Intro to Leap and Ocean

Alex Koszegi, Technical Sales Engineer

Allison MacDonald, Lead Experimental Physicist

Session Outline

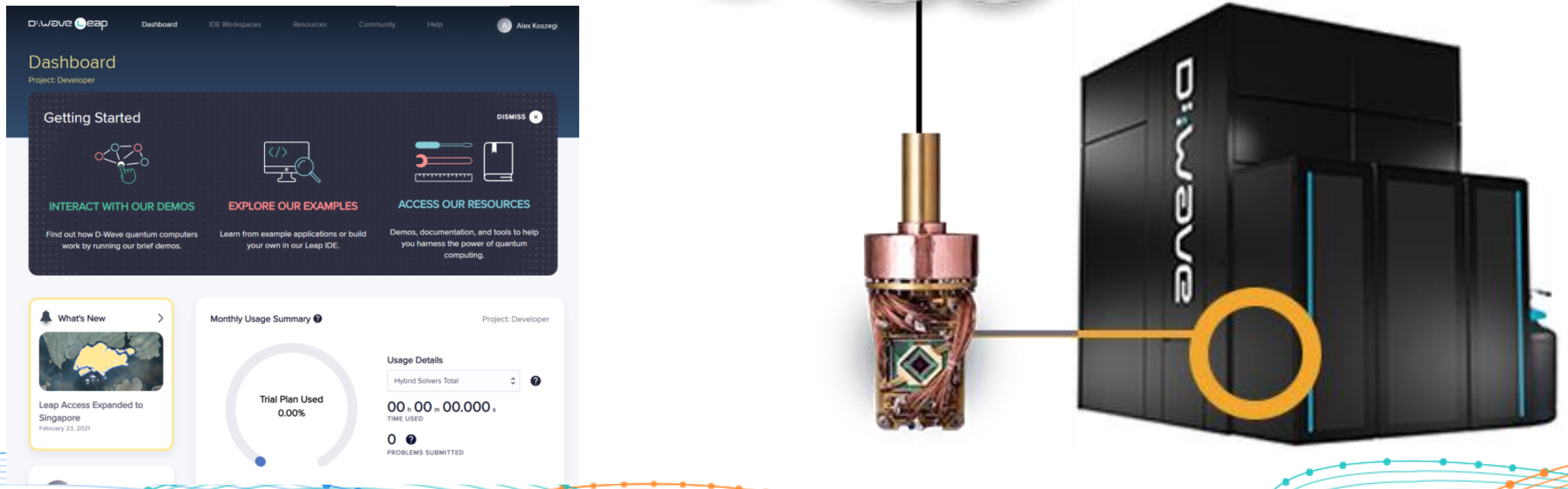
- **Leap tour**
- **Ocean SDK**
- **Review important concepts**
  - Quantum annealing
  - Problem Formulation
  - Chip topology
- **Example: 3 Qubit Problem**

Session Goals

1. Become familiar with Leap
2. Know what software tools are available
3. Understand the connection between quantum annealing, problem formulation and the chip topology

Leap Tour

Leap



Real-Time Cloud Access

Integrated Open Source ADE

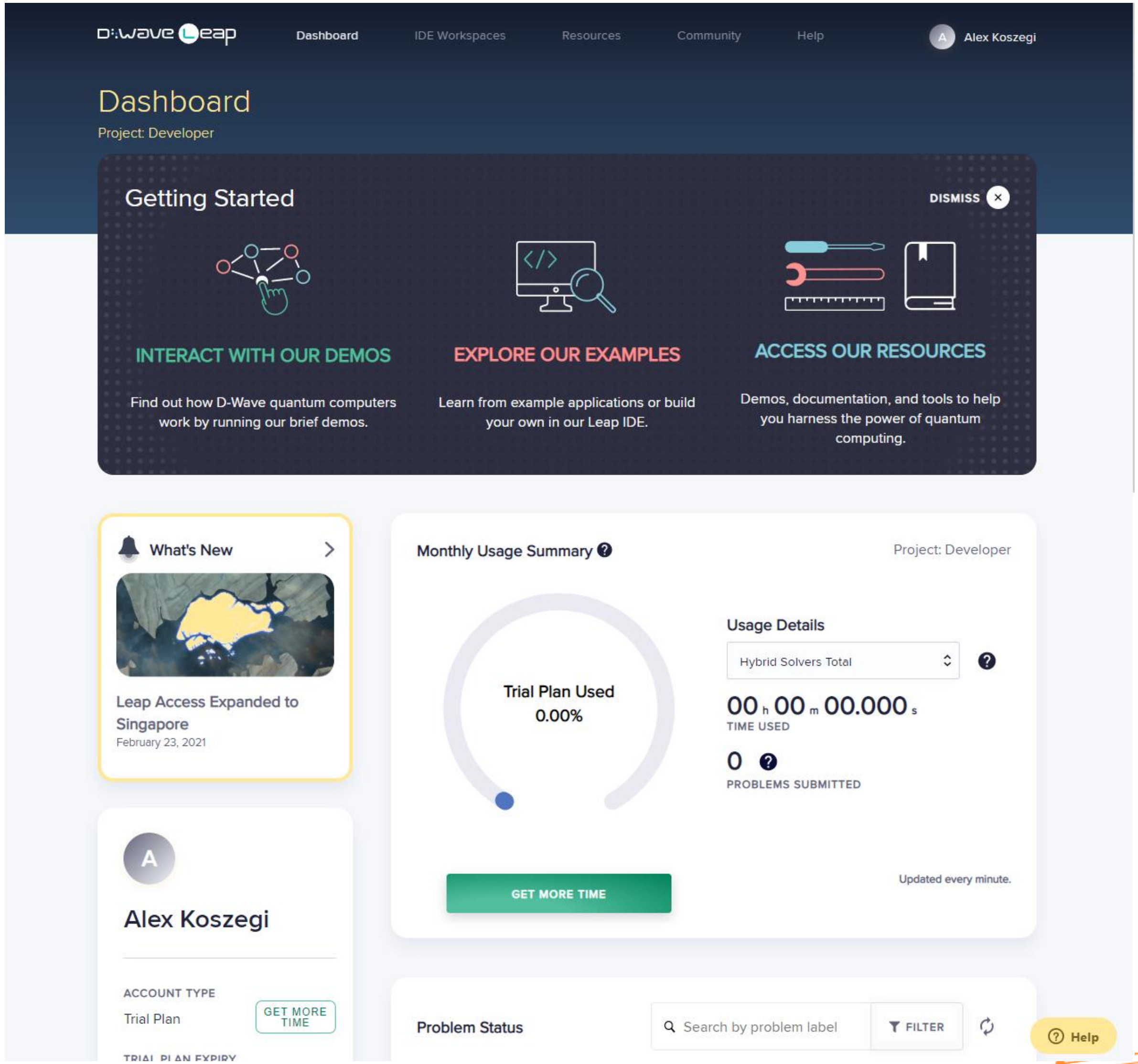
Demos and Reference Code

Community Support

Online Training

Dashboard

- Account details (name, project, API token)
- Quick links to demos, examples and resources
- Usage summary
- Problem status and details for your last 1000 problems
- Available solvers



Solvers

Hybrid

- Binary Quadratic Model (BQM) Solver
  - Up to 20 000 dense or 1 million sparse nodes
  - Binary variables
- Discrete Quadratic Model (DQM) Solver
  - Up to 3000 variables or 3 billion total biases
  - Discrete variables with up to 10 000 options each

QPU

- Advantage
  - Over 5000 qubits
  - Pegasus architecture (qubits are coupled to 15 other qubits)
- D-Wave 2000Q
  - Over 2000 qubits
  - Chimera architecture (qubits are coupled to 5 or 6 other qubits)

Available Solvers

Available solvers at Solver API endpoint:

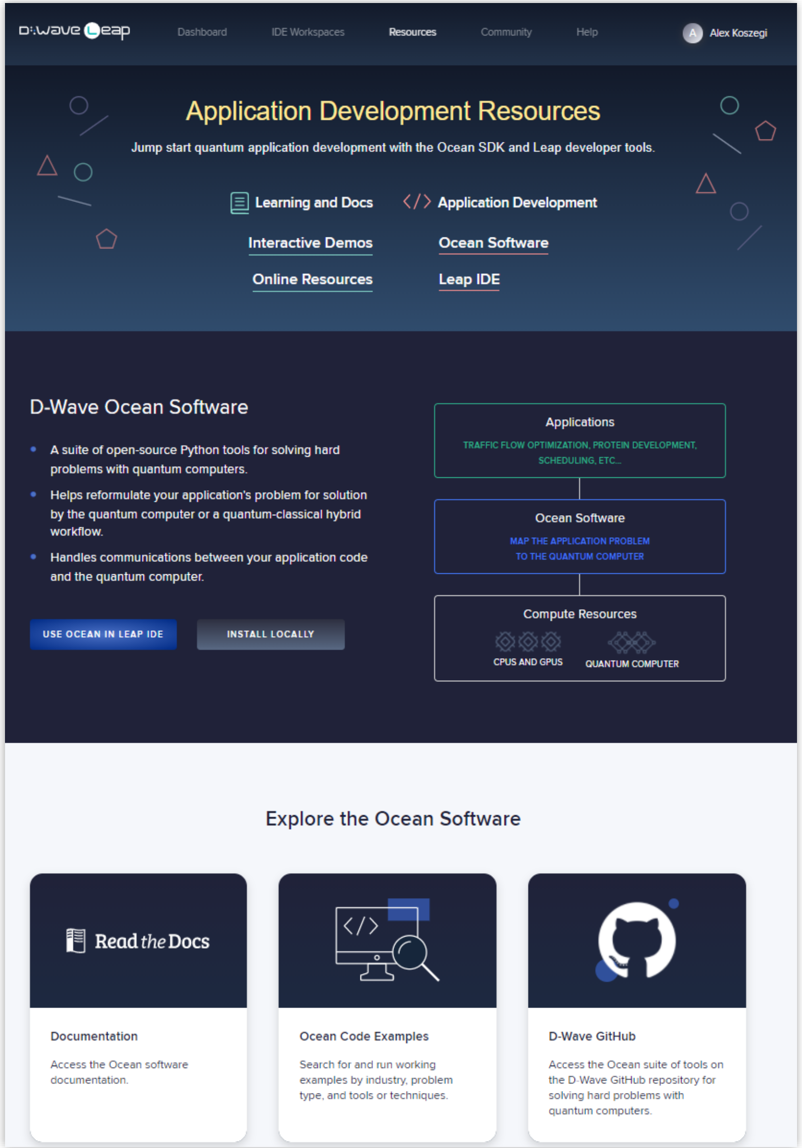
```
https://cloud.dwavesys.com/sapi/
```

Solver Name	Status	Description
Hybrid		
hybrid_binary_quadratic_model_version2	Online	Hybrid solver for general BQM problems, version 2.0
hybrid_discrete_quadratic_model_version1	Online	Hybrid solver for general DQM problems, version 1.0
QPU		
Advantage_system1.1	Online	Advantage system
DW_2000Q_6	Online	D-Wave 2000Q lower-noise system

Note: If you do not see the solver you are interested in, see the docs to find out how to query for available solvers.

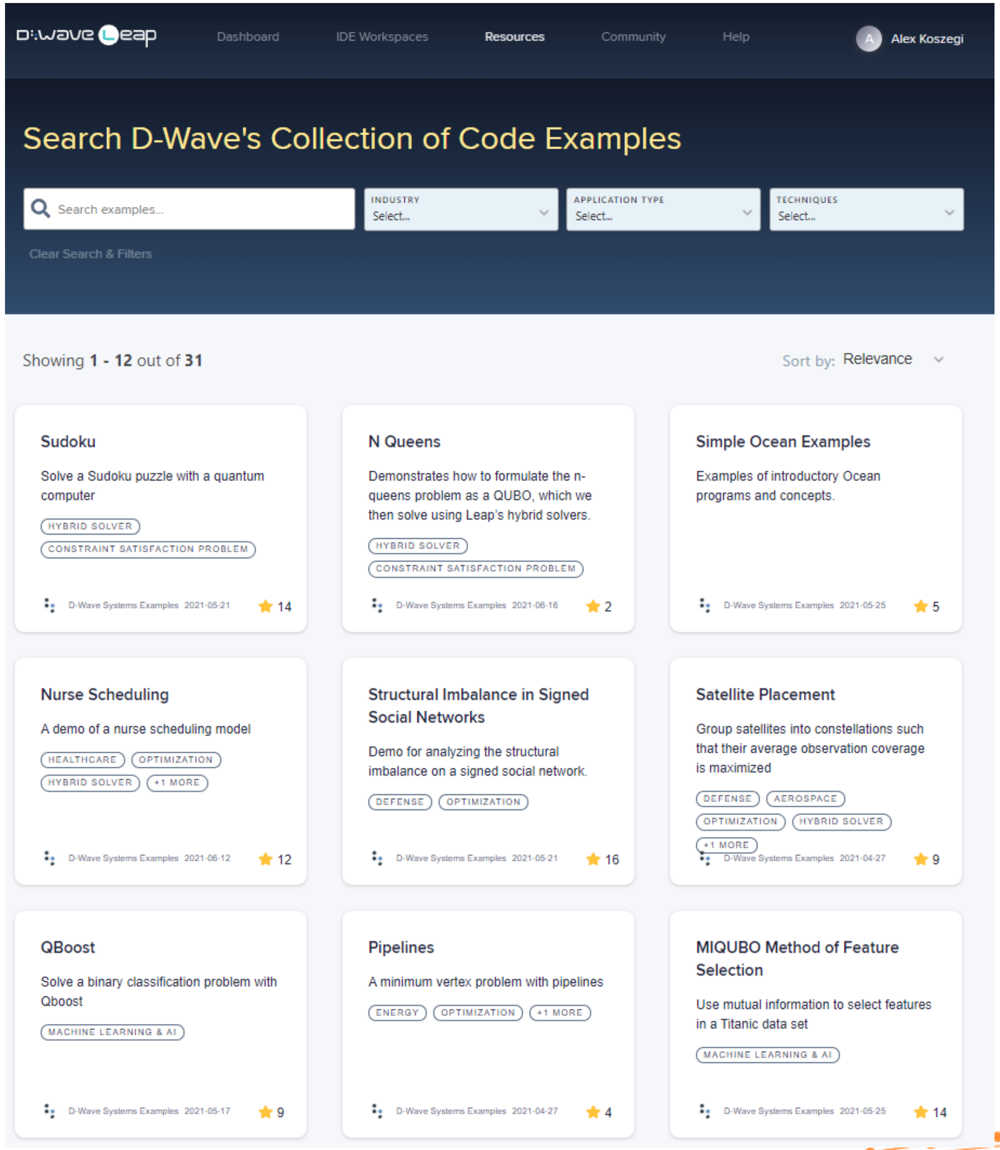
Resources

- Ocean installation instructions
- Ocean documentation
- Code examples
- Demos
- Problem Solving Handbook
- Jupyter Notebooks
- YouTube

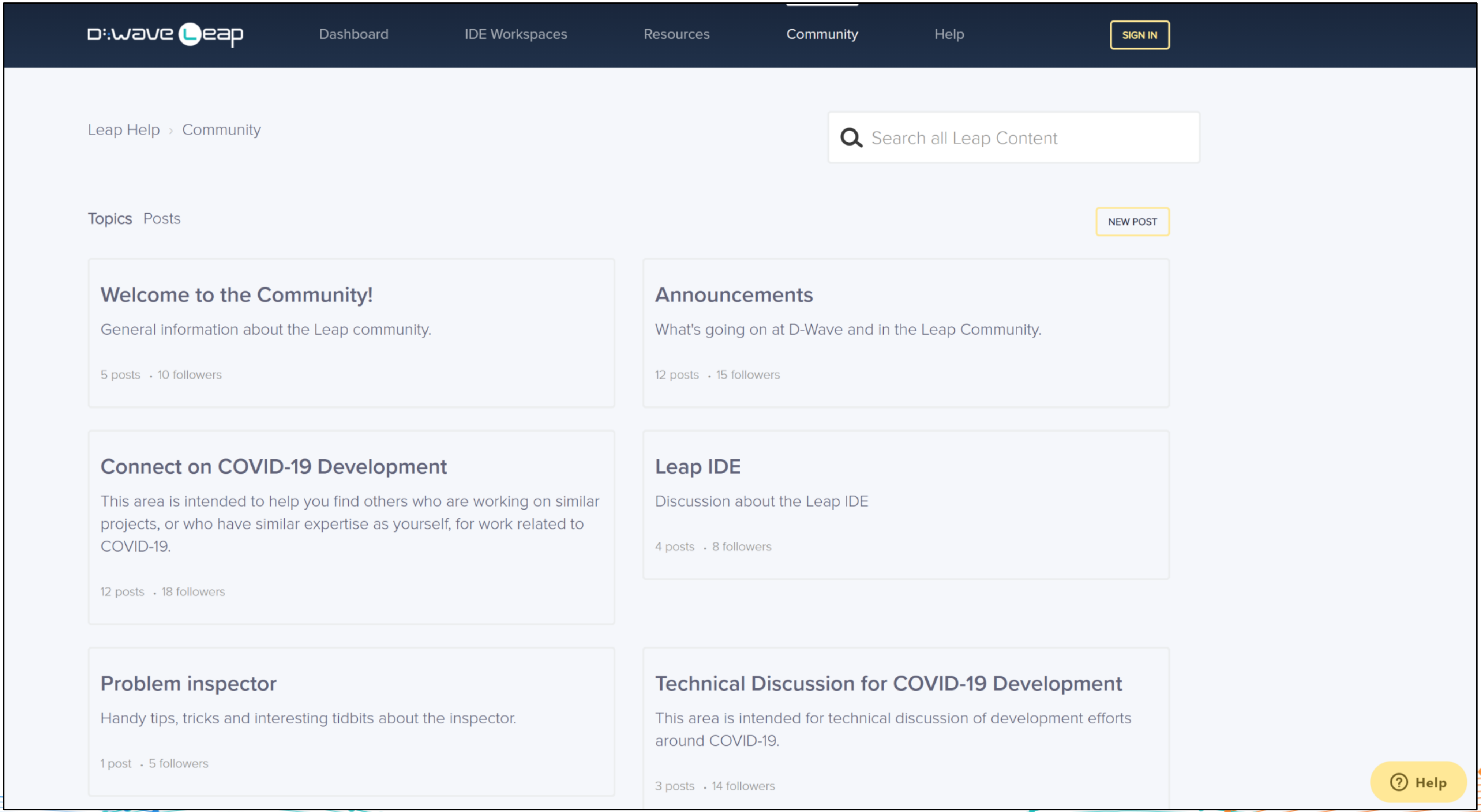


Resources – Collection of Examples

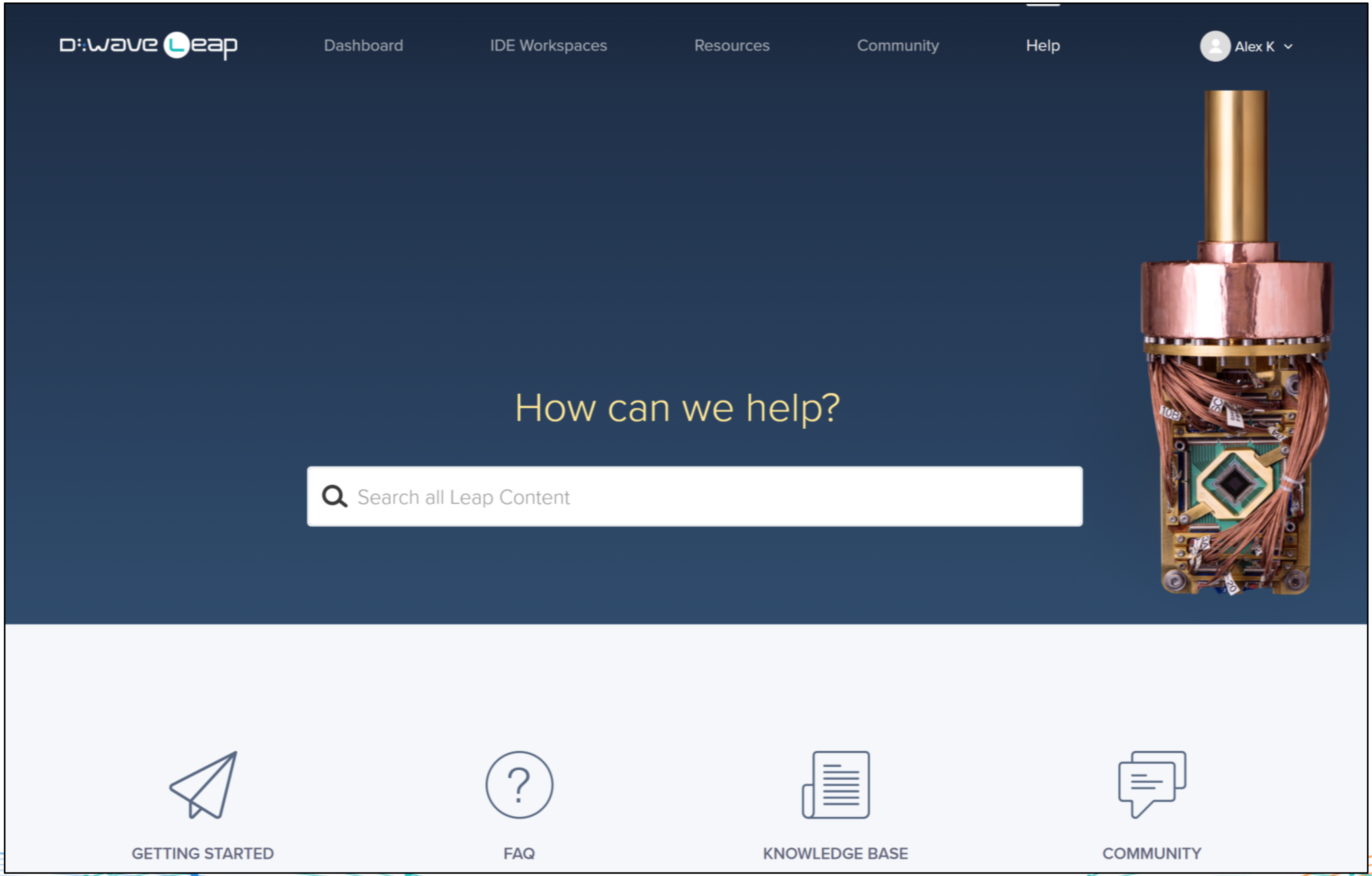
- Variety of code examples using the QPU, hybrid BQM, and hybrid DQM solvers
- Detailed explanations of the problem and the formulation
- Template included
- Access in the Leap IDE or GitHub



Community


Dashboard
IDE Workspaces
Resources
Community
Help
SIGN IN
Leap Help › Community
Search all Leap Content
Topics Posts
NEW POST
Welcome to the Community!
General information about the Leap community.
5 posts · 10 followers
Announcements
What's going on at D-Wave and in the Leap Community.
12 posts · 15 followers
Connect on COVID-19 Development
This area is intended to help you find others who are working on similar projects, or who have similar expertise as yourself, for work related to COVID-19.
12 posts · 18 followers
Leap IDE
Discussion about the Leap IDE
4 posts · 8 followers
Problem inspector
Handy tips, tricks and interesting tidbits about the inspector.
1 post · 5 followers
Technical Discussion for COVID-19 Development
This area is intended for technical discussion of development efforts around COVID-19.
3 posts · 14 followers
Help

Help and Search



Ocean SDK

Writing Quantum Programs

- Access the solvers through the Ocean SDK
- Ocean is a package of Python tools for accessing D-Wave's systems, preprocessing, postprocessing and building quadratic models
- Majority is open-source code available on GitHub
- Extensions and features from community welcome

D-Wave's Ocean Software

Ocean software is a suite of tools **D-Wave Systems** provides on the **D-Wave GitHub repository** for solving hard problems with quantum computers.

Ocean Software Stack

Use	Package	Description
Constructing quadratic models	dimod	Tools for working with BQMs, DQMs, and higher order models Also provides utilities for constructing new samplers and composed samplers
	dwavebinarycsp	Generates BQMs from constraint satisfaction problems
	penaltymodel	Maps constraints to BQMs
	Pyqubo	Creates quadratic models from mathematical expressions
Accessing quantum samplers	dwave-system	D-Wave samplers and composites
	dwave-cloud-client	API client to D-Wave solvers
Classical samplers	dwave-neal	Simulated annealing sampler
	dwave-tabu	Tabu sampler
Pre/post processing	dwave-preprocessing	Preprocessing tools for quadratic models
	dwave-greedy	Steepest descent solver
Hybrid tools/solvers	dwave-hybrid	Framework for building hybrid solvers
	qbsolv	Decomposing solver
Graph and visualization tools	minorminer	Tool for minor embedding graphs
	dwave-networkx	Networkx extension
	dwave-inspector	Visualizer for problems submitted to quantum computers

Program Structure

Ocean Program

- Establish sampler
- Define problem
- Send info to sampler
- Evaluate response

Program Example

```python
import dimod
from dwave.system import DWaveSampler, EmbeddingComposite

# 1. Define sampler
sampler = EmbeddingComposite(DWaveSampler())

# 2. Define problem
bqm = dimod.BQM({}, {'ab': -1, 'bc': -1, 'ca': -1}, 0, 'BINARY')

# 3. Submit problem and parameters to the solver
sampleset = sampler.sample(bqm, num_reads=10)

# 4. Evaluate the solution
print(sampleset)
```

```
   a  b  c energy num_oc. chain_.
0  1  1  1   -3.0      10     0.0
['BINARY', 1 rows, 10 samples, 3 variables]
```

Review – Bringing It All Together

Quantum Annealing

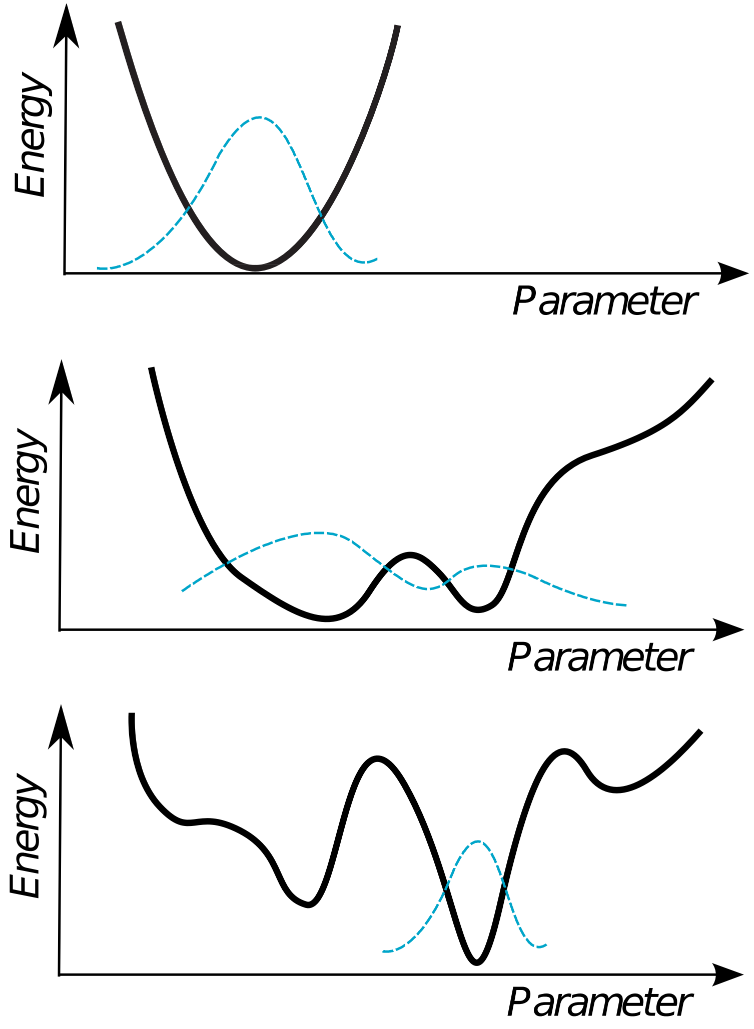
Start from H_i and anneal slowly to H_f:

$$H_i = -\sum_i \sigma_x^{(i)}$$

- The ground state of H_i will be superposition state of spin up and spin down
- This term drives quantum tunneling/spin flipping

$$H_f = -\sum_i h_i \, \sigma_z^{(i)} + \sum_{i,j>i} J_{ij} \, \sigma_z^{(i)} \sigma_z^{(j)}$$

Classical term representing the problem (ground state is the solution)



Problem Formulations

Binary Quadratic Model (BQM)

- General class of problems that can be mapped to the QPU

Ising Model

$$E_{ising} = \sum_{i} h_i\, s_i + \sum_{i>j} J_{i,j} s_i s_j$$

Binary variables:
$s_i \in \{-1,1\}$

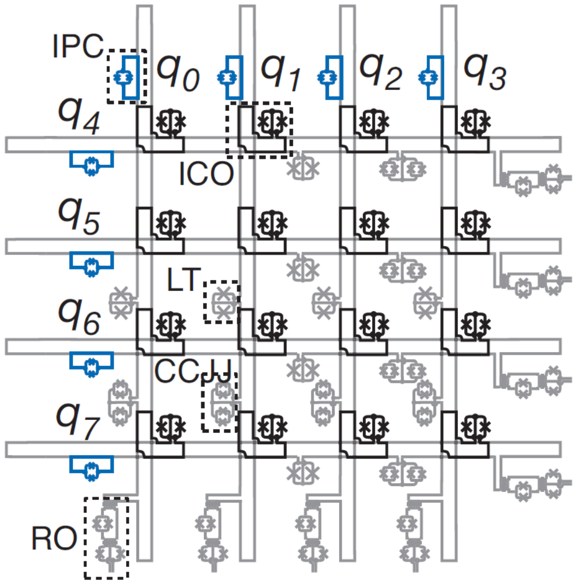
Quadratic Unconstrained Binary Optimization (QUBO)

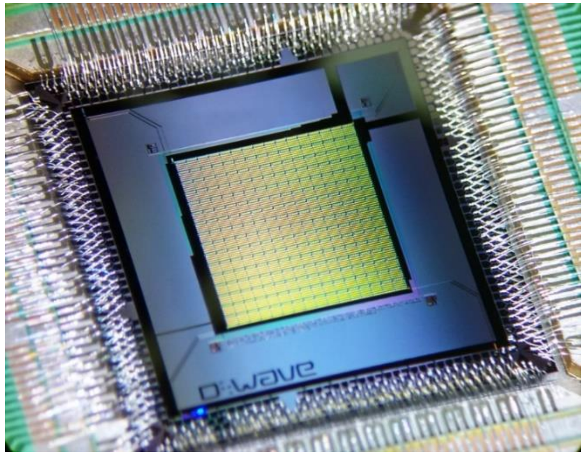
$$E_{qubo} = \sum_{i} a_i\, q_i + \sum_{i>j} b_{i,j} q_i q_j$$

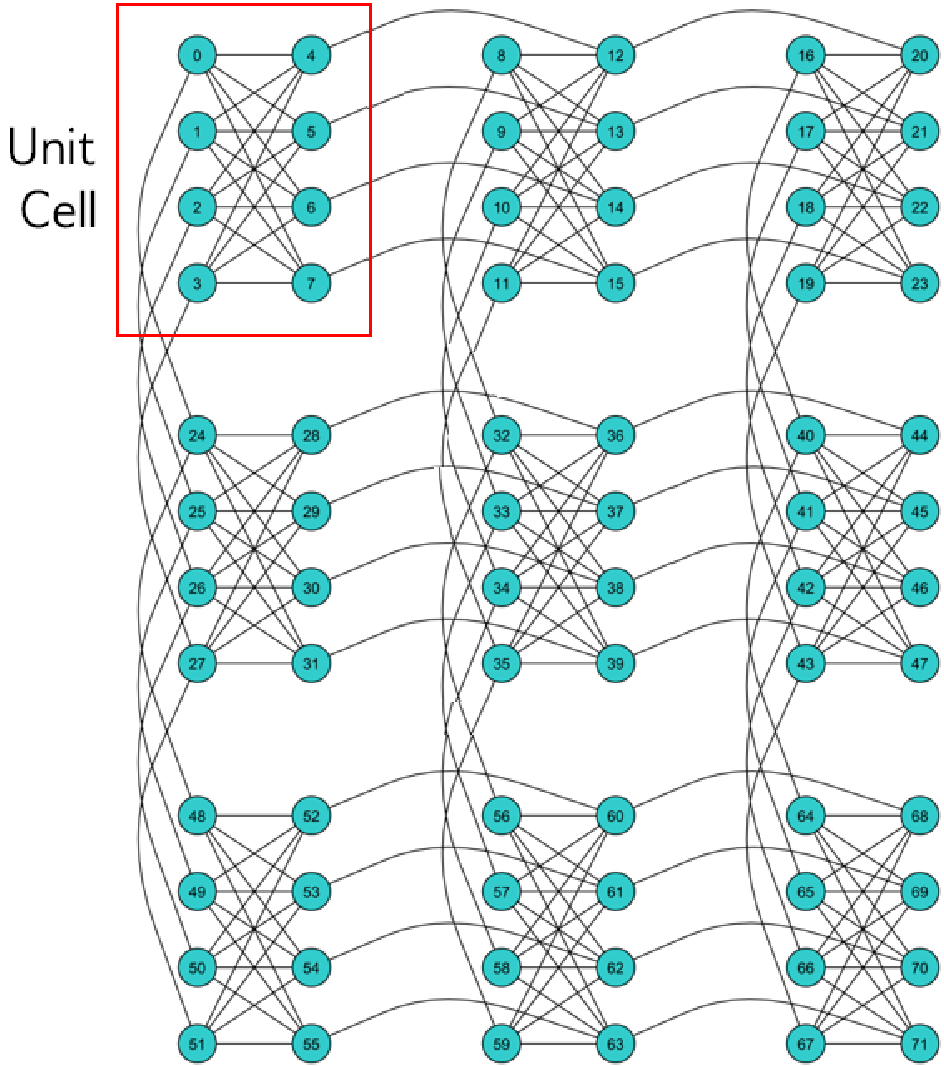
Binary variables:
$q_i \in \{0,1\}$

Converting between Ising and QUBO: $q_i = \dfrac{1 + s_i}{2}$

8-Qubit Unit Cell (D-Wave 2000Q)

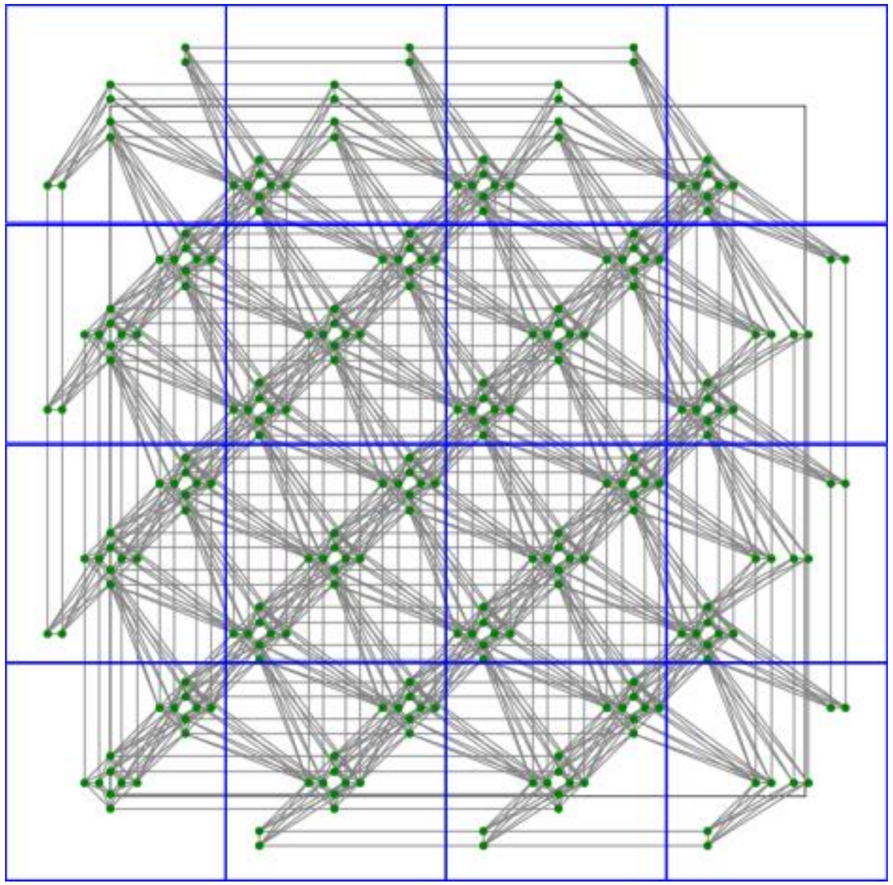




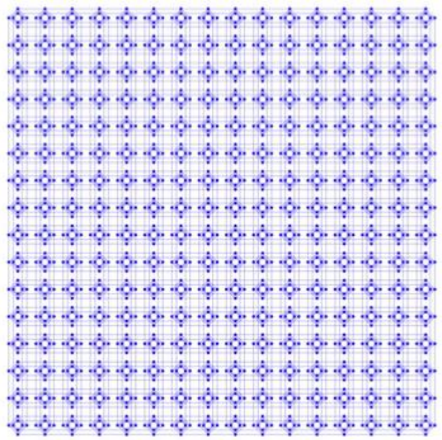
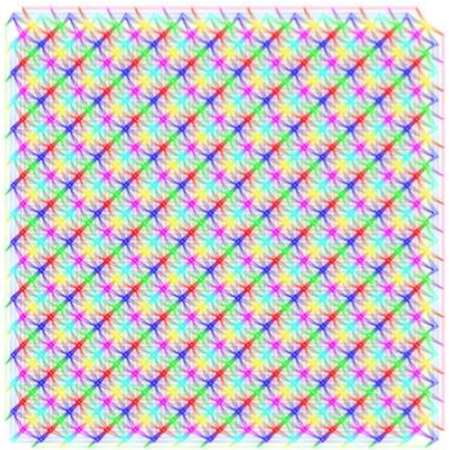


Advantage Architecture - Pegasus

- Unit cells of 24 qubits are tiled across the processor
- 16x16 grid of unit cells
- Each qubit couples to 15 other qubits



Embeddings

2000Q		Advantage
		
64 (17)	Complete Graph	180 (17)
64x64 (16)	Complete Bipartite	172x172 (15)
8x8x8 (4)	Cubic Lattice	15x15x12 (2)
16 bit (16)	Factoring Circuit	30 bit (15)

(chain length in parentheses)

3 Variable Programming Example

Example: 3 Variable Antiferromagnetic Chain

```python
import dimod
from dwave.system import DWaveSampler, EmbeddingComposite

# 1. Define sampler
sampler = EmbeddingComposite(DWaveSampler(solver={'topology__type': 'chimera'}))

# 2. Define problem: anti-ferromagnetic chain
#       E = a*b + b*c + c*a
bqm = dimod.BQM({}, {'ab': 1, 'bc': 1, 'ca': 1}, 0, 'SPIN')

# 3. Submit problem and parameters to the solver
sampleset = sampler.sample(bqm, num_reads=10)

# 4. Evaluate the solution
print(sampleset)
```

Note:

- We're using SPIN variables, so a, b and c are binary variables that can be +1 or -1
- We're running this problem on the D-Wave 2000Q (with the Chimera architecture)

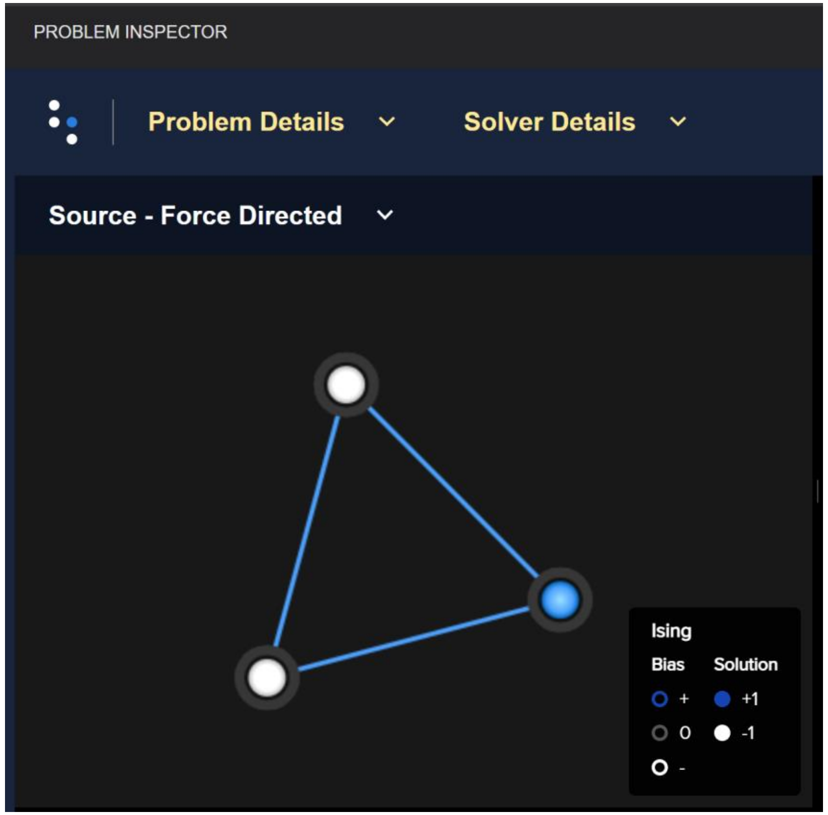
```
   a  b  c energy num_oc. chain_.
0 +1 -1 -1   -1.0       2     0.0
1 -1 -1 +1   -1.0       1     0.0
2 -1 +1 -1   -1.0       5     0.0
3 +1 -1 +1   -1.0       2     0.0
['SPIN', 4 rows, 10 samples, 3 variables]
```

Example: Three Variable Antiferromagnetic Chain

We can represent mathematical problems as graphs where

- Variables = nodes
- Quadratic interactions = edges
- Both nodes and edges can be weighted (weights represent linear and quadratic biases in the equation)

```
# 2. Define problem: anti-ferromagnetic chain
#        E = a*b + b*c + c*a
bqm = dimod.BQM({}, {'ab': 1, 'bc': 1, 'ca': 1}, 0, 'SPIN')
```

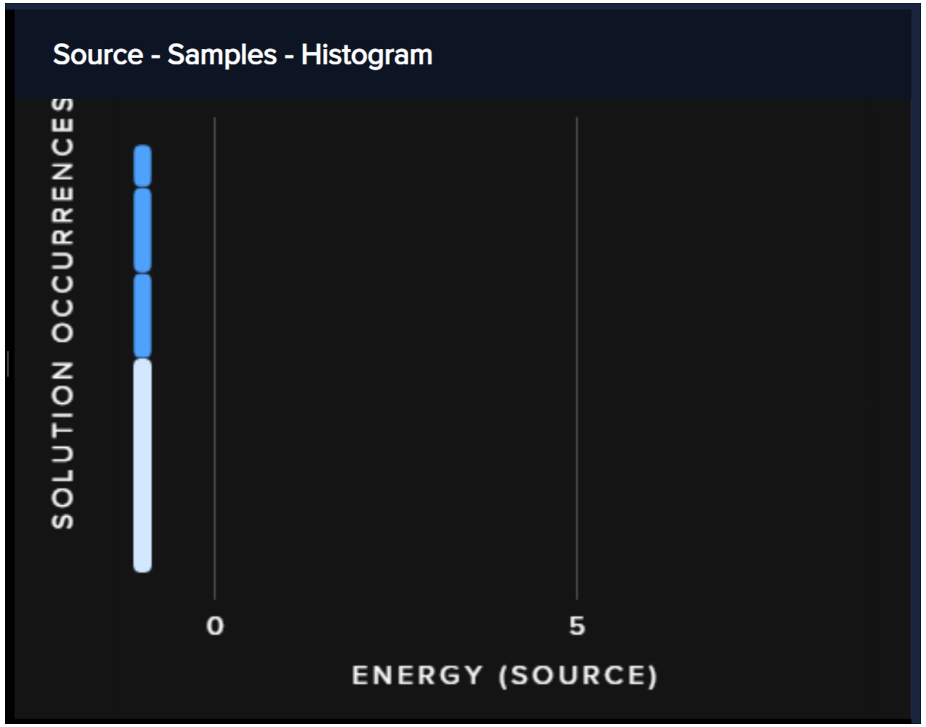


Example: Three Variable Antiferromagnetic Chain

The QPU is probabilistic so we need to sample many times from the problem's energy landscape.

These samples can be represented in a histogram of energies.

```
   a  b  c energy num_oc. chain_.
0 +1 -1 -1   -1.0       2     0.0
1 -1 -1 +1   -1.0       1     0.0
2 -1 +1 -1   -1.0       5     0.0
3 +1 -1 +1   -1.0       2     0.0
['SPIN', 4 rows, 10 samples, 3 variables]
```



Embedding

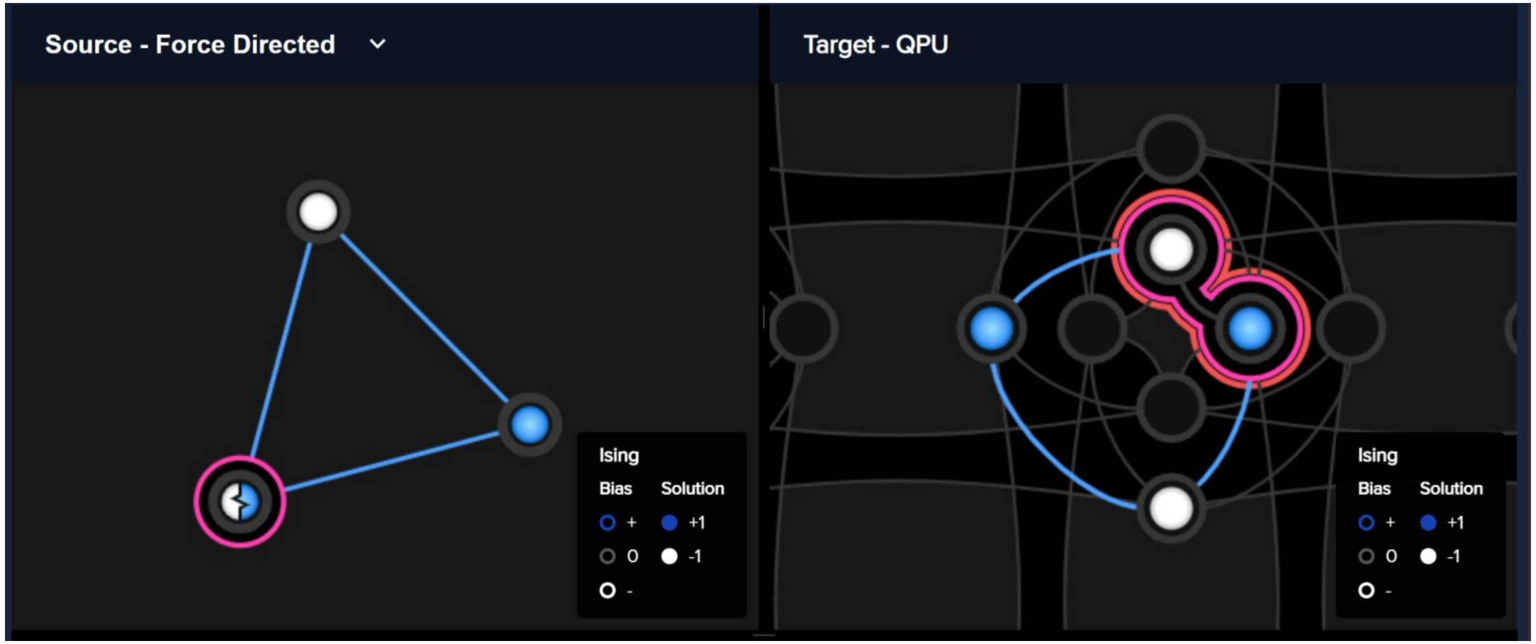
Example: Three Variable Antiferromagnetic Chain

To run a problem on the QPU we need to embed, or map the variables to qubits and interactions to couplers.

Example: Three Variable Antiferromagnetic Chain

When variables need to be embedded as chains of qubits, the chain strength parameter has to be tuned. It tells the QPU to treat qubits in a chain as a single variable.

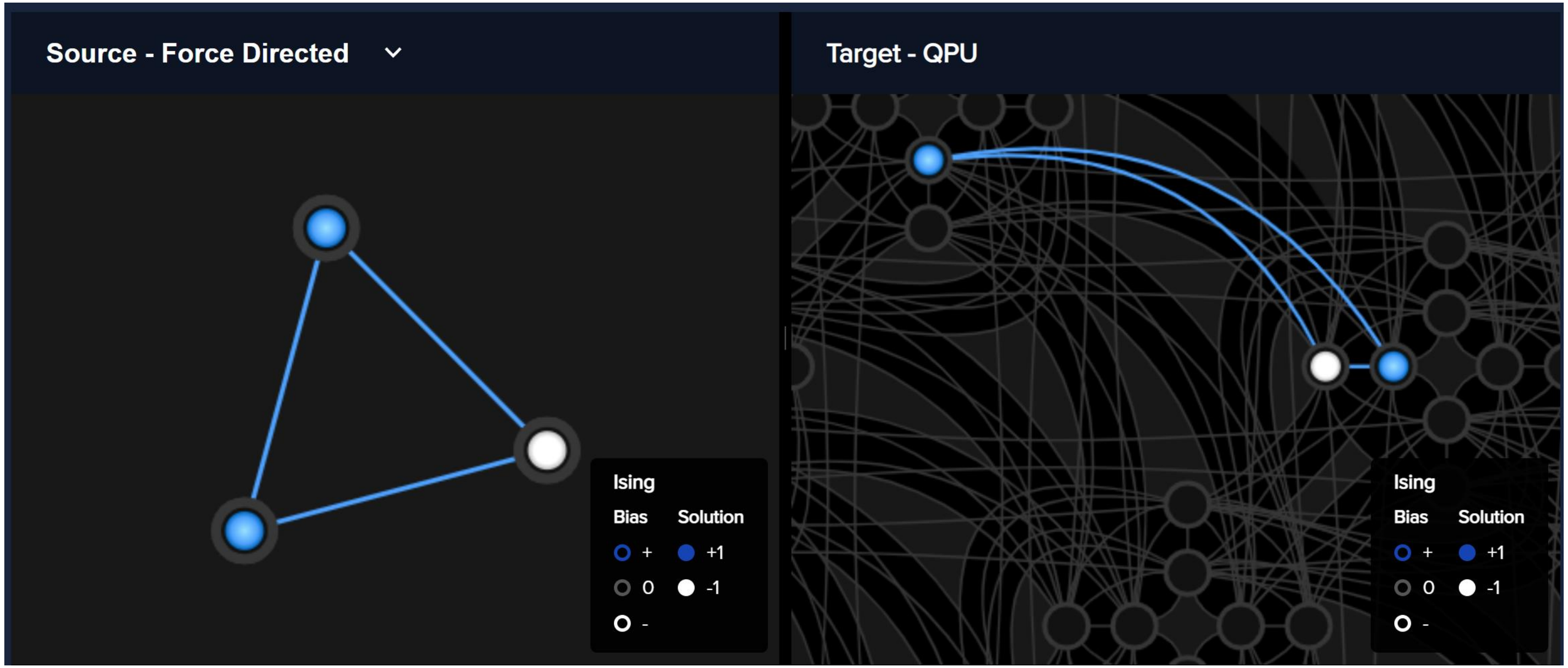
If it isn't strong enough the chain will 'break'



Example: Three Variable Antiferromagnetic Chain

Benefits of Advantage:

- Greater connectivity between qubits
- Greater number of qubits



Recap

Session Outline

- Leap is D-Wave's cloud platform
- The Ocean SDK contains open-source tools to submit problems to the QPU and hybrid solvers
- The QPU is probabilistic
- Problems need to be embedded onto qubits and couplers

Session Goals

1. Become familiar with Leap
2. Know what software tools are available
3. Understand the connection between quantum annealing, problem formulation and the chip topology