D:wave
The Quantum Computing Company™

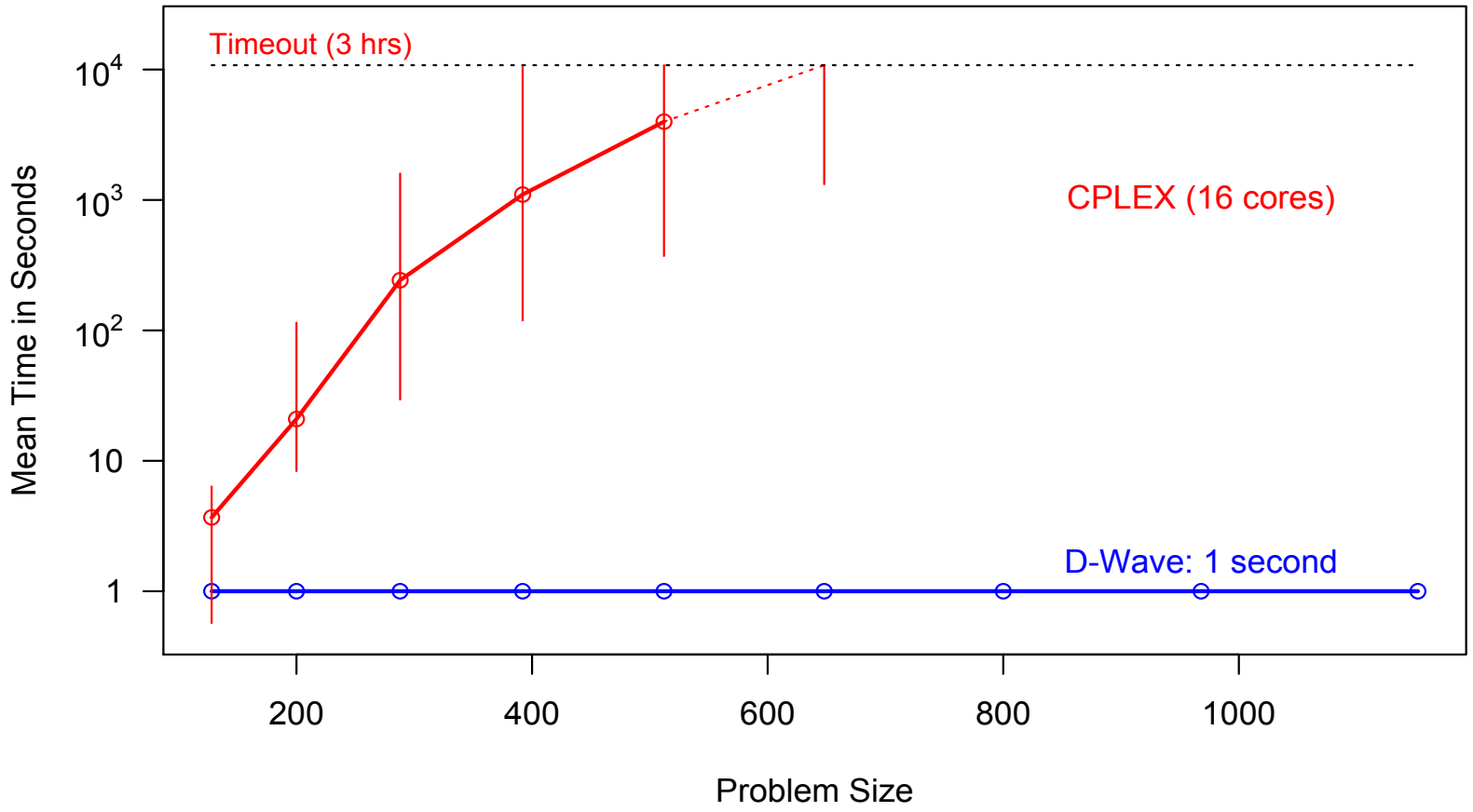
qbsolv: Creating Applications with QUBOs Bigger than Hardware

Overview

- Where the D-Wave solver fits in the universe
- D-Wave current software environment
- The need for qbsolv
- How to use qbsolv
- Performance
- How to make an application (lab)
- Summary
- qOp Installations

Collapsing to a solution (Classical v. Quantum Anneal)

Number of operations doesn't apply



D:wave
The Quantum Computing Company™

Thinking about thinking

- Reasoning

  1: the use of reason; especially : the drawing of inferences or conclusions through the use of reason

  2: an instance of the use of reason : argument

- "Classical" computer programming is designed around a logical/reasoned process of steps to arrive at an answer

- Problems arise when the logical processes have exponential operational counts when solving problems in the class of NP-hard

NP (Nondeterministic Polynomial Time)

- NP (-hard and -complete)

  NP is a complexity class that represents the set of all decision problems for which the instances where the answer is Yes have proofs that can be verified in polynomial time. To exhaustively search through all possible solutions requires exponential operations.

- Example: Traveling Salesman Problem (NP-hard)
  - No straight-forward algorithm to solve this problem in "polynomial time"
  - Brute force (compare all possible routes) ~ n! or for n=10 3,628,800 compares
  - Clever methods can reduce it to ~(n^2 * 2^n) or for n=10 102,400 compares
  - Web Demo

Problems like these are the most challenging for computers that use a series of logical reasoned steps

D:Wave
The Quantum Computing Company™

Thinking about thinking

- Intuition

1: quick and ready insight

2: a : immediate apprehension or cognition

b : knowledge or conviction gained by intuition

c : the power or faculty of attaining to direct knowledge or cognition without evident rational thought and inference

- Intuition can be used to rapidly find where to look for an answer; reasoning should then be employed to validate the result.
- NP problem-solving fits well into this (intuit <-> validate) process

Intuition needs logical validation

A bat and ball cost $ 1.10.
The bat costs one dollar more than the ball.
How much does the ball cost?

- A number came to your mind. The number, of course, is 10¢. The distinctive mark of this easy puzzle is that it evokes an answer that is intuitive, appealing, and **wrong**.
- Do the math, and you will see. If the ball costs 10¢, then the total cost will be $ 1.20 (10¢ for the ball and $ 1.10 for the bat), not $ 1.10.
- The correct answer is 5¢. It is safe to assume that the intuitive answer also came to the mind of those who ended up with the correct number— they somehow managed to resist the intuition.

  Kahneman,Daniel (2011-10-25). Thinking, Fast and Slow (p. 44). Farrar, Straus and Giroux. Kindle Edition.

D-Wave needs Classical computers

- Utilizing a D-Wave
  - Is like adding the speed of intuition to logical reasoned processes
  - The answer to the D-Wave QMI can be likened more to intuition than a reasoned process

- A classical computer, like reason, can stand alone
  - BUT there are classes of problems that will overwhelm a classical computer

- A D-Wave QMI, like intuition, cannot stand alone
  - Needs pre- and post-processing
  - Needs pre- and post-problem transformation

Approaches to NP problems

- Every NP-complete problem is reducible to every other NP-complete problem in polynomial time.

- To date, two methods have been the most successful in solving NP problems

    - One approach is to reduce the problem to a common NP problem then use a **multi-purpose solver**
    - The other approach is a hand-tailored application for each problem

- The D-Wave QMI is an NP-complete "solver" that solves the problem in "arguably" polynomial time

Approaches to NP problems

- Optimizations, like Traveling Salesman (TSP)
  - Approached with hand-tailored algorithms
    - Lowering the search space to a smaller exponential
    - Tailoring usually takes advantage of knowledge of the specific problem

- Constraint Satisfaction Problems (CSP)
  - Some approaches are hand-tailored
  - And **multi-purpose solvers**
    - SAT solvers (WalkSAT)
    - DIMACS input format

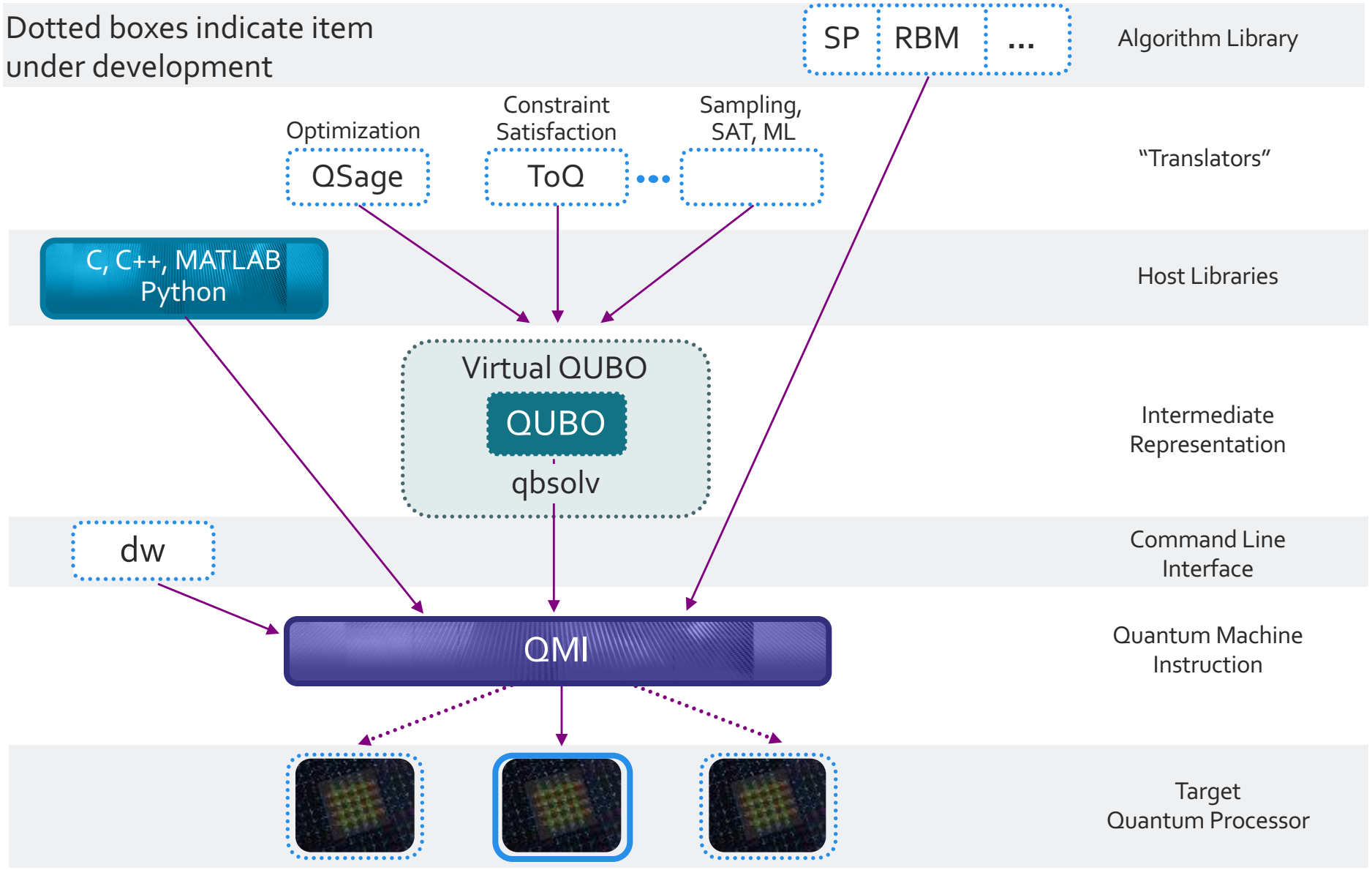
Approaches to NP problems

- To use the D-Wave, you must use the second method of converting your problem to another NP-complete problem (QUBO)

  Three issues:
  - The size is limited to currently manufactured number of qubits
  - The D-Wave system's Chimera topology is sparse
  - The D-Wave system's Chimera topology has specific connectivity

- The qOp tools are designed to assist the developer in using the D-Wave hardware.

D-Wave Software Environment

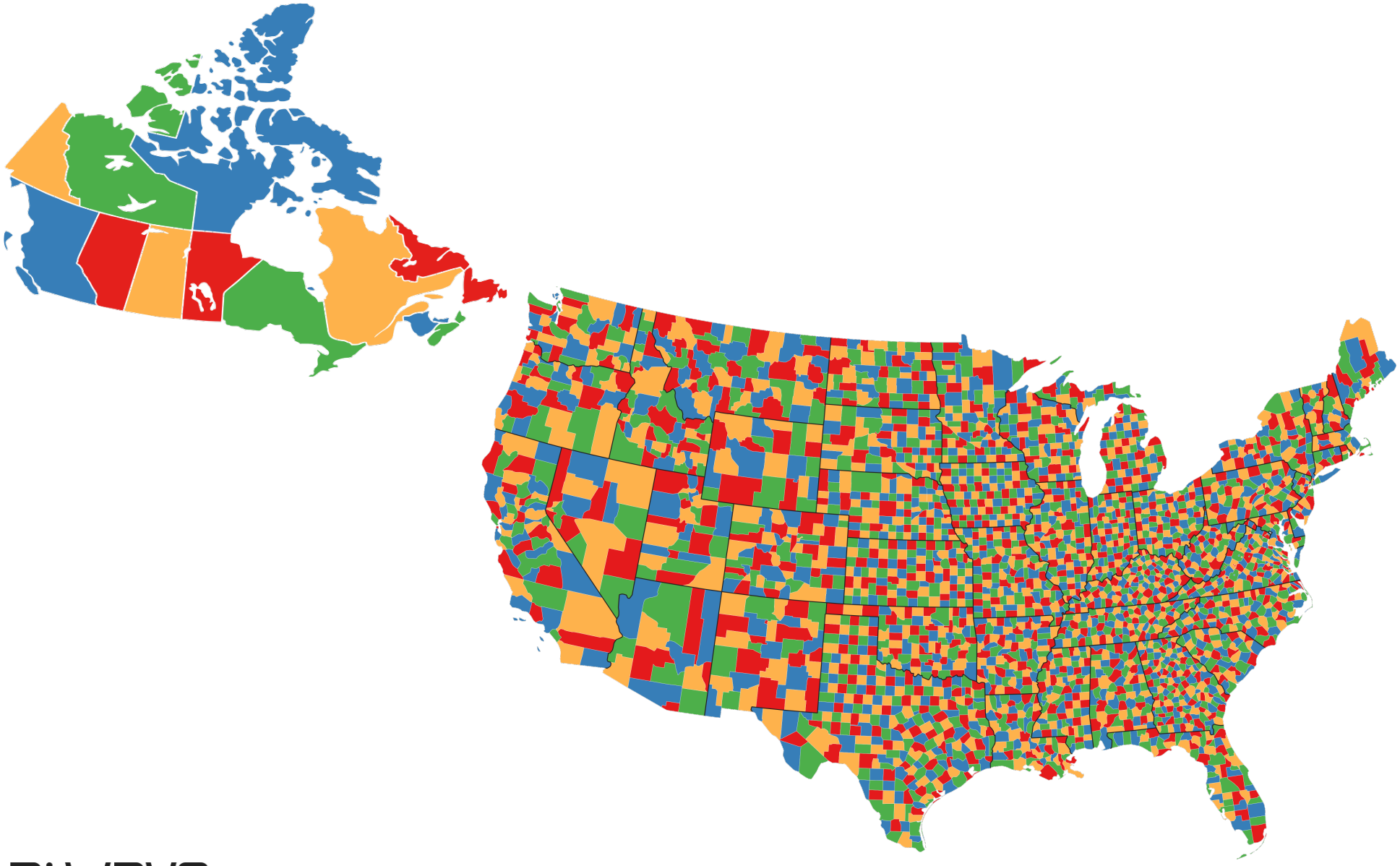


D-Wave The Quantum Computing Company™

D-Wave
The Quantum Computing Company™

The Need For qbsolv

From 12 to 3108!



D:wave
The Quantum Computing Company™

The Need for qbsolv

- Enabling users to solve such larger problems on D-Wave has application development value now
- Performance is strong, so also has application execution value
- Long-term, with strong performance, mapping problems to QUBOs could be a common path supported by tools
  - The way many SAT applications are built today with the DIMACS CNF format

qbsolv Is a Tool

- qbsolv accepts an arbitrary QUBO

  (bigger than, and more connected than hardware) and solves it in chunks on D-Wave system

  - *BQP is equivalent to a QUBO*
- Hybrid Algorithm
  - Identify the significant rows and columns of the larger problem
  - Create a smaller representative QUBO of that subset
  - Execute that smaller QUBO on the D-Wave system
  - Use the answer to guide the larger solver (new starting point, closer to the minimum)
- Ongoing research will yield more complementary power of D-Wave and classical computers

D-Wave
The Quantum Computing Company™

How To Use qbsolv

QUBO Objective function (written 2 ways)

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

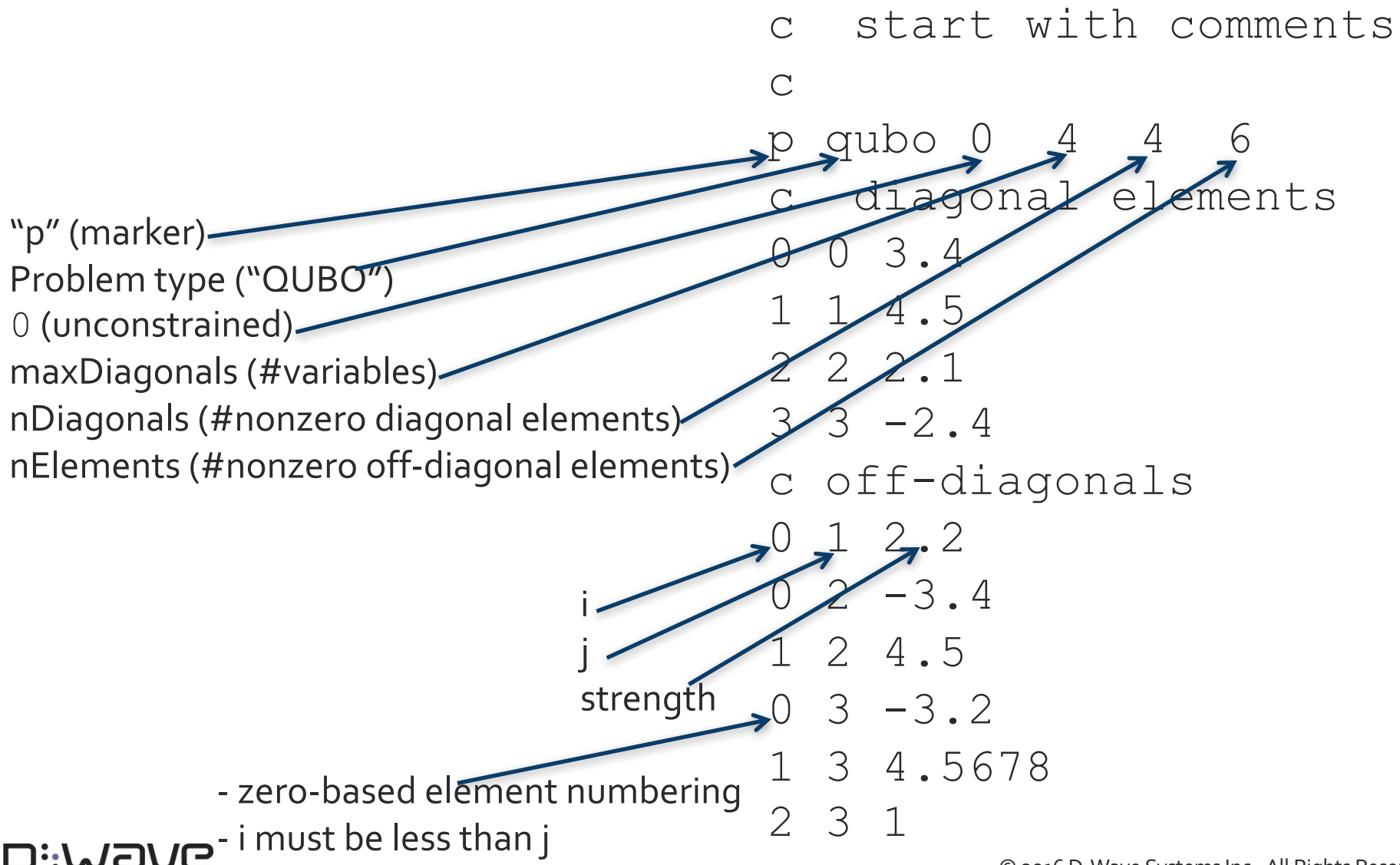
($q_l * q_l = q_l$) and $a_i = b_{ij}$ where $j==i$

$$\mathrm{Obj}(b_{ij}; q_i) = \sum_{i \le j} b_{ij} q_i q_j$$

D:wave
The Quantum Computing Company™

QUBO File Format

- Format is a variant of DIMACS CNF file format



D:wave
The Quantum Computing Company™

qbsolv Command Line

NAME
qbsolv – minimize the objective function represented by a QUBO
SYNOPSIS
qbsolv -i infile [-o outfile] [-m] [-T] [-t] [-n] [-S SubMatrix] [-w] [-h] [-v verbosityLevel] [-V] [-q] [-x]

DESCRIPTION
qbsolv executes a quadratic unconstrained binary optimization (QUBO) problem represented in a file, providing bit-vector result(s) that minimizes (or optionally, maximizes) the value of the objective function represented by the QUBO. The problem is represented in the QUBO(5) file format and notably is not limited to the size or connectivity pattern of the D-Wave system on which it will be executed.

The options are as follows:

-i infile The name of the file in which the input QUBO resides. This is a required option.

-o outfile This optional argument denotes the name of the file to which the output will be written. The default is the standard output.

-m This optional argument denotes to find the maximum instead of the minimum.

-T target This optional argument denotes to stop execution when the target value of the objective function is found.

-t timeout This optional argument stops execution when the elapsed CPU time equals or exceeds timeout value. This is only checked after completion of the main loop. Other halt values such as 'target' and 'repeats' will halt before 'timeout'. The default value is 2592000.0.

-n repeats This optional argument denotes, once a new optimal value is found, to repeat the main loop of the algorithm this number of times with no change in optimal value before stopping. The default value is 12.

-S subproblemSize This optional argument indicates the size of the sub-problems into which the QUBO will be decomposed. A subproblem size of 0 indicates to use the size known by **qbsolv** to be most effective for the D-Wave system targeted for execution. A subproblem size greater than zero indicates to use the given size and execute with **qbsolv**'s internal classical tabu solver rather than using the D-Wave system.

-w If present, in which case **-o** must also be specified, this optional argument will print the QUBO matrix and result in .csv format to **outfile**.

-h If present, this optional argument will print the help or usage message for qbsolv and exit without execution.

-v verbosityLevel This optional argument denotes the verbosity of output. A **verbosityLevel** of 0 (the default) will output the number of bits in the solution, the solution, and the energy of the solution. A **verbosityLevel** of 1 will output the same information for multiple solutions, if found. A **verbosityLevel** of 2 will also output more detailed information at each step of the algorithm.

-V If present, this optional argument will emit the version number of the qbsolv program and exit without execution.

-q If present, this optional argument triggers printing the format of the QUBO file.

qbsolv Output

```
qbsolv -i qubos/bqp50_1.QUBO

50 Number of bits in solution

10111111101001111101001101011111011111110110110

-5176.00000 Energy of solution

4 Number of Partitioned calls

 0.01195 seconds of classic cpu time
```

D-Wave
The Quantum Computing Company™

Performance

Performance on Beasley 2500-variable problems

bqp2500	HMA Time	qbsolv Time	Speed UP	HMA Energy	qbsolv Energy	Energy Improve
1	401.6	2.05	195.9	1,510,509	1,515,944	5,435
2	331.3	18.7	17.7	1,468,778	1,471,392	2,614
3	364.9	3.08	118.5	1,413,172	1,414,192	1,020
4	123.6	1.09	113.4	1,504,978	1,507,701	2,723
5	136.8	33.1	4.1	1,491,309	1,491,816	507
6	377.9	2.5	151.2	1,467,484	1,469,162	1,678
7	178.7	12.8	14.0	1,475,500	1,479,040	3,540
8	102.9	4	25.7	1,483,735	1,484,199	464
9	278.6	41	6.8	1,481,111	1,482,413	1,302
10	373.6	1.7	219.8	1,482,252	1,483,355	1,103
	2669.9	120.02	22.2	14,778,828	14,799,214	20,386

D:Wave
The Quantum Computing Company™

Problem Size, Execution Time

- Has been used for problems up to 12,000 variables on 1Kqb system
- Run-time varies between sub-second to hours
- Guidance
  - Better on optimization problems than satisfiability problems
  - Struggles with some (smaller) hard problems

D:wave
The Quantum Computing Company™

Lab: How to Make an Application

Exercise: "Viewing" a QUBO

- A feature of qbsolv to assist visualizing / debugging a QUBO matrix
- `qbsolv -w` creates a .csv file with the original QUBO and the QUBO multiplied row-/column-wise by the bit-vector result
- Viewing this file can give insight into the dynamics of the numerical solution

Creating the Excel File

- Get `canada.qubo` from qbsolv lab files on class site
- `qbsolv -i canada.QUBO -w -o canada.csv`
- <open `canada.csv` in Excel>
  - <SelectAll>, Format -> Column -> Autofit Selection so viewable
- Row2 and Column B have the resulting bit-vector
- C3:BB54 have the original problem
- C40:AL76 have the contributions from each element in this result
  - Sum(c40:AL76) == the result energy in A79
- The Excel sheet allows for visualizing the input QUBO and the result of the minimized objective function

Input QUBO and Result Bit-Vector

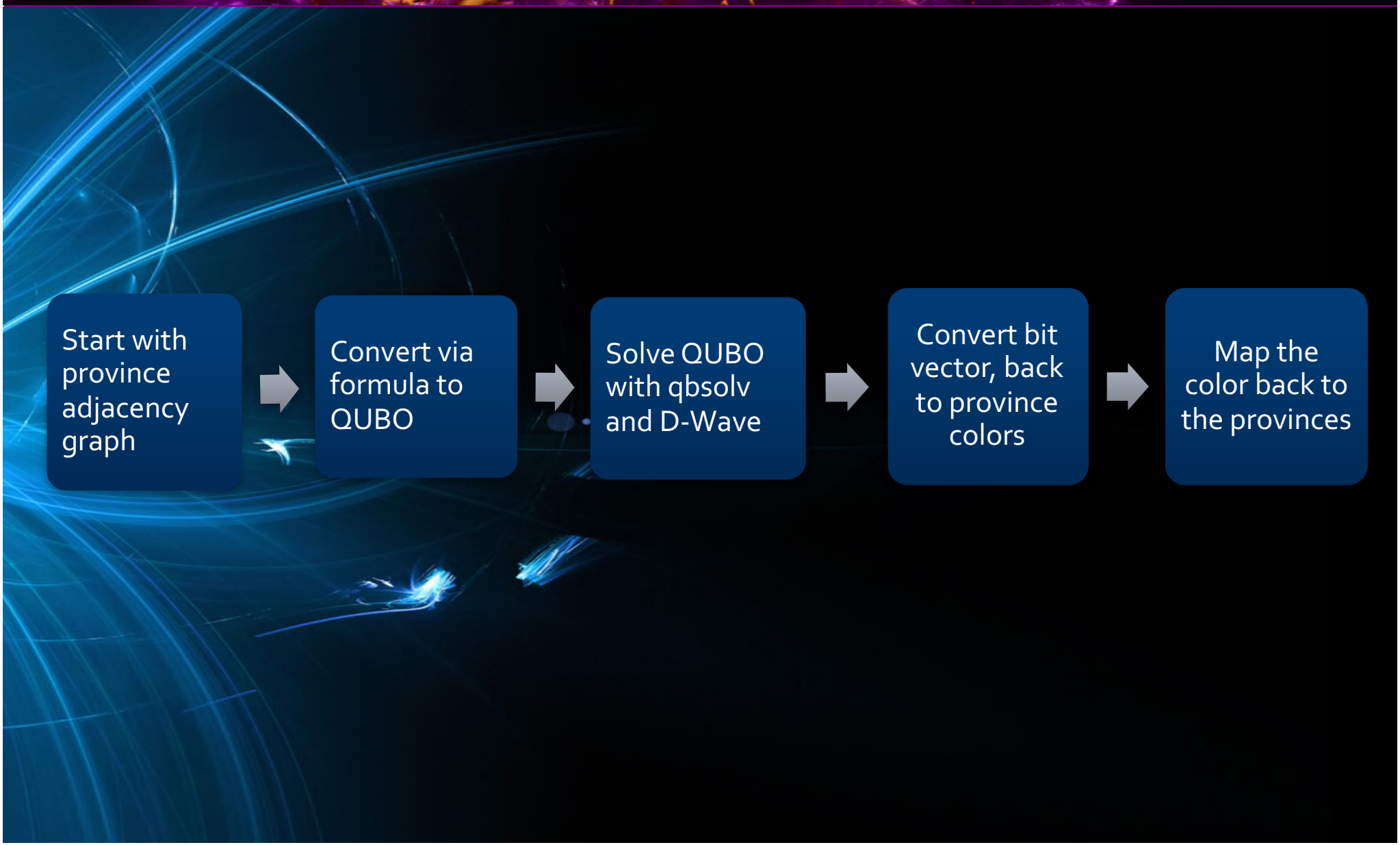
ij		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51
Q		1	0	0	0	0	0	1	0	0	1	0	0	0	0	0	1	0	1	0	0	1	0	0	0	0	0	1	0	0	1	0	0	0	0	0	1	1	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0
0	1	-1	2	2	2	1				1				1																																								
1	0		-1	2	2		1				1				1																																					
2	0			-1	2			1				1				1																																				
3	0				-1				1				1				1																																			
4	0					-1	2	2	2					1				1																																		
5	0						-1	2	2						1				1																																	
6	1							-1	2							1				1																																
7	0								-1								1				1																														
8	0									-1	2	2	2	1								1																												
9	1										-1	2	2		1																																			
10	0											-1	2			1																																		
11	0												-1				1																																	
12	0													-1	2	2	2	1				1																												
13	0														-1	2	2		1				1																											
14	0															-1	2			1				1																										
15	1																-1				1				1																									
16	0																	-1	2	2	2					1																							
17	1																		-1	2	2						1																						
18	0																			-1	2							1																					
19	0																				-1								1																			
20	1																					-1	2	2	2	1								1													
21	0																						-1	2	2		1								1												
22	0																							-1	2			1								1											
23	0																								-1				1								1										
24	0																									-1	2	2	2	1																	
25	0																										-1	2	2		1																
26	1																											-1	2			1															
27	0																												-1				1													
28	0																													-1	2	2	2	1												
29	1																														-1	2	2		1											
30	0																															-1	2			1										
31	0																																-1				1									
32	0																																	-1	2	2	2	1			1			1			1		
33	0																																		-1	2	2		1			1			1			1	
34	0																																			-1	2			1			1			1			1
35	1																																				-1				1			1			1	
36	1																																					-1	2	2	2			1			1	
37	0																																						-1	2	2			1			1	
38	0																																							-1	2			1			1	
39	0																																								-1			1			1
40	1																																									-1	2	2	2	1			1	
41	0																																										-1	2	2		1			1	
42	0																																											-1	2		1			1	
43	0																																												-1			1			1
44	0																																													-1	2	2	2	1			
45	1																																													-1	2	2		1			
46	0																																														-1	2			1			
47	0																																															-1				1	
48	0																																																-1	2	2	2
49	0																																																	-1	2	2
50	1																																																		-1	2
51	0																																																			-1

D-Wave
The Quantum Computing Company™

Evaluated QUBO (Canada unary Encoding): Notice only 13 -1's

Q		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51
		1	0	0	0	0	0	1	0	0	1	0	0	0	0	0	1	0	1	0	0	1	0	0	0	0	0	1	0	0	1	0	0	0	0	0	1	1	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0
0	1	-1																																																			
1	0																																																			
2	0																																																		
3	0																																																		
4	0																																																		
5	0																																																		
6	1							-1																																											
7	0																																																		
8	0																																																		
9	1										-1																																								
10	0																																																		
11	0																																																		
12	0																																																		
13	0																																																		
14	0																																																		
15	1																-1																																		
16	0																																																		
17	1																		-1																															
18	0																																																		
19	0																																																		
20	1																					-1																												
21	0																																																		
22	0																																																		
23	0																																																		
24	0																																																		
25	0																																																		
26	1																											-1																						
27	0																																																		
28	0																																																		
29	1																														-1																				
30	0																																																		
31	0																																																		
32	0																																																		
33	0																																																		
34	0																																																		
35	1																																				-1														
36	1																																					-1													
37	0																																																		
38	0																																																		
39	0																																																		
40	1																																																		
41	0																																																		
42	0																																																		
43	0																																																		
44	0																																																		
45	1																																																		
46	0																																																		
47	0																																																		
48	0																																																		
49	0																																																		
50	1																																																		-1
51	0																																																		

Application example flow: map coloring



Application flow to canada.qubo

canada.adj

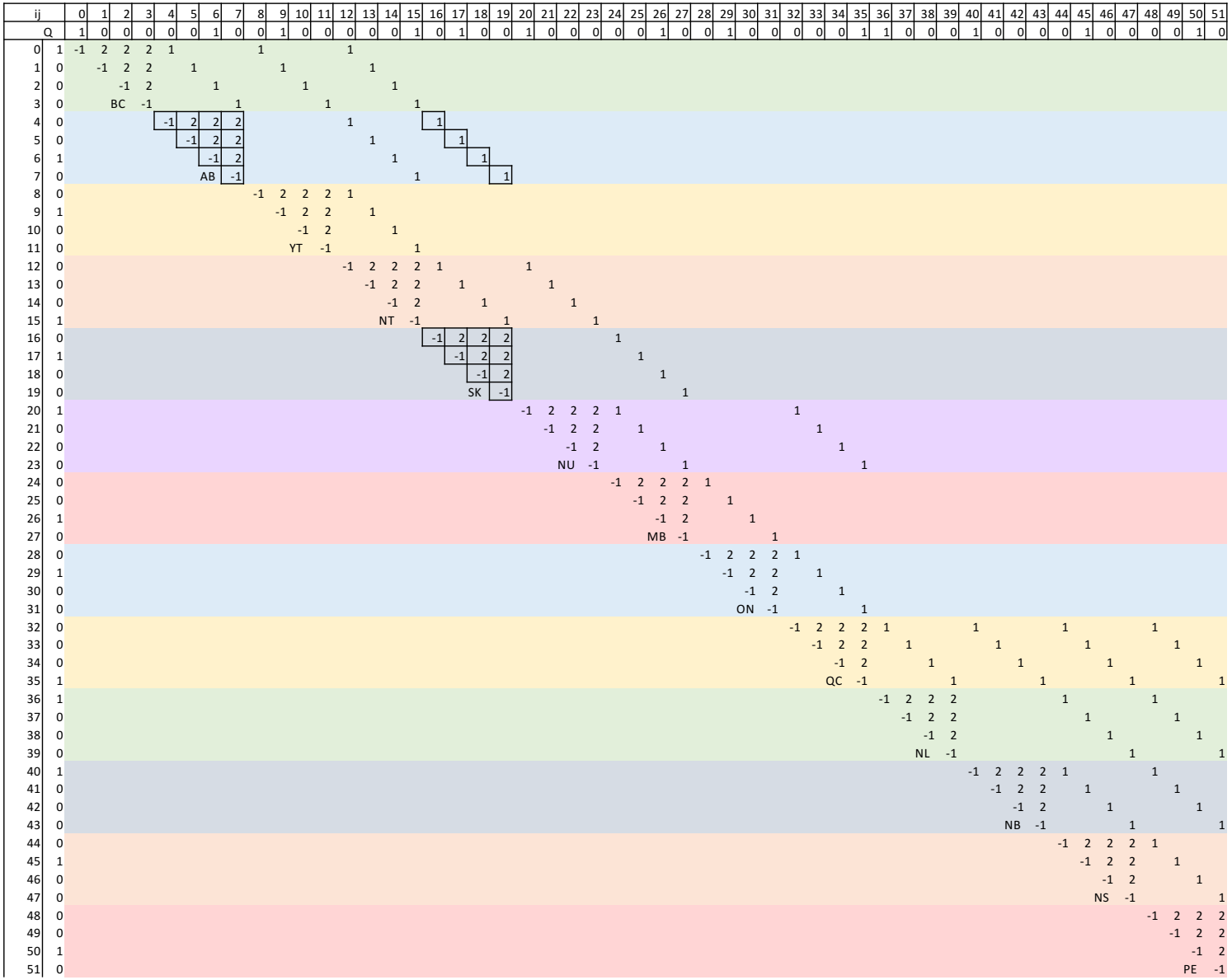
BC,AB,NT,YT
AB,BC,NT,SK
YT,BC,NT
NT,YT,BC,AB,NU,SK
SK,AB,NT,MB
NU,NT,MB,QC
MB,ON,NU,SK
ON,MB,QC,NU
QC,ON,NL,NB,NS,PE
NL,QC,PE,NS
NB,QC,NS,PE
NS,QC,NB,PE,NL
PE,NB,NS,NL,QC

canada.qubo

```
c
c  this QUBO was created by adj2QUBO.py for 4 color uninary encoding
c
p  QUBO  0  52 52 166
c BC
 0 0 -1
 1 1 -1
 2 2 -1
 3 3 -1
c AB
 4 4 -1
.......
c
c  Couplers
c
c BC   3 neighbors  12 external couplers
 0 1 2
 0 2 2
 0 3 2
 1 2 2
1 13 1
 2 14 1
 3 15 1
c BC linked to   YT
 0 8 1
 1 9 1
 2 10 1
 3 11 1
.....

 49 51 2
 50 51 2
```

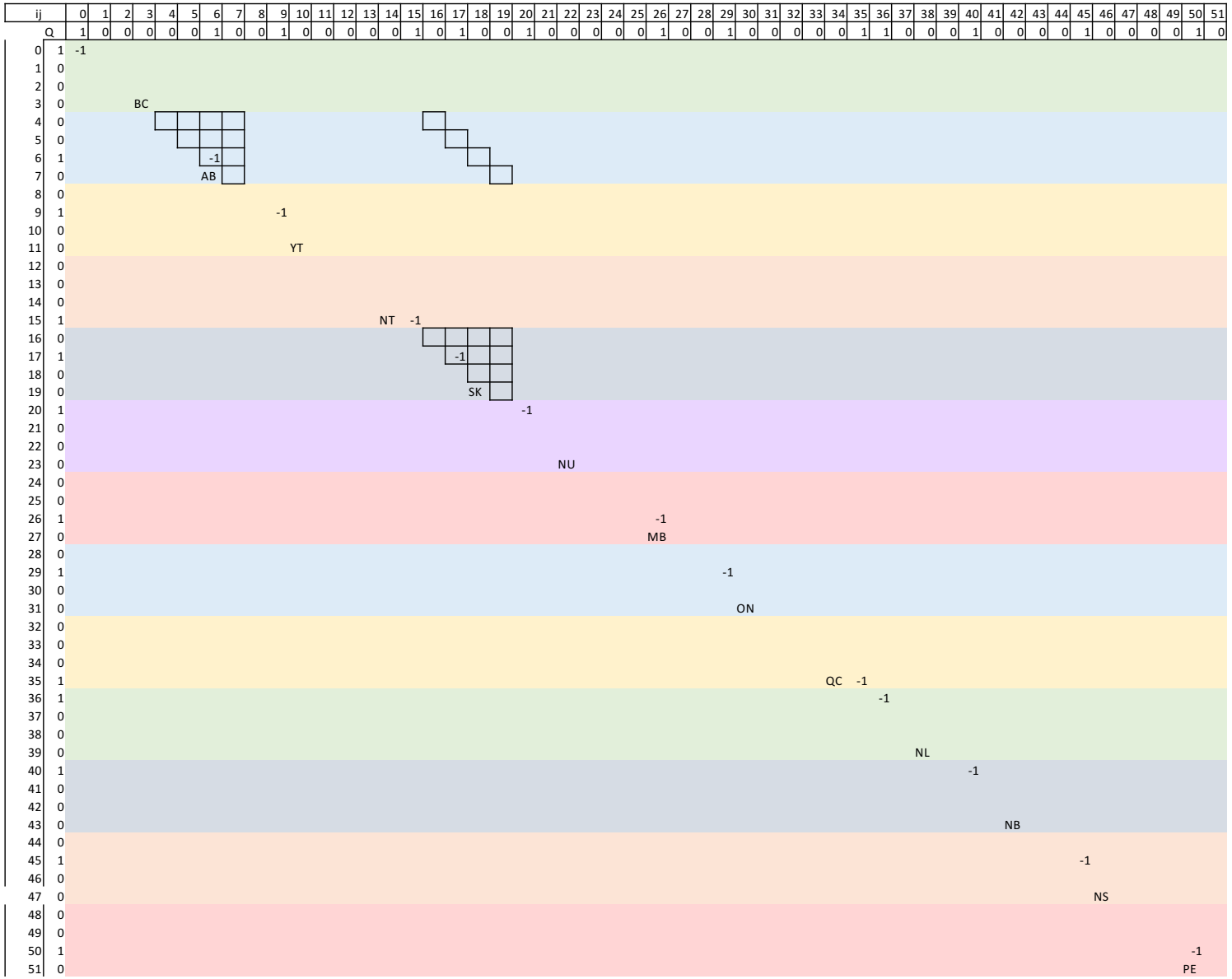
Canada QUBO from adj2qubo.py



BC,AB,NT,YT
AB,BC,NT,SK
YT,BC,NT
NT,YT,BC,AB,NU,SK
SK,AB,NT,MB
NU,NT,MB,QC
MB,ON,NU,SK
ON,MB,QC,NU
QC,ON,NL,NB,NS,PE
NL,QC,PE,NS
NB,QC,NS,PE
NS,QC,NB,PE,NL
PE,NB,NS,NL,QC

D-Wave
The Quantum Computing Company™

Result from qbsolv



BC,AB,NT,YT

AB,BC,NT,SK

YT,BC,NT

NT,YT,BC,AB,NU,SK

SK,AB,NT,MB

NU,NT,MB,QC

MB,ON,NU,SK

ON,MB,QC,NU

QC,ON,NL,NB,NS,PE

NL,QC,PE,NS

NB,QC,NS,PE

NS,QC,NB,PE,NL

PE,NB,NS,NL,QC

D-Wave
The Quantum Computing Company™

Whole application: Canada map coloring

demoCanada.sh:

```
#!/bin/bash
#  Make a QUBO from adj file

python adj2QUBO.py -i canada.adj -o canada.QUBO

# clean up old outputs if this script ran and aborted before

if [ -e canada.qbout ] ; then
      rm canada.q*.svg  canada.qbout
fi
if [ -e /c/Users/trainee/Desktop/CanadaMaps ] ; then
                  rm -r /c/Users/trainee/Desktop/CanadaMaps
fi

dw get connection
dw get solver

qbsolv -i canada.QUBO -o canada.qbout -v1 -n 8 -So
cat canada.qbout
echo Solved -- Ploting
./plotmap.sh
echo Plots complete,, copied to desktop to be displayed in Explorer

mkdir /c/Users/trainee/Desktop/CanadaMaps
cp canada.q*.svg /c/Users/trainee/Desktop/CanadaMaps
rm canada.q*.svg
```

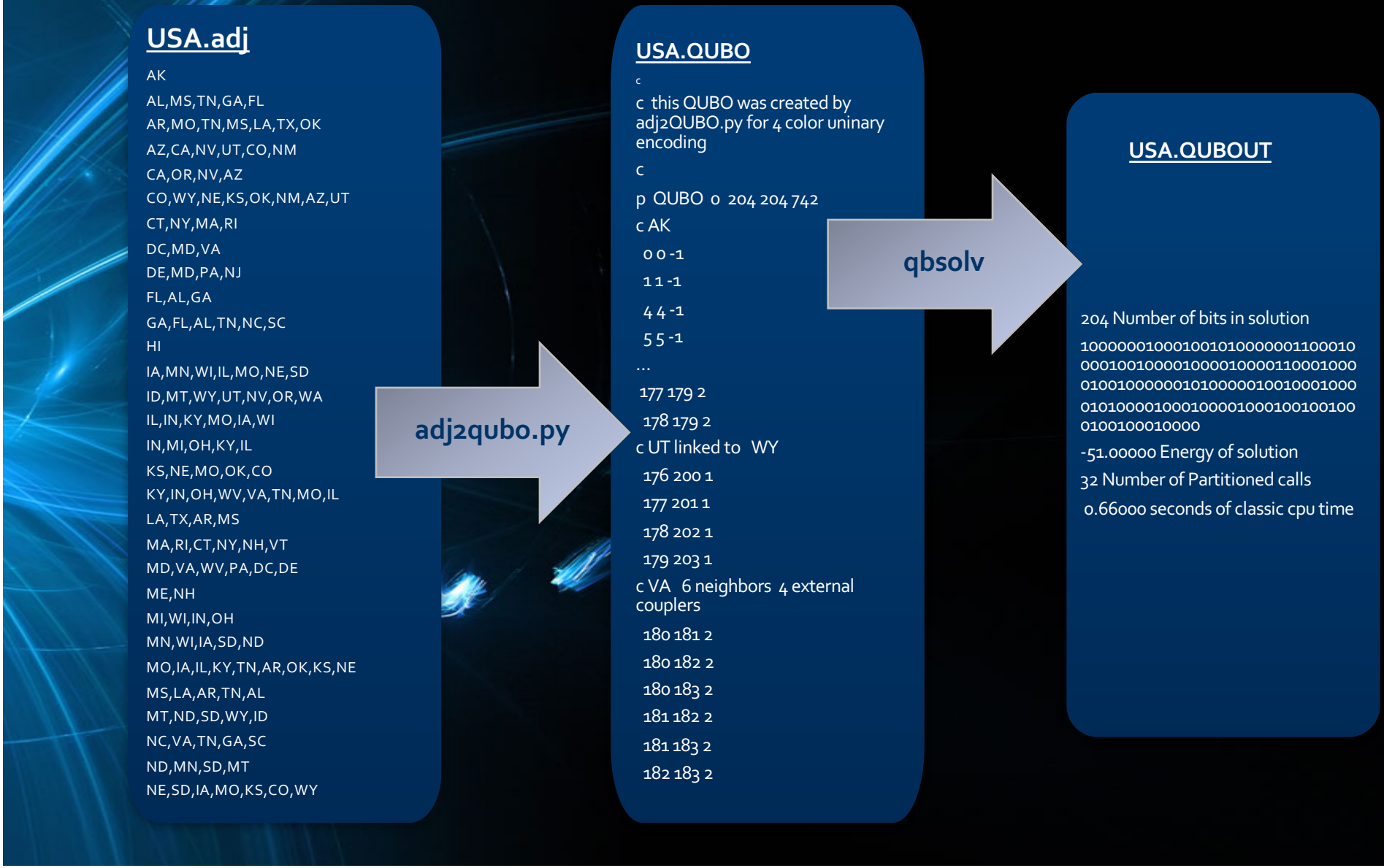
Files

canada.svg
color_canada.py
plotmap.sh
canada.adj
adj2QUBO.py
demoCanda.sh

Canada .svg file (converted to use Powerpoint)

D:wave
The Quantum Computing Company™

Application flow to USA.qubo



The same application flow to USA States Color

usa.qbout

204 Number of bits in solution

1000000100010010100000011000100001001
0000100001000011000100001001000000101
0000010010001000010100001000100000100
0100100100010010001000 0

-51.00000 Energy of solution

32 Number of Partitioned calls

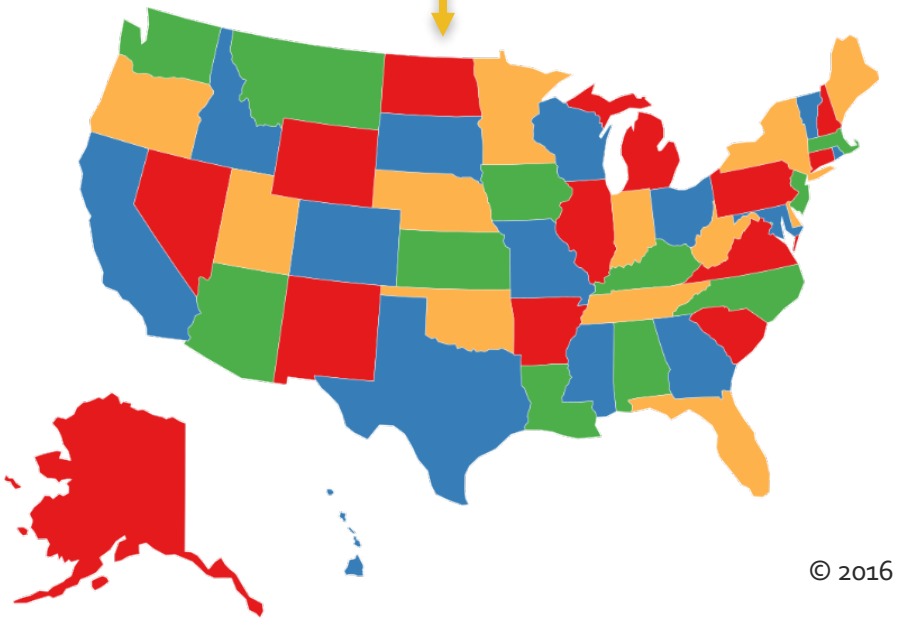
0.66000 seconds of classic cpu time

usa.adj

AK
AL,MS,TN,GA,FL
AR,MO,TN,MS,LA,TX,OK
AZ,CA,NV,UT,CO,NM
CA,OR,NV,AZ
CO,WY,NE,KS,OK,NM,AZ,UT
CT,NY,MA,RI
DC,MD,VA
DE,MD,PA,NJ
FL,AL,GA

usalow.svg

color_states.py



D:Wave
The Quantum Computing Company™

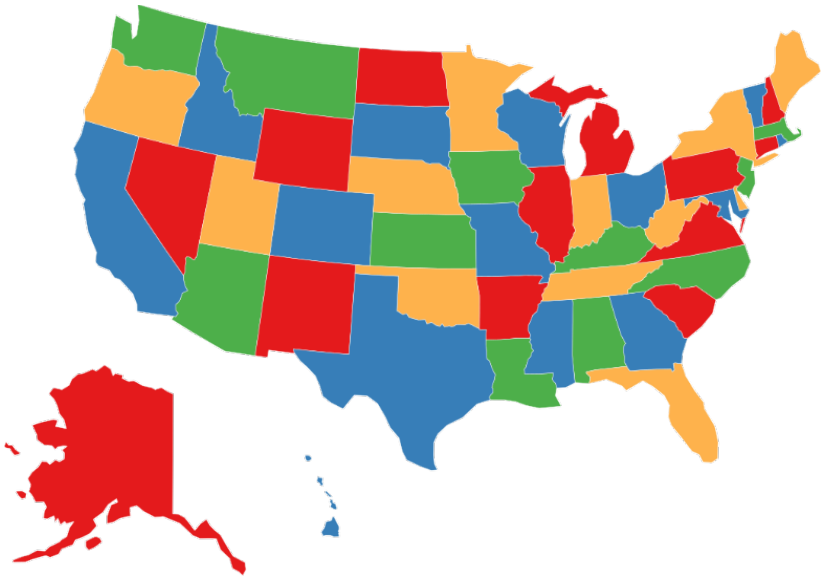
Exercise: Whole application USA

To run:
./demoStates.sh

Files	
adj2qubo.py	Converts .adj file to QUBO
color_states.py	Reads qbout segments and colors the map
demoStates.sh	Do everything script
plotmap.sh	Segments multiple solutions calls color_states
usa.adj	Text file , which states touch
usaLow.svg	svg file, uncolored, input to color_states

demoStates.sh (pseudo)

```
#!/bin/bash
./adj2QUBO.py -i usa.adj -o usa.QUBO
qbsolv -i usa.QUBO -o usa.qbout -v2 -So
./plotmap.sh
open by hand using explorer usa.q*.svg
```



D-Wave
The Quantum Computing Company™

Exercise: Add Constraints

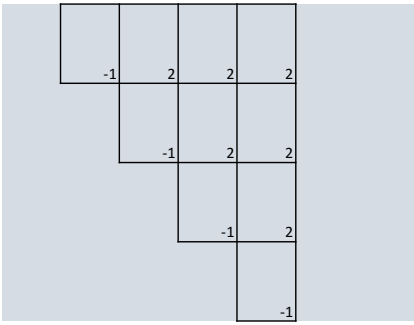
- Overconstrain the adj file
- Add constraints
  - Ex. TN not the same color as CA
  - Ex. HI, AK and TX as adjacent

Exercise: Lock a State into one color

- Modify the intermediate qubo file to limit a state to be one color (or Florida not blue?)
- Remove the row and column of 3 color choices for a state
  - Eliminate the node (diag element) for each color
  - Eliminate all couplers (off diag elements) with coupled to that diagonal
  - Reduce the nodes and couplers in the qubo program line
- Run the last parts of the script

Example application: Sudoku

- Connected qubo'let is like map coloring
- Unary encoding
  - A node bit for each possible value (1,2,3…9)
  - Couplers in the qubo'let force only one bit high
  - Couplers between qubo'lets to force unique qubo'let values



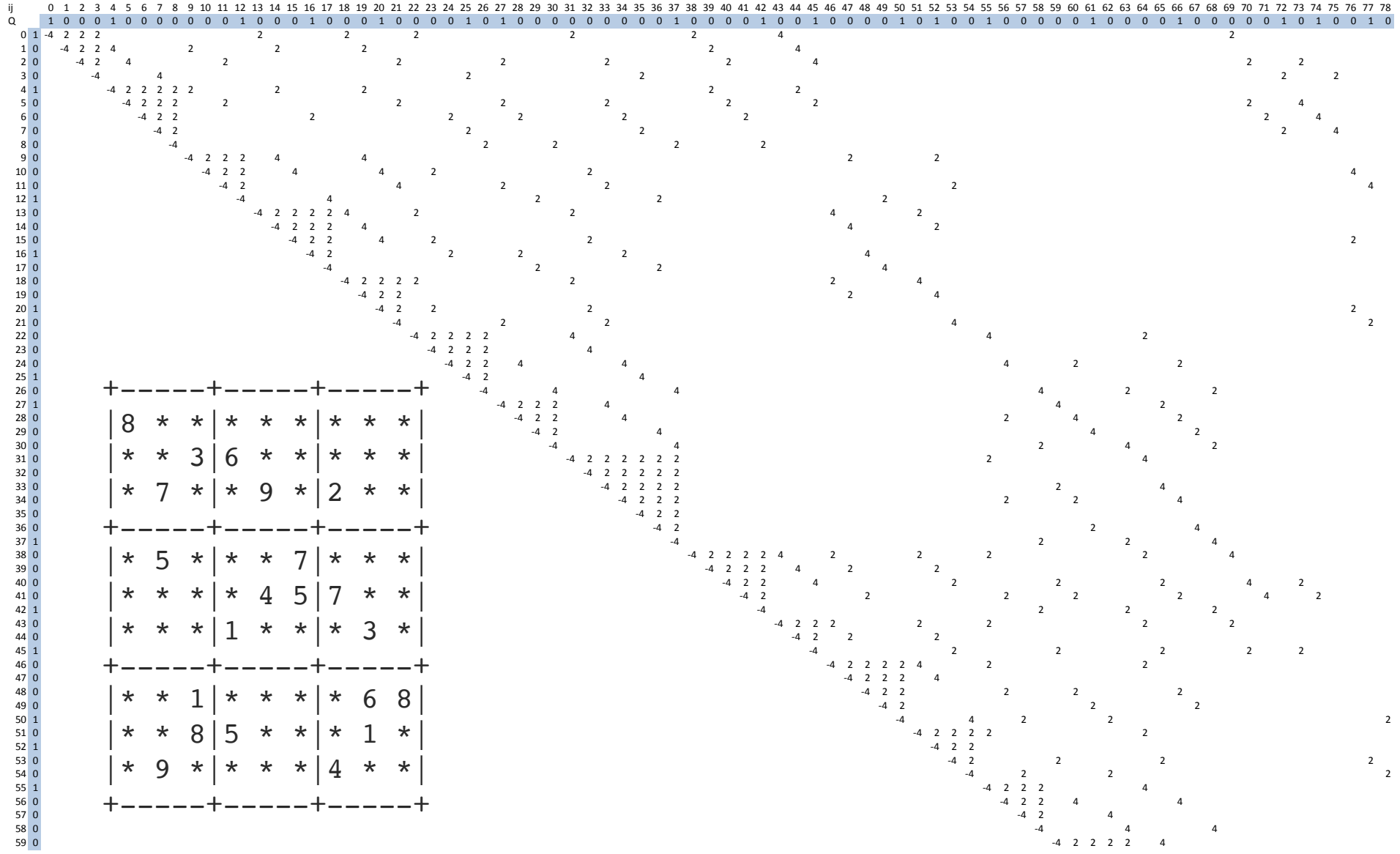
Example application: Sudoku

- Create QUBO for blank grid (10935 lines output)
- Identify fixed variables (517 lines output)
- Compute reduced QUBO (1407 lines output)
- Identify free variables ; Enumerate variables (1407 lines)
- Generate qbsolv input (1407 lines)
- Execute qbsolv (212 bits output)
- Obtain values of free variables from qbsolv output
- Combine fixed and free variables ; Generate output

Example application: Sudoku

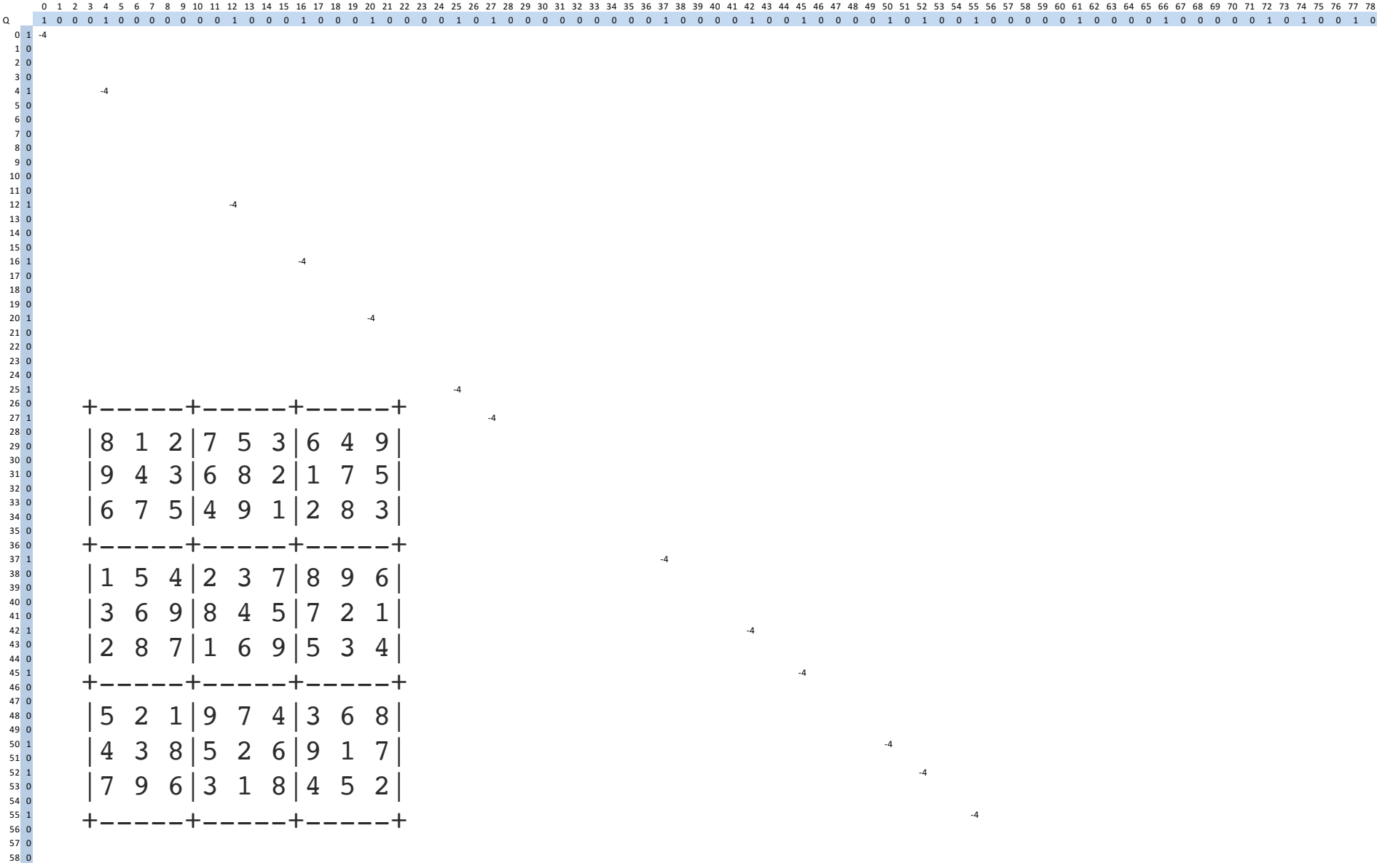
```
./driver.bash SUDOKU.1.txt
**************** Create QUBO for blank grid ****************
**************** Identify fixed variables ****************
**************** Compute reduced QUBO ****************
**************** Identify free variables ; Enumerate variables ****************
**************** Generate qbsolv input ****************
**************** Execute qbsolv ****************
**************** Obtain values of free variables from qbsolv output ****************
**************** Combine fixed and free variables ; Generate output ****************
+-----+-----+-----+
|* * *|1 * *|4 * *|
|* 1 *|* * *|* * 6|
|4 8 *|9 * *|2 * *|
+-----+-----+-----+
|* 5 *|* * 9|8 6 *|
|* * *|3 * *|* * 1|
|* 4 *|* * 2|5 7 *|
+-----+-----+-----+
|9 3 *|6 * *|1 * *|
|* 7 *|* * *|* * 5|
|* * *|2 * *|7 * *|
+-----+-----+-----+
+-----+-----+-----+
|3 9 6|1 2 7|4 5 8|
|7 1 2|5 8 4|3 9 6|
|4 8 5|9 6 3|2 1 7|
+-----+-----+-----+
|1 5 3|7 4 9|8 6 2|
|8 2 7|3 5 6|9 4 1|
|6 4 9|8 1 2|5 7 3|
+-----+-----+-----+
|9 3 8|6 7 5|1 2 4|
|2 7 1|4 9 8|6 3 5|
|5 6 4|2 3 1|7 8 9|
+-----+-----+-----+
```

Example: small cut of Sudoku



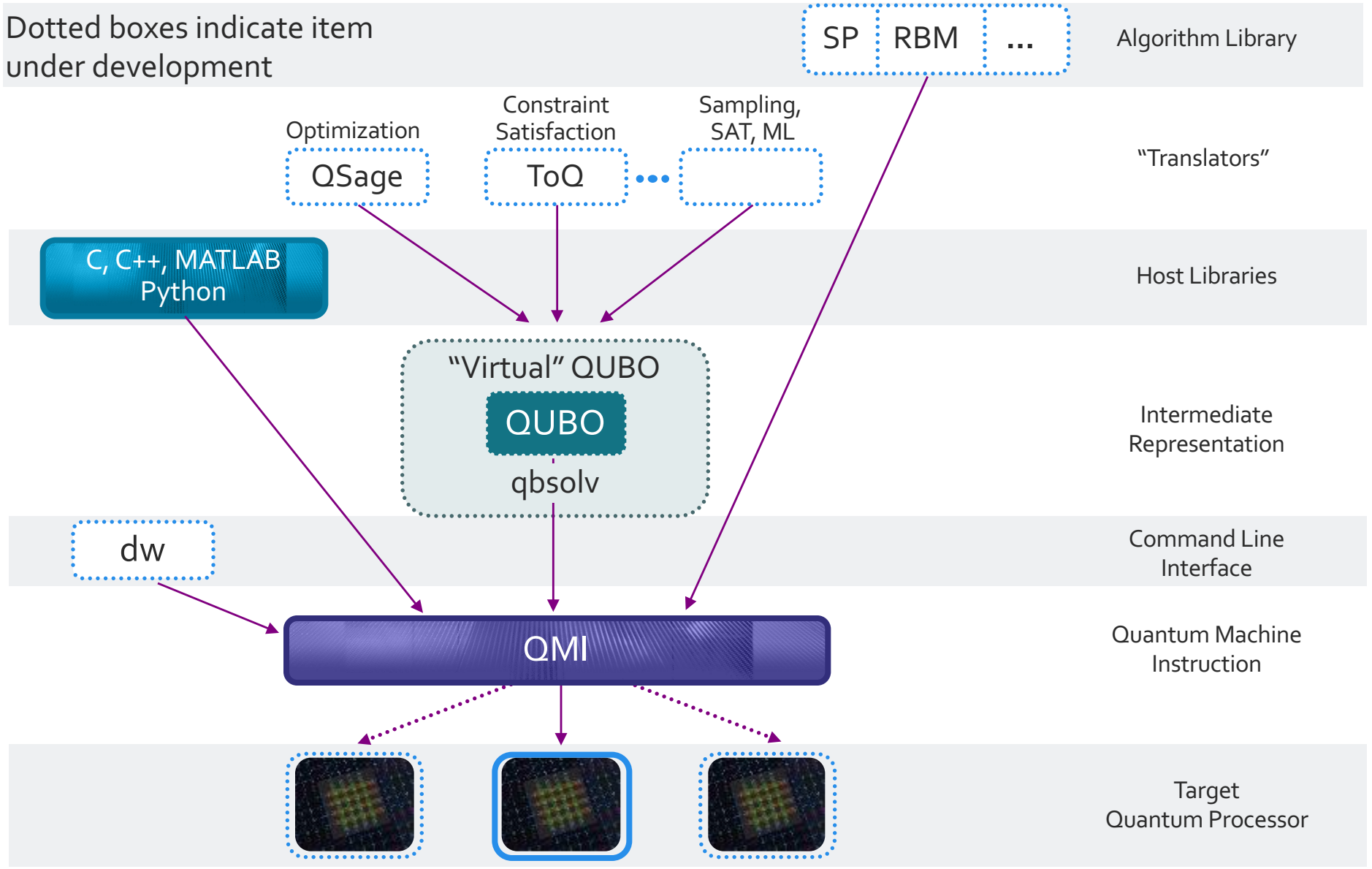
```
+-------+-------+-------+
| 8 * * | * * * | * * * |
| * * 3 | 6 * * | * * * |
| * 7 * | * 9 * | 2 * * |
+-------+-------+-------+
| * 5 * | * * 7 | * * * |
| * * * | * 4 5 | 7 * * |
| * * * | 1 * * | * 3 * |
+-------+-------+-------+
| * * 1 | * * * | * 6 8 |
| * * 8 | 5 * * | * 1 * |
| * 9 * | * * * | 4 * * |
+-------+-------+-------+
```

Example: SUDOKU
Solved or contributing elements in matrix



```
+-----+-----+-----+
|8 1 2|7 5 3|6 4 9|
|9 4 3|6 8 2|1 7 5|
|6 7 5|4 9 1|2 8 3|
+-----+-----+-----+
|1 5 4|2 3 7|8 9 6|
|3 6 9|8 4 5|7 2 1|
|2 8 7|1 6 9|5 3 4|
+-----+-----+-----+
|5 2 1|9 7 4|3 6 8|
|4 3 8|5 2 6|9 1 7|
|7 9 6|3 1 8|4 5 2|
+-----+-----+-----+
```

D-Wave Software Environment



D-Wave The Quantum Computing Company™

Summary

- qbsolv is a tool that solves QUBOs bigger or more densely connected than hardware
- Converting problems into unconstrained QUBOs is a good intermediate problem representation on the way to the D-Wave system
- Uses an algorithm that was recently the best available
- Other algorithms will prove better
- On benchmarks run, delivers best answers and fastest execution time
- Seeking users with big QUBOs for deeper understanding of algorithmic effectiveness

Next Steps for qbsolv

- Existence of qbsolv enables application development today with QUBOs
- To solve the NP-hard part of application, user focuses on conversion to QUBO
- We expect that this proof of a partitioning concept will provoke better solvers than qbsolv

  With app developers insulated by a standard input and output format