess the conditions of scaling for quantum annealing, both in simulation and in hardware. Early work with quantum Monte Carlo simulations showed the ability of QA to solve combinatorial optimization problem in some limits [44]. Additional work (in quantum Monte Carlo simulations) investigated the correlation between exponentially small energy



**Figure 2.2:** (a) Schematic of the evolution of the energy landscape for a combinatorial optimization problem. On the left, all states are minima, representing the equal superposition of the two basis states for  $\mathcal{H}_i(t = 0)$ . The red circle represents a single state in this space. On the right, a visualization of a  $\mathcal{H}_f(t = \tau)$ , where there are well-defined minima at the end of the evolution. The red circle represents a state corresponding to an optimum, having tunneled through an energy barrier from a higher-energy state. (b) Here, a diagram of the minimum gap shows the difference between the two lowest energy levels in the quantum system (as a function of evolution time), which defines the adiabatic condition of evolution.

## 2. COMBINATORIAL OPTIMIZATION AND QUANTUM ANNEALING

---

gaps and the probability of observing ground states using maximum independent set instances [45]. A similar experiment testing the scaling of annealing time in the presence of thermal excitations was performed using QA hardware in [46]. For a complete assessment of state-of-the-art theory and a thorough discussion of the scaling of QA in general, see [42].

### 2.3 Binary and combinatorial optimization

We now focus on the representation of general optimization problems in either of the admissible binary forms so they can be solved using quantum annealing. The first model is the classical Ising spin-glass model:

$$H_f = \sum_i h_i s_i + \sum_{i < j} J_{ij} s_i s_j, \quad (2.4)$$

where  $s_i \in \{-1, 1\}$  are the individual spin variables,  $h_i$  are the real-valued linear weights, and  $J_{ij}$  are the so-called quadratic interaction terms. Alternatively, for binary variables  $x_i \in \{0, 1\}$  the quadratic unconstrained binary optimization (QUBO) form is used:

$$\text{Obj}(\mathbf{Q}, x) = x^T \cdot \mathbf{Q} \cdot x, \quad (2.5)$$

where  $x = (x_0, x_1, \dots, x_{N-1})$  is a vector of  $N$  binary variables, and  $\mathbf{Q}$  is an  $N \times N$  real-valued matrix of interaction terms (diagonal elements are the variable weights, and off-diagonal elements are the quadratic interactions). Many canonical examples of combinatorial optimization have been studied extensively in literature using both of these forms, and often for research in computer science and quantum computing: the traveling salesperson problem (and related vehicle routing problem) [47], max-cut [36], satisfiability [48], graph coloring [49] and more. In general, all of these analyses in combinatorial optimization exploit binary programming, also known as 0 – 1 binary programming, or binary optimization (and in some cases pseudo-Boolean optimization). The task is to assign the optimal value for each binary variable in a set such that a particular objective function is minimized. Originally, as one of Karp’s 21 NP-complete problems, the objective function was represented as binary integer problem [50]. Formally, we are interested in a more general formulation, namely *pseudo-Boolean objective functions*. We define these as a family of functions whose domain is the set of  $N$  binary variables, which are then mapped to a real number:

## 2.3 Binary and combinatorial optimization

---

$$f : \mathbf{B}^N \rightarrow \mathbb{R}. \quad (2.6)$$

As an NP-hard problem itself, Boolean optimization can be used to represent all other NP-hard problems, and therefore is sufficient to describe these canonical combinatorial optimization problems (not just binary optimization problems), with some polynomial overhead. Therefore, the initial step for using QA in practice is that of problem definition: the objective function of the optimization problem must be represented entirely with binary variables.

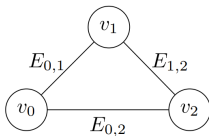
It is both common and useful in the context of quantum annealing (and quantum computing in general) to study combinatorial optimization from a graph representation perspective. Every Ising/QUBO problem can be represented as an undirected weighted graph, with every variable represented by a vertex  $v_i \in V$  and pairwise interaction term represented by an edge  $e_{ij} \in E$  between nodes  $v_i$  and  $v_j$ . In the Ising (QUBO) model, each  $h_i$  in Eq. (2.4) (diagonal elements of  $Q$  in Eq. (2.5)) is the corresponding weight for  $v_i$  in the graph representation, and likewise the  $J_{ij}$  (off-diagonal elements of  $Q$ ) for  $e_{ij}$ . A simple demonstration of the equivalence between QUBO, Ising, and weighted undirected graphs is shown in Fig. 2.3.

$$\mathcal{H} = h_0 s_0 + h_1 s_1 + h_2 s_2 + J_{0,1} s_0 s_1 + J_{0,2} s_0 s_2 + J_{1,2} s_1 s_2$$

(a)

$$\text{Obj} = x^T \begin{pmatrix} Q_{0,0} & Q_{0,1} & Q_{0,2} \\ 0 & Q_{1,1} & Q_{1,2} \\ 0 & 0 & Q_{2,2} \end{pmatrix} x$$

(b)



(c)

**Figure 2.3:** (a) A simple objective definition of a three-variable Ising model. (b) Equivalent representation of a QUBO matrix in upper-triangular form. The terms in the QUBO matrix which correspond to (a) can be derived using the change of basis  $s_i = 2x_i - 1$ . (c) A graph network representation of same system, with variables as nodes and interaction terms as edges. The specific weights in the graph depend on the choice of basis, (a) or (b).

Collectively, these are referred to as *binary quadratic models* (BQMs). To generalize binary models to arbitrary variable types, specific mathematical techniques are



used in practice. The relevant approaches used in the studies of this thesis (and the related works) are introduced next.

### 2.4 Generalizing QUBO and Ising

Although binary optimization is NP-hard, the polynomial overhead incurred when modeling more general problems means that certain classes of optimization are better (or conversely, less) suited for quantum annealing. Furthermore, different classes of real-world optimization problems require different categories of variables. Here we review the types of variables and general terms that are often used for quantum annealing in practice.

#### 2.4.1 Constrained optimization

In QUBO/Ising form, the real-valued constant variable coefficients are unconstrained. As quantum annealing is an analog process, there is also no way to explicitly constrain the variables during the evolution. Therefore, all equalities (and inequalities) must be implemented in a quadratic objective form in order to be included in the model. This means that constrained optimization cannot be performed directly, but is instead addressed by including an additional term scaled by a constant (known as a penalty factor) to separate *feasible* and *infeasible* solution spaces in the optima of the optimization problems. One example of an important constraint—the one-hot constraint, where exactly one binary variable is 1 and the rest are 0—is transformed as follows (with binary variables  $x_i$ ):

$$\sum_i x_i = 1 \longrightarrow \left(1 - \sum_i x_i\right)^2 = 0. \quad (2.7)$$

The left hand side of the equation above represents the equality constraint with a simple summation over the variables. Adding these factors to a QUBO would result in a minimum where all  $x_i = 0$ , obviously violating the purpose of the one-hot constraint. Thus, the right hand side is implemented to constrain the variable space by expanding the square, which has only linear and quadratic terms that can be added to the QUBO objective function. This now has the correct minimum with one binary variable set to 1 and the rest to 0. Arbitrary linear constraints are expressed in QUBO form as:

$$\lambda \left( b - \sum_i a_i x_i \right)^2 = 0, \quad (2.8)$$

where  $\lambda, b, a_i$  are all real-valued numbers, and  $x_i$  are a subset of all binary variables in the problem. Because the contribution of a satisfying configuration of the binary variables is zero, the cost of violating this condition can be set by the parameter  $\lambda$ . In pure constraint satisfaction problems, this isn't strictly necessary, since (by addition) the individual contributions of the constraints are zero, and thus the objective function value of a satisfying solution will also be zero. When mixing optimization terms and constraints in the objective function, it is necessary to set  $\lambda$  appropriately such that it is never energetically favorable to violate a constraint in favor of reducing the value of the objective function.

These concepts can be further extended to address inequalities as well, by introducing *slack variables*. In general, linear inequality constraints (on binary variables  $x$ ) are presented as:

$$\sum_i a_i x_i - b \leq 0, \quad (2.9)$$

where  $a_i, b$  are integer coefficients. To transform this inequality to QUBO, we must create an degenerate objective function such that all minima satisfy the inequality. We start by adding auxiliary variables such that the total slack is accounted for:

$$\lambda \left( \sum_i a_i x_i + \sum_j^W w_j y_j - b \right)^2 = 0. \quad (2.10)$$

The number of slack variables  $W$  and their coefficient  $w_j$  can be derived from the coefficients  $a_i$  and  $b$ , where at most  $W = b$  slack variables are needed<sup>1</sup>. Now the equality constraint is equivalent to the original inequality constraint, since it can be fulfilled for all values of the binary variables  $x_i$  satisfying the original inequality by choosing some appropriate values for the slack variables, which remain unconstrained. For values violating the original inequality constraint, the equation can never be fulfilled regardless of the values of the slack variables. The equality constraint written in the quadratic form is then added as a penalty term to the QUBO cost function.

---

<sup>1</sup>In general, it is possible to use binary encoding schemes such that only  $\lceil \log_2(n) \rceil$  auxiliary qubits are needed.

### 2.4.2 Discrete variables

Non-binary discrete variables can also be represented in Ising/QUBO, and have been investigated in the context of quantum annealing in scheduling problems [51, 52] and graph colouring [53, 54, 55], among others. Here, each discrete variable can be encoded by using multiple qubits to represent a logical integer variable subject to a single constraint [56, 57, 58]. Binary encoding, analogous to classical binary encoding, is efficient in the number of qubits in the sense that one can encode  $d$  discrete states using only  $\lceil \log_2(d) \rceil$  qubits. However, the binary encoding is not used much in practice for application problems, since the interactions necessary to enforce the validity of the encoding as well as the realization of the couplings between logical variables are rather complicated to implement with the binary encoding. Additionally, this encoding has been investigated in previous works and has been shown to be detrimental to the QPU's ability to find ground states [59, 49].

The one-hot encoding is a standard technique where a logical variable with  $d$  possible states is represented by  $d$  qubits. Each qubit's state corresponds to one possible value of the discrete variable if set to  $|1\rangle$ , and all other qubits are  $|0\rangle$ . The corresponding constraint is typically implemented as quadratic interaction terms between all  $d$  qubits, shown in Eq. (2.7). The advantage of this technique is the fact that the one-hot case can be trivially extended to the  $k$ -hot case, where a subset of exactly  $k$  qubits are  $|1\rangle$  and the rest  $|0\rangle$ — however, this requires all-to-all connectivity between the qubits. This is expressed in the form:

$$\left(k - \sum_i x_i\right)^2 = 0. \quad (2.11)$$

An alternative to the one-hot constraint is the *domain-wall encoding*, which can encode discrete variables with one fewer qubit per variable, i.e.,  $d - 1$  qubits for  $d$  states, and requires only a chain of couplings rather than all-to-all. It has been shown to be more efficient in different test problems [59, 49, 60]. However, the main drawback of this method is that it cannot extend directly to the  $k$ -hot constraint using the same technique. Rather,  $k$  copies of the  $d - 1$  chain must be connected (non-trivially) to enforce the same combinatorial landscape, which is no longer efficient in the number of qubits.

### 2.4.3 Continuous variables

Representing continuous numbers using binary variables is also possible using QUBO/Ising, although due to the binary encoding necessary it is not often used in practice. Much like in classical computing, these numbers are represented by binary encoding schemes. A single decimal variable  $\tilde{x}$  with  $N$  bits of precision would be encoded as follows (using binary values  $x$ ):

$$\tilde{x} = \sum_{i=0}^{N-1} 2^i x_i. \quad (2.12)$$

There are several drawbacks to using this encoding scheme. Firstly, this requires high precision in encoding the optimization problem, which makes the solution landscape more difficult to explore due to thermal noise in quantum hardware. Furthermore, the individual energy levels (or local optima) are exponentially spaced, making them more less likely to be explored. Lastly, the connectivity between the qubits is again quadratic, increasing the density of the problem being solved. However, expressing continuous variables is sometimes unavoidable, and this technique has been used in practice [61].

## 2.5 Quantum annealing in hardware

The core of a quantum processing unit (QPU) is a layout of qubits which are connected via a system of couplers. The QPUs used in the research in this thesis were from D-Wave Systems, which implement superconducting flux qubits to build their processors [62]. While the component physics is beyond the scope of this thesis, a technical description of the quantum processors can be found here [25]. In the absence of logical encoding schemes, the physical layout of the qubits is fixed in the QPU and is referred to as the *hardware graph*, denoted by  $U$ . This naming convention comes from the graph-theoretical description of the QPU, where each qubit (coupler) is represented by a node (edge) in an undirected graph. The exact topology of the hardware graph dictates the structure of graphs and possible  $\mathcal{H}_f$  that can be represented natively by the qubits' connectivity. The QPU topology is an engineering artifact arising from the design choices for the QPU. In general, due to engineering difficulties, connectivity between qubits comes at the cost of the total number of qubits in the QPU. Due to various technological limitations

## 2. COMBINATORIAL OPTIMIZATION AND QUANTUM ANNEALING

---

in manufacturing, calibration, or other anomalies, a small portion of the qubits and couplers in the QPUs may be defective and hence not programmable. The percentage of qubits and couplers that remain functional once fully exposed to users is referred to as the *hardware yield*. For D-Wave QPUs, the qubit yield is typically above 97% for current processors [63]. The graph representing the functional qubits and couplers is referred to as the QPU's *working graph*.

The two topologies used for studies presented in this thesis are the D-Wave 2000Q and Advantage QPUs. The topology of the D-Wave 2000Q (and earlier generations of QPUs) are referred to as *Chimera* topology. These graphs are composed of a 2D lattice of small complete bipartite graphs (each called a tile, or unit cell). These graphs are denoted as  $C_X$  (where  $X$  is the length of one side of the square lattice, for a total of  $X^2$  tiles), and each tile is in itself a  $K_{N,N}$  bipartite graph (full connectivity between left and right partitions). The D-Wave 2000Q has a size of  $C_{16}$  and  $K_{4,4}$  tiles. Each qubit in the Chimera topology has six couplers, where four couplers are inside the unit cell and two are to different unit cells. The Advantage QPUs implement a novel topology, called *Pegasus*, which differs in two major aspect from the Chimera graph: the graph degree is 15, and the hardware graph is not bipartite. Instances of the Pegasus topology that contain  $N \times N$  unit cells are referred to as  $P_N$  and consist of  $24N(N - 1)$  qubits. This results in a significantly more complex structure, allowing for denser graph structures of  $\mathcal{H}_f$  to be solved by the QPUs, and hence more difficult optimization problems (a technical description comparing Chimera and Pegasus can be found in Ref. [64]).

### 2.5.1 Minor-embedding for fixed topologies

Obviously not all interesting optimization problems can be defined by the hardware graph Chimera or Pegasus directly. However, it is possible to reformulate arbitrarily-structured Hamiltonians  $\mathcal{H}_f$  into a new hardware-compatible  $\mathcal{H}'_f$  via the technique of graph *minor-embedding*, a well-studied problem in graph theory (see, e.g. [65]). This process produces a mapping between one graph to another such that the relevant topological properties of the original graph are preserved. For QA, this involves encoding logical problem variables, or nodes in a graph, as *chains* of multiple physical qubits on a QPU so that they act as a single logical qubit [66, 67]. Different families of topologies may produce very different structures in hardware for the same input graph represented by  $\mathcal{H}_f$ . The impact of this, specifically in terms

of canonical problems and the various chain requirements when embedding certain graph families in both Chimera and Pegasus graphs are explored in [64].

There exist several difficulties when using minor-embedding techniques. First, deciding whether a graph can be minor-embedded into another is a known NP-complete problem, and so polynomial-time heuristics (or deterministic algorithms on constrained subspaces) are used in practice [68, 69]. Secondly, the use of embeddings requires an additional set of constraints to be imposed on the qubits representing a logical qubit (the magnitude of this constraint is called the *chain strength*). These enforce that the minimum energy of  $\mathcal{H}'_f$  is obtained only when all physical qubits representing a logical qubit are in the same state. It has been shown that the magnitude of the minimum chain strength increases with the degree of the graph of  $\mathcal{H}_f$  [67, 70], which can create distortions in the resulting Hamiltonian  $\mathcal{H}'_f$ . Generally, determining the best suited chain strength is a nontrivial task, and it is one of the parameters explored in more depth in the results of this thesis.

### 2.5.2 Noise and mitigation strategies

Building a quantum processor inherently involves the implementation of an open quantum system— meaning, the qubits can never be truly perfectly isolated from their operating environment. This makes it difficult to draw universal conclusions about the power of programmable QA [71]. However, many investigations into how these quantum system interact with the environment (known as background noise) have been published in the past. Dickson et al. [46] were able to demonstrate the effects of thermal noise and diabatic transitions with D-Wave QA processors. In this context, resilience against background noise was defined as the ability of the quantum system to yield the correct solution with acceptable probability within a time comparable to the closed-system adiabatic timescale. It was demonstrated that in the limit of weak coupling to the environment (i.e., relatively low levels of thermal noise), annealing diabatically across the minimum gap did not hinder QA, but rather enhanced its performance. This was demonstrated by manually raising the operating temperature of the chamber in which the QPU was operating, thus allowing for a controllable amount of noise to perturb the system. The specific point demonstrated was that thermal noise does not necessarily inhibit obtaining ground states at the end of the annealing, even with small minimum gaps and non-adiabatic conditions. However, the caveat remains that the annealing times

## 2. COMBINATORIAL OPTIMIZATION AND QUANTUM ANNEALING

---

need to be sufficiently long, and the system must be only weakly coupled to the environment<sup>1</sup>. Additional experimental studies of quantum annealing processors have investigated the effects of noise and its role in computation of solutions to optimization problems. It has been demonstrated that finding quantum speedups may be elusive in quantum annealing hardware with fixed topologies (tested on spin-glass instances) [72]. Furthermore, it has been shown in such tests that classical optimization algorithms can often match and exceed the performance of quantum annealing hardware on such test instances due to physical limitations of the devices [73]. Nonetheless, it has also been demonstrated that small clusters of qubits do indeed remain coherent for some duration of the quantum annealing protocol, which can contribute to solving specific kinds of (contrived) optimization problems [74]. The extent to which this contributes to hardware performance for large-scale problems remains an open question.

Noise can also directly perturb problem formulations in the analog system, i.e.,  $\mathcal{H}_f$ . The Hamiltonian parameters are subject to noise that can be modeled using a Gaussian distribution over the terms in the Hamiltonian. This noise (specifically, thermal noise whose timescale is much slower than the annealing dynamics) may cause a reshuffling of the energy levels in the problem Hamiltonian, which results in an incorrect encoding of the logical optimization problem in the eigenstates of the Hamiltonian. There are multiple ways to mitigate some of the effects of noise in quantum annealing. The most common technique is *spin-reversal transforms*, which are used to effectively average over the asymmetric final Hamiltonian distortion due to noise. This is done by creating random spin vectors  $S = \{-1, 1\}^N$  and recalculating a new  $\mathcal{H}'_f$  subject to this shift, where  $h'_i = h_i \cdot S(i)$ , and  $J'_{ij} = J_{ij} \cdot S(i)S(j)$ . These transformations do not change the distribution of ground states in the system, only the signs of a subset of terms. This new Hamiltonian  $\mathcal{H}'_f$  is then sampled using a QPU. To recover the samples subject to the original Hamiltonian  $\mathcal{H}_f$ , the solutions are multiplied by the vector  $S(i)$ . Typically, many such transforms are used when solving a single  $\mathcal{H}_f$ . While this method is effective in reducing the error due to static noise, each new  $\mathcal{H}'_f$  results in another programming cycle, increasing the total wall-clock time of using QA. Furthermore, it has been shown that increasing the number of transforms has

---

<sup>1</sup>Due to thermal relaxation, in the worst case the time to reach equilibrium in an open quantum system can grow orders of magnitude with respect to the time needed in a closed system.

diminishing returns for a fixed number of samples [75]. However, because of the technique’s effectiveness in averaging over the static noise, this method is often used in practice.

### 2.5.3 Workflow of solving problems with QPUs

In QA, as implemented in quantum hardware by D-Wave Systems, the process of solving a combinatorial optimization problem can be divided into discrete stages. It is important to realize that very little in the procedure can be changed, and in practice the freedom is in the tuning of parameters exposed to users within this system. It is then up to the users of QA to interpret the results relative to the problem. In the context of optimization, we can conceptually model the QPU as a black-box optimization algorithm with very specific parameters and constraints. The step-by-step process is as follows:

- **Definition of a QUBO or Ising formulation and graph representation.** QUBO problems and Ising models have become the standard input format for quantum annealers, to which the optimization problems of interest are converted. This problem is represented as an undirected weighted graph, where each node represents a variable and each edge denotes the interaction term between a pair of variables. Thus the problem statement is given as finding the correct assignment of either  $\{0, 1\}$  or  $\{-1, 1\}$  (depending on the choice of basis) to minimize the quadratic objective function.
- **Minor-embedding.** The logical interaction graph is translated to the physical hardware graph of the QPU. It is necessary to select sets of physical qubits to represent a single logical node and to identify the couplings between the physical qubits to realize the correct interactions between the logical variables. Provided the correct parameters are chosen to enforce the chains of physical qubits, this procedure does not alter the minimum energy solution landscape, and the QPU’s ability to optimize the embedded problem solved the original optimization problem of interest.
- **Programming and initialization.** Programming the quantum annealer requires setting the parameters that define the embedded problem to be solved,  $\mathcal{H}_f$ . This involves setting the weights for each qubit biases (controlling the magnetic field acting on the qubit) and coupler strengths (the interaction between qubits). To initialize the system, the qubits are set to a superposition



## 2. COMBINATORIAL OPTIMIZATION AND QUANTUM ANNEALING

---

of the two computational basis states. This is the lowest energy configuration of the easy-to-implement initial Hamiltonian.

- **Annealing process.** In this step, the Ising model is solved. The system transitions from the initial to the final Hamiltonian according to predefined annealing functions in an attempt to minimize the energy. The way that the functions  $A(t)$  and  $B(t)$  are evolved (called the *annealing path*) through this search space is not fully controllable, but rather the total physical time taken is set by the user of the QA QPU.
- **Readout of the solution.** At the end of the annealing phase, the qubits are measured in the computational basis, and their configuration represents a candidate minimum of the final Hamiltonian. The individual spin values of the final configuration are read out and stored externally representing a candidate solution to the original problem.
- **Resampling.** Because quantum annealing is a heuristic, there is only ever a non-zero probability that the computation results in a ground state of the system. Therefore, the anneal-readout cycle is repeated many times per problem to acquire multiple candidate solutions. The number of times this is performed is determined by the user.

It is important to note that, as a heuristic optimization routine, the *time* used by the quantum annealing algorithm must be distinguished from the time used by the QPU. Total runtime is a combination of engineering-specific timing that cannot be altered, user-specified parameters, and the number of independent trials. To represent wall-clock time,  $t_{\text{wall-clock}}$ , the most general timing model can be regarded as follows:

$$t_{\text{wall-clock}} = t_{\text{prog}} + N_{\text{reads}} \cdot (t_a + t_{\text{readout}}). \quad (2.13)$$

Here,  $t_{\text{prog}}$  is the programming time required to set the initial values of the qubits and couplers,  $t_a$  is the user-specified annealing time<sup>1</sup> (physical time of evolution from  $\mathcal{H}_i$  to  $\mathcal{H}_f$ ),  $t_{\text{readout}}$  is the time to read out all the qubits' states at the end of the annealing cycle, and  $N_{\text{reads}}$  is the number of trials (i.e., the number of samples obtained from the QPU). The importance of this model is exemplified

---

<sup>1</sup>This is essentially the same as  $\tau$  from Sec. 2.2. However, to distinguish the theoretical aspects of adiabatic evolution and the user-specified parameter for fixed annealing paths as implemented in D-Wave QPUs, we use  $t_a$  to refer to the annealing time in practice.

## 2.5 Quantum annealing in hardware

---

by the analysis of “runtime” for quantum annealing: clearly, the more samples are taken ( $N_{\text{reads}}$  is increased), the higher the probability that one of the samples is a ground state. However, the probability of the proportion of a single sample obtained at the end of the annealing is a ground state is controlled by  $t_a$ . For the research presented in this thesis, the distinction is made clearly in the appropriate contexts.