Building quantum applications with D-Wave's Leap

Alexander Condello
Before we start....

• (Almost) everything I will talk about is in open source
• Documentation is online and in LEAP
• We want feedback!
  • Community
  • Issues
  • Pull requests
• Your feedback helps us prioritize
Landscape metaphor

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

• Given:
  • Network of pipelines

• What do we want:
  • A (minimum) set of junctions from which we can monitor every pipeline segment
Example

Junctions for monitoring pipes

Goal:

Nodes that cover every edge

A vertex cover.
With the Ocean tools...

```python
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)
```
**Ocean**™ software development kit
Suite of open-source Python tools on the D-Wave GitHub repository

**Optimization**
- Portfolio optimization
- Traffic flow
- New application

**Constraint satisfaction**
- Scheduling
- Circuit fault detection
- New application

### Applications
- New application
- New mapping method

### Mapping methods
- Uniform sampler API

### Samplers
- Simulated annealing
- D-Wave API
- Hybrid sampler
- New sampler

### Compute resources
- CPUs and GPUs
- QPUs

Problem Suitable for QPU: Binary Quadratic Model (BQM)
With the Ocean tools...

```python
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)
```
Binary Quadratic Model

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

where \( a_{i,j}, b_i, c \in \mathbb{R} \)

\( v_i \in \{-1, +1\} \) or \( v_i \in \{0, 1\} \)
Binary Quadratic Model and pipelines

Binary Variables
- Each junction either has a sensor or no

Pairwise interactions
- Every edge needs a sensor

Linear optimization
- Minimum number of sensors

$$E(v) = \alpha \sum_{v \in V} v + \sum_{u,v \in E} (1 - u)(1 - v)$$
Materials Properties
- Atomic magnetometer
- Solid state materials simulation
- Quantum molecular dynamics
- Quantum chemistry computation

Machine Learning
- Finding Higgs Boson
- Image recognition
- Tree cover classifier
- DNA binding
- Individual cancer drugs

Optimization
- Radiotherapy
- Multi-period portfolios
- Satellite placement
- Traffic flow
- Internet ad placement

Cyber Security & Fault Detection
- Formation of Terrorist Networks
- Fault detection in circuits
- Facial recognition

EARLY APPLICATIONS
- 200+

Copyright © D-Wave Systems Inc.
BQM to QMI
The D-Wave System
Quantum Processing Unit (QPU)
Ising Hamiltonian

\[ H_{\text{ISING}} = -\frac{A(s)}{2} \left( \sum_i \sigma_x^{(i)} \right) + \frac{B(s)}{2} \left( \sum_i h_i \sigma_z^{(i)} + \sum_{i>j} J_{i,j} \sigma_z^{(i)} \sigma_z^{(j)} \right) \]

Quantum Machine Instruction (QMI)
Binary Quadratic Model

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

\[ a_{i,j}, b_i, c \in \mathbb{R} \]

\[ v_i \in \{-1, +1\} \text{ or } v_i \in \{0, 1\} \]
Quantum Machine Instruction (QMI)

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

+ Rules

\[ v_i \in \{-1, 1\} \]

\[ a_{i,j} \in [-2, 1], b_i \in [-1, 1], c \in \mathbb{R} \]

Hardware structured
Qubit Layout

2000Q
Working Graphs

2000Q

Advantage

Qubits:
- 2000
- 5000

Couplers:
- 6000
- 40000
A Closer Look...

2000Q

![Chimera](image)

![D-Wave](image)

**6**  **Average Degree**  **15**

Advantage

![Pegasus](image)

![D-Wave](image)
Hybrid Algorithm Development
dwave-hybrid

• Hybrid Asynchronous Decomposition Sampler framework
  • Minimal, Python, solver/sampler-building framework, built atop Ocean tools
  • Leverages quantum and classical resources
  • Independent parts are executed concurrently
  • Problems are broken into pieces that fit the compute resources
  • Uses sample sets (probabilistic approach)
Motivation

Algorithm 1: Partitioning algorithm implemented by qbsolv

1. Input: QUBO instance
2. If best_energy is the lowest value found to date
3. best_energy, best_solution ← TabuSearch(QUBO, solution)
4. index ← OrderByImpact(QUBO, best_solution)
5. passCount ← 0
6. solution ← best_solution
7. while passCount < numRepeats do
8. change ← false
9. for i = 0, i < fraction * size(QUBO), i = subQUBOSize do
10. # select subQUBO with other variables clamped
11. sub_index ← i; i+subQUBOSize-1
12. subQUBO ← Clamp(QUBO, solution, index, sub_index)
13. (sub_energy, sub_solution) ← DWaveSearch(subQUBO)
14. if (solution | sub_index) ≠ sub_solution then
15. solution | sub_index ← sub_solution
16. change ← true
17. end if
18. end for
19. if not change then
20. Randomize(solution[0 : i − 1])
21. end if
22. best.energy ← energy
23. best.solution ← solution
24. passCount ← 0
25. else
26. passCount += 1
27. end if
28. index ← OrderByImpact(QUBO, solution)
29. end while
30. Output: best.energy, best_solution

Loop(RacingBranches(
  InterruptableTabuSampler(),
  EnergyImpactDecomposer(size=50)
  | QPUSubproblemAutoEmbeddingSampler()
  | SplatComposer()
) | ArgMin())
Example

- Find a sub-problem with a high energy impact and solve that on the QPU

EnergyImpactDecomposer(size=50) | QPUSubproblemAutoEmbeddingSampler() | SplatComposer()
Pre-processing

Example

- Use roof duality to determine and fix the assignments of some variables in polynomial time

References:


Post-processing

Example

• Use the QPU to seed another classical algorithm

QPU Sampler \rightarrow \text{Tabu Sampler}

QPUSubproblemAutoEmbeddingSampler() \mid \text{TabuSubproblemSampler()}

Copyright © D-Wave Systems Inc.
Meta-algorithms

Example

• Use a quantum computer to build a strong classifier from a selection of weak classifiers

• Co-developed with Google to train image classifiers for cars


Demo of Qboost

The D-Wave quantum computer has been widely studied as a discrete optimization engine that accepts any problem formulated as quadratic unconstrained binary optimization (QUBO). In 2008, Google and D-Wave published a paper, Training a Binary Classifier with the Quantum Adiabatic Algorithm, which describes how the QBoost ensemble method makes binary classification amenable to quantum computing: the problem is formulated as a thresholded linear superposition of a set of weak classifiers and the D-Wave quantum computer is used to optimize the weights in a learning process that strives to minimize the training error and number of weak classifiers.

This code demonstrates the use of the D-Wave system to solve a binary classification problem using the Qboost algorithm.

Disclaimer

This demo and its code are intended for demonstrative purposes only and are not designed for performance.

Usage

A minimal working example using the main interface function can be seen by running:

```
python demo.py --datadir --dataset
```
Example

• Run tabu while waiting on the result from the QPU

RacingBranches(
    InterruptableTabuSampler(),
    EnergyImpactDecomposer(size=50)
    | QPUSubproblemAutoEmbeddingSampler()
    | SplatComposer()
) | ArgMin()