D:wave
The Quantum Computing Company™

Practical Quantum Computing

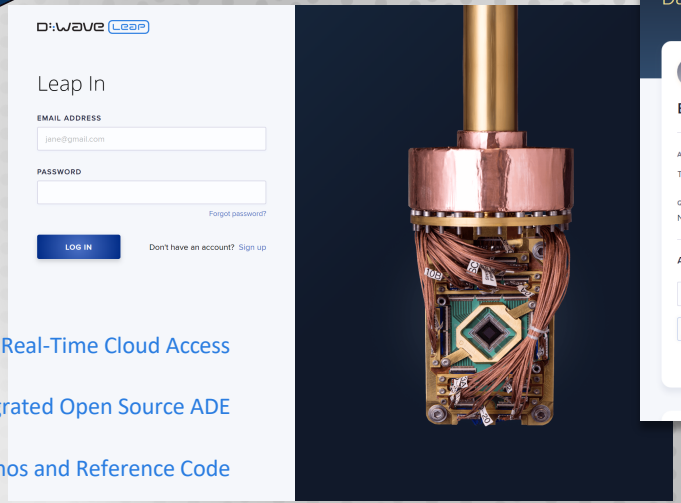
An Introduction

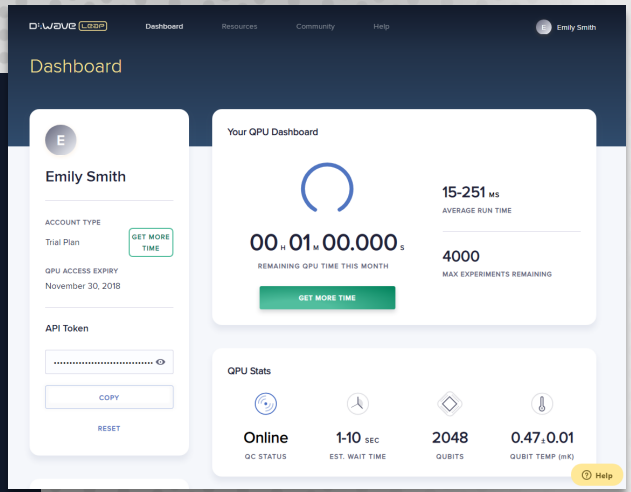
Alexander Condello
D-Wave

D-Wave Leap™

The *Only* Real-Time
Quantum Application Environment

Enabling a New Developer Community





Free Real-Time Cloud Access

Integrated Open Source ADE

Demos and Reference Code

Community Support

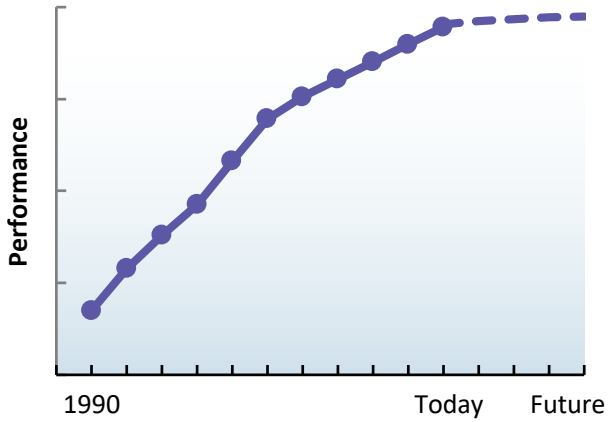
Online Training

D-Wave
The Quantum Computing Company™

Why Quantum Computing Gets So Much Attention

- End of Moore's Law
- Classical systems' power consumption is reaching its limits
- Potential to address NP-hard problems
- Huge potential speedup over classical approaches on some problems

Classical computers are reaching limits…



1 E.g., Speed, compactness, power consumption

SOURCE: Intel presentation; IEEE Transactions on Semiconductor Manufacturing May 2011, McKinsey

Practical

Adiabatic Quantum Computing (AQC) [Farhi et al. `01]

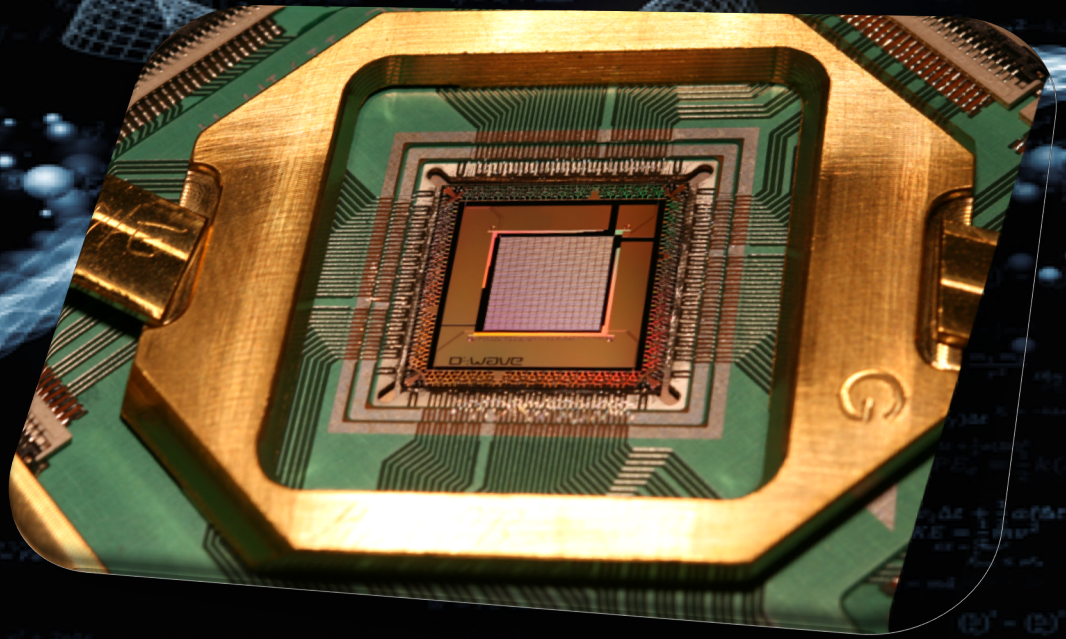
- System is evolved from the lowest-energy state of an easy problem to the lowest-energy state of a target problem
- If evolution is gradual enough, system remains in the lowest energy state
- Initially, the state is a superposition of all classical states

Why AQC?

- Computationally equivalent to gate-model [Aharonov et al. `08]

How To Build a Quantum Computer

Quantum Processor Unit (QPU)



Quantum Machine Language Programming

QUBIT	q_i	**Quantum bit** which participates in annealing cycle and settles into one of two possible final states: $\{0,1\}$
COUPLER	$q_i q_j$	Physical device that allows one **qubit to influence another qubit**
WEIGHT	a_i	Real-valued **constant associated with each qubit**, which influences the qubit's tendency to collapse into its two possible final states; controlled by the programmer
STRENGTH	b_{ij}	Real-valued **constant associated with each coupler**, which controls the influence exerted by one qubit on another; controlled by the programmer
OBJECTIVE	Obj	Real-valued **function which is minimized** during the annealing cycle

$$Obj(a_i, b_{ij}; q_i) = \sum_i a_i q_i + \sum_{ij} b_{ij} q_i q_j$$

Binary Quadratic Model

$$E(v) = \sum_{i,j} b_{i,j} v_i v_j + \sum_i a_i v_i + c$$

$$a_i, b_{i,j}, c \in \mathbb{R}$$

$$v_i \in \{-1, +1\} \text{ or } v_i \in \{0, 1\}$$

Synonyms

- Binary Quadratic Model
- Ising Model
- Quadratic Unconstrainted Binary Optimization Problem
- Probabilistic Graphical Model
- Restricted Boltzman Machine

Binary Quadratic Model

$$E(v) = \sum_{i,j} b_{i,j} v_i v_j + \sum_i a_i v_i + c$$

$$a_i, b_{i,j}, c \in \mathbb{R}$$

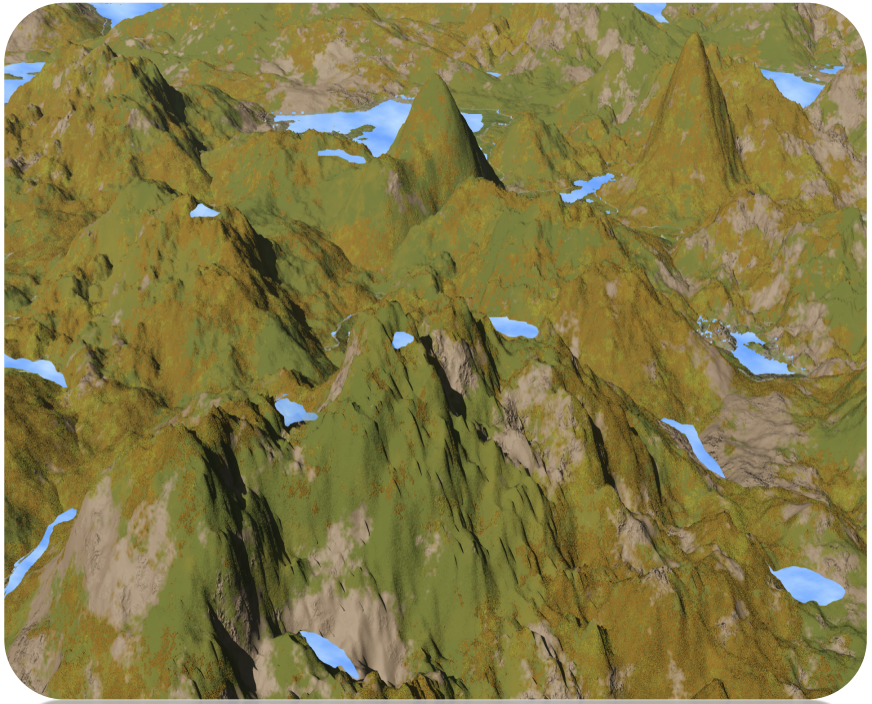
$$v_i \in \{-1, +1\} \text{ or } v_i \in \{0, 1\}$$

Landscape metaphor

Space of solutions defines an energy landscape and the best solution is the lowest valley

Classical algorithms can only walk over this landscape

Quantum annealing uses quantum effects

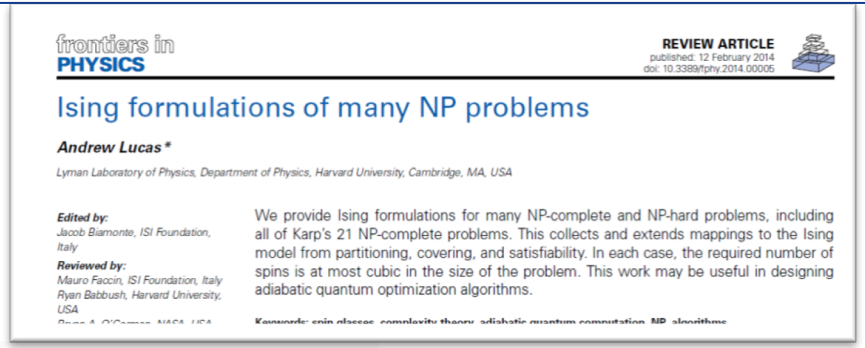


Sampling

QPU is probabilistic

- Finding a ground state
- Sampling ground states
- Characterizing the landscape
  - Machine learning
  - Material simulation

Is it possible to express problems as BQMs?



frontiers in PHYSICS

REVIEW ARTICLE
published: 12 February 2014
doi: 10.3389/fphy.2014.00005

Ising formulations of many NP problems

Andrew Lucas*

Lyman Laboratory of Physics, Department of Physics, Harvard University, Cambridge, MA, USA

Edited by:
Jacob Biamonte, ISI Foundation, Italy

Reviewed by:
Mauro Faccin, ISI Foundation, Italy
Ryan Babbush, Harvard University, USA

We provide Ising formulations for many NP-complete and NP-hard problems, including all of Karp's 21 NP-complete problems. This collects and extends mappings to the Ising model from partitioning, covering, and satisfiability. In each case, the required number of spins is at most cubic in the size of the problem. This work may be useful in designing adiabatic quantum optimization algorithms.

Keywords: spin glasses, complexity theory, adiabatic quantum computation, NP, algorithms

- Partitioning Problems
  - Number Partitioning
  - Graph Partitioning
  - Cliques
  - Reducing N TO logN Spins in Some Constraints
- Binary Integer Linear Programming
- Covering and Packing Problems
  - Exact Cover
  - Set Packing
  - Vertex Cover
  - Satisfiability
  - Minimal Maximal Matching
- Problems with Inequalities
  - Set Cover
  - Knapsack with Integer Weights

- Coloring Problems
  - Graph Coloring
  - Clique Cover
  - Job Sequencing with Integer Lengths
- Hamiltonian Cycles
  - Hamiltonian Cycles and Paths
  - Traveling Salesman
- Tree Problems
  - Minimal Spanning Tree with a Maximal Degree Constraint
  - Steiner Trees
  - Directed Feedback Vertex Set
  - Undirected Feedback Vertex Set
  - Feedback Edge Set
- Graph Isomorphisms

Customer's Early Applications

Optimization

Multi-period portfolios

Internet ad placement

Satellite Placement

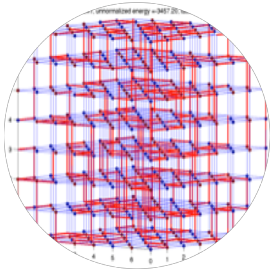
Machine Learning

Image recognition

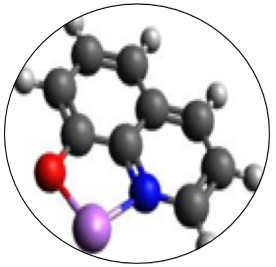
Tree cover classifier

Higgs Boson Detection

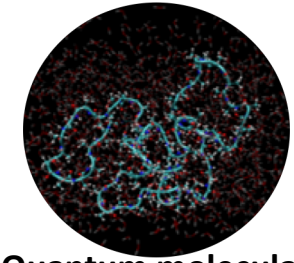
Quantum Materials



Quantum solid state

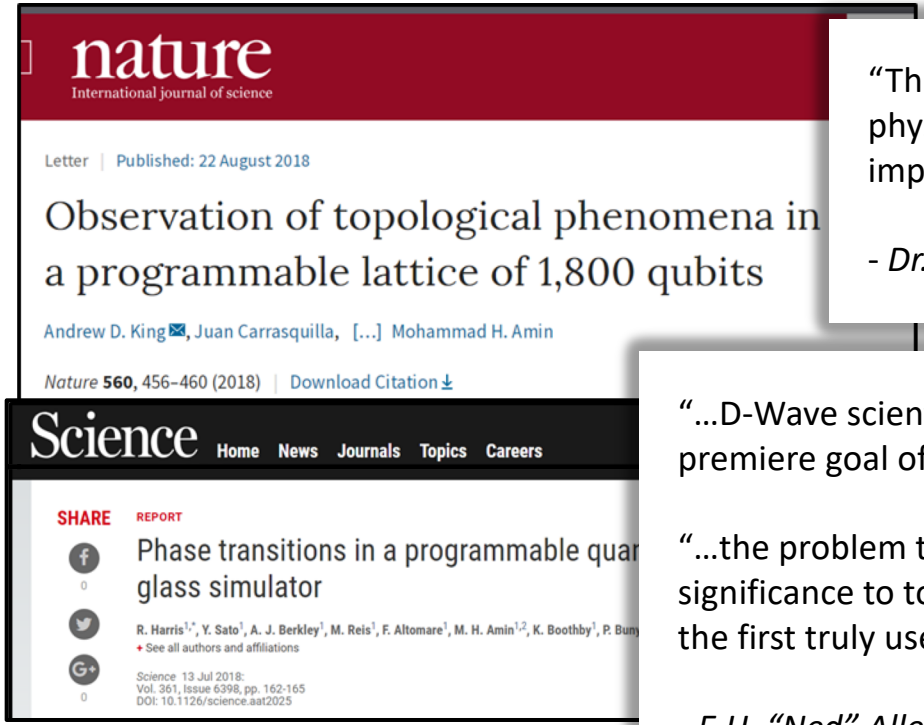


Quantum Molecules



Quantum molecular dynamics

Results At Scale



"This paper represents a breakthrough in the simulation of physical systems which are otherwise essentially impossible,"

- Dr. J. Michael Kosterlitz, ***2016 Nobel laureate in Physics***

"…D-Wave scientists and engineers have accomplished a premiere goal of scientific computing…"

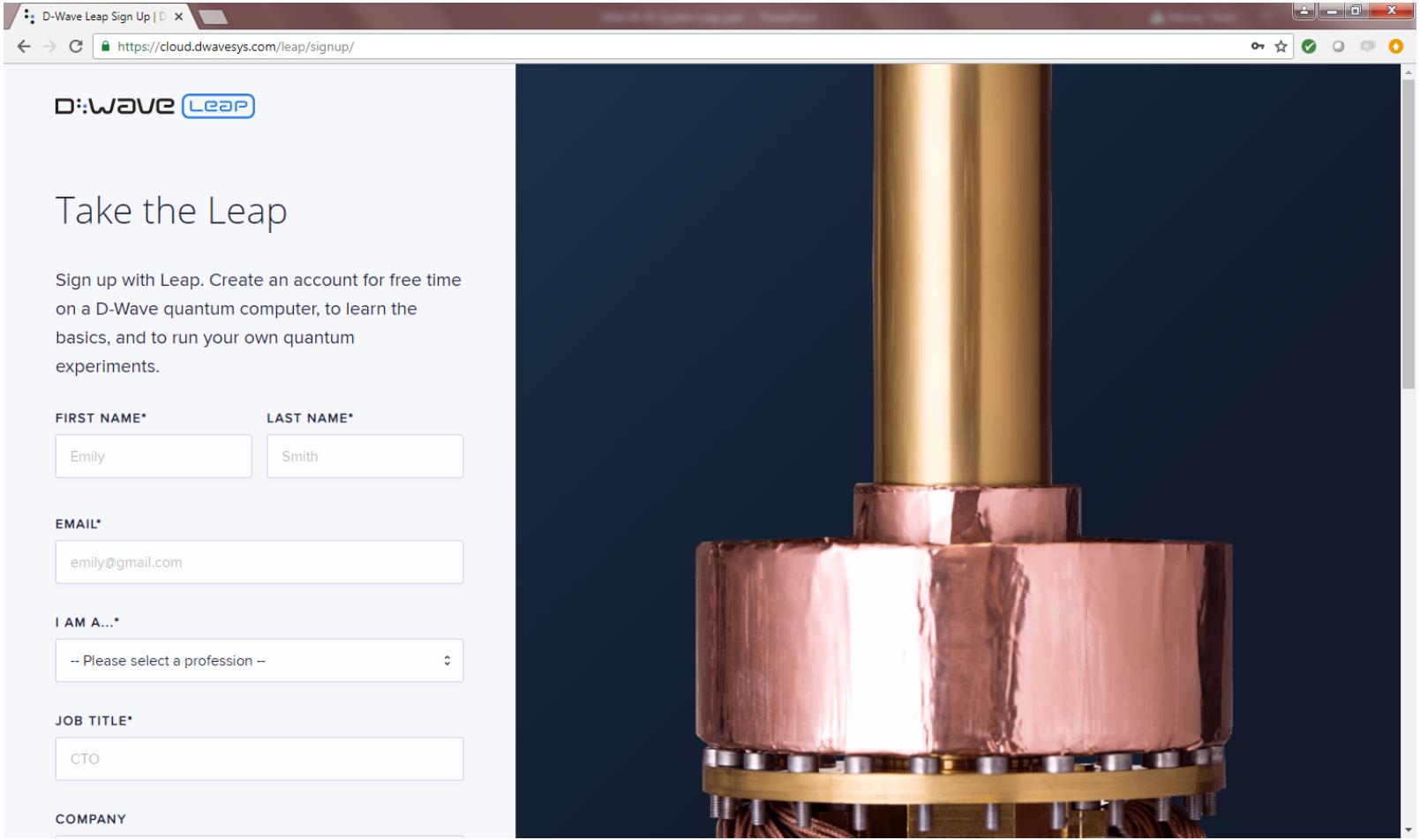
"…the problem they've attacked is one of immediate significance to today's advanced technology sectors and it is the first truly useful application of a quantum computer."

- E.H. "Ned" Allen Ph.D., ***Chief Scientist and Corporate Senior Fellow at Lockheed Martin***

D-Wave
The Quantum Computing Company™

Why is this interesting to the open source community

- New and exciting field for computing
- There are problems best solved on classical resources
  - Need developers on the classical side
- You can try these things for yourself
  - We in the open source space thrive on growth and debate


D-Wave Leap Sign Up |
https://cloud.dwavesys.com/leap/signup/
D:Wave Leap
Take the Leap
Sign up with Leap. Create an account for free time on a D-Wave quantum computer, to learn the basics, and to run your own quantum experiments.
FIRST NAME*
LAST NAME*
Emily
Smith
EMAIL*
emily@gmail.com
I AM A...*
-- Please select a profession --
JOB TITLE*
CTO
COMPANY

Demo 2 > Try it on the D

https://cloud.dwavesys.com/leap/demos/socialnetwork/try-it

D:Wave Leap

LEAVE THE DEMO

SOCIAL NETWORK ANALYSIS

Try it on the D-Wave quantum computer

SYRIAN DATASET

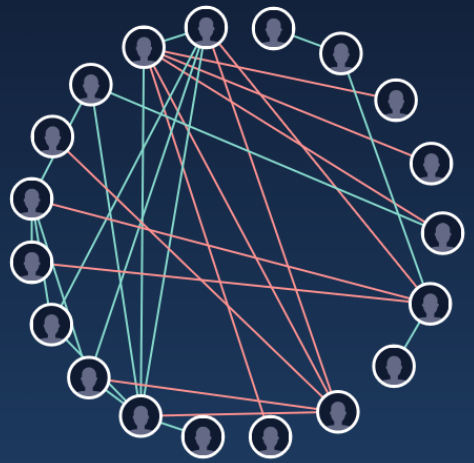
A study of the violent extremist network in Syria found that the network was balanced in 2012. However, in 2013 an increase in active groups in the Syrian theatre changed the existing landscape significantly.

Here is the network for the Syrian theatre in 2013. Click to run on the quantum computer to compute its structural balance.

SOLVE THIS PROBLEM USING THE D-WAVE QUANTUM COMPUTER

RUN

SYRIA 2013

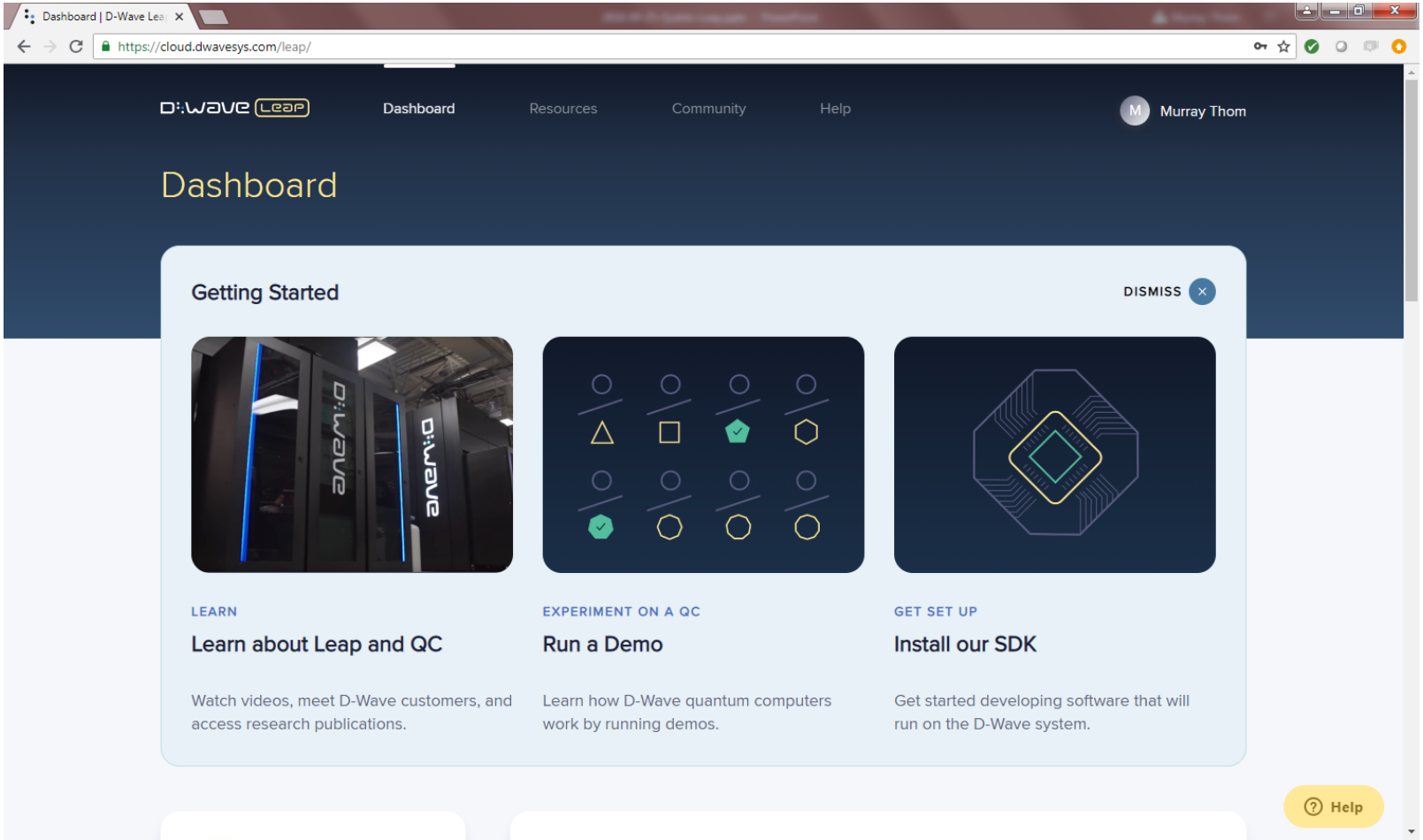


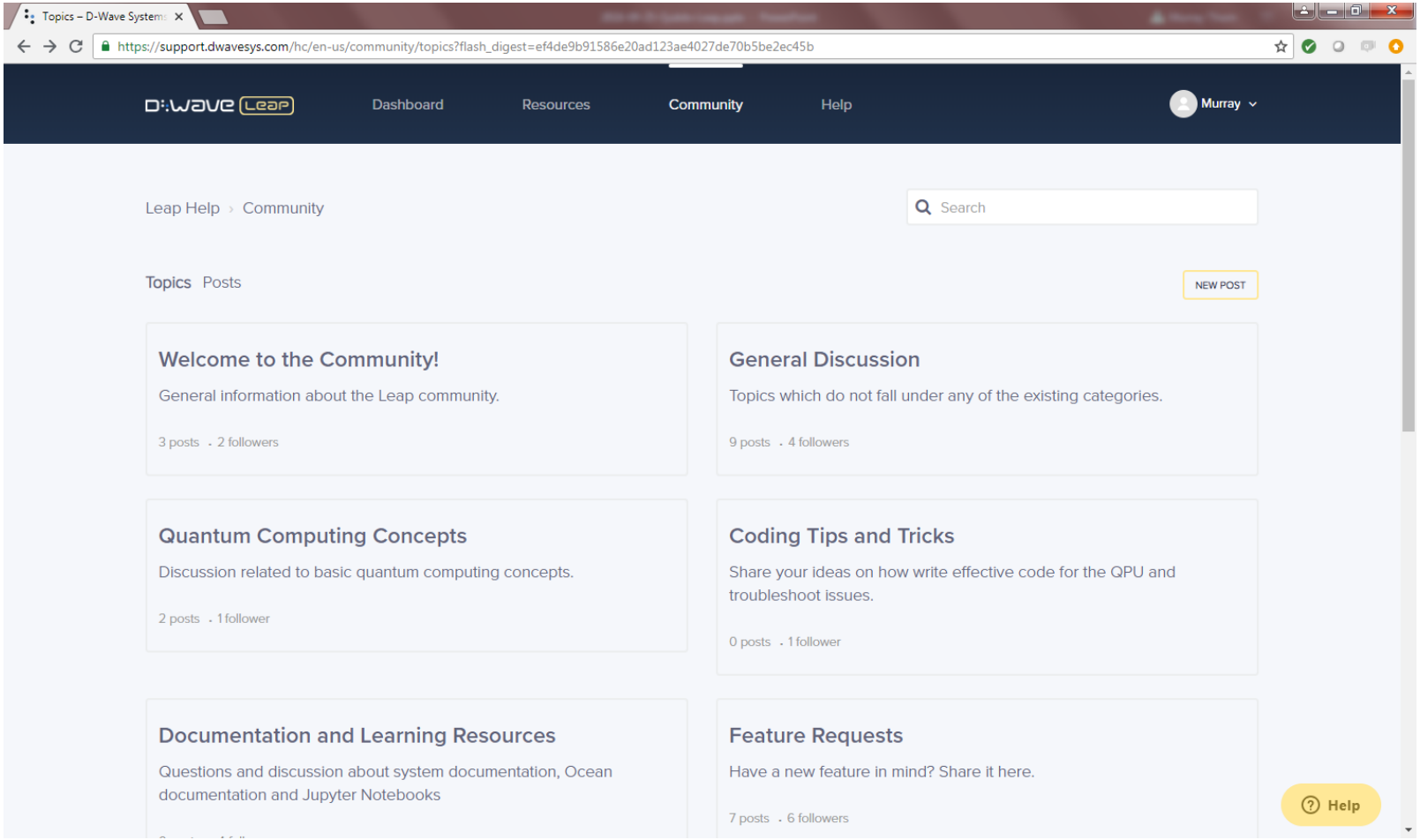
PREVIOUS STEP

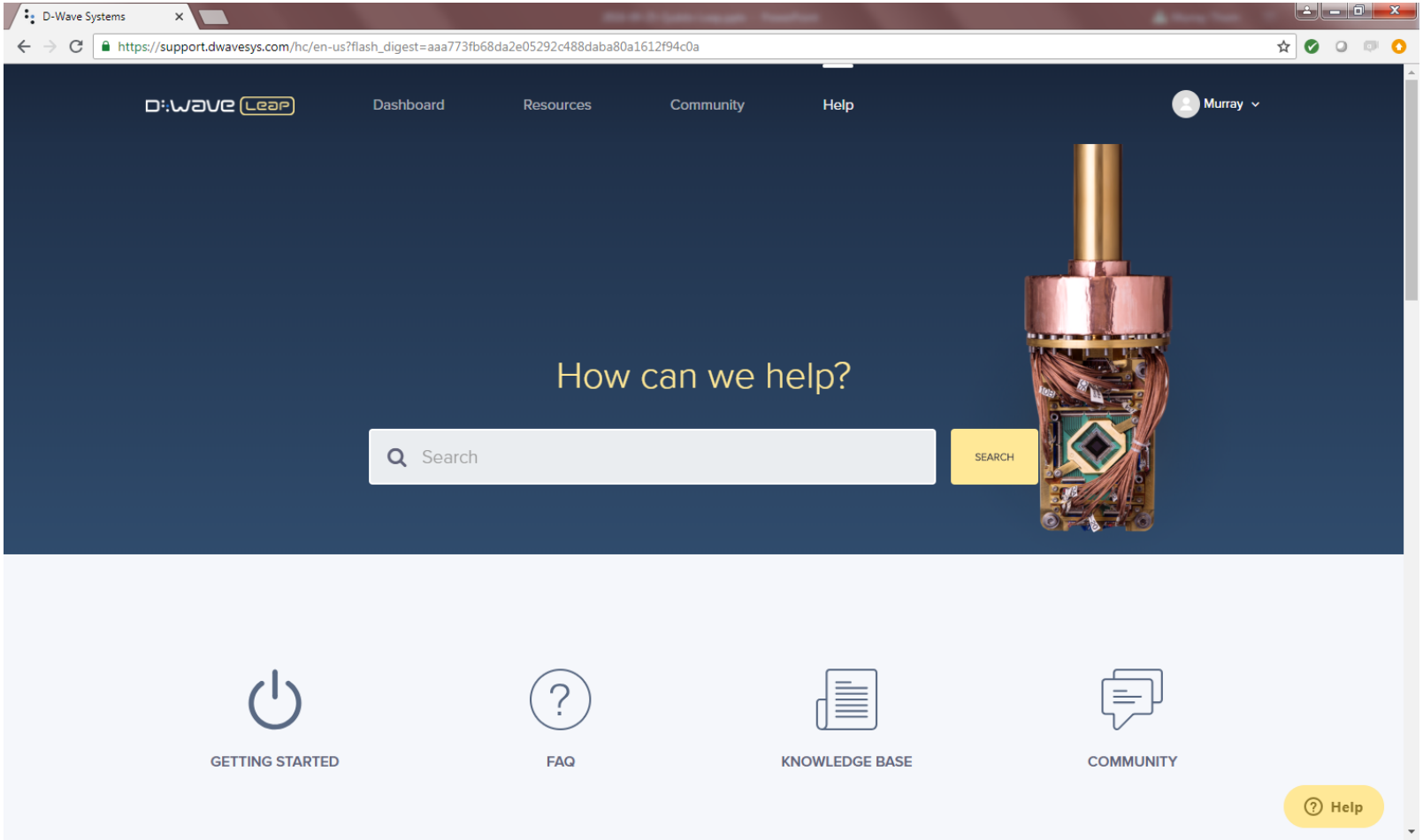
NEXT: Performance

CONTINUE

D:Wave The Quantum Computing Company™







D:Wave
The Quantum Computing Company™

D-Wave's Software Development Kit

Ocean

Alexander Condello

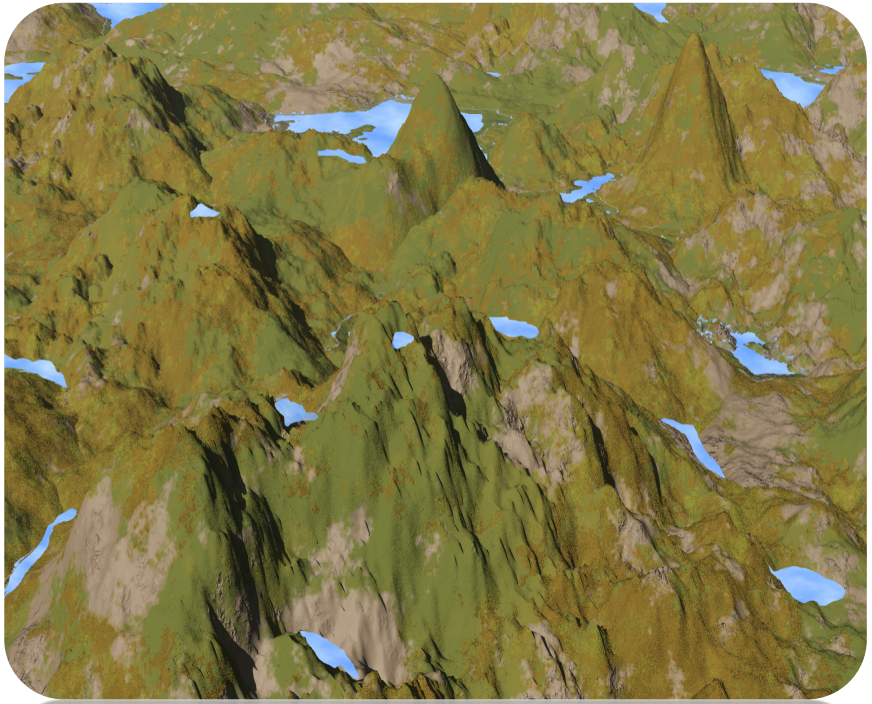
D-Wave

Landscape metaphor

Space of solutions defines an energy landscape and the best solution is the lowest valley

Classical algorithms can only walk over this landscape

Quantum annealing uses quantum effects



Binary Quadratic Model

$$E(v) = \sum_{i,j} v_i v_j a_{i,j} + \sum_i v_i b_i + c$$

$$a_i, b_{i,j}, c \in \mathbb{R}$$

$$v_i \in \{-1, +1\} \text{ or } v_i \in \{0, 1\}$$

Quantum Machine Instruction

$$E(s) = \sum_{i,j} s_i s_j J_{i,j} + \sum_i s_i h_i$$

$$J_{i,j} \in [-2, 1], h_i \in [-2, 2]$$

$$s_i \in \{-1, +1\}$$

Has a particular graph, defined by the hardware

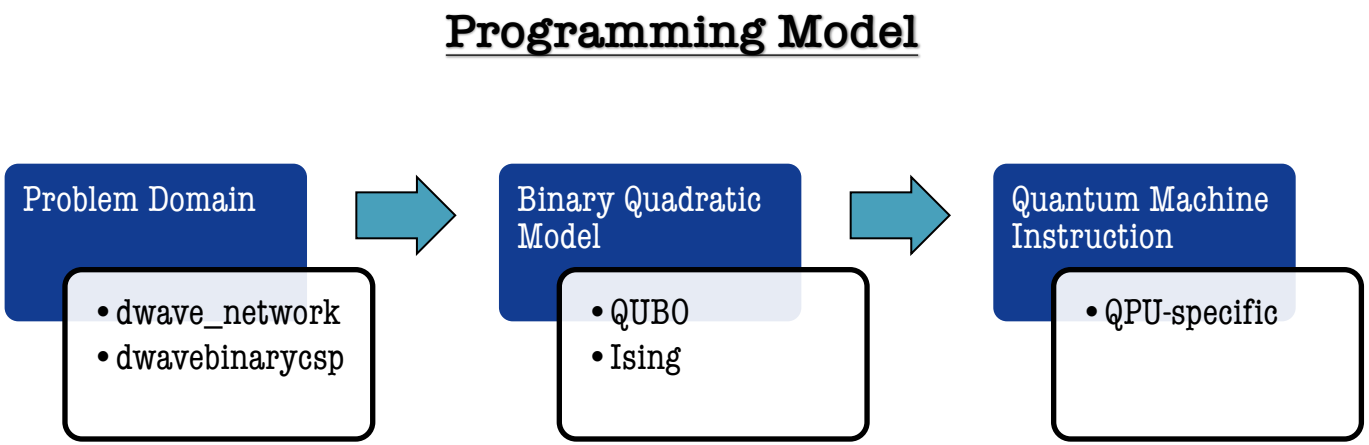
How do I program with BQMs?

D-Wave's Ocean Software

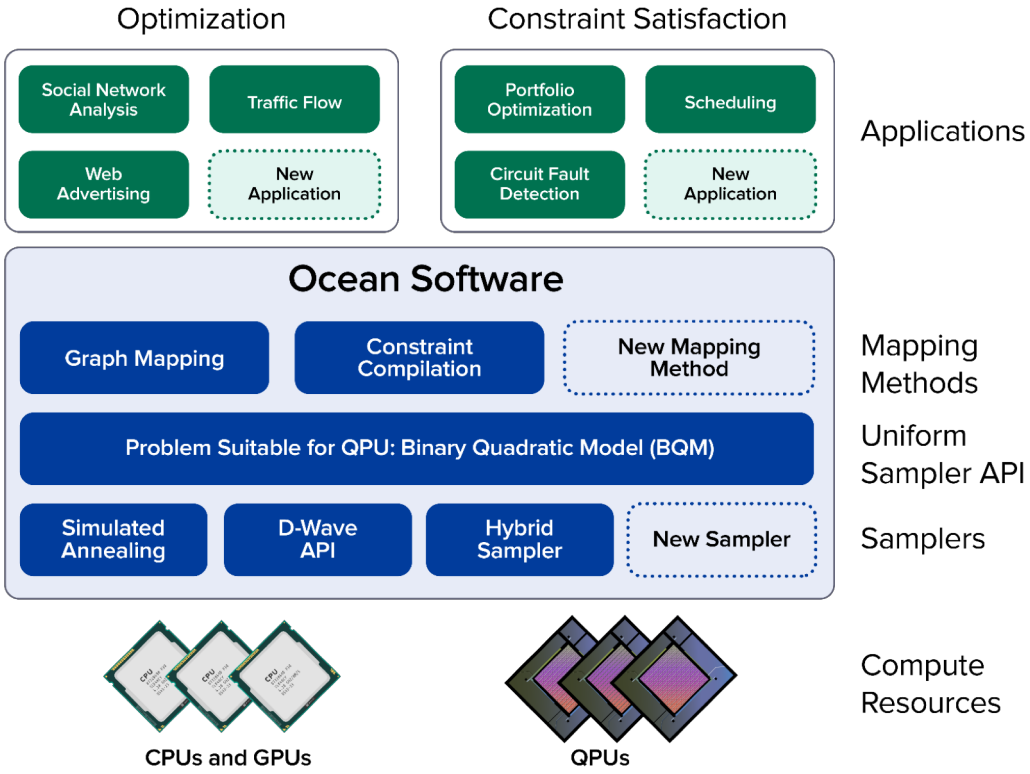
Ocean software is a suite of tools **D-Wave Systems** provides on the **D-Wave GitHub repository** for solving hard problems with quantum computers.

- Tools for solving BQM problems.
- Python front-end, some C++ components
- Majority is open-source code available on GitHub.
- Extensions and features from community welcome!

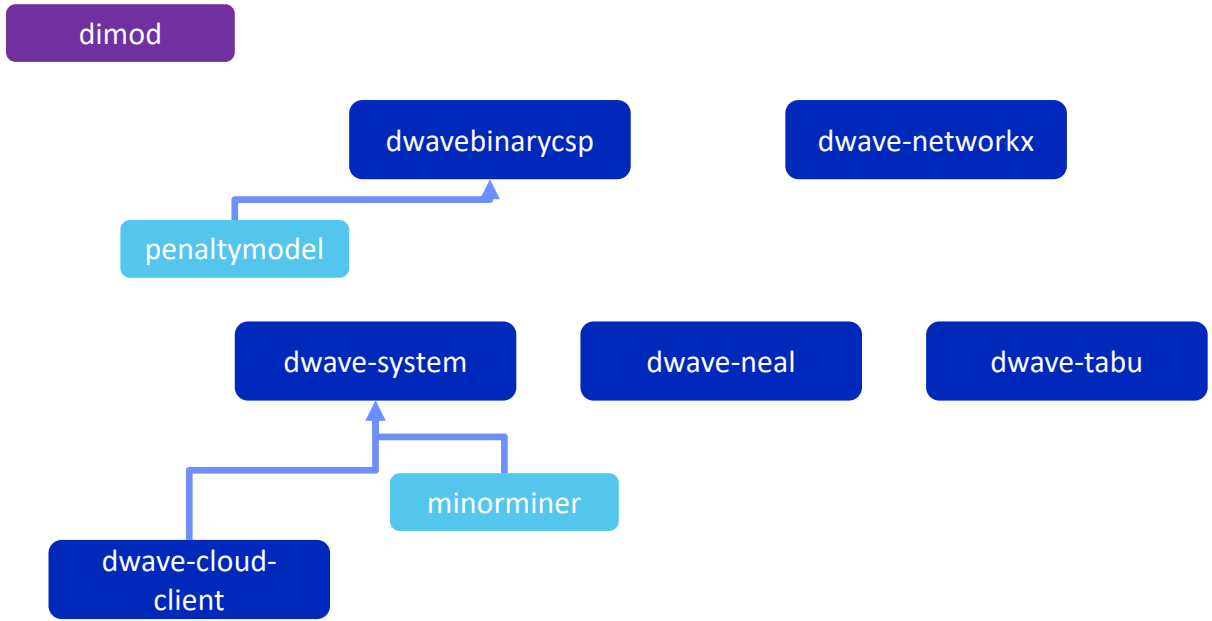
How does it all fit together?



Ocean Software Stack



Ocean Packages



Mapping Methods

Uniform Sampler API

Compute Resources

Ocean Packages

dimod

dimod

- Provides objects used throughout the Ocean stack
  - BQM
  - Sampler ABC

Ocean Packages

dimod

Mapping Methods

Uniform Sampler API

Compute Resources

Ocean Packages

dimod

dwave-cloud-client

- Handles communication with the QPU
- Stores token locally for reuse

Mapping Methods

Uniform Sampler API

dwave-cloud-client

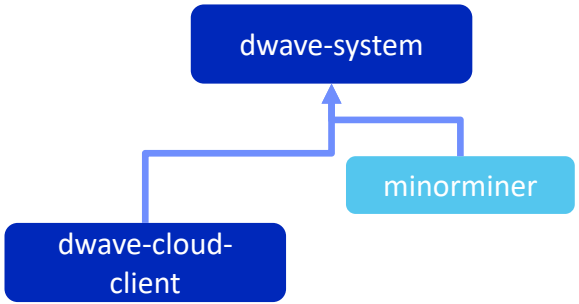
Compute Resources

Ocean Packages

dimod

dwave-system
- Access QPU as a Sampler

minorminer
- Tools for embedding problems onto the QPU



Mapping Methods

Uniform Sampler API

Compute Resources

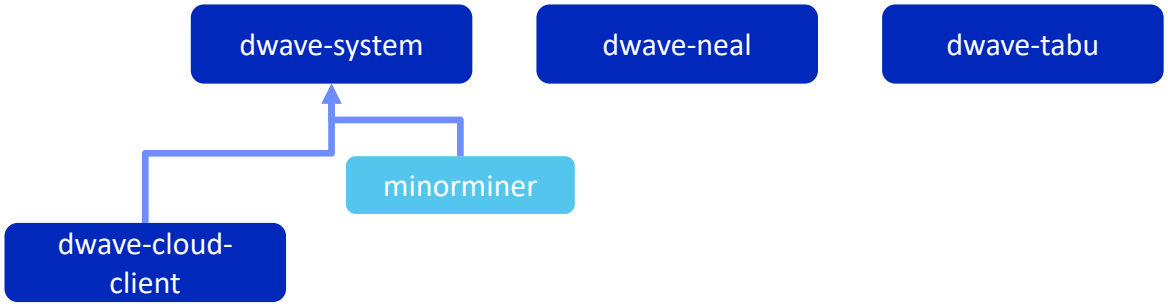
Ocean Packages

dimod

dwave-neal
- Simulated annealing

dwave-tabu
- Tabu search

Mapping Methods



Uniform Sampler API

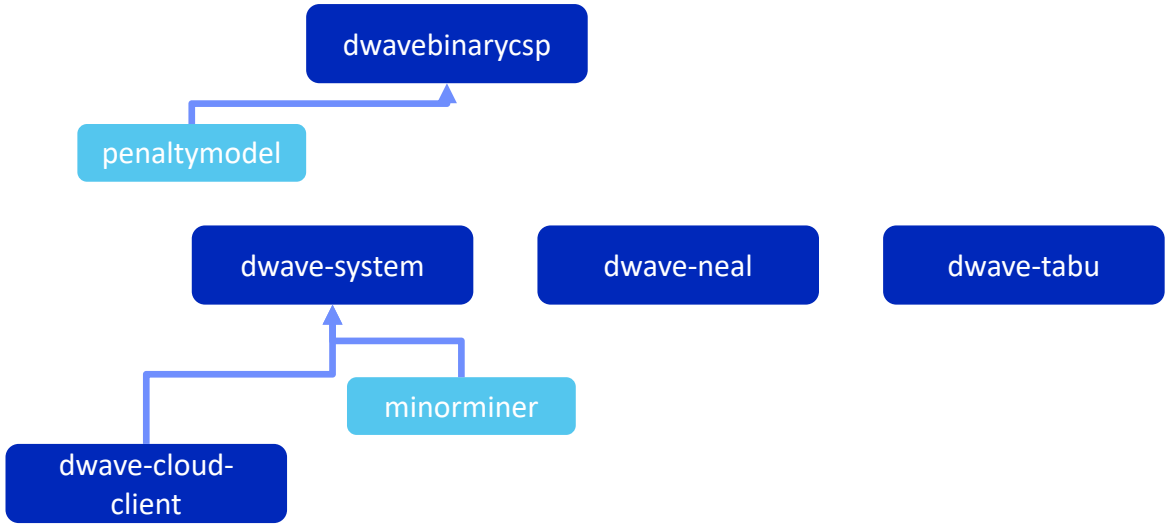
Compute Resources

Ocean Packages

dimod

dwavebinaryscp

- Construct a BQM from a constraint satisfaction problem



Mapping Methods

Uniform Sampler API

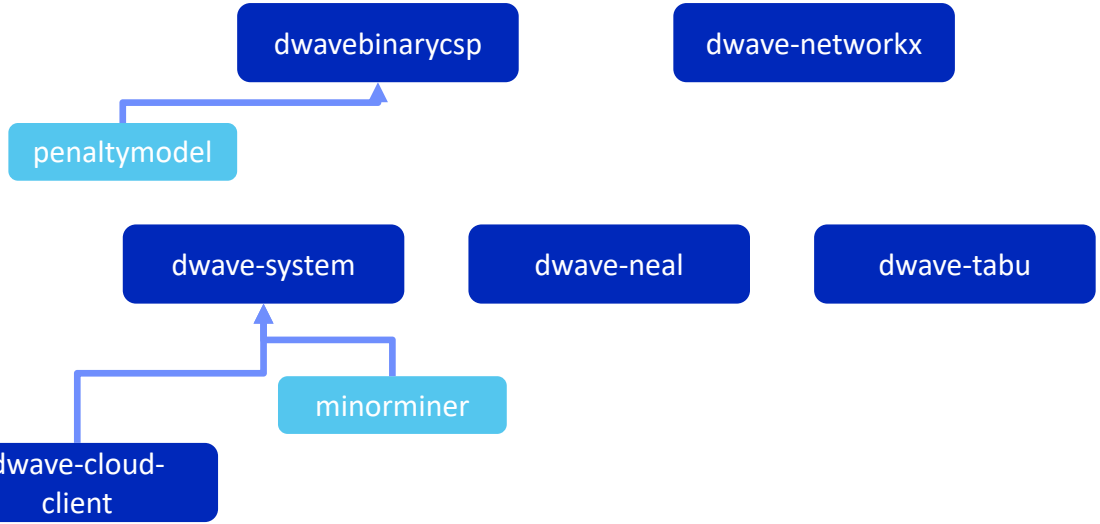
Compute Resources

Ocean Packages

dimod

dwave-networkx

- Solves graph problems using the QPU

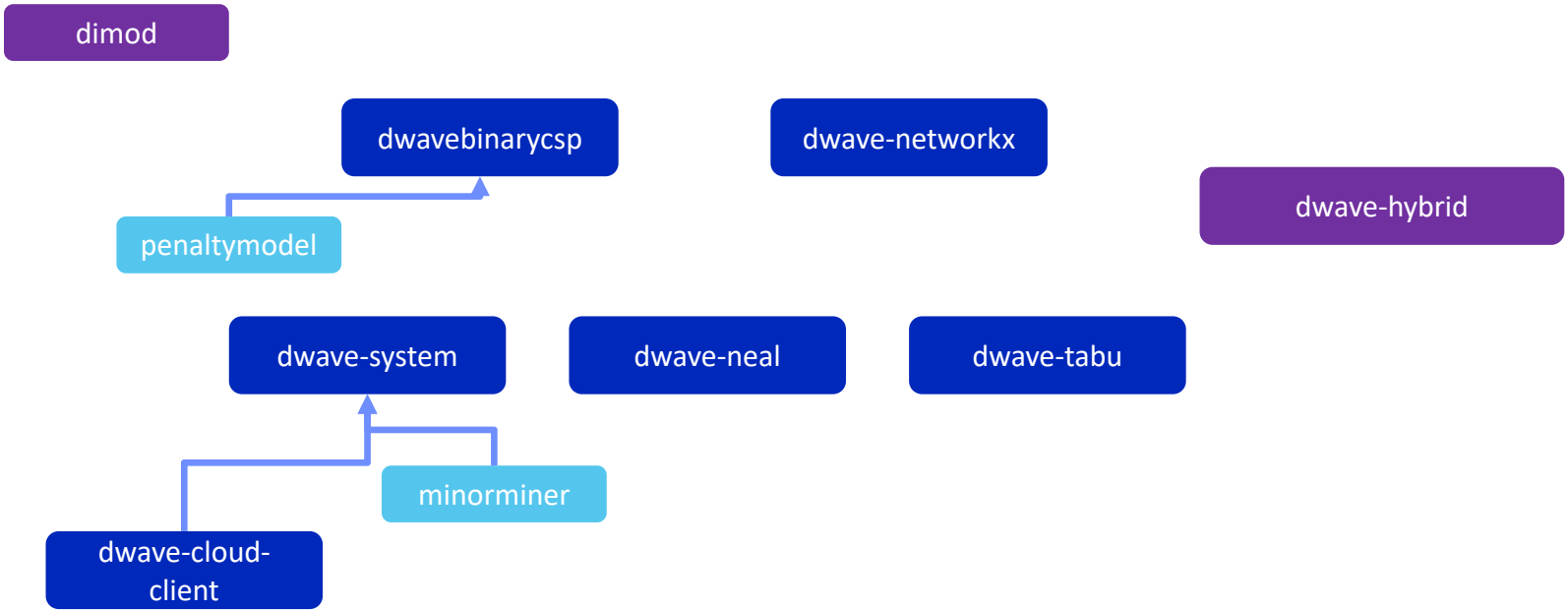


Mapping Methods

Uniform Sampler API

Compute Resources

Ocean Packages



QPU Programming Method

1. Translate your problem to binary optimization
2. Define your BQM function
3. Put the coefficients in matrix form
4. Run the matrix through a sampler
5. Post-process to interpret the results

Example

- Given:
  - Network of pipelines
- What do we want:
  - A set of junctions from which we can monitor every pipeline segment

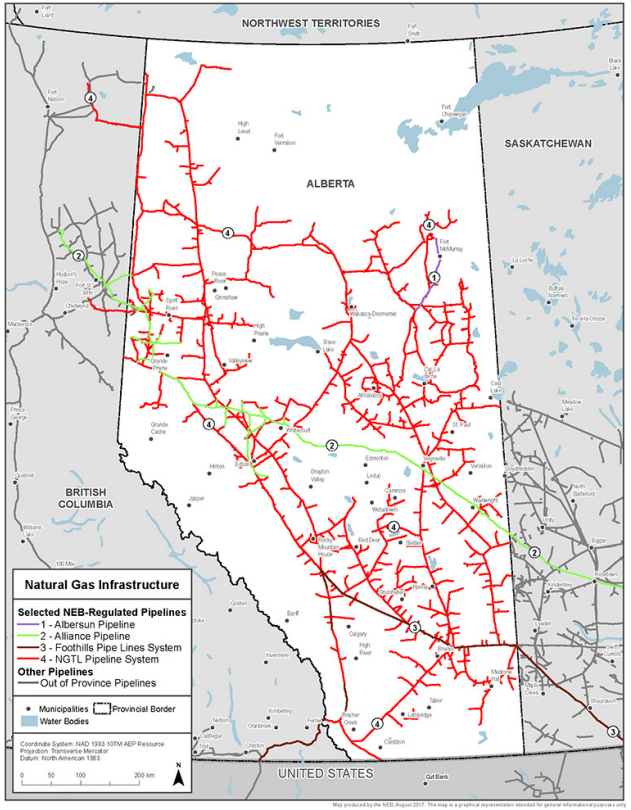
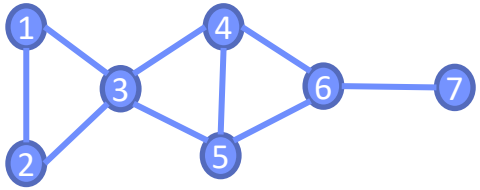
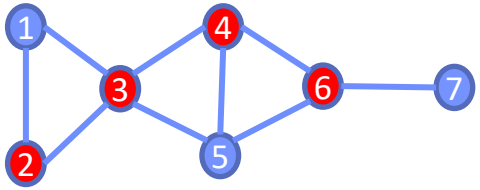


Image from National Energy Board of Canada,
https://www.neb-one.gc.ca/nrg/ntgrtd/mrkt/nrgsstmprfls/ab-eng.html

Example





Goal:

Junctions for monitoring pipes

Goal:

Nodes that *cover* every edge

In graph theory, this is called a **vertex cover**.

Without Ocean algorithm tools….

1. Write our problem in binary optimization form

$$\min \quad \sum_i x_i$$

$$\text{s.t.} \quad \sum_{(u,v)\in E} (x_u \cdot x_v - x_u - x_v) > 0$$

2. Translate to a QUBO

$$\min \left(\sum_i x_i \right) + \gamma \left(\sum_{(u,v)\in E} (x_u \cdot x_v - x_u - x_v) \right)$$

3. Simplify down to a QUBO matrix

$$\begin{matrix} -1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & -1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & -3 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & -2 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & -2 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & -2 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{matrix}$$

Without Ocean algorithm tools….

1. Write our problem in binary optimization form

$$\sum_i x_i$$

$$(x_u \cdot x_v - x_$$

2. Translate to a QUBO

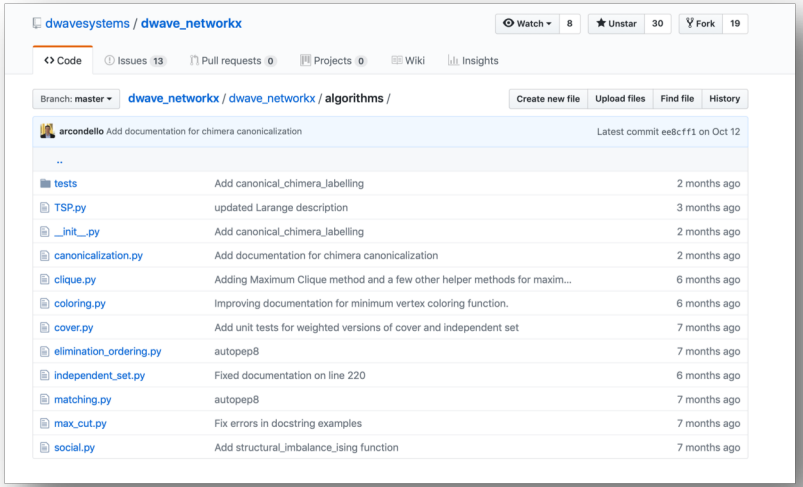
$$\min\left(\sum_i x_i\right) + \gamma \cdot x_v - x_u - x_v)\Big)$$

3. Simplify down to a QUBO matrix

$$\begin{matrix} & & 1 & 0 & 0 & & \\ -1 & 1 & 0 & 0 & & & \\ 1 & -3 & 1 & 1 & 0 & & \\ 0 & 0 & 1 & -2 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & -2 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & -2 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{matrix}$$

With Ocean algorithm tools!

- Identify the type of problem (e.g. min vertex cover)
- Search the Ocean Tool Suite for a tool



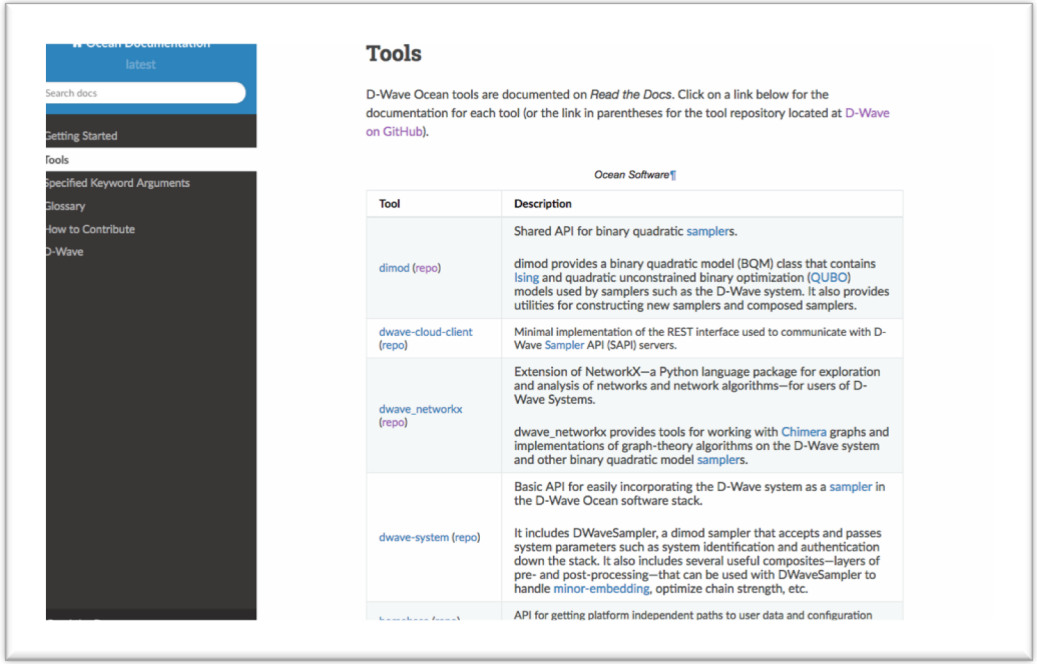
- Write a Python program ***in our problem domain*** and call the tool

Python Program using QPU

```
import networkx as nx
import dwave_networkx as dnx
from dwave.system import DWaveSampler, EmbeddingComposite

sampler = EmbeddingComposite(DWaveSampler())

G = nx.Graph()
G.add_edges_from([(1,2),(1,3),(2,3),(3,4),(3,5),(4,5),(4,6),
                  (5,6),(6,7)])

cover = dnx.min_vertex_cover(G, sampler=sampler)
```

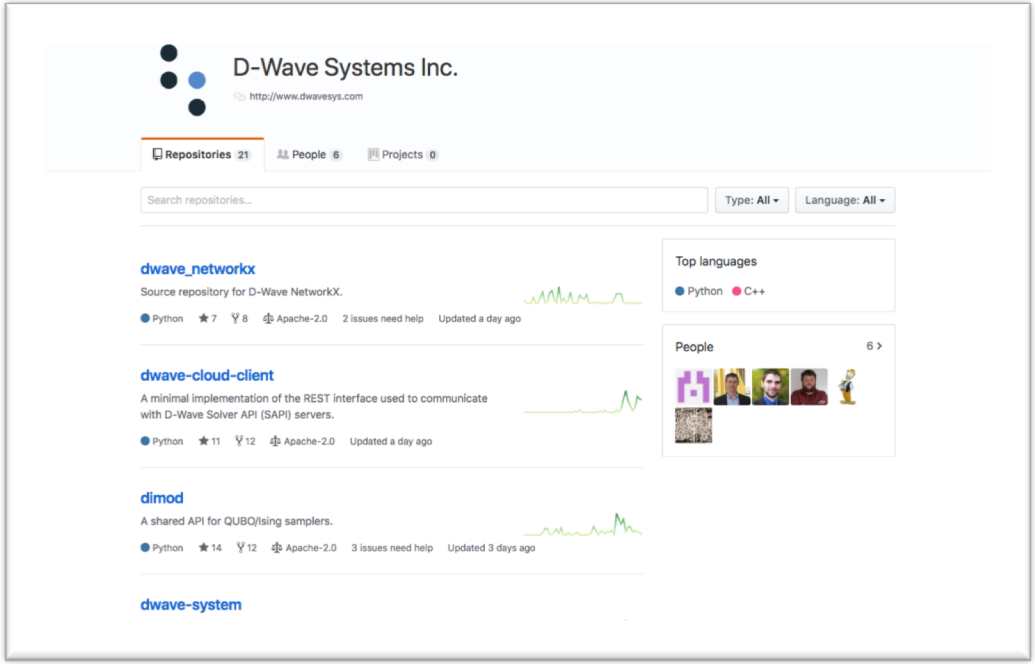
Where to find more information

- Read-the-docs
  - https://docs.ocean.dwavesys.com



Where to look at code and contribute

- D-Wave's Github
  - https://github.com/dwavesystems



What if there isn't a tool for my application?

OPTION 1

1. Develop BQM

2. Write Python program.
   - Set sampler to be used
   - `sampler.sample(bqm)`
   - Interpret results

What if there isn't a tool for my application?

OPTION 1

1. Develop BQM
2. Write Python program
   - Set sampler to be used
   - `sampler.sample(bqm)`
   - Interpret results

OPTION 2

1. Develop BQM
2. Write a reusable tool.
3. Write Python program that calls that tool.
4. Submit your tool to be included in our library.

How to contribute to Ocean

- Anyone can contribute to Ocean by making a pull request in GitHub.

- For a guide on how to make a pull request, check out this site:

  https://www.digitalocean.com/community/tutorials/how-to-create-a-pull-request-on-github

How to install

Installing the Ocean Tool Suite:

```
pip install dwave-ocean-sdk
```

- D-Wave's Github
  - https://github.com/dwavesystems
- Read-the-docs
  - https://docs.ocean.dwavesys.com/en/latest/