D:Wave
The Quantum Computing Company™

Map Coloring

Algorithms into Tools: ToQ & qbsolv

LANL / D-Wave Quantum Programming

June 9, 2016

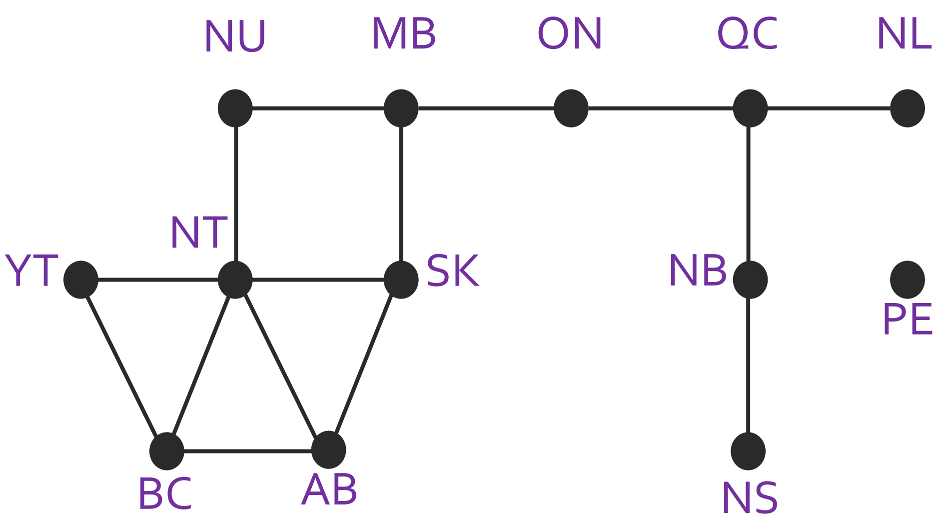
D-Wave Systems Inc.

Denny Dahl

Example: 4-coloring Canada’s provinces

D:Wave
The Quantum Computing Company™

Canada represented as a graph



AB Alberta
BC British Columbia
MB Manitoba
NB New Brunswick
NL Newfoundland and Labrador
NS Nova Scotia
NT Northwest Territories
NU Nunavut
ON Ontario
PE Prince Edward Island
QC Quebec
SK Saskatchewan
YT Yukon

Needle & Haystack : Coloring Canada

(Not to scale)

# of colors	Needle	Haystack	N/H
3	1728	$3^{13} = 1.6\text{x}10^6$	0.0011
4	653184	$4^{13} = 6.7\text{x}10^7$	0.0097

Encode colors and provinces via qubits

Pick unary encoding for simplicity:

- 13 regions
- 4 colors (Blue, Green, Red, Yellow)
- Create 13x4 = 52 logical qubits

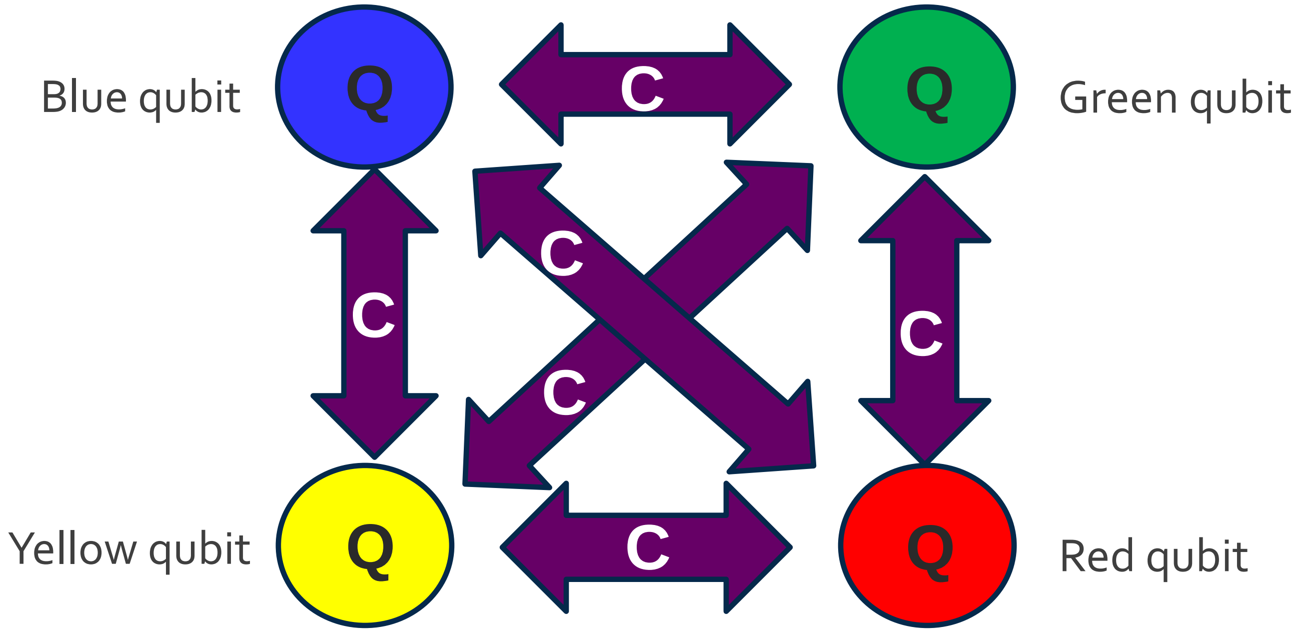
Build QMI with these four tasks:

1. Turn on exactly one of the four color qubits for each region
2. Map logical color qubits for a region to physical qubits of a unit cell
3. Use intercell couplers to enforce neighbor constraints
4. Clone regions as necessary so that Canada can embed into a planar grid

Each task contributes a portion of the final QMI

Add individual contributions to get the total QMI

Task 1: turn on one of four color qubits



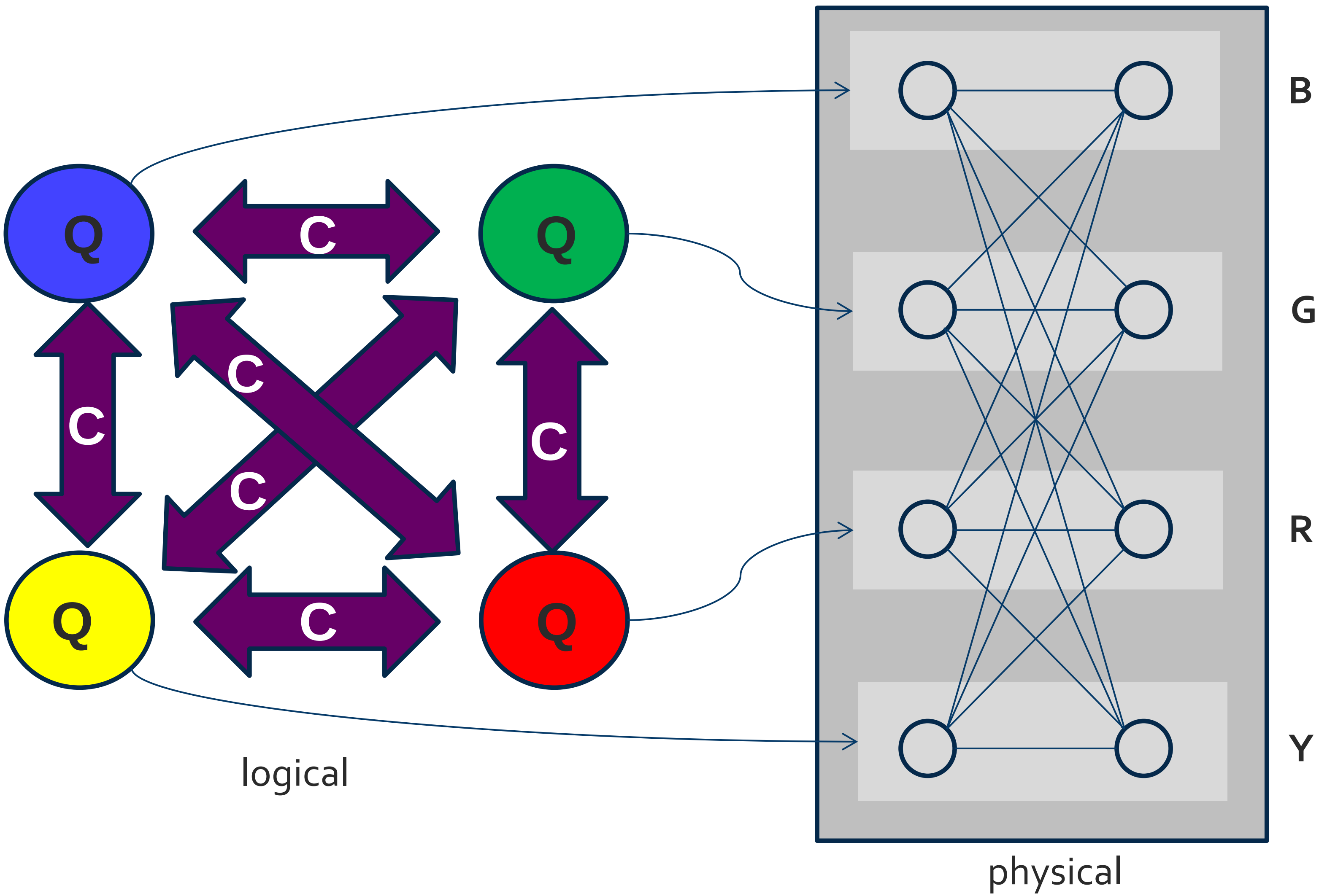
Objective : $O(q_b, q_g, q_r, q_y) = (q_b + q_g + q_r + q_y - 1)^2 \cong$

$$-1(q_b + q_g + q_r + q_y)$$

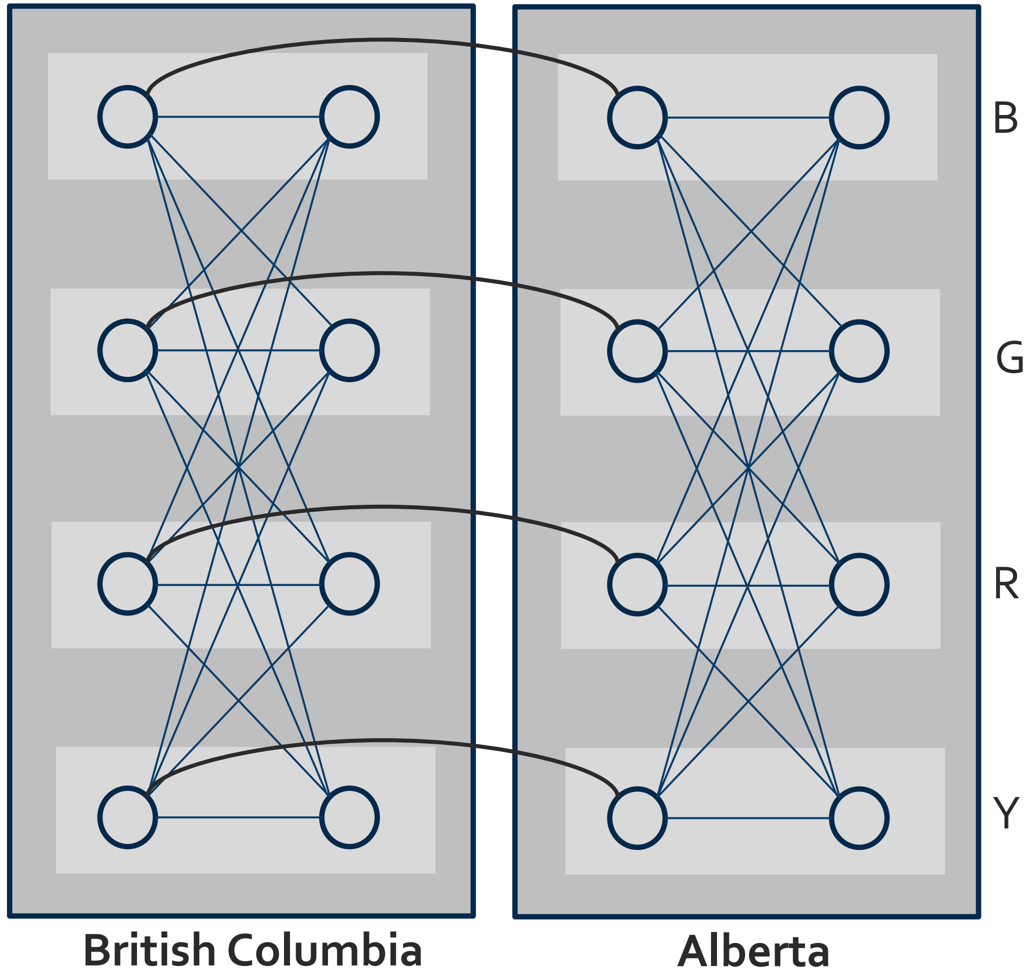
$$+2(q_b q_g + q_b q_r + q_b q_y + q_g q_r + q_g q_y + q_r q_y)$$

D-Wave The Quantum Computing Company™

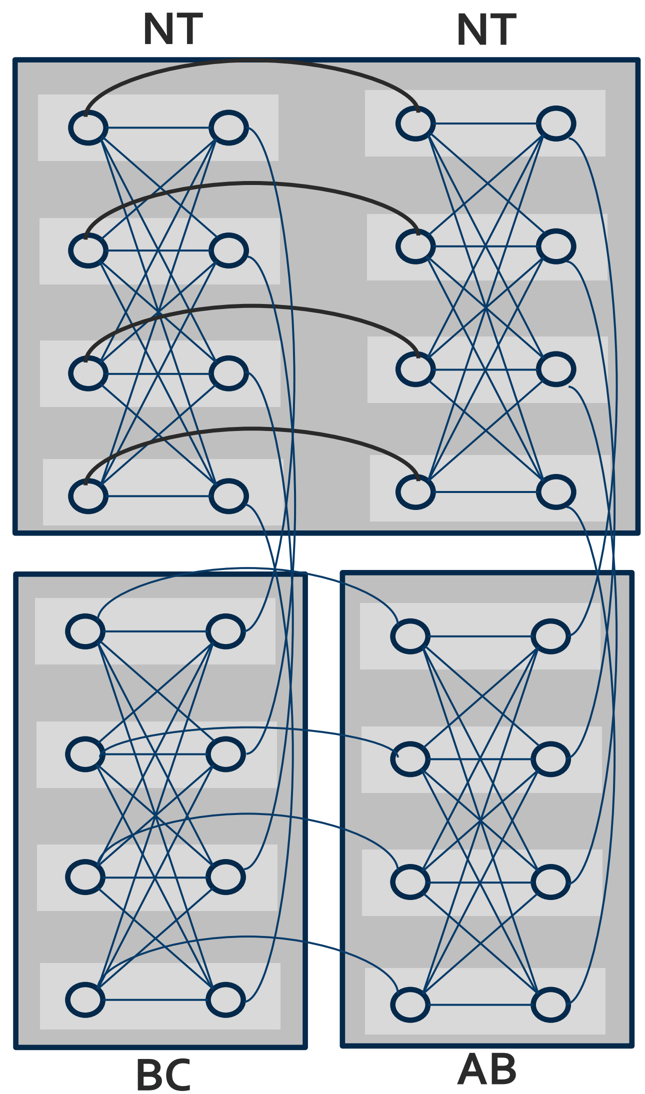
Task 2: embed logical to physical qubits



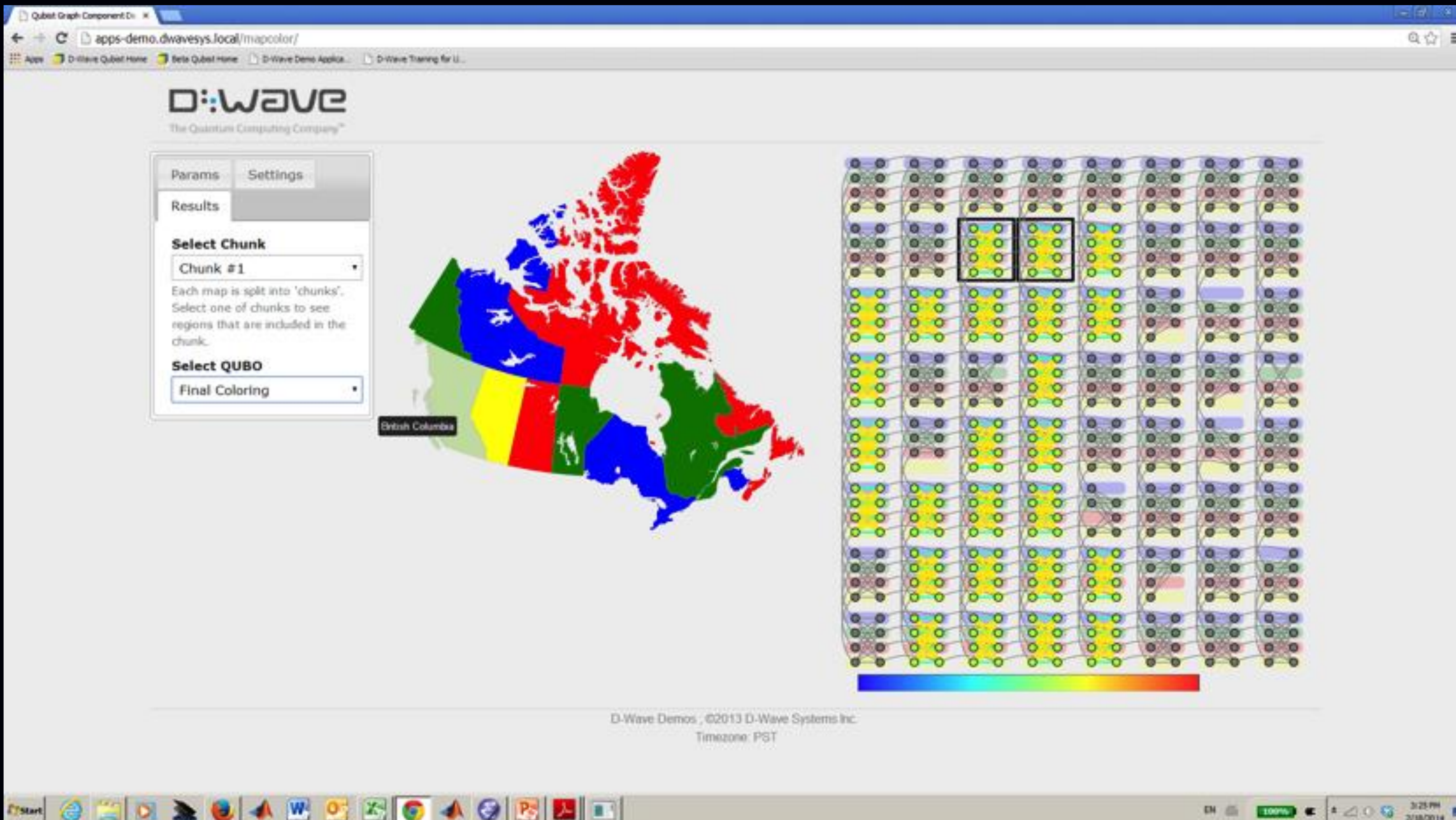
Task 3: Intercell couplers constrain neighbors



Task 4: Clone regions for planar embedding



Colors encoded in unit cells



The Quantum Computing Company™

Implementations of map coloring

C

```c
void setup_unit_cell(int row, int col)
{
  int i, j;

  if (cell_region[row][col] == UNDEF)
    return;

  /* STEP 1: turn on one of C qubits */

  for (i=0; i<C; ++i)
    {
      weight[DW_QUBIT(row,col,'L',i)] += -0.5;
      weight[DW_QUBIT(row,col,'R',i)] += -0.5;
    }

  for (i=0; i<C; ++i)
    for (j=0; j<C; ++j)
      if (i != j)
        strength[DW_INTRACELL_COUPLER(row,col,i,j)] += 1;

  /* STEP 2: create chains */

  for (i=0; i<C; ++i)
    {
      weight[DW_QUBIT(row,col,'L',i)] += 1;
      weight[DW_QUBIT(row,col,'R',i)] += 1;
      strength[DW_INTRACELL_COUPLER(row,col,i,i)] += -2;
    }
```

Snippet (28 of 596 LOC)

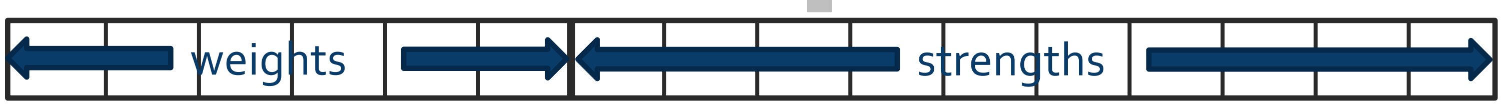
ToQ

```
mbool: 1, 4, @AB
mbool: 1, 4, @BC
mbool: 1, 4, @MB
mbool: 1, 4, @NB
mbool: 1, 4, @NL
mbool: 1, 4, @NS
mbool: 1, 4, @NT
mbool: 1, 4, @NU
mbool: 1, 4, @ON
mbool: 1, 4, @QC
mbool: 1, 4, @SK
mbool: 1, 4, @YT

assert: @AB != @BC
assert: @AB != @NT
assert: @AB != @SK
assert: @BC != @NT
assert: @BC != @YT
assert: @MB != @NU
assert: @MB != @ON
assert: @MB != @SK
assert: @NB != @NS
assert: @NB != @QC
assert: @NL != @QC
assert: @NT != @NU
assert: @NT != @SK
assert: @NT != @YT
assert: @ON != @QC
```

entire program

QMI :



D:WAVE The Quantum Computing Company™

ToQ *(pronounced "too-kew")*

- High Level Language interpreter of optimization problem assertions
- Works as a standalone program, or as a HLL-callable library routine from a user's program (C, C++, Fortran, Python)
- Permits users to "speak" in the language of their problem domain
- Run-time control of assertions via variables from user's program
- Provides exhaustive error management
- Communicates directly with the D-Wave System, and sends results back to the user
- Includes internal documentation and optional reports back to the user

D-Wave The Quantum Computing Company™

More difficult: Coloring the map of the US

# of colors	Needle	Haystack	N/H
3	0	$3^{49} = 2.4\text{x}10^{23}$	0
4	25623183458304	$4^{49} = 3.2\text{x}10^{29}$	$8\text{x}10^{-17}$

Suppose that:

- a classical computer can execute 4B instructions per second
- each instruction examines a random map coloring

It would take around one month to find a valid coloring

If you're attempting to find the minimum number of colors required by this map, you might stop before the month was up and draw an incorrect conclusion!

D:WAVE
The Quantum Computing Company™

 |

Scaling up…

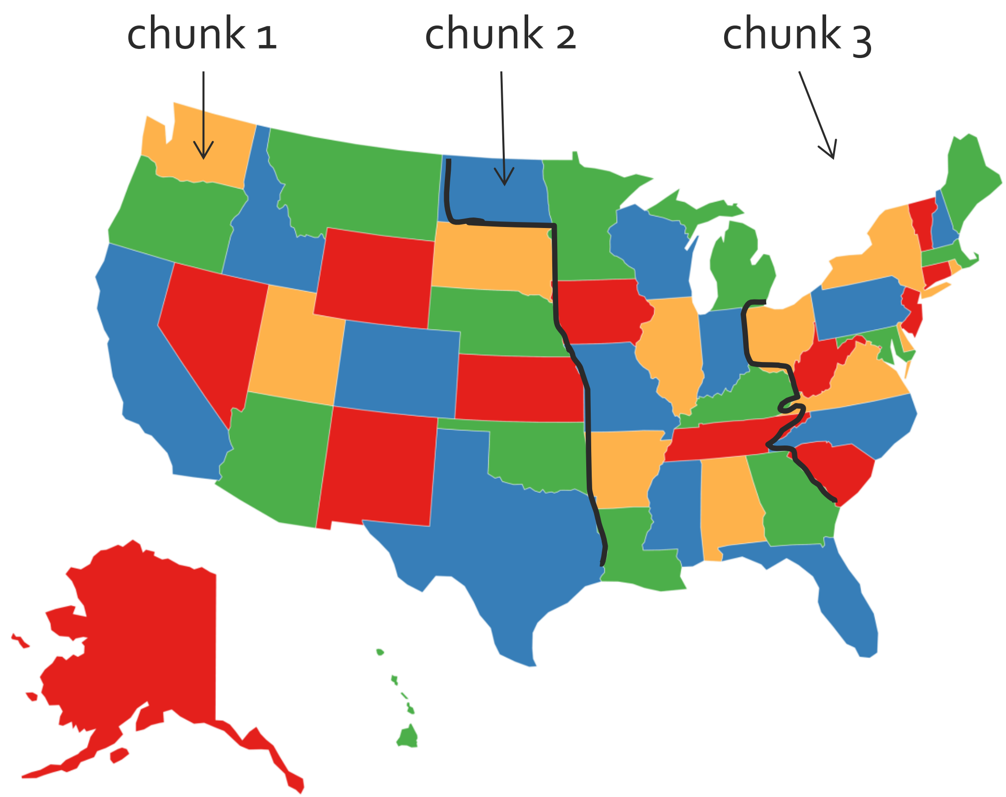
- We cannot fit all the states into unit cells of the chip…
- …so we adopt a divide-and-conquer strategy

Divide the US map into chunks.

Process the first chunk and get valid colorings for the first chunk of states.

Use these colorings to *bias* the second chunk.

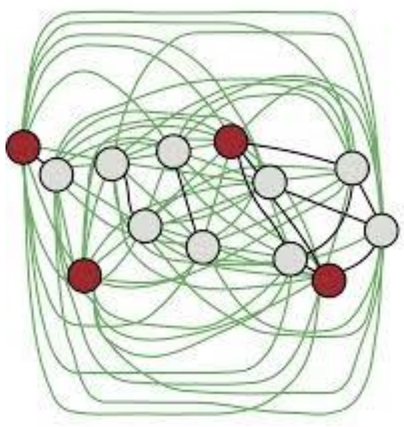
Repeat.



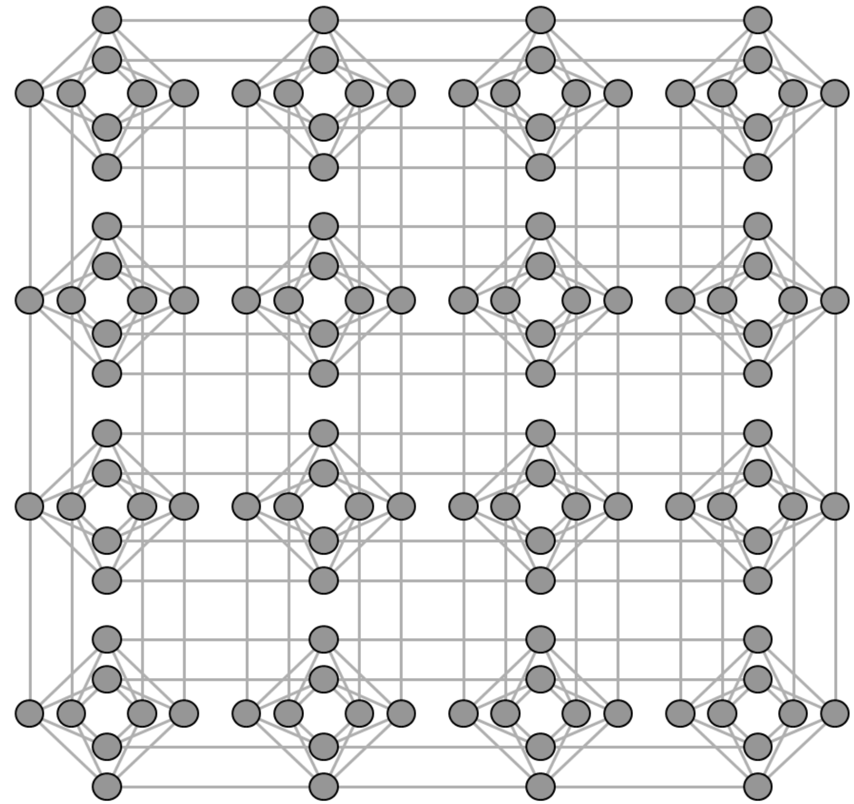
D-Wave
The Quantum Computing Company™

 |

Embedding: using the SAPI heuristic



logical



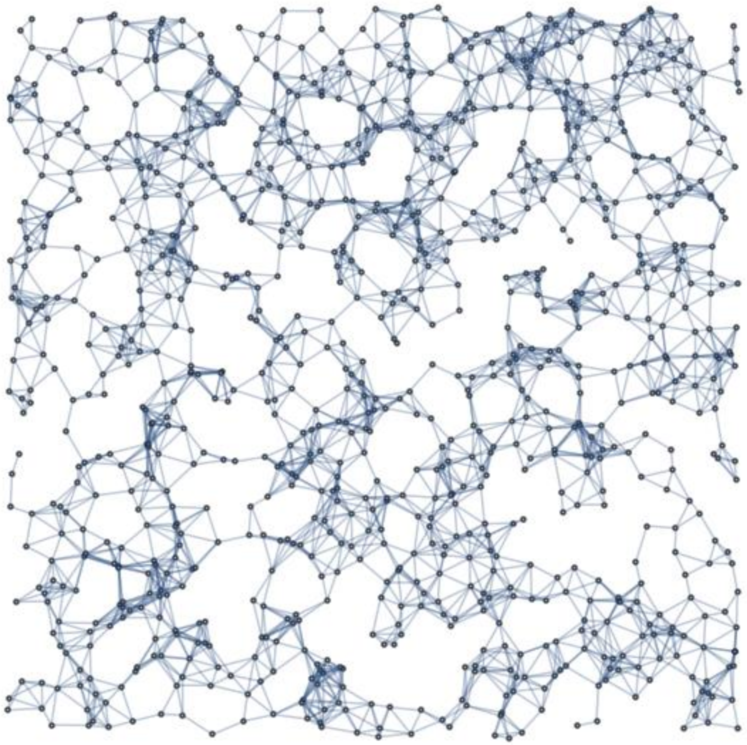
physical

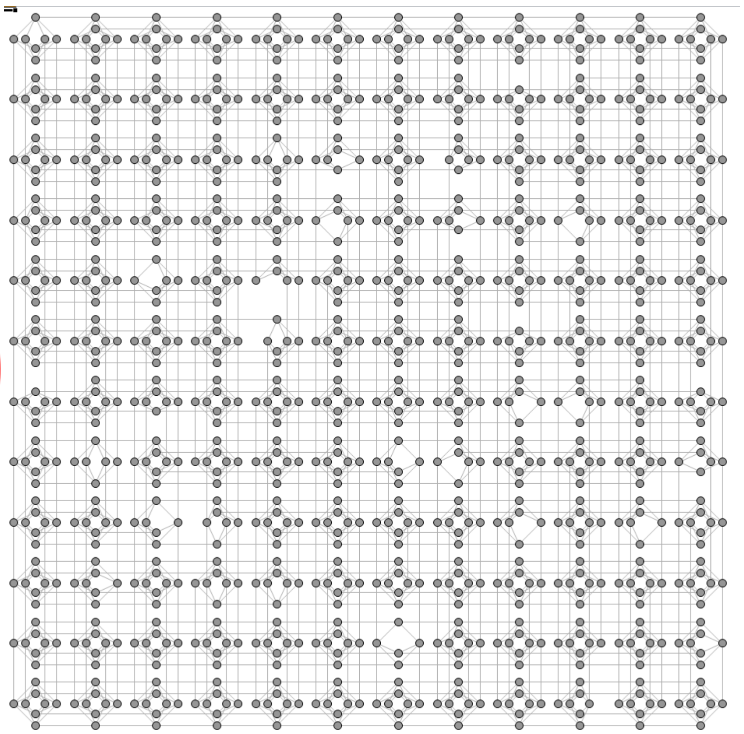
See the D-Wave embedding algorithm reference

The Quantum Computing Company™

One more fly in the ointment

Most QUBOs are too big to embed!





D-Wave The Quantum Computing Company™

Decomposition technique

A Multilevel Algorithm for Large Unconstrained Binary Quadratic Optimization

Yang Wang[1], Zhipeng Lü[2], Fred Glover[3], and Jin-Kao Hao[1]

[1] LERIA, Université d'Angers, 2 Boulevard Lavoisier, 49045 Angers Cedex 01, France
[2] School of Computer Science and Technology, Huazhong University of Science and Technology, 430074 Wuhan, China
[3] OptTek Systems, Inc., 2241 17th Street Boulder, CO 80302, USA
{yangw,hao}@info.univ-angers.fr, zhipeng.lv@hust.edu.cn, glover@opttek.com

Abstract. The unconstrained binary quadratic programming (UBQP) problem is a general NP-hard problem with various applications. In this paper, we present a multilevel algorithm designed to approximate large UBQP instances. The proposed multilevel algorithm is composed of a backbone-based coarsening phase, an asymmetric uncoarsening phase and a memetic refinement phase, where the backbone-based procedure and the memetic refinement procedure make use of tabu search to obtain improved solutions. Evaluated on a set of 11 largest instances from the literature (with 5000 to 7000 variables), the proposed algorithm proves to be able to attain all the best known values with a computing effort less than any existing approach.
Keywords: multilevel approach; unconstrained binary quadratic optimization; hybrid method; memetic algorithm; tabu search

qbsolv

- Shell utility
- Hybrid quantum/classical solver for large QUBOs
- Allows specification and solution of QUBOs with more variables than qubits
- Relies on pre-compiled set of QUBOs for complete graphs
- Layered on dw
- Integrated component of qOp tool suite

QUBO File Format

- Format is a variant of DIMACS CNF file format

```
c  start with comments
c
p qubo 0  4  4  6
c  diagonal elements
0 0 3.4
1 1 4.5
2 2 2.1
3 3 -2.4
c off-diagonals
0 1 2.2
0 2 -3.4
1 2 4.5
0 3 -3.2
1 3 4.5678
2 3 1
```

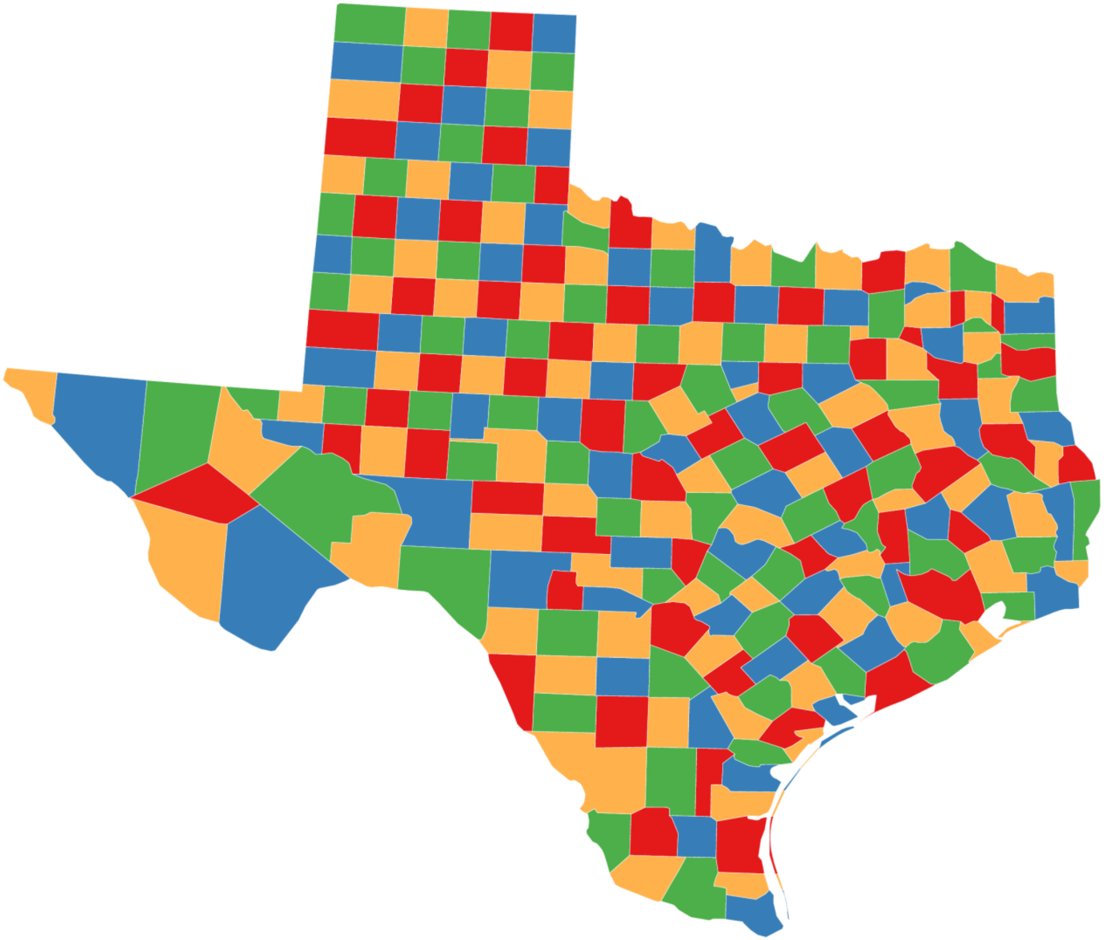
"p" (marker)
Problem type ("qubo")
0 (unconstrained)
maxDiagonals (#variables)
nDiagonals (#nonzero diagonal elements)
nElements (#nonzero off-diagonal elements)

i
j
strength

- zero-based element numbering
- i must be less than j

D-Wave The Quantum Computing Company™

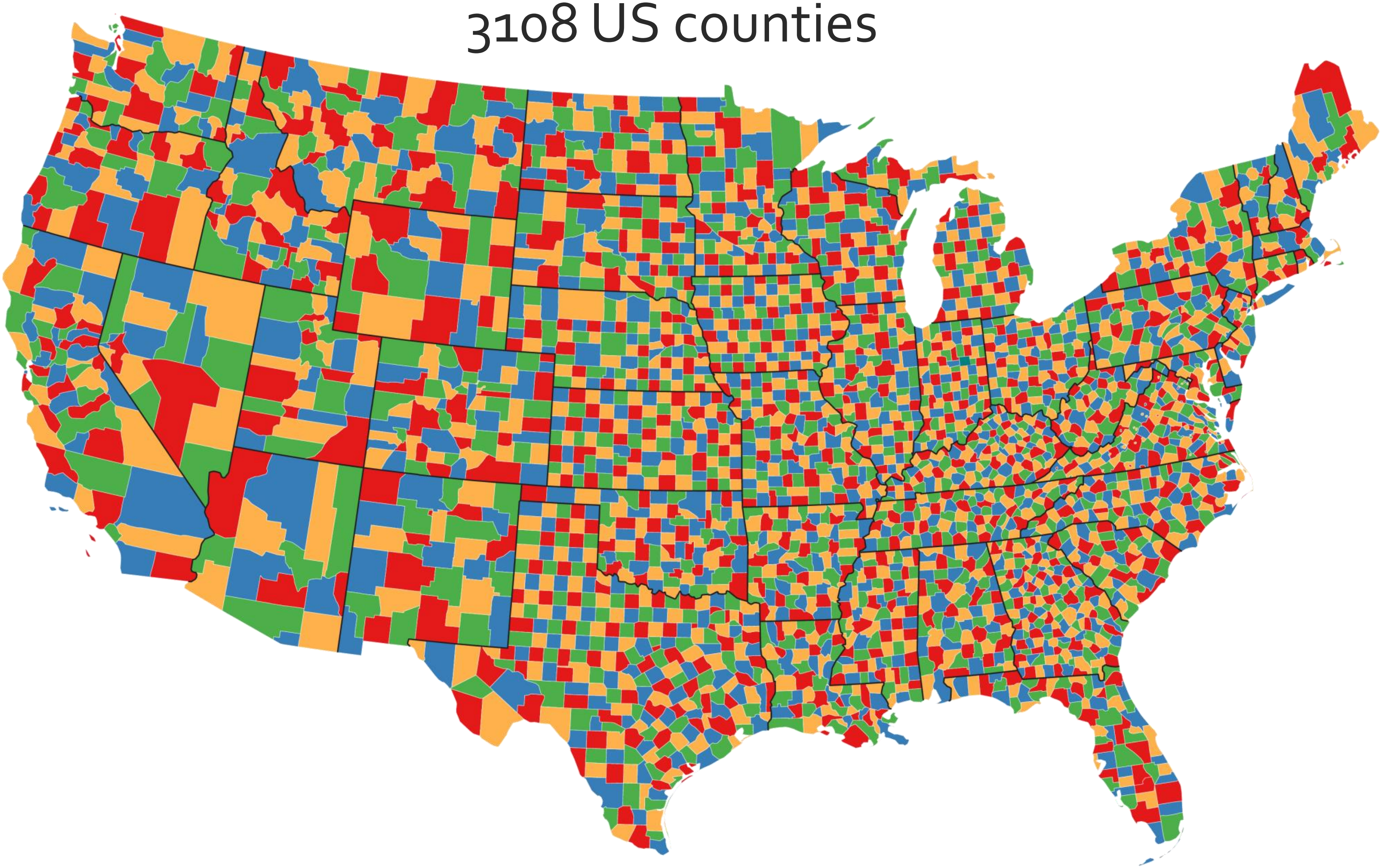
Decomposition allows bigger…



254 counties in Texas

...and bigger problems

3108 US counties



Conclusions

- Individual constraints can be translated into QUBOs
- Sum QUBOs to combine constraints
- An aggregate QUBO (or QMI) can represent many constraints
- Transformations are necessary to enable ***decomposition, parametrization, degree lowering, …***
- It is now possible to imagine combining these steps to begin to build a rudimentary ***quantum compiler***