The Hybrid Solver Service

D-Wave
The Quantum Computing Company™

Copyright © D-Wave Systems Inc.

Agenda

- Introduction to The Hybrid Solver Service
- Problem – Knapsack
- Conclusions

Copyright © D-Wave Systems Inc.

Introduction to the Hybrid Solver Service

Copyright © D-Wave Systems Inc.

Leap^{TM} Hybrid Solver Service: Expands Problem Size & Ease Of Use

Submit complex problems and get good answers without having to know anything about the D-Wave quantum processing unit (QPU):

- Run problems with up to 20,000 variables fully connected
- Problems run on classical and quantum computing resources using an algorithm portfolio
- Easily accessed in Leap user interface as a hybrid solver
- Shorter learning curve
  - No embedding or processor parameters to worry about

Copyright © D-Wave Systems Inc.

Hybrid Quantum-Classical Approach

The portfolio approach:

- Hybrid solvers target different categories of inputs and use cases
- During its operation, each heuristic solver formulates some number of quantum queries, which are sent to the QPU
- The classical and quantum components in each solver communicate asynchronously so that one part of the system does not block another

Copyright © D-Wave Systems Inc.

Hybrid Quantum-Classical Approach: Quantum Acceleration

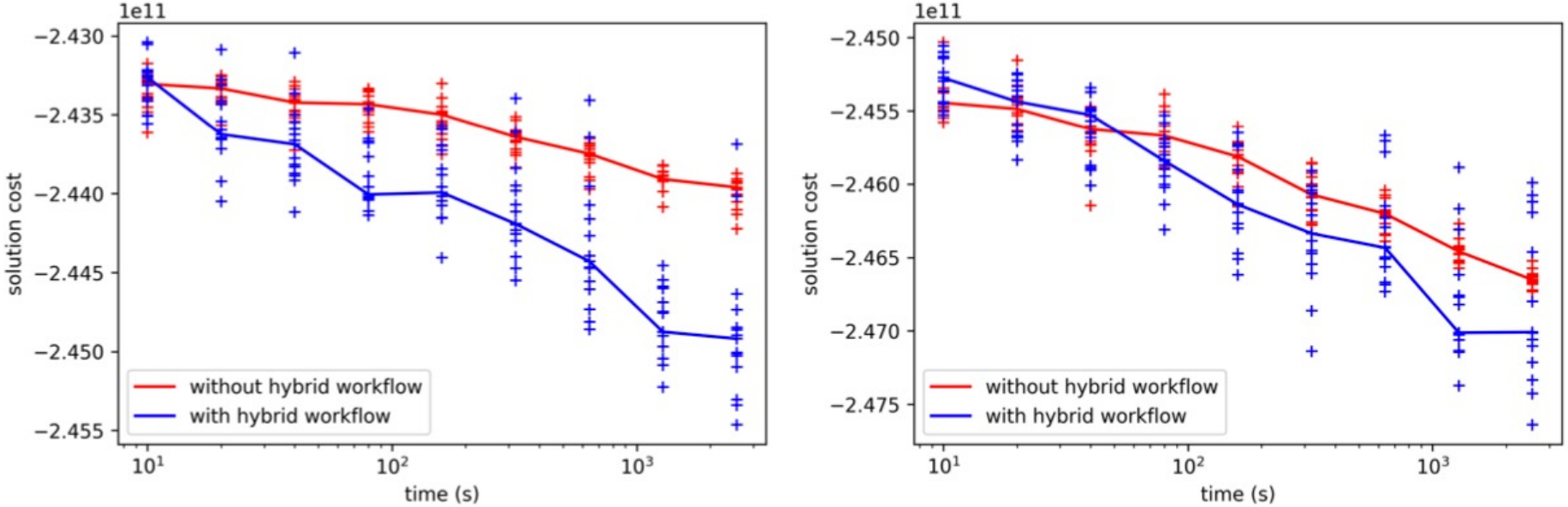
At the end of the time limit:

- All solvers stop
- Best solution is returned to the user
- The user did not have to decide which solution approach is best for any given input

Copyright © D-Wave Systems Inc.

Hybrid Approach Wins Over Heuristic-Only (White Paper)

- Reduced solution cost (measures solution quality)
- Both dense (left graph) and sparse (right graph)
- Quantum queries lead to more rapid gains in solution quality



Copyright © D-Wave Systems Inc.

The Time Limit Parameter *T*

There is only one user parameter: time limit *T*, in units of seconds

- The system calculates a minimum time limit that scales with input problem size (and it may be used by default)
- The minimum time limit ensures that each hybrid solver has enough time to both perform a first step and to query and have a good chance to receive at least one response from the D-Wave QPU
- Currently the minimum time limit for all problems is 3 seconds
- Currently the maximum time is 24 hours

Copyright © D-Wave Systems Inc.

How To Solve A QUBO Problem?

Formulate a QUBO (quadratic unconstrained binary optimization) for the problem

Write a Python program using the Ocean API, which implements the QUBO

Run the Python program on a sampler/solver

Copyright © D-Wave Systems Inc.

Using BQM Solver Is Easy

```
from dwave.system import LeapHybridSampler
sampler = LeapHybridSampler()
response = sampler.sample(bqm, time_limit=5)
```

Things to notice:

- No embedding to worry about
- No *chain_strength* to think about
- Starting *time_limit* at the minimum; increase if needed

Copyright © D-Wave Systems Inc.

Knapsack problem

Copyright © D-Wave Systems Inc.

Problem Specifications

- We want to pack a shipping container or a knapsack or a backpack…
- We have a collection of objects
- Each object has a weight w_i and a value v_i
- Fill up the container so that we respect a maximum weight W
- Maximize the total value in the container

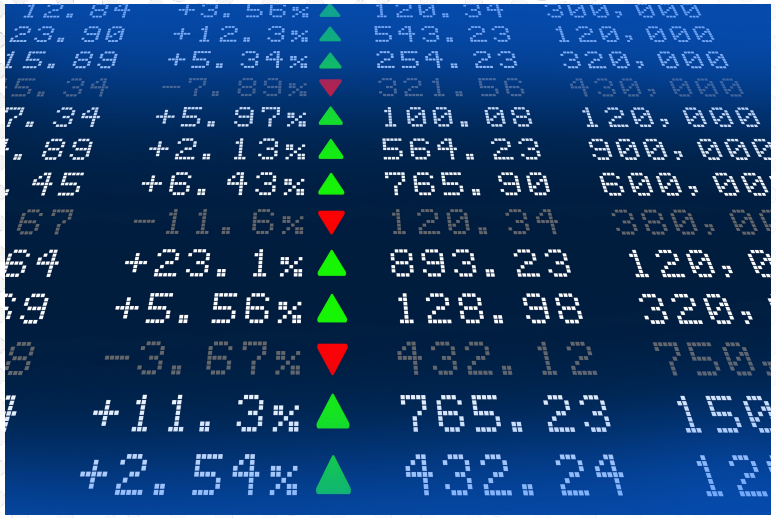
Copyright © D-Wave Systems Inc.

Example Knapsack Problems

Choosing which ads to bid on in Internet ad auctions, on fixed budget

Forestry service wants to bid on land packages with fixed budget

Financial portfolios: maximize your profits on a fixed budget



Copyright © D-Wave Systems Inc.

QUBO Step 1

Objective:

– Maximize the total value of the selected objects

Constraints:

– The total weight must be less than or equal to *W*

Copyright © D-Wave Systems Inc.

QUBO Step 2: Binary Variables

Binary Variables:

For each object we ask: “Is this object in the knapsack or not?”

We can use the binary variables to “label” the objects.

$$x_i = \begin{cases} 1, \text{if object } i \text{ is in the knapsack} \\ 0, \text{if object } i \text{ is not in the knapsack} \end{cases}$$

Copyright © D-Wave Systems Inc.

The QUBO – Write The Objective First

$$H = -\sum_{\alpha} v_{\alpha}\, x_{\alpha}$$

Minimizes the sum of values – makes it as negative as possible – hence the most value possible

Remember that the x_{α} are binary variables, and will be returned by the solver (output)

The v_{α} are the values of the individual objects (input)

Copyright © D-Wave Systems Inc.

The QUBO – Try To Write The Constraint

$$H = A\,(W - \sum_\alpha w_\alpha\, x_\alpha)^2 - \sum_\alpha v_\alpha\, x_\alpha$$

What's wrong with this?

This assumes *W* is *equal* to the sum of the weights, but we want it to be less than or equal to

We need to add *auxiliary variables* – so that it may be less than by one, or less than by two …

Copyright © D-Wave Systems Inc.

The QUBO Formulation*

$$H = -\sum_i v_i x_i \tag{1}$$

$$\text{s.t.} \tag{2}$$

$$\sum_i w_i x_i \leq W \tag{3}$$

$$H = -\sum_i v_i x_i + A \left(\sum_i c_i y_i - \sum_i w_i x_i \right)^2 \tag{4}$$

$$c_i = 2^i \quad \text{for} \quad i = 1, ..., M-1 \tag{5}$$

$$c_M = W + 1 - 2^M \tag{6}$$

$$M = \lceil \log_2 W \rceil \tag{7}$$

- Andrew Lucas, Ising formulations of many NP problems, Front. Phys., 12 Feb. 2014.
- Andrew Lucas, "NP-hard combinatorial problems as Ising spin glasses", Workshop on Classical and Quantum Optimization; ETH Zuerich - August 20, 2014

Copyright © D-Wave Systems Inc.

Python Code

```
df = pd.read_csv(data_file_name, header=None)
df.columns = ['cost', 'weight']
bqm = knapsack_bqm(df['cost'], df['weight'], weight_capacity)
sampler = LeapHybridSampler()
response = sampler.sample(bqm)
process_results(response)
```

Copyright © D-Wave Systems Inc.

Run Using HSS On Small Data Set With Max Weight 50

Value	Weight(kg)
35	12
85	27
30	11
50	17
70	20
80	10
55	15

Solution:

Choose [20, 10, 15]

Energy = -205

HSS finds the solution promptly with time_limit = 3

The energy looks good – matches the total value

The weights add up to less than or equal to 50 kg

Copyright © D-Wave Systems Inc.

Run Using HSS On Larger Data Set With Max Weight 70

Value	Weight(kg)
84	4
78	20
91	1
96	19
83	14
92	10
77	95
44	70
…	

68 objects in data set
Problem cannot be embedded on DW-2000Q

The hybrid solver finds solution with run_time and charge_time close to 3 seconds

Solution is weights [4, 20, 1, 19, 14, 10]
Solution satisfies constraints (less than 70 kg)
Energy equals the summed objective – values add to -524

The hybrid solver finds the known optimal solution (found elsewhere by classical methods)

This example is in the github.com/dwave-examples repository

Copyright © D-Wave Systems Inc.

Analysis

- This problem is too big for the QPU – does not embed
- The hybrid solver makes use of the QPU
- Research question: will the hybrid solver find the optimal solution every time if we give it longer run times

Copyright © D-Wave Systems Inc.

Conclusions

Copyright © D-Wave Systems Inc.

Conclusions

- HSS is very easy to use
- HSS runs on very large problems
- We saw it do very well on Knapsack

Copyright © D-Wave Systems Inc.