Quantum optimization and applied algorithms

1 Introduction

Quantum computing has advanced significantly in recent years, and while no conclusive quantum advantage has been demonstrated at the time this document has been written, real world applications appear to be on the horizon. There are several potential areas for near term applications of quantum computing, of which optimization is one promising area. For this chapter, we will investigate how quantum tools can be used to solve classical optimization problems.

The first step in quantum optimization is to map the problem to a physical model which can be encoded on a quantum computer. In practice, the most common choice is the Ising model, which is known to be able to map all NP-hard[1] problems:

$$H_{\text{Ising}} = \sum_{k=1}^{n} \sum_{j=k+1}^{n} J_{kj} \sigma_k^z \sigma_j^z + \sum_{j=1}^{n} h_j \sigma_j^z, \tag{1}$$

where $\sigma_j^z = \left(\bigotimes_{k=1}^{j-1} I_2\right) \otimes \sigma^z \otimes \left(\bigotimes_{k=j+1}^{n} I_2\right)$ where $I_2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ is the 2x2 identity matrix and $\sigma^z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$. The $\otimes$ symbol indicates a tensor product, which is reviewed in Appendix 1, along with the $\bigotimes_{k=1}^{j-1}$ notation. The coupling strengths J_{kj} form a matrix, and the field strengths h_j form a vector specifying a particular Ising model. Optimization problems can be encoded in the matrix J and the vector h in ways which will be explained in section 3.

When J and h encode a problem, they encoded it in such a way that the minimum energy with respect to H_{Ising} is the best solution. When represented as a matrix, H_{Ising} is a $2^n \times 2^n$ diagonal matrix where each diagonal entry gives the energy of a classical bitstring from $|00...0\rangle$ to $|11...1\rangle$, but writing it out this way is only possible for very small problems. To see why, consider a 100 bit optimization problem, which is actually still quite small compared to what one typically encouters in the 'real world', the number of solutions will be $2^{100} \sim 10^{30}$. In physics terms, this is more than the number of water molecules in a large tanker truck full of water, counting the molecules in such a truck is clearly not possible. Finding the solution to such an optimization problem by just checking all possible solutions and taking the lowest is similarly impossible, if you had a fast computer which could check one solution every nanosecond, even if you started at the beginning of the universe, you would be less than $\frac{1}{1000}$th of the way to being done.

[1] Non-deterministic polynomial-time hard (often mistakenly referred to as 'non-polynomial hard'), the exact definition of this problem class is not important for this project, but this is the 'hardest' class of conventional optimisation problems, and it is suspected (but not proven) that no efficient algorithms exist for this class of problems.

2 The Quantum Adiabatic Algorithm

Clearly, there are better ways (i.e. *algorithms*) to solve optimization problems than just checking every possibility. Also, in most cases you just need a good solution, not the *best* solution. Coming up with clever classical ways to solve optimization problems (e.g. genetic algorithms, swarm algorithms, and many others) is a huge area of active research, but these problems could also potentially be solved with the help of quantum mechanics. So far, all we have shown is how to state an optimization problem as a Hamiltonian, we have not shown how to add quantum effects. Consider the effect of adding single bit flip operations to H_{Ising}, to obtain the *transverse field Ising model*:

$$H_{\text{transverse Ising}} = -A(t)\sum_j \sigma_j^x + B(t)H_{\text{Ising}} \tag{2}$$

where A and B are time dependent controls, and $\sigma_j^x = \left(\bigotimes_{k=1}^{j-1} I_2\right) \otimes \sigma^x \otimes \left(\bigotimes_{k=j+1}^{n} I_2\right)$ where $\sigma^x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ is the usual Pauli matrix. If $B = 0$ and $A > 0$, then the ground state $|\phi_0(A > 0, B = 0)\rangle$ of Eq. (2) will just be the lowest energy state of each σ^x individually, in other words, $|\phi_0(A > 0, B = 0)\rangle = \bigotimes_{j=1}^{n} |+\rangle = \frac{1}{\sqrt{2^n}} \sum_{j=0}^{2^n-1} |j\rangle$ where $|+\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix}$. On the other hand, when $B > 0$ and $A = 0$, then by definition $|\phi_0(A = 0, B > 0)\rangle$ will be the solution[2] to the optimization problem which has been encoded into J and h in Eq. (1).

Based on the statements in the previous paragraph, and some basic quantum mechanics, there is a general quantum algorithm for optimization problems, assuming a physical system which can implement the transverse field Ising model with sufficient controls is available. The *adiabatic theorem of quantum mechanics* states that a quantum system initialized in the ground state of one Hamiltonian can remain in the ground state even if the Hamiltonian is changed. For the adiabatic theorem to hold, the change must be slow enough, and some other technical criteria—which transverse Ising systems always satisfy—are met. If we start in $|\phi_0(A > 0, B = 0)\rangle = \bigotimes_{j=1}^{n} |+\rangle$ with $B = 0$ and $A > 0$, and change A and B *slowly enough* until we end with $B > 0$ and $A = 0$, we will find a final state which is very similar to $|\phi_0(A = 0, B > 0)\rangle$, which is the solution to our problem. This techniques is known as *adiabatic quantum computing* (AQC) or the *quantum adiabatic algorithm* (QAA). Note that, while AQC definitely works, if you change the Hamiltonian slowly enough, it won't be useful in practice if it takes much longer than classical computers solving the same problem. In practice we would be happy even if it doesn't find the absolute best solution, but finds a better solution than the classical algorithms, or if it finds one the classical methods do find, but finds it much faster.

[2] if there is a tie for the best solution, then it could be a superposition state

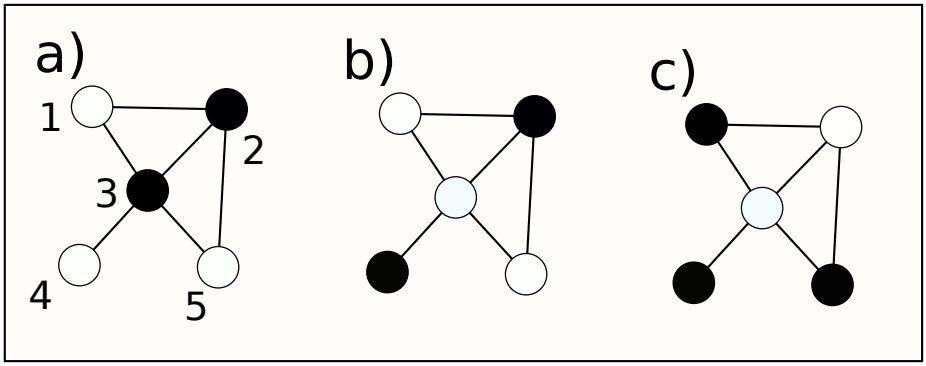

Figure 1: A graph with different sets of vertices coloured in black. a) A set of vertices which are not independent. b) A set of vertices which are independent, but is not the maximum independent set. c) The maximum independent set of vertices for this graph. The numbering scheme used in Eq. (4) is shown in a).

3 Encoding Optimization problems into Hamiltonians

To explain how optimization problems can be mapped to the J and h terms in Eq. (1), we will consider a simple problem, known as maximum independent set. The easiest way to think of maximum independent set is to think of a *graph* with vertices connected by edges, for example, the one shown in Fig. 1. The goal of maximum[3] independent set is to colour in as many vertices as possible such that no two coloured vertices share an edge.

While maximum independent set may seem like a silly colouring problem, it could actually come up in the real world. For instance, imagine a stock broker wants to build a large diverse portfolio of stocks and the presence of an edge represents pairs of stocks for which the values are likely to be highly correlated (maybe two companies which produce the same product or rely on the same resource). Finding the largest portfolio of uncorrelated stocks in this example is exactly the maximum independent set problem.

To express the maximum independent set problem as an Ising model, we first need to think how to encode the concept of an independent set. To do this, first construct a two qubit Hamiltonian which penalizes (higher energy) the $|11\rangle$ state relative to the $|00\rangle$, $|01\rangle$, and $|10\rangle$ states. This constraint is achieved by

$$H_{2\,\text{independent}} = -\sigma^z \otimes I_2 - I_2 \otimes \sigma^z + \sigma^z \otimes \sigma^z = -\sigma_1^z - \sigma_2^z + \sigma_1^z \sigma_2^z, \quad (3)$$

where the second form drops the identity operators for brevity. This Hamiltonian ensures the lowest energy state does not contain two ones.

Consider a graph defined by the upper triangular adjacency matrix M, such that a 1 entry indicates an edge and a 0 entry indicates no edge, between

[3]Not to be confused with *maximal* independent set, which is not a hard problem.

corresponding vertices. If the qubits represent vertices, and state $|1\rangle$ indicates the vertex is coloured, we can use terms like Eq. (3) to penalise adjacent vertices that are both coloured. By defining $J = M$ and $h_k = -\sum_j (M_{kj} + M_{jk})$, a Hamiltonian of the form of Eq. (1) will enforce independent sets. In other words, when expressed as a bitstring, if the ones in a state form an independent set on the graph defined by M, they will get one energy, but they will get a higher energy if they do not. Try writing it out by hand for a three qubit example, such as o–o–o, see figure 3 in Appendix 2.

Finally, to construct a *maximum* independent set Hamiltonian, rather than just an independent set Hamiltonian, we need to give a lower energy to states with more qubits in the 1 state. To do this, we now set $h_k = -(\sum_j M_{kj} + M_{jk}) + \kappa$ (a greek kappa, not k) where $0 < \kappa < 1$.

There are many more complicated ways of encoding optimization problems into Ising Hamiltonians, but maximum independent set is one of the simplest, and gives a flavour of how this process works.

4 Time-dependent Hamiltonian simulation

For the milestone, you will be simulating the quantum adiabatic algorithm solving a small instance of maximum independent set. To start with, consider the graph depicted in Fig. 1, which is defined by the following adjacency matrix (see Appendix 2)

$$M = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}. \tag{4}$$

First, construct H_{Ising} for this graph, and verify that the lowest energy state is indeed the maximum independent set $|10011\rangle = |19\rangle$ when written using the convention that the first bit is the most significant. In vector form, this is a column vector of length 2^5 with a one in the 19^{th} position $(18\,\text{zeros}, 1, 13\,\text{zeros})^T$ (see Appendix 2).

Once you have verified that you have constructed the classical problem Hamiltonian correctly, the next step is to simulate an adiabatic algorithm solving the problem. To simulate the adiabatic algorithm, you have to simulate evolution with a time dependent Hamiltonian. This time evolution can be written as an infinite product of matrix exponentials:

$$\int_0^{t_{max}} Dt\, \mathcal{T} \exp\{-iH(t)\} = \lim_{q\to\infty} \mathcal{T} \prod_{j=1}^{q} \exp\left\{-i\frac{t_{max}}{q}\, H\left(\frac{j\, t_{max}}{q}\right)\right\} \tag{5}$$

where $\mathcal{T}$ indicates that the integral (or product) is time ordered[4]. If we are willing to accept some numerical error, we can set q to be large but not strictly

[4]Note that the $\frac{t_{max}}{q}$ comes from the term Dt outside of the exponent, this follows from the general definition of the Feynman path integral $\int_0^{t_{max}} Dt\mathcal{T} f(t) =$

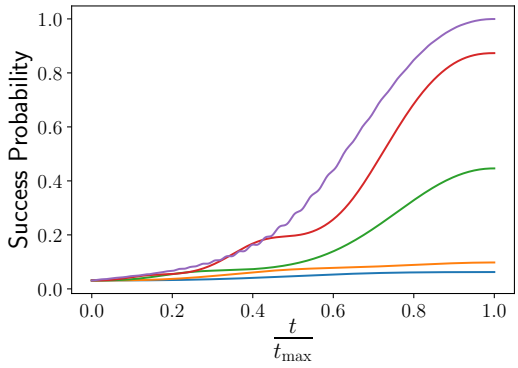

Figure 2: Success probability versus time for solving the maximum independent set problem on the graph given in Fig. 1 and Eq. (4) for different total runtimes $t_{\max} = 1$ (blue), $t_{\max} = 2$ (yellow), $t_{\max} = 5$ (green), $t_{\max} = 10$ (red), $t_{\max} = 100$ (magenta). This plot was created using $\kappa = \frac{1}{2}$.

infinite, thus obtaining a discrete version of the continuous-time evolution in a form convenient for numerical calculations. If we start in state $|\psi(t=0)\rangle$, then the state at time $\frac{k\,t_{max}}{q}$ can be written as

$$\left|\psi\left(t=\frac{k\,t_{\max}}{q}\right)\right\rangle \approx \mathcal{T}\prod_{j=1}^{k}\exp\left\{-i\frac{t_{\max}}{q}\,H\left(\frac{j\,t_{\max}}{q}\right)\right\}|\psi(t=0)\rangle\,, \qquad (6)$$

where we use the convention that each subsequent term of the product is multiplied to the left of the previous terms. Mathematically, exponentiating a matrix is well defined (at least for square matrices) through the power series expansion of the exponential function. Fortunately, Python has a very efficient function to exponentiate matrices: `scipy.linalg.expm`.

5 Milestone project

For the milestone you will simulate an adiabatic algorithm which solves the maximum independent set problem on the graph defined by the adjacency matrix given in Eq. (4). For this example, we will define $A(t) = 1 - \frac{t}{t_{\max}}$ and $B(t) = \frac{t}{t_{\max}}$ where $t_{\max}$ is the total runtime of the algorithm. Plot the probability of 'success', i.e., the probability that the measured result will be the maximum independent set, versus $\frac{t}{t_{\max}}$ for several different values of $t_{\max}$. This plot should include both values of $t_{\max}$ for which the success probability is barely changed from random guessing, values for which the success probability is greater than 0.95, and at least one value where the probability is somewhere

$\lim_{q\to\infty}\mathcal{T}\prod_{j=1}^{q} f(\frac{jt_{\max}}{q})^{\,t\,\frac{\max}{q}}$.

in between. Find an acceptable value of q by trial and error simulating Eq. (6), q does not have to scale with $t_{\max}$. Fig. 2 depicts the success probabilities versus time for different total runtimes, that you should aim to reproduce.

6 Milestone extensions

There are a wide variety of possible extensions which would be appropriate, including many not listed here. All of the listed ones are are suitable for level 3 computer projects, but those which are likely to be more challenging are marked with a †):

- Map other problems such as maximum 2 satisfiability using the mapping from this experimental paper [1] and † more complicated problems such as those described in [2].
- Apply sparse matrix techniques to simulate adiabatic quantum computing on larger systems using `scipy.sparse.linalg.expm_multiply`, possibly also looking at problems beyond maximum independent set.
- Investigate the effect of different annealing schedules, in other words different functional forms of $A(t)$ and $B(t)$ rather than just the linear example used for the milestone, for example:

$$A(\alpha, s', t) = \frac{1 - \frac{t}{t_{\max}}}{\alpha + (1-\alpha)\left(1 - \frac{t}{t_{\max}}\right)(1-s')^{-1}}$$
$$B(\alpha, s', t) = \frac{\frac{t}{t_{\max}}}{\alpha + (1-\alpha)\frac{t}{t_{\max}}(s')^{-1}}, \tag{7}$$

  where $0 < s' < 1$ controls where the algorithm slows down (it translates to a value of $\frac{t}{t_{\max}}$ in the original $A = \frac{t}{t_{\max}}$, $B = 1 - \frac{t}{t_{\max}}$ parametrization) and $0 < \alpha \leq 1$ controls how much it slows down. How does this affect the performance? Is it better to slow down when the energy gap between the ground and first excited state is large, or when it is small?
- Reproduce some of the results seen in early proof-of-principle problems for adiabatic quantum computing [3]. Note that some of these problems are not formulated as Ising models.
- Explore QAOA (quantum approximate optimisation algorithm) rather than adiabatic protocols. In QAOA, $-\sum_j \sigma_j^x$ and H_{Ising} are applied in an alternating fashion rather than simultaneously see: [4, 5]
- Explore quantum walks on graphs [6](† possibly also including marked states and decoherence [7]). Quantum walks have recently been considered as a tool to solve optimization problems [8], but more results are known for walks on graphs.

- † Write code to perform path integral quantum annealing (PIQA) [9], which can be applied to much larger problems than the matrix simulation methods used here. Potentially reproduce a version of the PIQA plots in [10].

Warning: the use of the terms *adiabatic quantum computing* versus *quantum annealing* is not standardized in the literature, and different authors use these terms differently. Read carefully, to understand what is actually being done in a paper.

References

[1] S. Santra, G. Quiroz, G. Ver Steeg, and D. A. Lidar *Max 2-SAT with up to 108 qubits* New Journal of Physics, 16, **4** pp. 045006 (2014).

[2] V. Choi *Different Adiabatic Quantum Optimization Algorithms for the NP-Complete Exact Cover and 3SAT Problems* arXiv:1010.1221 (2010).

[3] E. Farhi, J. Goldstone, S. Gutmann, and M. Sipser *Quantum Computation by Adiabatic Evolution*, arXiv:quant-ph/0001106 (2000).

[4] E. Farhi, J. Goldstone, S. Gutmann *A Quantum Approximate Optimization Algorithm*, arXiv:1411.4028 (2014).

[5] S. Hadfield *Quantum Algorithms for Scientific Computing and Approximate Optimization* (PhD. thesis) arXiv:1805.03265 (2018).

[6] B. Tregenna, W. Flanagan, R. Maile, and V. Kendon *Controlling discrete quantum walks: coins and initial states* New Journal of Physics, 5, pg. 83 (2003).

[7] V. Kendon *Decoherence in quantum walks – a review* Mathematical Structures in Computer Science, 17, **6** pp. 1169-1220, (2007).

[8] A. Callison, N. Chancellor, F. Mintert, V. Kendon *Finding spin-glass ground states using quantum walks* arXiv:1903.05003 (2019).

[9] R. Martonak, G. E. Santoro, and E. Tosatti *Quantum annealing by the path-integral Monte Carlo method: the two-dimensional random Ising model* Phys. Rev. B **66** 094203 (2002).

[10] N. Chancellor *Modernizing quantum annealing using local searches* New Journal of Physics 19, **2** 023024 (2017).

Appendix 1: A review of tensor products

Tensor products provide a powerful mathematical tool for a range of physics models. The basic idea behind a tensor product is that it separates operations occurring on different degrees of freedom of a physical system. Tensor products are required to complete the milestone. Note that the Python function `numpy.kron` performs tensor products. Terminology in Python for matrix operations is not the same as usually used in physics, and you should always check that it is actually doing what you want. In particular, matrix multiplication using * is usually elementwise; you need to use something like `numpy.dot` to do what we consider normal matrix multiplication.

While Python has functions to perform all the operations you need (use them, they are very efficient implementations), it is necessary to understand how they work, in order to check that your code is correct. Effectively, the action of the tensor product $a \otimes b$ is to replace each element of a, a_{jk} with the matrix $a_{jk} \times b$. Since each element is replaced by a matrix, the total size of the resulting matrix is length$(a) \times$length(b). Unlike standard matrix multiplication, where the multiplication dimension needs to match, tensor products can be performed between pairs of matrices (or vectors) of any size. Like all types of multiplication, tensor products are *associative*, $a \otimes b \otimes c = a \otimes (b \otimes c) = (a \otimes b) \otimes c$.

As an example, consider the smallest non-trivial matrix tensor product, which is a tensor product of two 2×2 matrices, which in this case can be thought of as representing quantum mechanical operators on qubits (blue states added as a visual aid) :

$$a \otimes b = \begin{matrix} \\ \langle 0| \\ \langle 1| \end{matrix} \begin{matrix} \begin{matrix} |0\rangle & |1\rangle \end{matrix} \\ \begin{pmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{pmatrix} \end{matrix} \otimes \begin{matrix} \\ \langle 0| \\ \langle 1| \end{matrix} \begin{matrix} \begin{matrix} |0\rangle & |1\rangle \end{matrix} \\ \begin{pmatrix} b_{00} & b_{01} \\ b_{10} & b_{11} \end{pmatrix} \end{matrix} =$$

$$\begin{matrix} \\ \langle 00| \\ \langle 01| \\ \langle 10| \\ \langle 11| \end{matrix} \begin{matrix} \begin{matrix} |00\rangle & |01\rangle & |10\rangle & |11\rangle \end{matrix} \\ \begin{pmatrix} a_{00}b_{00} & a_{00}b_{01} & a_{01}b_{00} & a_{01}b_{01} \\ a_{00}b_{10} & a_{00}b_{11} & a_{01}b_{10} & a_{01}b_{11} \\ a_{10}b_{00} & a_{10}b_{01} & a_{11}b_{00} & a_{11}b_{01} \\ a_{10}b_{10} & a_{10}b_{11} & a_{11}b_{10} & a_{11}b_{11} \end{pmatrix} \end{matrix} \tag{8}$$

The action of the tensor product denoted by $\otimes$ is to combine the spaces which the operator acts on, such that the indices of the first matrix become the first digit of the binary number representing the state and the second becomes the second. In this way, a sort of composite representation can be constructed where the state of two qubits can be written as a single state vector $|\alpha\beta\rangle = |\alpha\rangle \otimes |\beta\rangle$. Each degree of freedom can be addressed independently using the tensor product, for instance, if we replace matrix a in Eq. (8) with a 2×2 identity matrix, the

resulting matrix

$$\begin{array}{c} \\ \langle 00| \\ \langle 01| \\ \langle 10| \\ \langle 11| \end{array} \begin{array}{c} \begin{array}{cccc} |00\rangle & |01\rangle & |10\rangle & |11\rangle \end{array} \\ \begin{pmatrix} b_{00} & b_{01} & 0 & 0 \\ b_{10} & b_{11} & 0 & 0 \\ 0 & 0 & b_{00} & b_{01} \\ 0 & 0 & b_{10} & b_{11} \end{pmatrix} \end{array} \tag{9}$$

can only flip or apply phase differences to the second qubit, similarly, if b were replaced by an identity the resulting matrix,

$$\begin{array}{c} \\ \langle 00| \\ \langle 01| \\ \langle 10| \\ \langle 11| \end{array} \begin{array}{c} \begin{array}{cccc} |00\rangle & |01\rangle & |10\rangle & |11\rangle \end{array} \\ \begin{pmatrix} a_{00} & 0 & a_{01} & 0 \\ 0 & a_{00} & 0 & a_{01} \\ a_{10} & 0 & a_{11} & 0 \\ 0 & a_{10} & 0 & a_{11} \end{pmatrix} \end{array} \tag{10}$$

can only act on the first qubit.

This generalises for tensor products of more degrees of freedom. The matrices quickly become unwieldy to write out explicitly, but computers can store and manipulate very large matrices. An operation a on the jth qubit of an n qubit system can be written $a_j = \left(\bigotimes_{k=1}^{j-1} I_2\right) \otimes a \otimes \left(\bigotimes_{k=j+1}^{n} I_2\right)$ where I_2 is a 2×2 identity matrix and $\bigotimes_{k=1}^{j-1}$ indicates repeated tensor products in the same way $\prod_{k=1}^{j-1}$ would represent repeated multiplication. In particular $\bigotimes_{k=n-q+1}^{n} I_2 = I_2 \otimes I_2 ... (\text{total of q times}) ... \otimes I_2 = I_2^{\otimes q}$.

Any Hermitian operator of size 2^n can be constructed from sums and products of Pauli operations on different sites $\sigma_j^{\{x,y,z\}}$ plus the identity operation. The operational meaning of $\sigma_j^{\{x,y,z\}}$ is to perform a Pauli $\{x, y, z\}$ operation on the jth qubit while doing nothing (the identity) to the others, simultaneous operators on qubits j and k where $j \neq k$ can be represented as $\sigma_j^{\{x,y,z\}} \sigma_k^{\{x,y,z\}}$. Since these Pauli operations are acting on different qubits they will commute, i.e., $\left[\sigma_j^{\{x,y,z\}}, \sigma_{k \neq j}^{\{x,y,z\}}\right] = 0$.

Coding tip: write a function (or three) to create $\sigma_j^{\{x,y,z\}}$ once, and call it in later functions, rather than trying to 'hard code' the creation of each term from tensor products. Such functions can be written using less than 10 lines of code, if written well.

Appendix 2: Graphs, adjacency matrices, and Hamiltonians

Graphs are formal mathematical objects consisting of vertices (usually represented visually by circles) connected by edges (usually represented by line segments). Graphs can be drawn on a two dimensional plane by choosing (x, y)

coordinates for the vertices and drawing appropriate edges between them. This positioning of the vertices is important for human understanding, but is not part of the definition of the graph itself. Rearranging the vertices keeps the graph the same. The power of graphs is that they allow an abstract representation of the relationships between different entities, a family tree for instance is one type of (directed) graph which provides information about ancestry and lineage. In general graphs can have different types of edges (relationships between vertices) or weights assigned to each edge, and can either be directed (relationship has a direction, e.g., parent to child in family tree) or undirected. The maximum independent set problem is defined on a very simple class of graphs where edges are undirected and for which there is only one type of (unweighted) edge, vertices can only be connected by 0 or 1 edges, and vertices are not allowed to connect to themselves (no 'self loops').

While graphs provide a powerful tool for humans to visualize real problems and systems, they are not a way of representing information which computers can easily operate on directly. Fortunately, since a graph is defined by the connections between vertices, rather than the positioning of those vertices, a graph can efficiently be represented by a matrix, and matrices can be efficiently manipulated by computers. Consider the simple graphs which are used to define maximum independent sets. Each pair of vertices either share an edge or do not share an edge, therefore the information contained in this kind of graph with n vertices can be expressed as $n\,(n-1)$ binary variables (numbers which can only be 0 or 1). In practice however, it is more convenient to organize these variables into an $n \times n$ *adjacency matrix*, an array of numbers which has zeros for everything below the diagonal and has ones in the $(j, k > j)$ position if there is an edge between two vertices and zero otherwise. Note that, depending on the application, adjacency matrices may be defined with entries below the diagonal, too, for traversing the edge from k to j. The upper triangular definition is convenient for application to maximum independent set Hamiltonians, but we could have used either convention.

To construct an adjacency matrix, one must assign labels in some order to the vertices, as was done in Fig. 1 a). This ordering is arbitrary, but once chosen must be used consistently. The matrix in Eq. (4) is reproduced below, with the rows and columns labelled as a reference

$$M = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array}. \tag{11}$$

As an example, if we look in the first row and second column, there is a 1 entry in this matrix, and indeed, there is an edge between vertex 1 and vertex 2 in Fig. 1. On the other hand, there is a zero in the first row and fourth

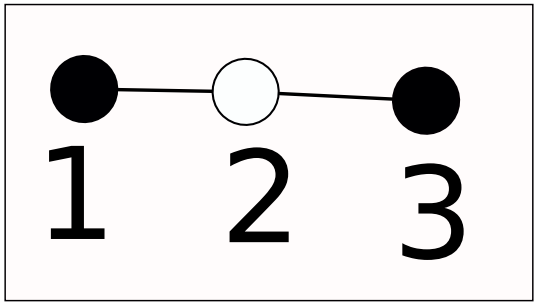

Figure 3: A simple graph with three vertices, coloured to show maximum independent set and numbered to match Eq. (12).

column, and there is indeed no edge shared between vertex 1 and vertex 4. From this graph, the vector h and matrix J, which define the problem in the Ising Hamiltonian, can be defined using Eq. (3). Mathematically, this Hamiltonian can be expressed as a $2^n \times 2^n$ matrix, although for large problems this matrix will not be practical to construct simply because of how large it is. Already for the maximum independent set problem defined in the main text, the matrix will be $2^5 \times 2^5 = 32 \times 32$, rather large to write out explicitly on the page.

Instead, consider a three qubit problem defined by the graph depicted in Fig. 3, which has the adjacency matrix

$$M' = \begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} \end{array} \tag{12}$$

. The Hamiltonian for the maximum independent set is therefore represented by

$$J = \begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} \end{array}, \quad h = \begin{array}{c} 1 \\ 2 \\ 3 \end{array} \begin{pmatrix} -1+\kappa \\ -2+\kappa \\ -1+\kappa \end{pmatrix}. \tag{13}$$

The total 8×8 problem Hamiltonian for this problem can be expressed as

$$H_{\text{problem}} =$$

		$\|0\rangle$	$\|1\rangle$	$\|2\rangle$	$\|3\rangle$	$\|4\rangle$	$\|5\rangle$	$\|6\rangle$	$\|7\rangle$
		$\|000\rangle$	$\|001\rangle$	$\|010\rangle$	$\|011\rangle$	$\|100\rangle$	$\|101\rangle$	$\|110\rangle$	$\|111\rangle$
$\langle 0\|$	$\langle 000\|$	$-2+3\kappa$	0	0	0	0	0	0	0
$\langle 1\|$	$\langle 001\|$	0	$-2+\kappa$	0	0	0	0	0	0
$\langle 2\|$	$\langle 010\|$	0	0	$-2+\kappa$	0	0	0	0	0
$\langle 3\|$	$\langle 011\|$	0	0	0	$2-\kappa$	0	0	0	0
$\langle 4\|$	$\langle 100\|$	0	0	0	0	$-2+\kappa$	0	0	0
$\langle 5\|$	$\langle 101\|$	0	0	0	0	0	$-2-\kappa$	0	0
$\langle 6\|$	$\langle 110\|$	0	0	0	0	0	0	$2-\kappa$	0
$\langle 7\|$	$\langle 111\|$	0	0	0	0	0	0	0	$6-3\kappa$

(14)

Note that it is diagonal, because it has been constructed entirely from σ_z operators plus identity terms. The σ_x terms in the transverse field in Eq. (2) will appear as off-diagonal entries (see below). The bras and kets labelling the rows and columns depict two different ways of expressing the states, either by listing the states of each of the three qubits (blue), or by converting this state to a decimal number (red). For visual clarity, the lowest energy eigenvalue has been coloured magenta (recall that $0 < \kappa < 1$). By examining the problem Hamiltonian, we can observe that the maximum independent set is the state $|101\rangle = |5\rangle = (0,0,0,0,1,0,0)^T$, where the last expression is the state expressed as a vector. Finally, the total Hamiltonian for the adiabatic evolution Hamiltonian can be constructed, setting $\kappa = \frac{1}{2}$ and plugging into Eq. (2), we obtain:

$$H(t)_{\text{total}} =$$

	$\|000\rangle$	$\|001\rangle$	$\|010\rangle$	$\|011\rangle$	$\|100\rangle$	$\|101\rangle$	$\|110\rangle$	$\|111\rangle$
$\langle 000\|$	$-\frac{1}{2}B(t)$	$-A(t)$	$-A(t)$	0	$-A(t)$	0	0	0
$\langle 001\|$	$-A(t)$	$-\frac{3}{2}B(t)$	0	$-A(t)$	0	$-A(t)$	0	0
$\langle 010\|$	$-A(t)$	0	$-\frac{3}{2}B(t)$	$-A(t)$	0	0	$-A(t)$	0
$\langle 011\|$	0	$-A(t)$	$-A(t)$	$\frac{3}{2}B(t)$	0	0	0	$-A(t)$
$\langle 100\|$	$-A(t)$	0	0	0	$-\frac{3}{2}B(t)$	$-A(t)$	$-A(t)$	0
$\langle 101\|$	0	$-A(t)$	0	0	$-A(t)$	$-\frac{5}{2}B(t)$	0	$-A(t)$
$\langle 110\|$	0	0	$-A(t)$	0	$-A(t)$	0	$\frac{3}{2}B(t)$	$-A(t)$
$\langle 111\|$	0	0	0	$-A(t)$	0	$-A(t)$	$-A(t)$	$\frac{9}{2}B(t)$

(15)