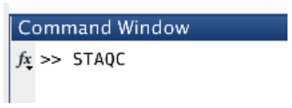
PHYC90045 Introduction to Quantum Computing

Lab Session 9

9.1 Introduction

Welcome to Lab-9 of PHYC90045 Introduction to Quantum Computing. The purpose of this lab session is to get some experience with Adiabatic Quantum Computing (AQC) through the use of a Matlab based simulator STAQC (AQC simulator written by Sam Tonetto) and the D-Wave devices.

Instructions to access STAQC:
Download the "STAQC files" files from LMS. Open Matlab and navigate to directory with files. Type in "STAQC" in console. Refer to Appendix 1 in these lab notes for documentation.



```
>> STAQC
```

9.2 Single-qubit system: eigenstates and eigenvalues

Before we embark on solving problems using the Adiabatic Quantum Computing (AQC) framework with STAQC, we need to review a few basics about eigenvalues and eigenvectors of operators. The eigensystem for a 2x2 Hermitian matrix M is defined as: $M\chi = E\chi$ where χ is a 2x1 column vector (the "eigenvector", or "eigenstate" in QC parlance), and E is a number (eigenvalue), which is real when M is Hermitian (the case we consider here). In the case of a 2x2 matrix M there will be 2 eigenvectors $\chi_{1,2}$ with corresponding eigenvalues $E_{1,2}$. When the eigenvalues are the same (distinct eigenvectors), the system is called "degenerate". (NB. We use "E" for the eigenvalues because in terms of the physics, the eigenvalues of a Hamiltonian (energy) operator, which is what our M matrix will represent, are the allowed energies of the system).

Suppose, we have a single-qubit system described by a Hamiltonian operator involving the Z operation only, i.e. we write $H_P = Z$ (we will use H_P to denote what will be the Hamiltonian operator encoding the "problem" to be solved (refer to lectures)). Let's solve the "eigensystem" $H_P\chi = E\chi$, i.e. $Z\chi = E\chi$. We already know the eigenvalues and eigenvectors from lectures – here they are using this notation.

Hamiltonian: $H_P = Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$

Eigensystem: $Z\chi = E\chi$

$$\chi_1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \qquad Z\chi_1 = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

$$\text{i.e. } Z\chi_1 = \chi_1 \rightarrow E_1 = +1$$

$$\chi_2 = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \qquad Z\chi_2 = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}\begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ -1 \end{pmatrix}$$

$$\text{i.e. } Z\chi_2 = -\chi_2 \rightarrow E_2 = -1$$

In this example, χ_2 has the lowest eigenvalue (energy -1). We can re-cast into our ket notation which should look familiar:

$$|0\rangle \to \begin{pmatrix} 1 \\ 0 \end{pmatrix} \qquad |1\rangle \to \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

$$Z\,|0\rangle = +\,|0\rangle \qquad Z\,|1\rangle = -\,|1\rangle$$

Let's look at another Hamiltonian operator that we will use in this lab, $H_P = X$. We have seen the eigensystem for this as well in lectures:

$$H_P|\pm\rangle \to X|\pm\rangle = \pm|\pm\rangle$$

where $|\pm\rangle = (|0\rangle \pm |1\rangle)/\sqrt{2}$. So the eigenvectors and eigenvalues of X are $\chi_{1,2} \to |\pm\rangle$ and $E_{1,2} \to \pm 1$. Here, the state $|-\rangle$ has the lowest energy.

Now we will make a Hamiltonian H(s), depending on a parameter s, which is a simple linear combination of X and Z :

$$H(s) = (1-s)H_X + s\,H_P$$
$$H_X = X$$
$$H_P = Z$$

Clearly, at s = 0, we have $H(0) = H_X$ and at s = 1 we have $H(1) = Z$. Here, s is our "scaled" time, i.e. s = t/T, where T will be the total time of the AQC evolution (later) where we evolve from $H(0) = H_X$ to our "problem" Hamiltonian $H(1) = H_P$ (in this case $H_P = Z$). Let's find the eigenstates of $H(s)$ for any s, i.e. in ket notation we want to solve:

$$H(s)|\chi\rangle = E|\chi\rangle$$

Notes:

- The Energies will be dependent on s, i.e. E(s), but we usually supress that.
- For eigenvalues, we write E_i instead of the more customary λ_i, because the eigenvalues of a Hamiltonian are its energies.
- Hamiltonians are Hermitian, not unitary. They are not a quantum gate. (when we get to the QAOA lab we will see the difference more explicitly).

Exercise 9.2.1

Generally, to solve an eigensystem we first calculate the eigenvalues through the characteristic equation, and then find the eigenvectors. To calculate the eigenvalues, we can re-arrange the above equation in matrix form as:

$$(H(s) - E)\chi = 0$$

and then solve the determinant (characteristic) equation $\det((H(s) - E)) = 0$.

i.e. $\begin{vmatrix} [H(s)]_{11} - E & [H(s)]_{12} \\ [H(s)]_{21} & [H(s)]_{22} - E \end{vmatrix} = 0$ where in matrix form we have $H(s) = \begin{bmatrix} [H(s)]_{11} & [H(s)]_{12} \\ [H(s)]_{21} & [H(s)]_{22} \end{bmatrix}$.

This is a quadratic equation for E with two solutions, $E_{1,2}$. Using the fact that $\begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc$, the eigenvalues of $H(s) = (1-s)X + s\,Z$ as a function of s are (to check your understanding you can verify this):

E_1	$-\sqrt{1-2s+2s^2}$
E_2	$\sqrt{1-2s+2s^2}$

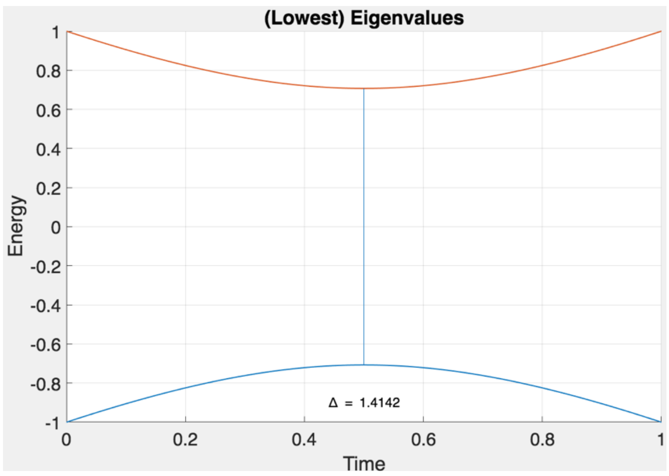
(Here E_1 is the defined as the lowest eigenvalue)

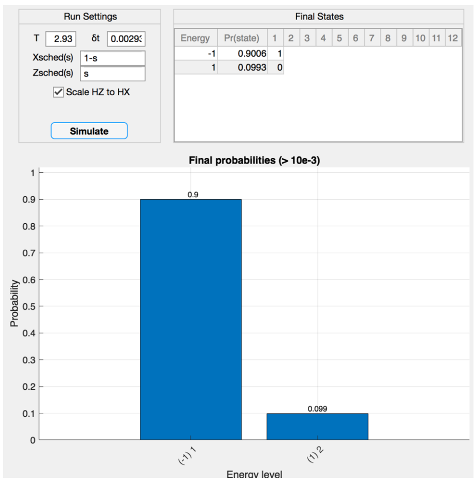
Exercise 9.2.2
Hence, find $\Delta = \min(E_2(s) - E_1(s))$, the minimum gap between the two eigenvalues of $H(s)$ at s = 1/2. (the answer is $\Delta = \sqrt{2}$).

Exercise 9.2.3
Using the eigenvalues $E_{1,2}$ you have found, verify that the eigenstates of $H(s)$ at $s = \frac{1}{2}$ are:

$\vert\chi_1\rangle$	$\frac{1-\sqrt{2}}{\sqrt{4-2\sqrt{2}}}\vert 0\rangle + \frac{1}{\sqrt{4-2\sqrt{2}}}\vert 1\rangle$
$\vert\chi_2\rangle$	$\frac{1+\sqrt{2}}{\sqrt{4+2\sqrt{2}}}\vert 0\rangle + \frac{1}{\sqrt{4+2\sqrt{2}}}\vert 1\rangle$

Exercise 9.2.4
STAQC assumes that the Hamiltonian is of the form $H(s) = (1-s)H_X + s\,H_P$ (H_P is synonymous with "HZ" in the STAQC definition), so we only need to specify H_P (refer to appendix). Program the problem Hamiltonian $H_P = Z$ into the STAQC, run with default parameters (T = 1, $\delta t = 0.001$) and verify the minimum gap you have calculated.




Run Settings

T 2.93 δt 0.0029

Xsched(s) 1-s

Zsched(s) s

[✓] Scale HZ to HX

Final States

Energy	Pr(state)	1	2	3	4	5	6	7	8	9	10	11	12
-1	0.9006	1											
1	0.0993	0											

Using your solutions for the states, verify the probability distribution given by STAQC for the system at $t = T = 1$ ("Final" state, corresponding to $s = 1$).

<u>9.3 Two-Qubit Hamiltonian</u>

In AQC for multi-qubit systems we stick to the same basic form of the Hamiltonian, $H(s) = (1-s)H_X + s\,H_P$. First, we will extend to a simple example of two qubits, before we start contemplating a practical problem Hamiltonian case. For two qubits, our matrix representation will be 4x4, and there will be in general 4 eigenvalues, some of which may be degenerate (i.e. the same value).

Exercise 9.3.1

Consider the two-qubit Hamiltonian:

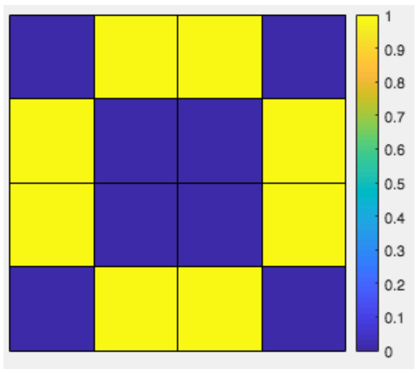
$$\begin{aligned} &H(s) = (1-s)H_X + s\,H_P \\ &H_X = X_1 + X_2 \\ &H_P = Z_1 + Z_2 \end{aligned}$$

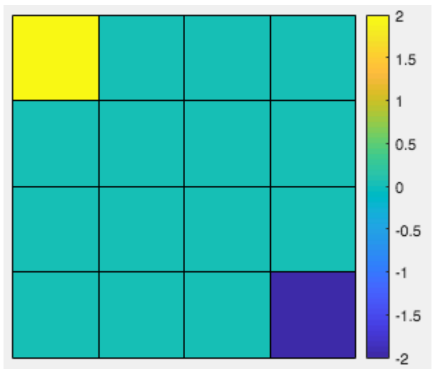
Noting that $H_X = X_1 + X_2$ is really $H_X = X \otimes I + I \otimes X$ etc, we use the definition of outer products to get the matrices corresponding to H_X and H_P:

$$H_X = X\otimes I + I\otimes X = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

$$H_P = Z\otimes I + I\otimes Z = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -2 \end{bmatrix}$$

Program this H_P into STAQC and verify your matrices for H_X and H_P – you should obtain:





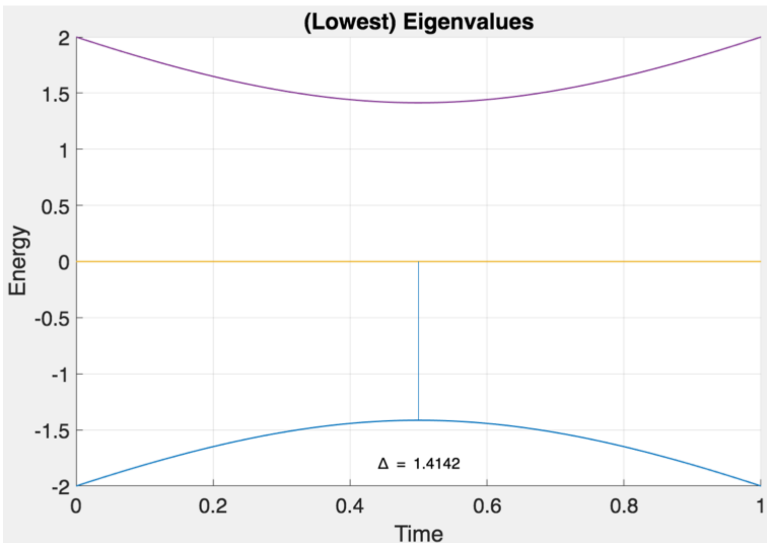
Left: HX: $H_X = X\otimes I + I\otimes X$ Right: HP=HZ: $H_P = Z\otimes I + I\otimes Z$

Compare with your matrices (NB pay attention to the colour scales).

Exercise 9.3.2
The eigenvalues of $H(s)$ are, in ascending order, given by:

E_1	$-2\sqrt{1-2s+2s^2}$
E_2	0
E_3	0
E_4	$2\sqrt{1-2s+2s^2}$

(can you derive these?). Simulate the problem on STAQC with default parameters (T = 1, $\delta t = 0.001$) to verify the eigenvalues. STAQC produces all the eigenvalues as function of s, from s = 0 to s = 1, i.e.



Note again that "Time" is the actual time t = sT. This is important, as we will now see.

9.4 Introducing evolution

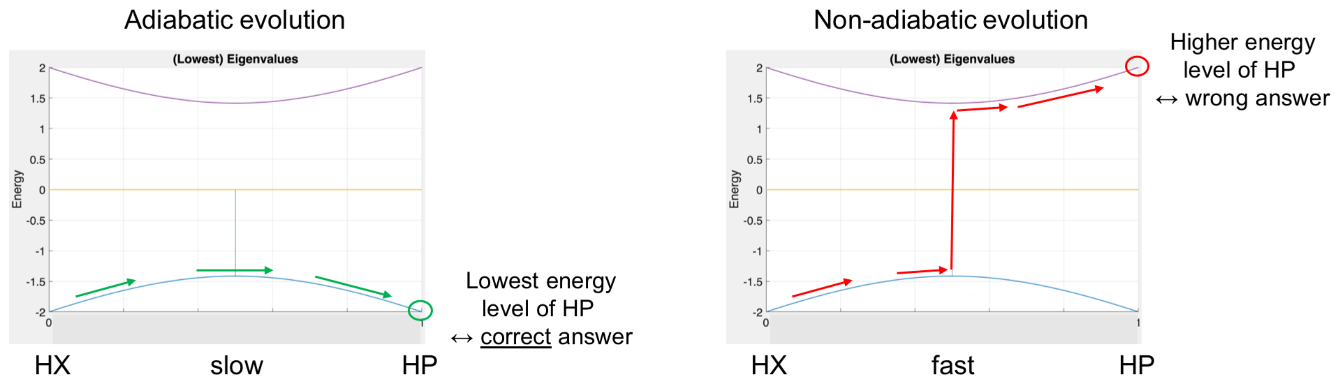
Now that you understand the eigenspectrum of energies (eigenvalues) as a function of the scaled "time" parameter s, we will look at the real magic of AQC – adiabatic evolution. The thing we are trying to do it evolve the Hamiltonian system from time t=0 (s=0), to the final time t=T (s=1) so that:

$$H(s=0) = H_X \rightarrow H(s=1) = H_P$$

Generally we encode the answer to our original problem in the lowest eigenvalue (energy) state of H_P. While we can calculate that eigenvalue for small numbers of qubits (as we do in STAQC), for bigger systems that becomes impossible on a conventional computer (e.g. N ~ 40 is about the limit), hence the promise of AQC as an alternative form of QC.

Remember from lectures: *adiabatic* means varying slowly. That is, if we vary the parameter s slowly enough the system will evolve staying in or close to the lowest eigenvalue (energy) state. *Hence, when we get to the end of the adiabatic evolution at s=1 we should be in* *the lowest state of the H_P system and can read off the values of each qubit to get our problem solution (see figure below, left).*

But – how slow is slow enough? Basically, if the minimum gap is small it is easier it is to bump the system into the next highest (energy) eigenvalue state by going too fast…and hence when you get to the end of the evolution and read off the qubit values you won't be in the right solution state (see below, right). The gap is an important factor (but not the only one as we will see) in defining how slow the evolution needs to be in order to be in the adiabatic regime, and hence the value of T we must set.

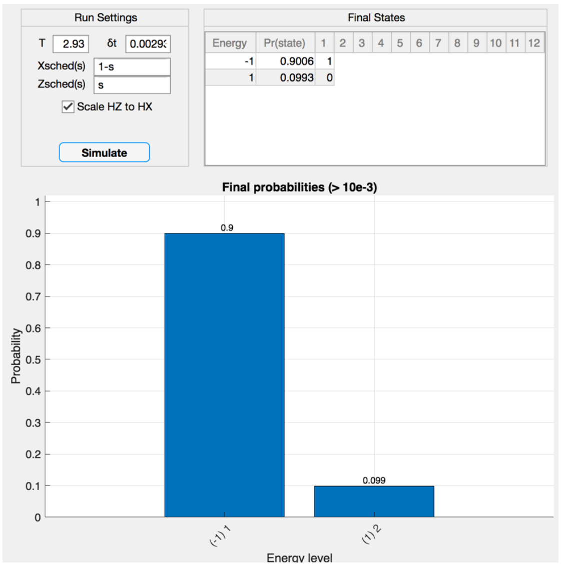


Exercise 9.4.1

To see what happens when we evolve the system too quickly, or just slow enough, let's go back to the 1 qubit case:

$$H(s) = (1-s)H_X + s\,H_P$$
$$H_X = X$$
$$H_P = Z$$

Program this system into STAQC and run the simulation for values of T from small to large. Inspect the results of the evolution by looking at the "Final Histogram" tab. How large must the timescale T be to get >90% probability of being in the ground-state of H_P at the end of the evolution (s = 1)? i.e. the plot would look something like this:



Run Settings

T: 2.93 δt: 0.0029:

Xsched(s): 1-s

Zsched(s): s

☑ Scale HZ to HX

Final States

Energy	Pr(state)	1	2	3	4	5	6	7	8	9	10	11	12
-1	0.9006	1											
1	0.0993	0											

Exercise 9.4.2
Try this for the two qubit Hamiltonian in section 9.3.

9.5 Introducing couplings

Now that we can handle more than 1 qubit and evolved the system, we can introduce the next component that gets us closer to programming and solving real problems in the AQC framework.

Exercise 9.5.1
In STAQC, program in the problem Hamiltonian

$$H_P = Z_1 + 2Z_2 + 2Z_1 Z_2$$

(remember to think about this in the outer-product framework, which we suppress here for simplicity). STAQC will now run the eigensystem based on $H(s) = (1-s)H_X + s\, H_P$ with $H_X = X_1 + X_2$ as the default. Run the simulation with default parameters and study the results. What is the minimum gap? (answer: 0.633)

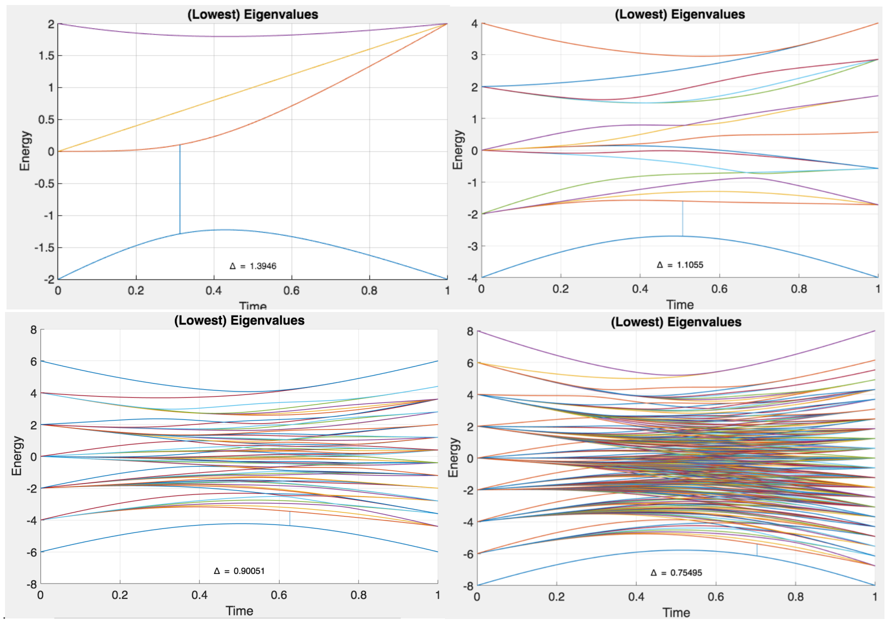
Exercise 9.5.2
For the following Hamiltonians (with non-degenerate ground states), with increasing numbers of qubits, program in STAQC and tabulate the minimum gap:

$$H_P = -Z_1 + Z_2 + Z_1 Z_2$$
$$H_P = -Z_1 + Z_2 - 2Z_3 + 2Z_4 + \sum_{i=1}^{4}\sum_{j>i} Z_i Z_j$$
$$H_P = -Z_1 + Z_2 - 2Z_3 + 2Z_4 - 3Z_5 + 3Z_6 + \sum_{i=1}^{6}\sum_{j>i} Z_i Z_j$$
$$H_P = -Z_1 + Z_2 - 2Z_3 + 2Z_4 - 3Z_5 + 3Z_6 - 4Z_7 + 4Z_8 + \sum_{i=1}^{8}\sum_{j>i} Z_i Z_j$$

#Qubits	Min Gap	T to achieve >90% (use $\delta t = T/100$)
2	1.3946	
4	1.1055	
6	0.90051	
8	0.75495	

Marvel at how complicated the eigenvalue system becomes as the number of qubits grows (especially with couplings). Remember, STAQC only plots the lowest set of eigenvalues (energies) of the system as these are the relevant ones for the AQC computations. For N qubits, how many eigenvalues are there in total?

For the systems given above, you should see the following "eigen-spectra":



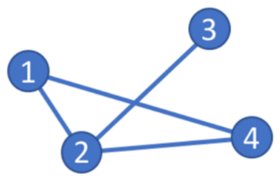
9.6 Graph problems in AQC

Luckily we have covered this in the lectures. Recall that the Hamiltonian for the Graph Partitioning problem is:

$$H_P = A\left(\sum_i Z_i\right)^2 + B\left(\sum_{\langle ij\rangle} \frac{I - Z_i Z_j}{2}\right)$$

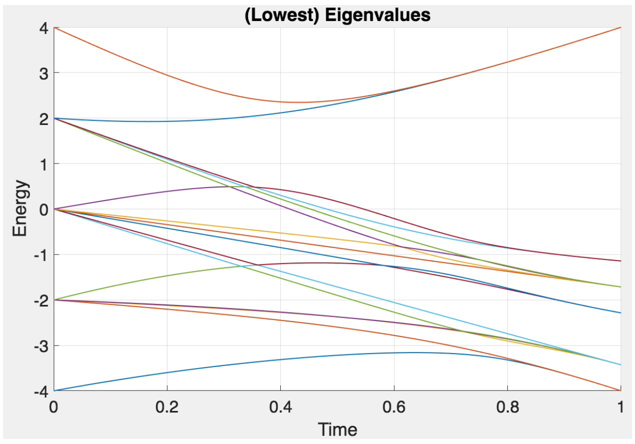
For A=1, B=1, the problem Hamiltonian for the graph shown right is

$$H_p = \frac{3}{2}(Z_1Z_2 + Z_2Z_4 + Z_1Z_4 + Z_2Z_3) + 2Z_1Z_3 + 2Z_3Z_4$$



Exercise 9.6.1
Derive the problem Hamiltonian for the graph partitioning problem above, understand how the solution is represented. Program this into STAQC, run it with default parameters, inspect the outputs, compare with the solution(s).

NB. In the eigenspectrum, note that a higher eigenvalue curve starting at E = -2 in H_X ends up in the ground-state of H_P, because the solution is doubly-degenerate. Although this might seem to imply that the minimum ground-excited gap is 0, it isn't that simple when H_P has degenerate ground-states (multiple valid solutions). It is usually the case that eigenvalue curves that end in degenerate ground-states of H_P are associated with eigenstates that do not couple to each other. So the minimum gap in this context would be between the ground-state eigenvalues and the lowest eigenvalue curve that doesn't end in the ground energy.



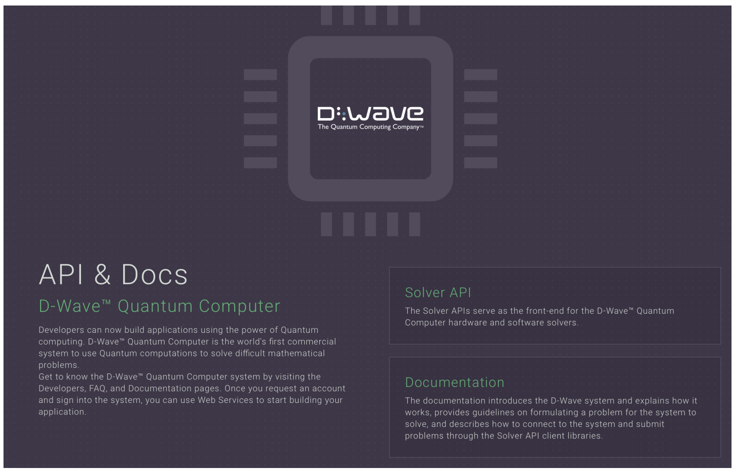
9.7 Programming AQC problems on D-Wave Machines

D-Wave is a company that commercialised the first quantum processor technology focussing on the AQC or "quantum annealing" model (Dwavesys.com). The physical hardware is based on superconducting qubits, similar to IBMQ, but are comparatively noisy. Because the quantum coherence times of the qubits are short relative to the computation (evolution) time it is not yet known whether such systems can outperform a classical computer – a question which is an active area of research complicated not just by the degree of quantum coherence and noise in the system, but the connectivity of the qubit array itself. Nonetheless, D-Wave systems are not just an engineering marvel (systems have reached 2000 qubits and counting), but their very existence has spurred a huge amount of work in learning how to program problems on such machines. The very fact that you will be able to access and program a D-Wave system on-line in this teaching lab is testament to the sophistication of the AQC approach and the generosity of D-Wave.

We will first organise you into small groups (3-4 students): we have 7 D-Wave accounts for students, corresponding accounts of the form PHYC90045_**Group1**@ph.unimelb.edu.au. When you do anything on D-Wave an authentication token corresponding to the account will be sent to our 90045 central physical email – your group will need to let us know when an API token has been sent on your group's behalf (i.e. let us know when your group is about to sign-in).

To start using the D-Wave systems you would normally go first to the "Leap" interface (www.dwavesys.com/take-leap), but you're a bit more advanced. We will connect directly with the preparatory STAQC work we have done and go straight to the "qubist" page.

Go to "qubist" page at https://cloud.dwavesys.com/qubist.



API & Docs

D-Wave™ Quantum Computer

Developers can now build applications using the power of Quantum computing. D-Wave™ Quantum Computer is the world's first commercial system to use Quantum computations to solve difficult mathematical problems.
Get to know the D-Wave™ Quantum Computer system by visiting the Developers, FAQ, and Documentation pages. Once you request an account and sign into the system, you can use Web Services to start building your application.

Solver API

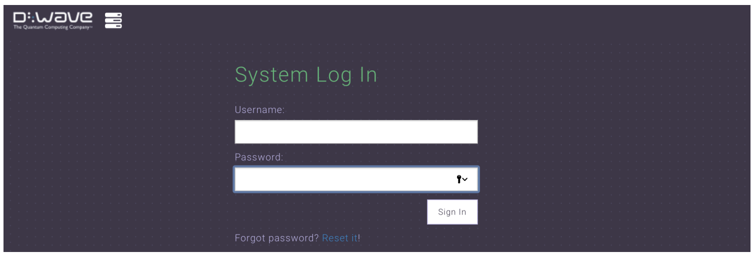
The Solver APIs serve as the front-end for the D-Wave™ Quantum Computer hardware and software solvers.

Documentation

The documentation introduces the D-Wave system and explains how it works, provides guidelines on formulating a problem for the system to solve, and describes how to connect to the system and submit problems through the Solver API client libraries.

Click on "Solver API"

Login with your Group's email address, get password from Lab tutor, authentication will be required.



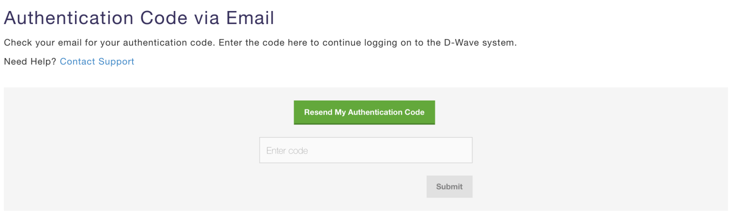
System Log In

Username:

Password:

Sign In

Forgot password? Reset it!



Authentication Code via Email

Check your email for your authentication code. Enter the code here to continue logging on to the D-Wave system.

Need Help? Contact Support

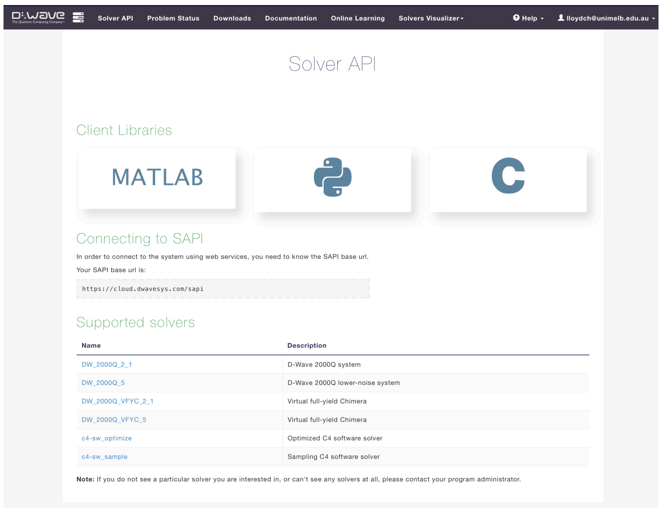
Resend My Authentication Code

Enter code

Submit

Tell the Lab tutor you're about to Click Resend Authentication Code. Get the authentication code from the Lab tutor. Login.

You should be at this page.



Solver API

Client Libraries

Connecting to SAPI

In order to connect to the system using web services, you need to know the SAPI base url.

Your SAPI base url is:

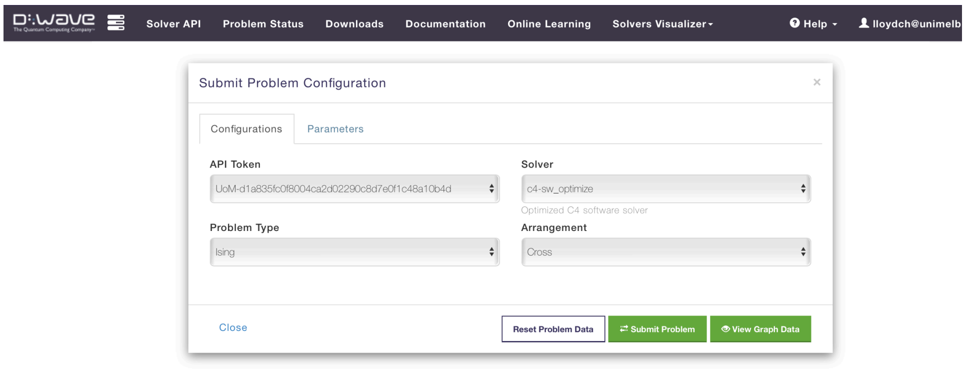
```
https://cloud.dwavesys.com/sapi
```

Supported solvers

Name	Description
DW_2000Q_2_1	D-Wave 2000Q system
DW_2000Q_5	D-Wave 2000Q lower-noise system
DW_2000Q_VFYC_2_1	Virtual full-yield Chimera
DW_2000Q_VFYC_5	Virtual full-yield Chimera
c4-sw_optimize	Optimized C4 software solver
c4-sw_sample	Sampling C4 software solver

Note: If you do not see a particular solver you are interested in, or can't see any solvers at all, please contact your program administrator.

Go to Solvers Vizualiser -> Submit Problems (top right)



Submit Problem Configuration

Configurations | Parameters

API Token: UoM-d1a835fc0f8004ca2d02290c8d7e0f1c48a10b4d

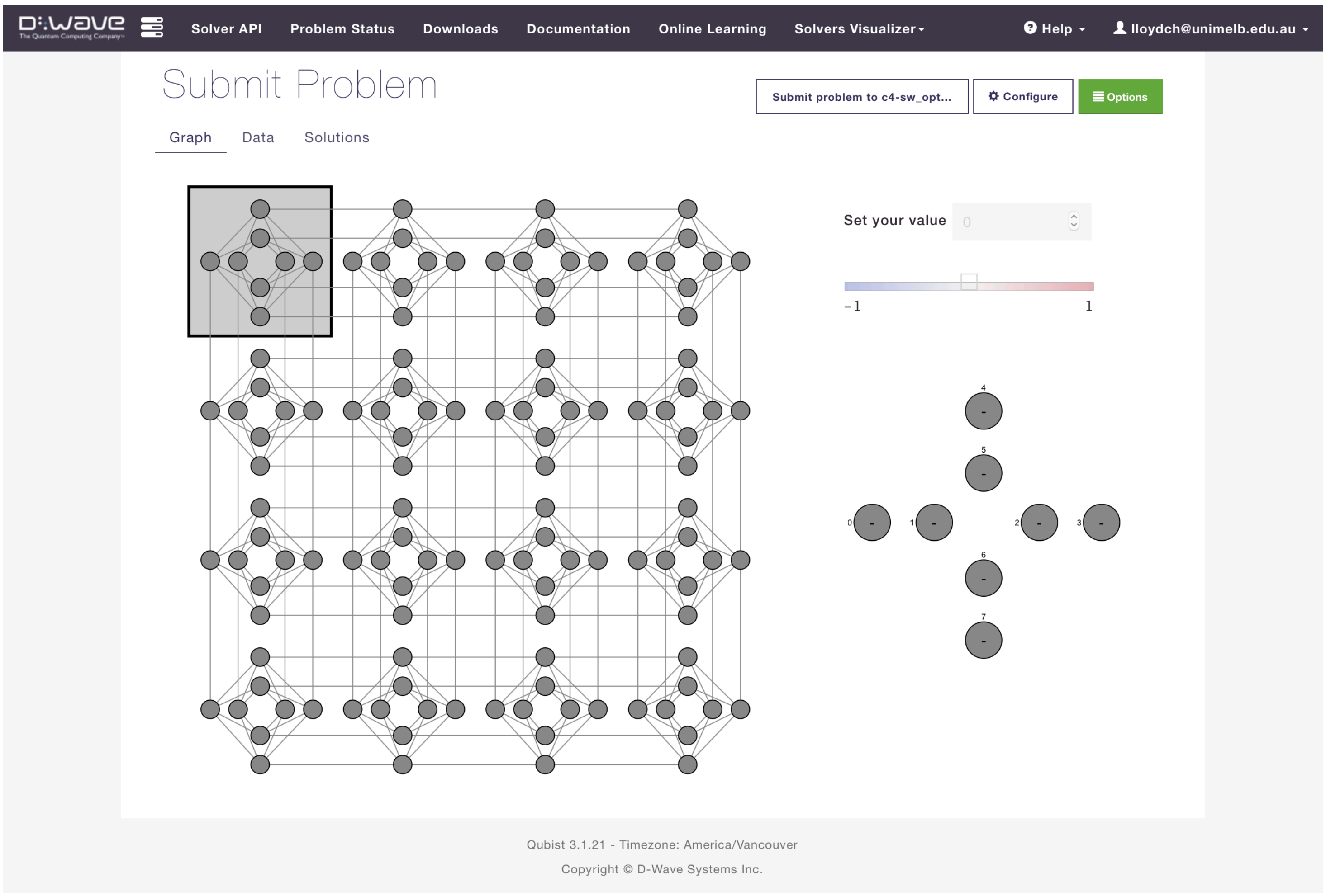
Solver: c4-sw_optimize

Optimized C4 software solver

Problem Type: Ising

Arrangement: Cross

Finally, you should see the "Submit Problem" page showing the Chimera architecture:



Qubist 3.1.21 - Timezone: America/Vancouver

Copyright © D-Wave Systems Inc.

You can program a problem in one of two ways:

1. In the **Graph** tab (shown above), click on a qubit and then in the zoomed-in view, click on either an coupling or a qubit and adjust the slider to change the coupling/local-field value respectively. Note that the slider only goes from -1 to 1, so you will have to rescale every coefficient in your Hamiltonian to within that range.

2. In the **Data** tab (next to the Graph tab), enter (or paste in) an edge-list where every row is of the form:

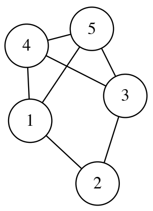
First row:	#qubits #terms
Subsequent rows:	FirstQubit SecondQubit Coupling

To get a local field on a particular qubit, set FirstQubit=SecondQubit (explained below).

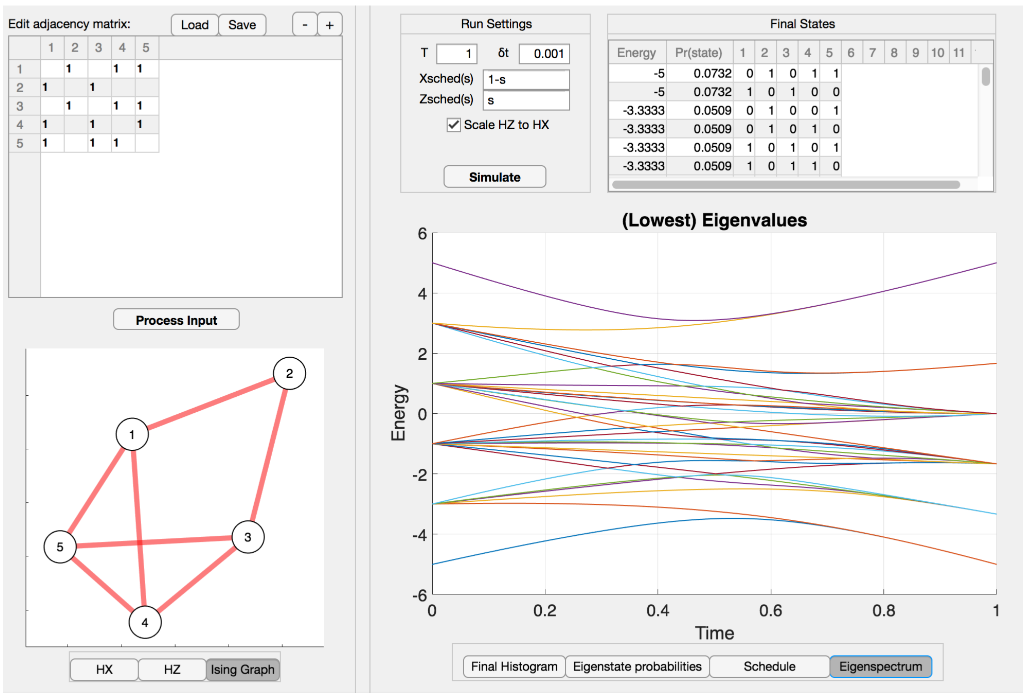
Exercise 9.7.1
We will first use STAQC to program a simple problem and then program it into D-Wave. Recall from lectures that the Max-Cut Hamiltonian is

$$H_P = \sum_{\langle ij \rangle} Z_i Z_j$$

Program into STAQC the max-cut problem Hamiltonian for the graph shown below.



Run for different values of total timescale, T, and inspect the solutions.



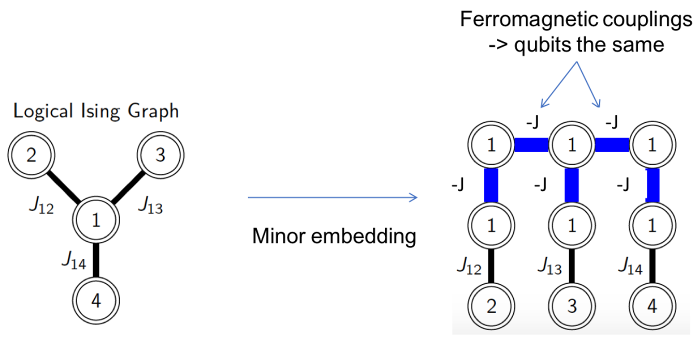
Edit adjacency matrix:

	1	2	3	4	5
1		1		1	1
2	1		1		
3		1		1	1
4	1		1		1
5	1		1	1	

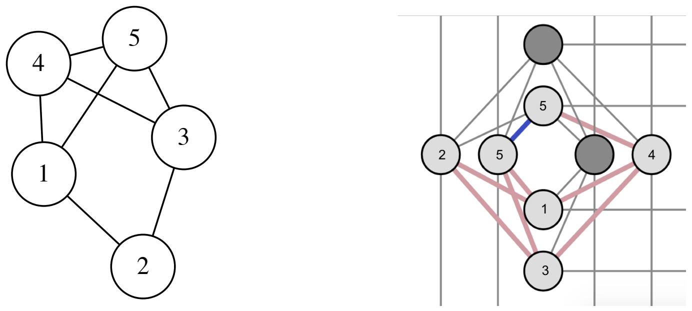
T 1 δt 0.001
Xsched(s) 1-s
Zsched(s) s
✓ Scale HZ to HX

Energy	Pr(state)	1	2	3	4	5	6	7	8	9	10	11
-5	0.0732	0	1	0	1	1						
-5	0.0732	1	0	1	0	0						
-3.3333	0.0509	0	1	0	0	1						
-3.3333	0.0509	0	1	0	1	0						
-3.3333	0.0509	1	0	1	0	1						
-3.3333	0.0509	1	0	1	1	0						

<u>**Exercise 9.7.2 Follow on from the STAQC example**</u>
Try programming in the Max-Cut example in Exercise 9.7.1. This will require a minor-embedding. Recall from lectures how to set couplings for minor embeddings:



Try the minor-embedding into one unit cell as shown below. Set the coupling strengths as necessary to ensure that the physical qubits representing Qubit 5 act as one logical qubit. As you are programming the problem into the Chimera Graph, make a note of which physical qubit indices are associated with your logical qubit indices, as we have done below (right).



For reference, here are the minor embedding details in text form:

Minor Embedding Qubit Map

1 -> 71
2 -> 64
3 -> 70
4 -> 67
5 -> 65 69

Minor embedding: Edge List (NB. header <-> 128 qubits, 8 terms for D-Wave simulator)

```
128 8
71 64 0.3
71 67 0.3
71 65 0.3
64 70 0.3
70 67 0.3
70 65 0.3
67 69 0.3
65 69 -1
```

(i) First run on the D-Wave simulator and understand the results. Do the following: Press **Configure**, and in the **Configurations** tab, set the **Solver** to be **c4-sw_sample**, and set **Number of Reads** to 100. Then **Submit Problem**.

To see the solutions go back to the **Graph** tab and press **Browse Solutions** you can inspect the outputs. You can also check the **Solutions** tab and select **Reads by Energy Chart** to see a histogram how many times a solution of a particular energy was sampled.

(ii) To run on the physical device, do the following: Press **Configure**, and in the **Configurations** tab, set the **Solver** to be either DW_2000Q_2_1 or DW_2000Q_5, and set **Number of Reads** to 100. Then **Submit Problem**.

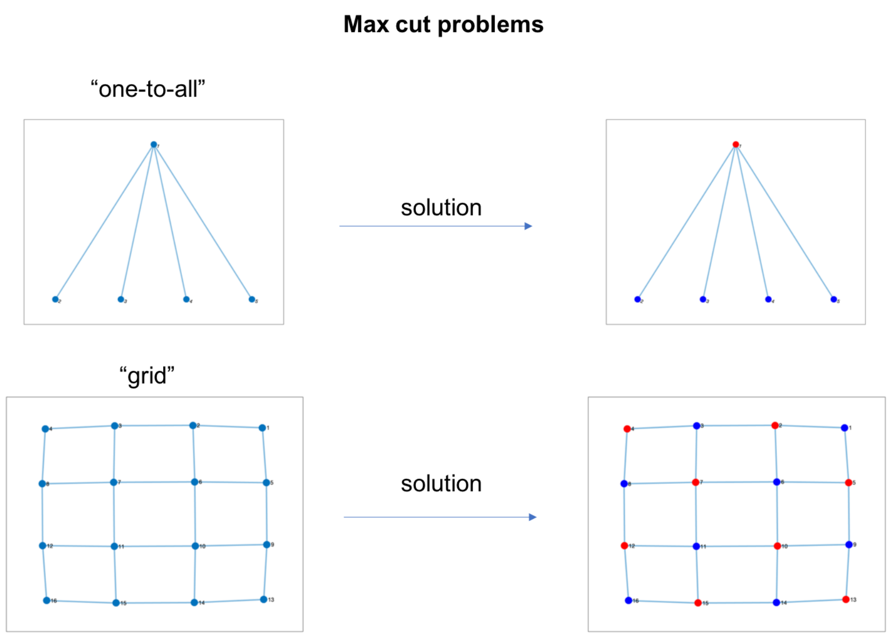
To see the solutions go back to the **Graph** tab and press **Browse Solutions** you can inspect the outputs. You can also check the **Solutions** tab and select **Reads by Energy Chart** to see a histogram how many times a solution of a particular energy was sampled.
- Note that the energies in the histogram will not be the same as the energies listed in the STAQC, since the minor-embedding has a different Hamiltonian. Instead, look at the spin configurations of the solution.
- Note also that in the D-Wave histogram, the spin eigenvalues are shown for spins participating in the problem, while for the STAQC, the qubit states are shown. Which qubit states correspond to particular spin eigenvalues?

Exercise 9.7.3 Big Random Problems

Set up a large random problem instance on D-Wave. In **Options**, under **Test Data**, click **Generate Random Problem**. This will initialize all Hamiltonian weights randomly for the entire Chimera Graph. Try running this for different annealing times and observe how the distribution changes.

Exercise 9.7.4 Big Max-Cut – demon on screen.

In this exercise we will define some larger Max Cut instances (see below) and solve them on a D-Wave machine.



Exercise 9.7.5 Exploring more examples on D-Wave

Go to https://cloud.dwavesys.com/leap/resources/demos/
Take a look at the demos, especially demo 2 on Network Analysis Using Optimization, which gives a good overview of kinds of network problems solvable by Quantum Annealing. The problem of Structural Balance is a slight generalization of Max Cut.

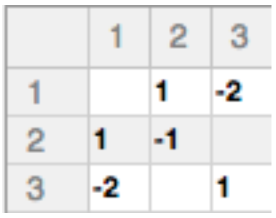
Next, from https://cloud.dwavesys.com/qubist/submit_problem/, click on "Online Learning". Here there are many different kinds of Jupyter Notebooks you can explore and run online. Pick any topics that interest you to see how they are solved on D-Wave machines. In particular "Structural Imbalance in Signed Social Networks" is a good follow-on from the demos.

Appendix 1: STAQC documentation

Problem Input Panel

- In this panel you can specify your problem Hamiltonian. You don't have to worry about creating H_X – this is done automatically.
- Problem Hamiltonians can be specified through an *Adjacency Matrix*.
  - The off-diagonal (i,j) elements of the matrix represent coefficients of Z_iZ_j
  - The on-diagonal elements represent coefficients of single-qubit Z terms.

  For example,



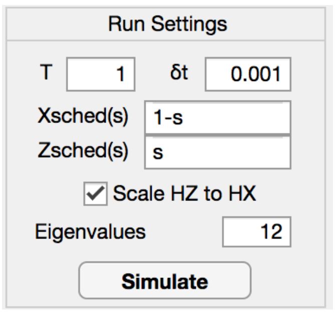
	1	2	3
1		1	-2
2	1	-1	
3	-2		1

  represents the Hamiltonian $H_Z = Z_1Z_2 - 2Z_1Z_3 - Z_2 + Z_3$

- You can also **load** or **save** adjacency matrices to text files. If you load your own adjacency matrix, keep in mind that it must be symmetric.
- After loading an adjacency matrix, click **Process Input** to construct the problem Hamiltonian. This also display outputs in the bottom-left:
  - **Ising Graph** shows the basic connectivity of the problem Hamiltonian specified. Anti-ferromagnetic (positive) couplings are shown in red, while Ferromagnetic (negative) couplings are shown in blue. You can use the scroll wheel to zoom if it is cut off.
  - **HX** shows a heat map of the initial transverse Hamiltonian, as a matrix.
  - **HZ** shows a heat map of the final problem Hamiltonian, as a matrix.

Simulation Panel

- Once a problem Hamiltonian has been loaded, simulation parameters can be set in the "Run Settings" panel.



Run Settings

T: 1　δt: 0.001

Xsched(s): 1-s

Zsched(s): s

[✓] Scale HZ to HX

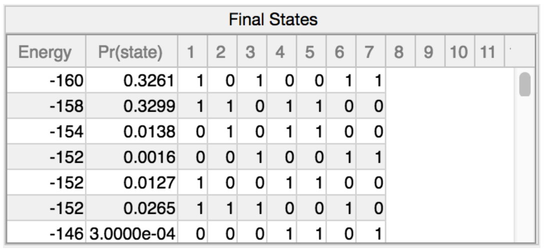
Eigenvalues: 12

  The settings are
  - **T** is the timescale of Adiabatic Evolution. This determines the length of time over which the Adiabatic Quantum Computation occurs.
  - **δt** is the timestep of the numerical simulation. By default, whenever T is changed, δt will be adjusted to be T/1000. For larger simulations, you may want to increase the timestep to reduce simulation runtime.
  - **Xsched(s)** and **Zsched(s)** are functions of the adiabatic parameter $s \in [0,1]$, which specify the factor multiplying the HX and HZ Hamiltonians for every value of s. Any valid matlab expression is allowable, so long as Xsched(1) = 0.

    - **Scale HZ to HX** re-scales the problem Hamiltonian HZ during simulation, so that the maximum and minimum eigenvalues of HZ match the maximum and minimum eigenvalues of HX.
    - For $n \geq 9$ qubits, it is too inefficient to get every eigenvalue of $H(s)$, so **Eigenvalues** specifies how many lowest-lying eigenvalues to compute.

- Press **Simulate** to run the simulation. You can see the progress of the simulation and any error messages in the Matlab console.

- Outputs:
    - In the "Final States" panel, basis states are listed alongside their energies and probabilities of being measured. The basis states are ordered in energy from lowest-energy to highest.



Final States

Energy	Pr(state)	1	2	3	4	5	6	7	8	9	10	11
-160	0.3261	1	0	1	0	0	1	1				
-158	0.3299	1	1	0	1	1	0	0				
-154	0.0138	0	1	0	1	1	0	0				
-152	0.0016	0	0	1	0	0	1	1				
-152	0.0127	1	0	0	1	1	0	0				
-152	0.0265	1	1	1	0	0	1	0				
-146	3.0000e-04	0	0	0	1	1	0	1				

    - In the bottom-right window, you can toggle between
        - **Final Histogram** – The probabilities of measuring states of a certain energy at the end of the evolution.
        - **Eigenstate Probabilities** – This plots the probability of being in the lowest-lying eigenstates of $H(s)$ during evolution.
        - **Schedule** – Plots the control schedules for HX and HZ.
        - **Eigenspectrum** – Plots the eigenvalues of $H(s)$ throughout the evolution. If the ground state of H_Z is unique, this will also display the minimum gap between the ground and first excited levels.

A note on units

The Matlab Simulator uses *atomic units*, which are suitable for describing the physics at very small scales. You don't need to worry too much about this, but if you're interested, and for a sense of perspective:

- If 1 unit of energy is 1.65×10^{-5} eV, then 1 unit of time will be 0.04 nanoseconds.
- Scaling the energy unit by a factor K will rescale the time unit by $\hbar$/K, where $\hbar = 6.58 \times 10^{-16}$eVs is Planck's reduced constant.

Appendix 2: D-Wave documentation

The two D-Wave devices are **DW_2000Q_2_1** and **DW_2000Q_5**. Both have 2048 qubits (although some qubits are disabled), arranged in a 16x16 grid of $K_{4,4}$ unit cells (i.e. [16, 16, 4] topology) and have couplings in the range

- **h_range:** [-2, 2]
- **j_range**: [-1, 1]

The total duration of a problem run is $(P1 + P2) \times P3 + P4$, where

- P1 = **Annealing time** (default = 20, range = [1, 2000])
- P2 = **Readout thermalization** (default = 0, range = [0, 10000])
    - Time to wait after annealing before reading out solution.
- P3 = **Number of Reads** (range = [1, 10000])
- P4 = **Programming thermalization** (default = 1000, range = [0, 10000])
    - Time to wait after programming before beginning annealing.