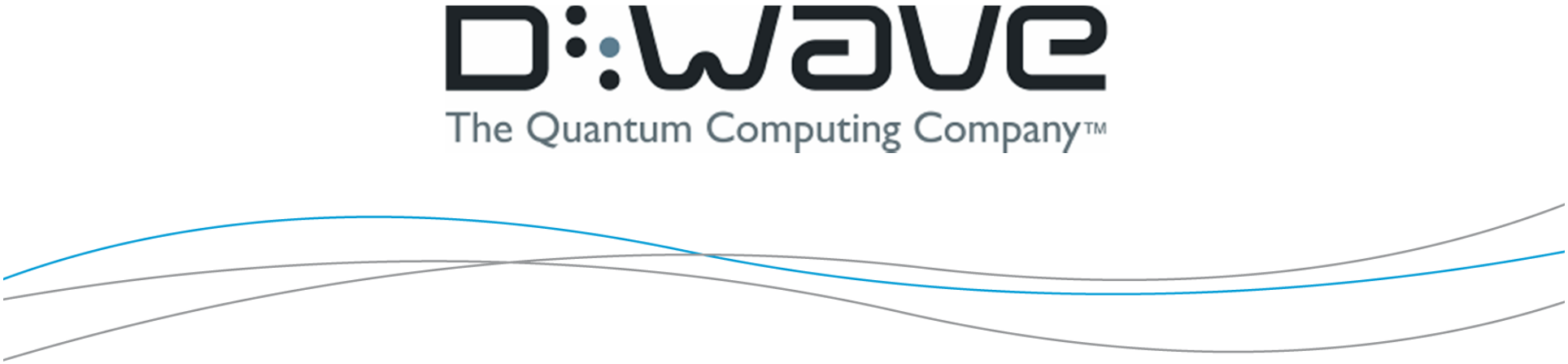


Mapping discrete optimization problems to sparse Ising models

Aidan Roy

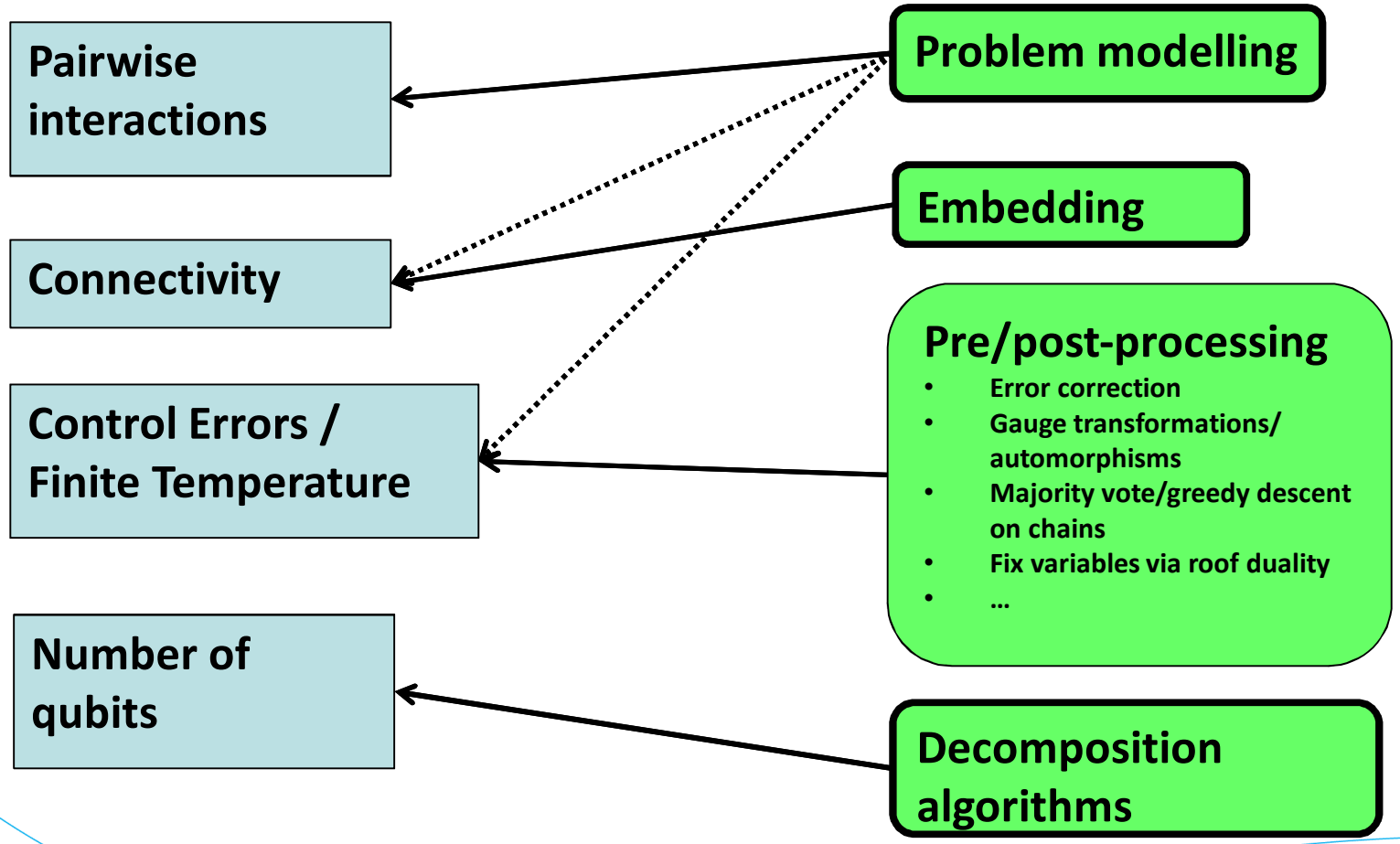
Joint work with
Bill Macready (lead), Zhengbing Bian, Jun Cai, Fabian Chudak, Patrick Hagerty, Robert Israel, Andrew King, Brad Lackey

Current D-Wave hardware specs

- **Pairwise interactions**
  - Final Hamiltonian has form $H_P = \Sigma_i h_i {\sigma_i}^z + \Sigma_{i,j} J_{ij} {\sigma_i}^z \otimes {\sigma_i}^z$ with $h,\ J$ and annealing time programmable

    i.e. minimizes Ising model energy $E(s|h,J) = \Sigma_i h_i s_i + \Sigma_{i,j} J_{ij} s_i s_j, \quad s_i \in \{-1,1\}$
- **Connectivity, number of qubits**
  - 512 qubits, maximum degree 6
- **Control Errors**
  - Finite dynamic range $h, J \in [-1,1]$
  - Input misspecification up to 4%
- **Finite Temperature**
  - Current energy scales are about $\frac{E}{T} \sim$ 3-4

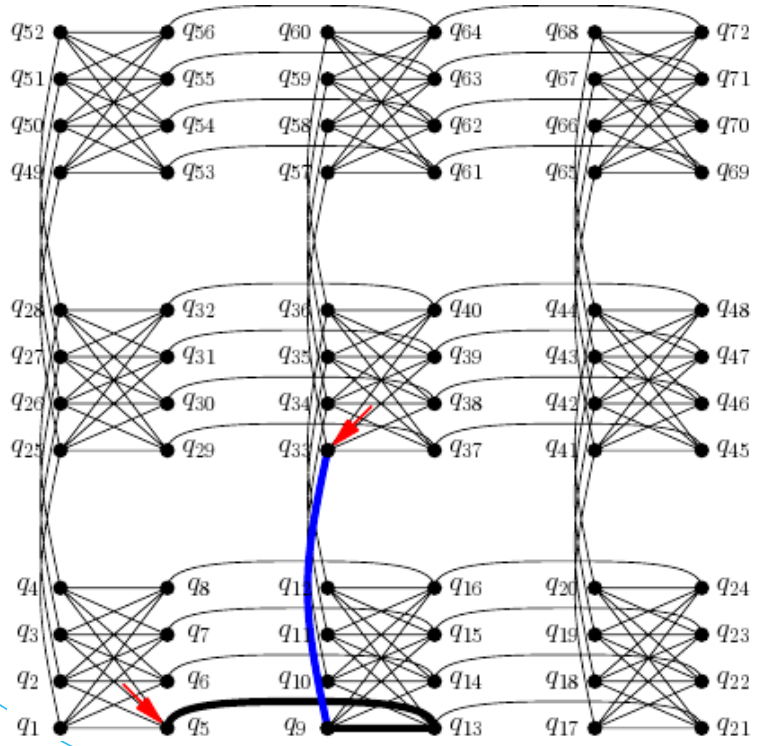
Hardware limitations / Software solutions



Mapping Ising models to the hardware

Embedding

- **In general D-Wave's hardware is too sparse (degree 6) to represent the interactions of a given Ising model**
- **One solution: let multiple qubits represent the same variable**



eg) to model an interaction between qubits q_{33} and q_5:

- Apply strong ferromagnetic couplings to ensure that qubits q_5, q_{13}, and q_9 all take the same value:

$$\boldsymbol{J_{5,13} = J_{13,9} = -1}$$

- Now the required $\boldsymbol{q_{33}q_5}$ term can be implemented on the blue edge

D-Wave
The Quantum Computing Company™

The minor-embedding problem

- **Every variable is represented by a connected subset of qubits (a *chain*)**
  - If variables x and y share an interaction in the Ising model, there must be an edge between their chains
  - Chains must be disjoint
- **Embedding problem depends only on graph H of problem variable interactions and graph G of qubit interactions**
- **This is the *minor-embedding problem:***

H can be represented in G using chains if and only if H is a minor of G.

Embedding is hard in general

- Graph-minors problem is NP-complete
- Best exact algorithm for embedding H in G, k = branchwidth(G):

$$O(2^{(2k+1)\log k}\,|H|^{2k}\,2^{|H|^2}\,|H|)$$

- For fixed H, determining if G has an H-minor is technically P-time, but algorithm is not known or practical.

But…

- We do not require optimal embedding
- If embedding exists, probably many exist
- Therefore probabilistic algorithms are OK (but not studied!)

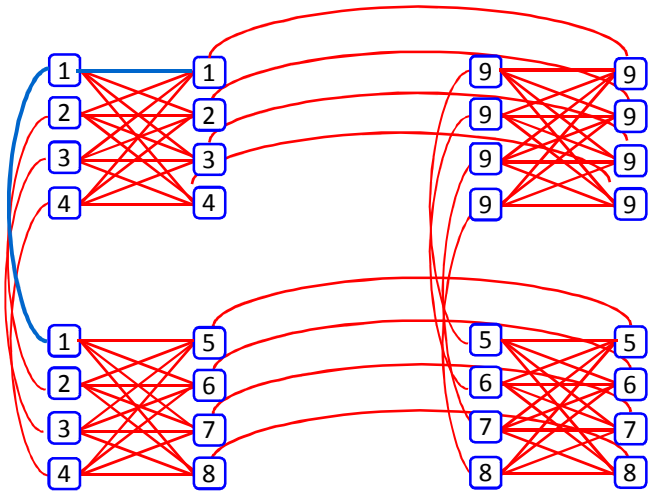
Chimera specific structure can be exploited

- **Complete graph of $4N + 1$ vertices in C_N**

  Any QUBO on N variables maps to a hardware graph of size O(N^2) with chains of size O(N)

- **Tile structure makes the problem easier**
  - Planar-like embeddings may be easier than general embeddings

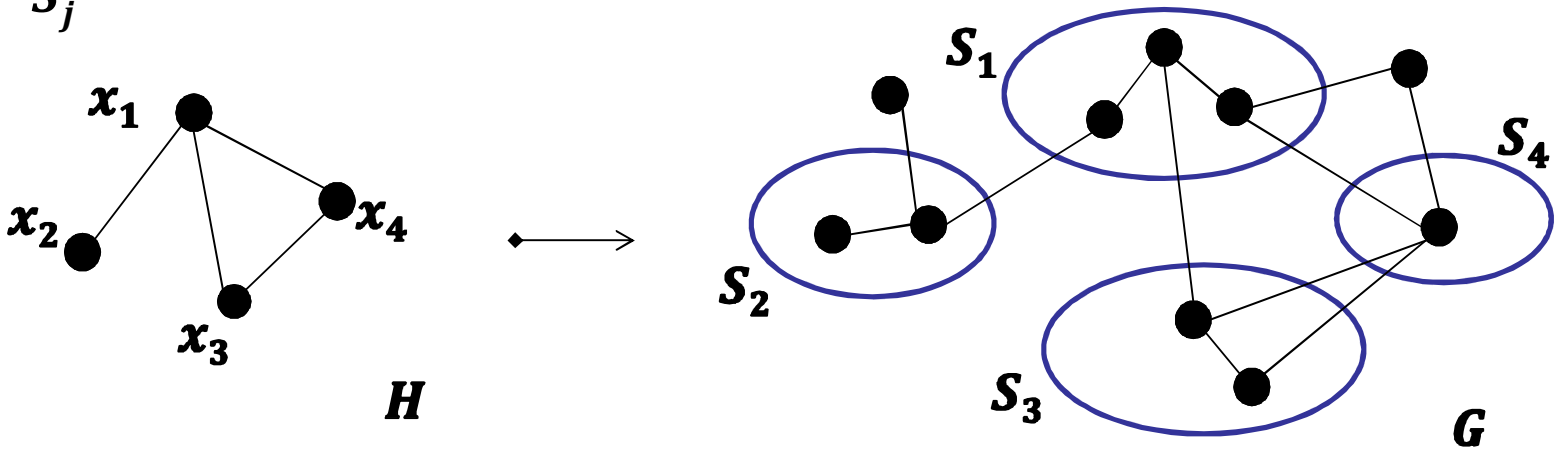
K_9 in C_2



- **Large automorphism group: $|\mathrm{Aut}(C_N)| = 8(4!)^{2N}$**
  - Implies many possible embeddings if an embedding exists
- **Sparsity is conducive to shortest path algorithms**

First greedy heuristic for minor-embedding H into G

- **Represent each variable x_i of H by a connected subgraph (chain) S_i of G such that if $x_i \sim x_j$ then there is an edge between S_i and S_j**



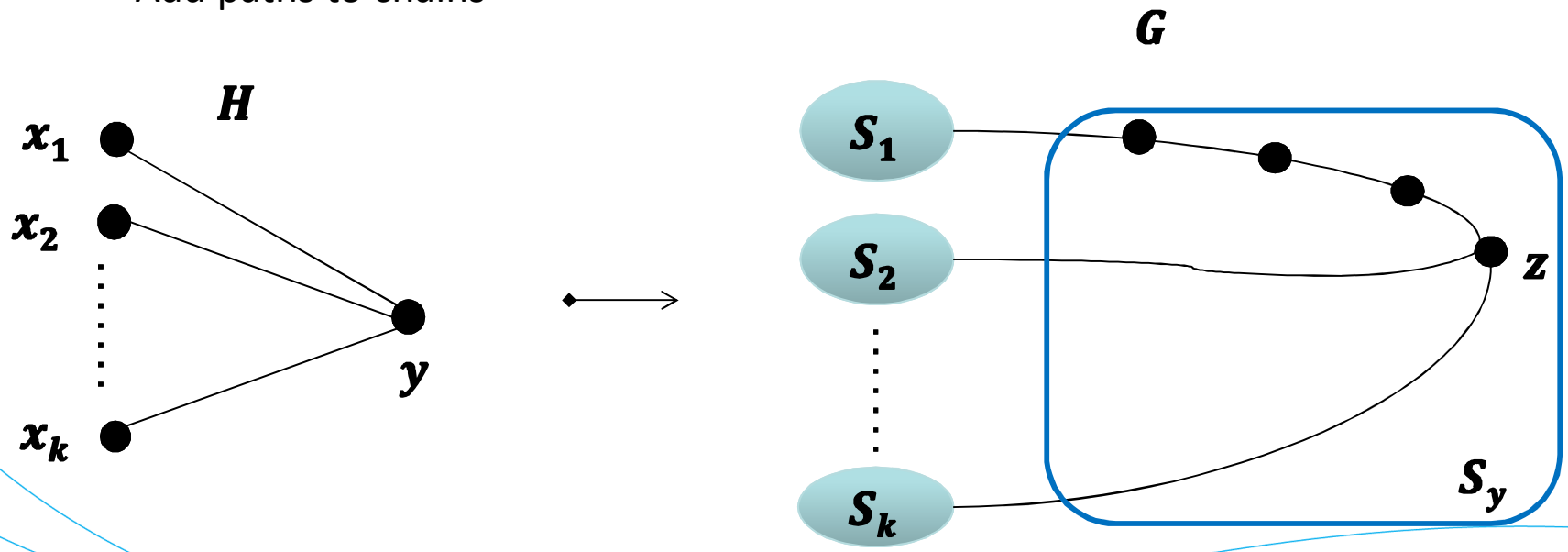
- **Add $x_1, \cdots, x_n$ to an embedding in order, ensuring that a variable is adjacent to all its earlier neighbours and S_i's are disjoint**

- **For each x_i, try to minimize $|S_i|$.**
  - Related to minimal Steiner Tree problem, itself NP-hard.
  - However, polynomial approximations exist.

First greedy heuristic for minor-embedding H into G

To embed variable y of H with neighbours $x_1, \cdots, x_k$ represented by chains $S_1, \cdots, S_k$:

- Find shortest path from S_i to every unused qubit in G (Dijkstra's algorithm)
- Represent y by root qubit z with smallest sum of path lengths
- Add paths to chains



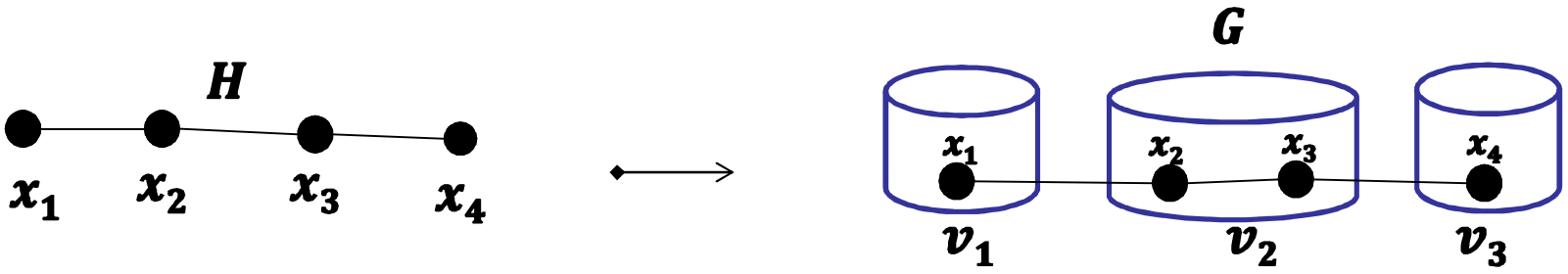
A better heuristic for minor-embedding H into G

- **Simply greedy vertex addition fails quickly**

  Eg) If x_1 has degree 7 but is embedded at a qubit of degree 6

- **So, allow multiple variables of H to be embedded at the same qubit of G**

  "bags": B_v = {variables of H embedded at v in G}



- **After all variables added, remove and try to re-embed x_1, x_2 , …**

- **Repeat until an embedding with bags of size 1 is found or you get stuck.**

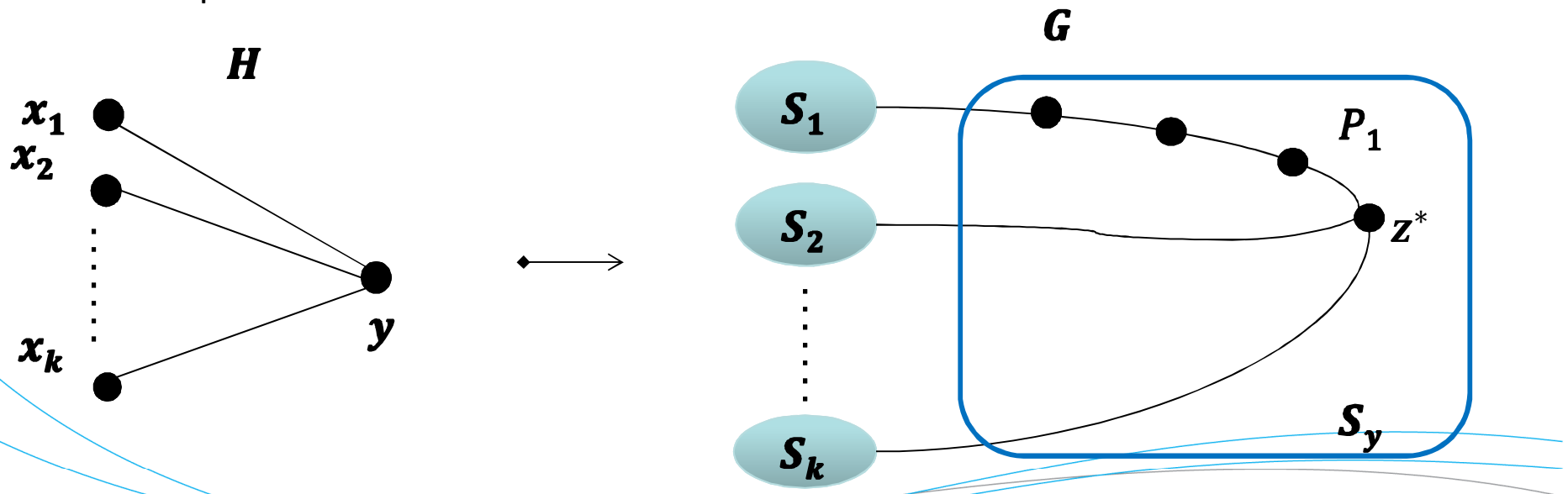
A better heuristic for minor-embedding H into G

To embed variable y of H with neighbours $x_1, \cdots, x_k$ represented by chains $S_1, \cdots, S_k$:

- Find *weighted* shortest path from S_i to every qubit z in G:

$$\text{weight}(P) = \sum_{v \in P} \exp(|\{x_i \neq y : v \in S_i\}|)$$

- Represent y by root vertex z^* with smallest sum of path weights
- Add paths to chains



Demo

Summary

- **Many ways to map to the hardware**
  - **Minor-embedding is the cleanest**
- **Global algorithms are tractable at current scale**
  - **Each iteration costs** $\mathrm{O}(|\mathrm{E}(H)|^*|\mathrm{V}(G)|\log|\mathrm{V}(G)|)$
- **Number of qubits / chain size matter**
  - **Embedding algorithms need to account for this**

Mapping boolean constraints to Ising models

Constraint Satisfaction Problems

- NP problems have simple verification circuits
- If we encode the verification circuit as an optimization objective we can reverse the flow of information from outputs to inputs
  - E.g. factor by representing multiplication circuits
- To carry this out in hardware we need to represent constraints as penalty functions
  - Focus on sparse constraints

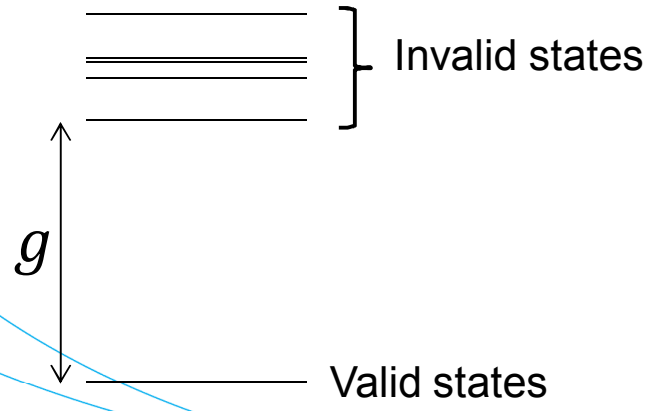
Penalty functions for local constraints

- **A local constraint is a list of feasible configurations F**
  - **For a constriant on K variables, $F \subseteq \{0,1\}^K$**

- **Represent F by a penalty Ising model $E(x)$ defined so that**

$$E(x) = \begin{cases} 0 & \text{if } x \in F \\ \geq g & \text{if } x \notin F \end{cases}$$

where $g > 0$ is a gap



- **Large g mitigates the effects of control errors and finite temperature**
- **May also increase the quantum gap**

Ancillary variables

- **Typically we will require additional ancillary variables which are minimized over:**

$$\boldsymbol{E}(\boldsymbol{x}) = \min_{\boldsymbol{a}} \boldsymbol{P}(\boldsymbol{x}, \boldsymbol{a})$$

  - Introducing more ancilla than necessary can increase g
- **Ancillary variables can also accommodate hardware connectivity:**
  - **Impose hardware connectivity constraints on z = [x,a]**

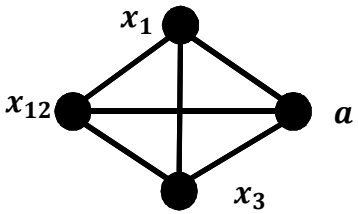
Example: Parity check

$$x_1 + x_2 + x_3 = 0 \bmod 2, \quad x_i \in \{0, 1\}$$

- **Feasible states:** $F = \{000, 011, 101, 110\}$

- **"Obvious" representation:**

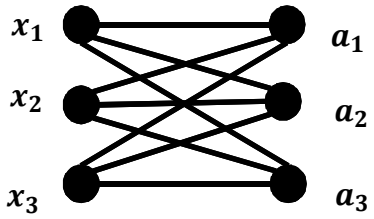
$$P(x, a) = (x_1 + x_2 + x_3 - 2a)^2$$



Energy gap = 1

- **Better representation:**

$$P(x, a) = -a_1 + a_2 - a_3 + x_1(a_1 + a_2 + a_3) + x_2(-a_1 + a_2 + a_3) + x_3(a_1 + a_2 - a_3)$$



Energy gap = 2

Finding penalties: problem formulation

- **Let $\boldsymbol{\theta} = \boldsymbol{c} \cup \{\boldsymbol{h_v}\} \cup \{\boldsymbol{J_{v,v'}}\}$ (c constant offset)**
- **Identify $\boldsymbol{\theta}$ by solving:**

$$\max_{\boldsymbol{\theta},\boldsymbol{g}} \quad \boldsymbol{g}$$

$$\begin{aligned} \text{subject to:} \quad & \boldsymbol{P(x,a|\theta)} \geq \boldsymbol{0} && \forall \boldsymbol{x} \in \boldsymbol{F}, \forall \boldsymbol{a} \\ & \boldsymbol{P(x,a|\theta)} \geq \boldsymbol{g} && \forall \boldsymbol{x} \notin \boldsymbol{F}, \forall \boldsymbol{a} \\ & \exists \boldsymbol{a}\ \boldsymbol{P(x,a|\theta)} \leq \boldsymbol{0} && \forall \boldsymbol{x} \in \boldsymbol{F} \\ & \boldsymbol{-1 \leq \theta \leq 1} \end{aligned}$$

- **Note: we need to determine the state of the ancillary**

$$\boldsymbol{a_*(x)} = \arg\min_{\boldsymbol{a}} \boldsymbol{P(x,a)} \qquad \forall \boldsymbol{x} \in \boldsymbol{F}$$

- **This is a hard problem**
  - **Optimizing for $\boldsymbol{a_*(x)}$ for any given problem defined by $\boldsymbol{\theta}$, and then optimizing over all problems**

Observations

- **Energy function is linear in θ for fixed $z = [x, a]$:**

$$P(x, a|\theta) = c + \Sigma_i h_i z_i + \Sigma_{i,j} J_{ij} z_i z_j$$

- ***If* we knew the ancillae settings $a_*(x)$ then finding θ reduces to linear program:**

$$\max_{\theta, g} g$$

subject to:

$$\begin{aligned} P(x, a|\theta) &\geq 0 && \forall x \in F, \forall a \\ P(x, a|\theta) &\geq g && \forall x \notin F, \forall a \\ P(x, a_*(x)|\theta) &\leq 0 && \forall x \in F \\ -1 \leq \theta &\leq 1 \end{aligned}$$

- **This LP is solvable at scales of interest**

Strategies for finding ancilla values

- **MILP approach:**
  - **solve with additional variables** $a_*(x) = \arg\min_a P(x, a|\theta)$:

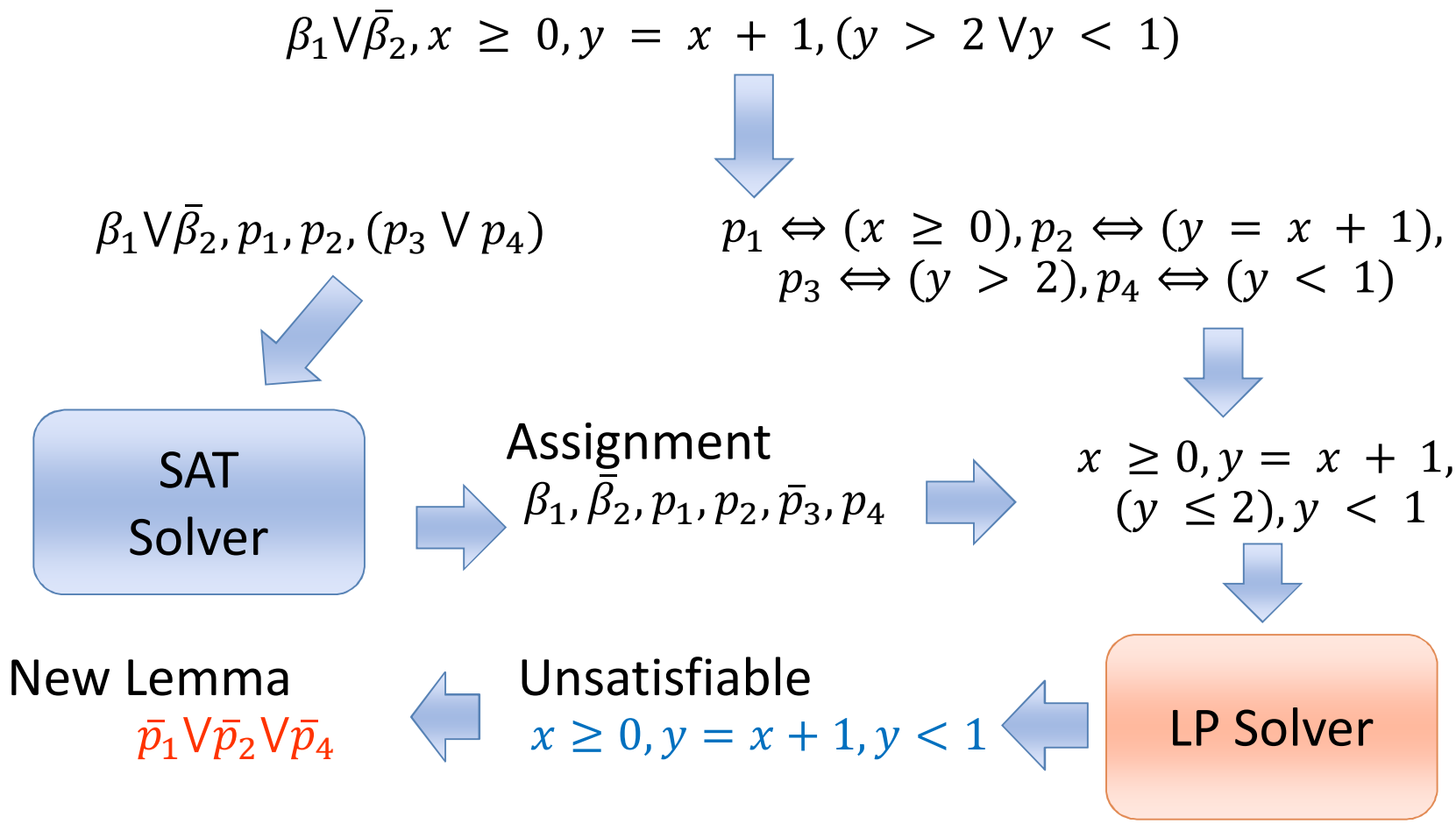
$$P(x, a\,|\theta) \leq 100 \parallel a - a_*(x) \parallel^2 \qquad \forall x \in F, a$$

- **Probabilistic approach:**
  - **Replace min over a with a softmin lower bound**

$$\min_a P(x, a|\theta) \longrightarrow F(x|\theta) = -\frac{1}{\beta} \ln \sum_a exp[-\beta P(x, a|\theta)]$$

  - **Tends to get stuck in local optima with no energy gap**
- **SMT solvers (best so far)**

Satisfiability Modulo Theory (SMT) solvers



D-Wave
The Quantum Computing Company™

Using SMT solvers to find penalty functions

- **Introduce predicate variables indicating settings of ancillary variables:**

$$\beta(x, i) = \text{true} \qquad \text{iff} \qquad [a_*(x)]_i = 1$$

- **Check if penalty function can be realized for $g = 1, 2, \ldots$**
- **We have found SMT to be faster than CPLEX MILP, and easier to model in**
  - **Can incorporate symmetries and low tree-width**

Summary

- **Methods works well at small scales**
  - **constraints embeddable in up to 50 qubits often solved in less than an hour**
  - **SMT solver effective at searching the entire space**
- **Methods can encapsulate both Ising model *and* embedding**
  - **by imposing Chimera structured constraints on ancillary variables**
- **Better models of the same problem give different classical energy gaps**
  - **resulting in different hardware performance**

Decomposition techniques and real-world problems

Decomposition Techniques

- Used to solve problems that are larger than the hardware
- Split the problem into regions, then communicate between regions to coordinate solutions
- Test application: LDPC decoding
- Test algorithms:
  - Belief propagation
  - Dual decomposition

LDPC Decoding

- **Classical channel error decoding:**

Input: parity check matrix $H \in \{0,1\}^{m \times n}$,

received message $y \in \{0,1\}^{n \times 1}$

Output: codeword x **closest to** y

$$\min_{x \in \{0,1\}^{n \times 1}} ||y - x||^2 \quad \text{s.t.} \quad Hx = 0 \pmod 2$$

- **LDPC:** H **is *low density parity check***
  - **Each check has at most 5 bits**
  - **Each bit in at most 4 checks**
  - **Checks generated randomly**
  - **Can get close to Shannon limit of channel capacity**
  - **Has an effecitve classical decoding scheme (BP)**

LDPC as an Ising model

- **Write each check as a quadratic boolean function:**

$$F_C(x_C) = (x_1 + x_2 + x_3 - 2a_1)^2$$

$$= \begin{cases} 0, & x_1 + x_2 + x_3 = 0 \pmod 2 \\ > 0, & x_1 + x_2 + x_3 = 1 \pmod 2 \end{cases}$$

- **Each variable also has a linear term:**

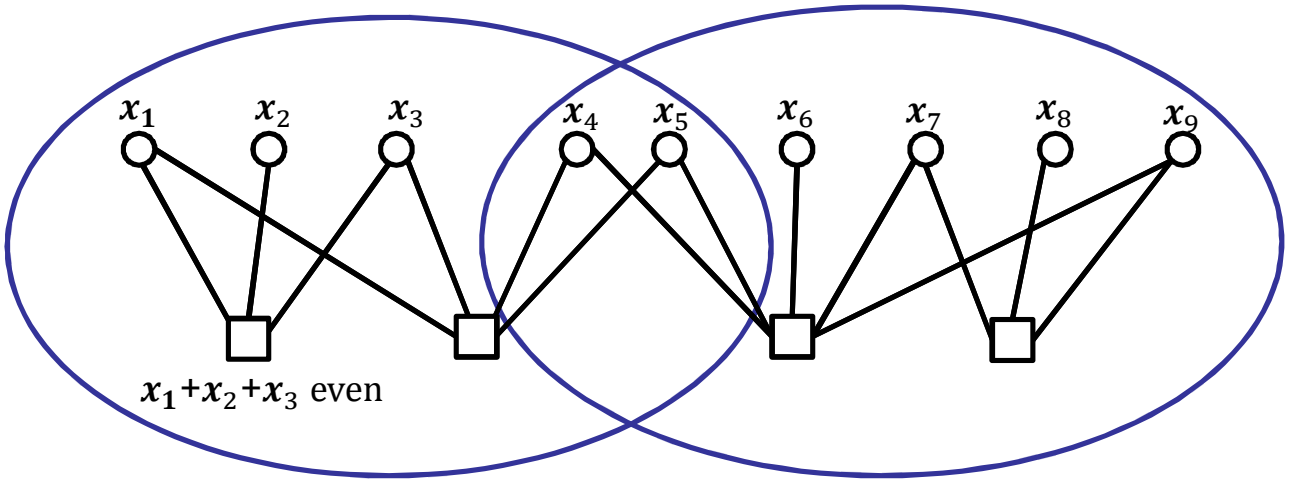
$$F_i(x_i) = (y_i - x_i)^2, \qquad y_i \text{ received value for bit } i$$

- **Overall decoding:**

$$\min_{x} \; F(x) = \sum_i (x_i - y_i)^2 + M \sum_C F_C(x_C)$$

(for some fixed M)

Region-based solutions



Regions should…

- **Be as large as possible while still embeddable in hardware**
  - **Optimize many variables simultaneously, avoid local optima**
- **Have overlap of variables between regions as small as possible**
  - **Minimize communication between regions**

Region-based Belief Propagation

- **Objective function:** $F(x) = \sum_{\alpha} F_{\alpha}(x_{\alpha})$ **where** α **labels the regions which depend on a subset of the variables**
- **Each message is a belief of a "score" (lower score = better) for value** x_i **at variable** i
- **Messages from region to variable nodes:**

$$\mu_{\alpha \to i}(x_i) = \min_{x_\alpha \setminus x_i} \left\{ F_{\alpha}(x_{\alpha}) + \sum_{j \in N(\alpha) \setminus x_i} \mu_{j \to \alpha}(x_j) \right\}$$

- **Messages from variable to region nodes:**

$$\mu_{i \to \alpha}(x_i) = \sum_{\beta \epsilon N(i) \setminus \alpha} \mu_{\beta \to i}(x_i)$$

Dual decomposition

- **To minimize $F(x) = \sum_\alpha F_\alpha(x_\alpha)$ (each α labels a region):**

$$\min_x \; F(x) = \min_{x,\, x^{(\alpha)}} \; \sum_\alpha \; F_\alpha(x^{(\alpha)})$$

$$\text{s.t.} \quad x^{(\alpha)} = x \quad \text{for all } \alpha$$

- **Add Lagrange multipliers:**

$$L(\lambda) = \min_{x,\, x^{(\alpha)}} \; \sum_\alpha \; F_\alpha(x^{(\alpha)}) + \sum_\alpha \; \lambda^{(\alpha)}(x^{(\alpha)} - x)$$

and solve the dual $\max_\lambda L(\lambda)$

- **Subgradient optimization: solving** $\min_{x^{(\alpha)}} F_\alpha\left(x^{(\alpha)}\right)$ **in hardware gives descent moves for** $L(\lambda)$

Region-based algorithms

- **Both Belief Propagation and Dual Decomposition are general algorithms**
  - **can be applied to any QUBO problem**
- **We have tried other algorithms, but BP and DD were the most effective for LDPC decoding**
  - **Correct choice of region–based algorithm is very problem specific**
  - **Both algorithms have their weaknesses**
- **Both BP and DD re-use regions embedded in the hardware**
  - **No need to re-embed at each iteration**

LDPC problems studied

- LDPC instances: 100, 200, 400, 600, 800 and 1000 bits, randomly generated checks
  - # of checks = 70% of # of bits
  - variables in 2-4 checks,
  - checks have 3-5 variables
- Pick a random code word x, and flip 8%, 10%, 12%, 14% of its bits
- Make sure:
  - code word x is the unique solution to decoding problem
  - Standard decoding algorithm fails after 1000 iterations
- Then generate 4-25 regions (20 checks each) and embed each region
  - 150 – 200 variables per region
  - 300 – 350 qubits after embedding
  - 30 – 50 boundary variables in each region

Results: LDPC decoding

Problem			Belief propagation			Dual decomposition		
Bits	Regions	Error rate	Solved?	Iterations	Minimizations	Solved?	Iterations	Minimizations
100	**4**	**8%**	✓	2	296	✓	4	16
		10%	✓	4	452	✓	4	16
		12%	✓	4	561	✓	11	52
200	**7**	**8%**	✓	17	6419	✓	13	105
		10%	✓	7	1975	✓	41	385
		12%	✓	6	1891	✓	81	777
400	**14**	**8%**	✓	6	3057	✓	55	1050
		10%	✓	12	8047	✓	4	56
		12%	✓	8	5724	✓	170	3304
600	**21**	**8%**	✓	7	5386	✓	61	1743
		10%	✓	9	8663	✓	56	1596
		12%	✓	9	9200	×	200	5838
800	**28**	**8%**	✓	6	7417	✓	133	5124
		10%	✓	11	7915	✓	115	4452
		12%	✓	13	13070	×	200	7784
1000	**35**	**8%**	✓	10	10272	✓	139	6685
		10%	✓	9	12935	×	200	9730
		12%	✓	29	18410	×	200	9730
--------	----------	----------						
100		**14%**	✓	19	2413			
200			✓	8	2105			
400			×	20	13421			
600			×	4	4914			
800			×	8	10002			
1000			×	15	61493			

D-Wave
The Quantum Computing Company™

References

- Z. Bian, F. Chudak, P. Hagerty, R. Israel, B. Lackey, W. G. Macready and A. Roy. "*Discrete optimization using Quantum Annealing on sparse Ising models*", to be submitted to "Frontiers in Physics", 2014.
- J. Cai, W. G. Macready, A. Roy. "*A practical heuristic for graph minors*", 2014. arxiv.org/1406.2741.

V6 vs. V7

Vesuvius 7 online in Burnaby since January:

- **New coupler design to boost maximum coupling strength**
- **Removed several key crosstalks to improve h ICE**
- **New coupler design reduces J ICE from DAC round-off**
- **Added passivation layer to the top of fabrication stack to reduce 1/f noise**
- **481 qubits (low yield typical for new version)**

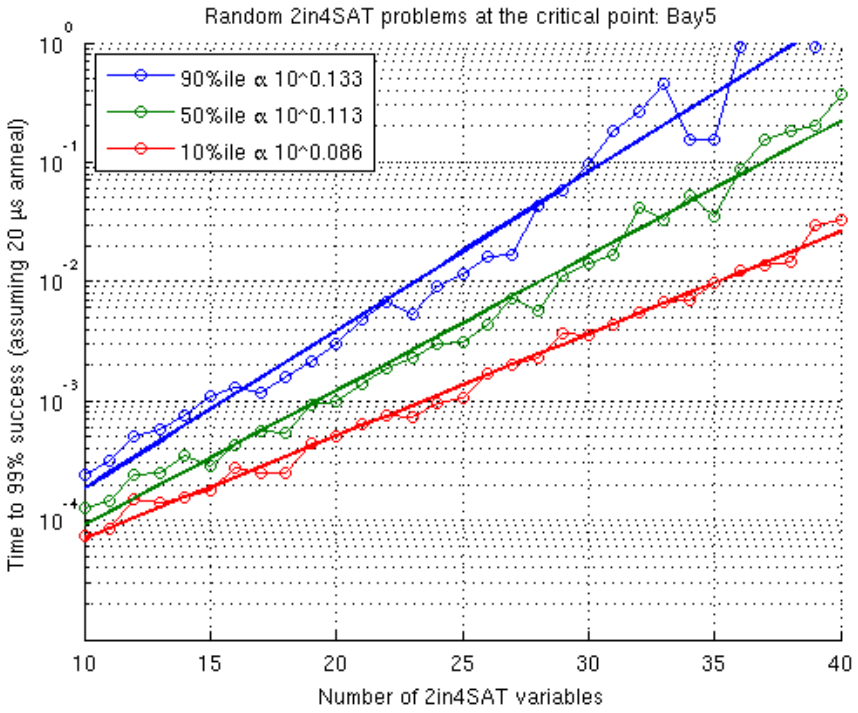
V6 vs. V7 performance on chains: 2in4-SAT tests

- **2in4-SAT: exactly 2 out of 4 literals in every clause are satisfied**
- **Tested at critical clause/variable ratio of 0.7 (phase transition from satisfiable to not satisfiable)**
- **Tested problems: 50 instances of each size with 10-40 variables, embedding with maximum chain length 4-16**
- **Problems embedded on the intersection of working graphs, chain strength 3.5, majority vote post-processing**
- **20 gauge transforms, 1000 samples each**

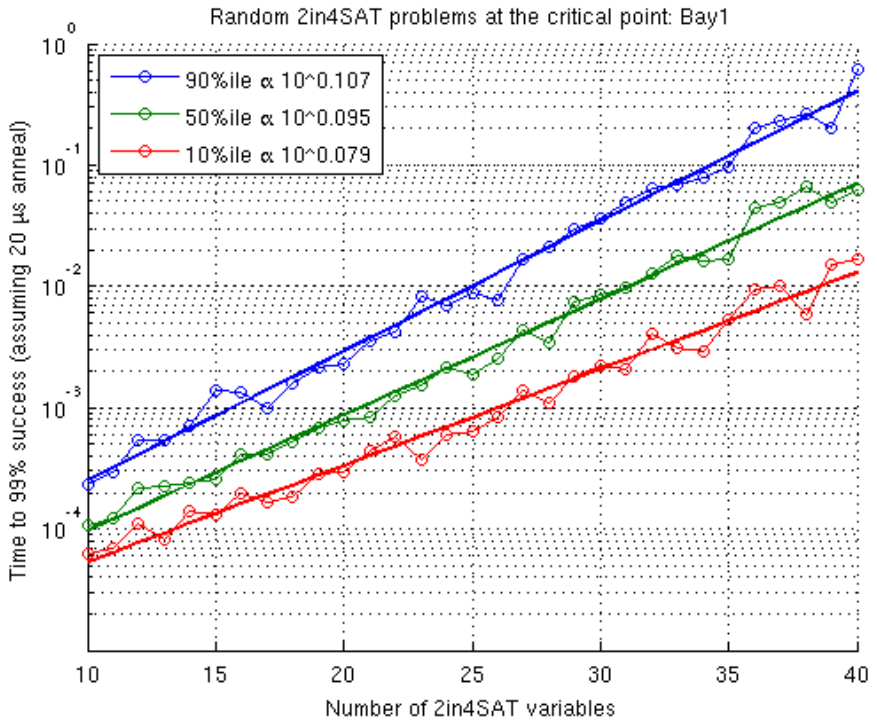
V6 vs. V7 performance on chains

- **2-in-4-SAT: exactly 2 out of 4 literals are true in every clause.**

V6



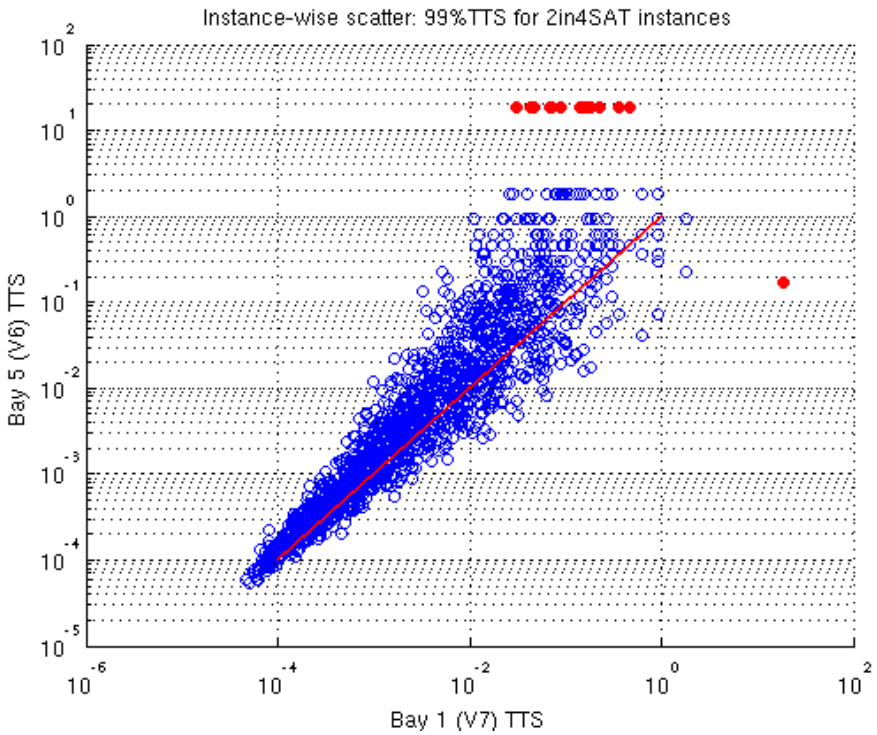
V7

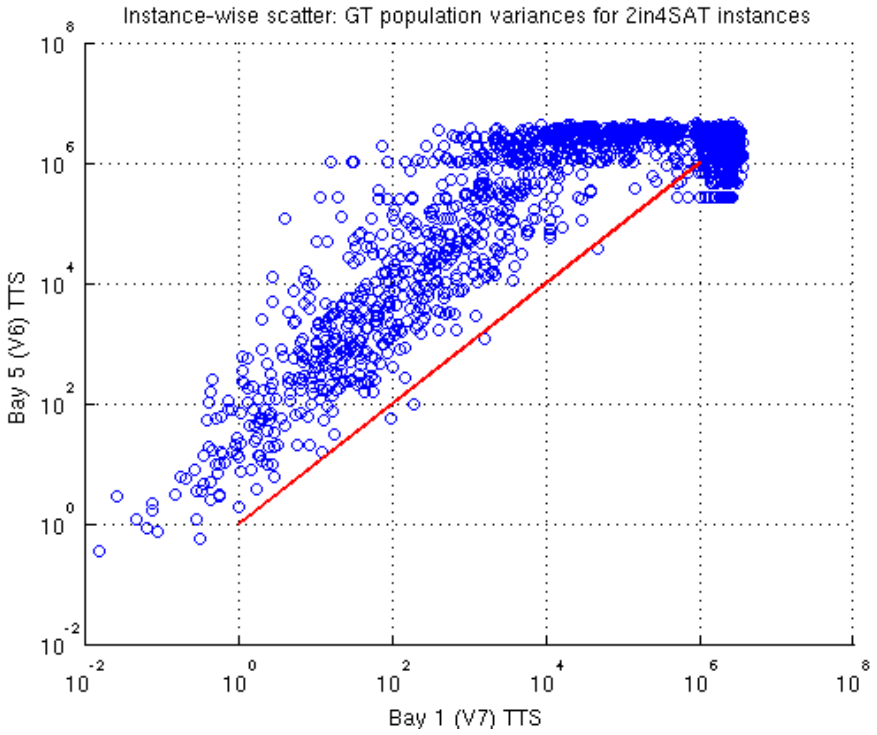


- **Scaling performance is dominated by ICE, not annealing gap**

D-Wave The Quantum Computing Company™

V6 vs. V7 performance on chains





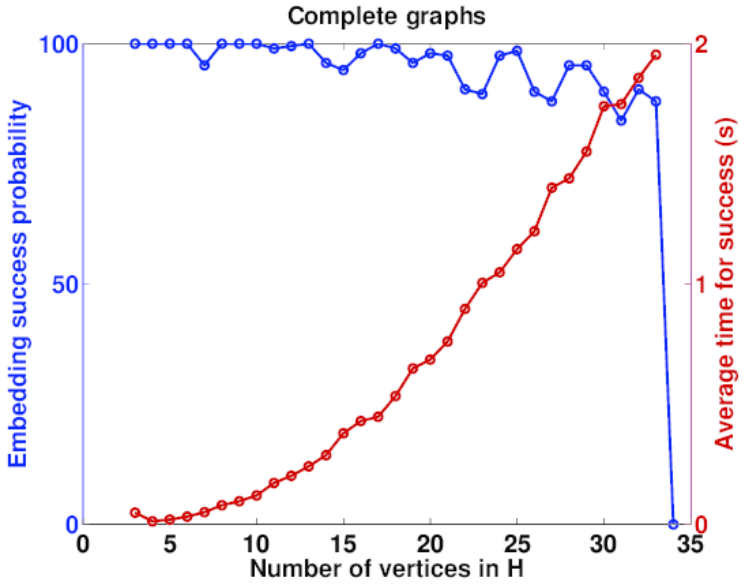
- **Without gauge transformations, performance difference is even more pronounced**

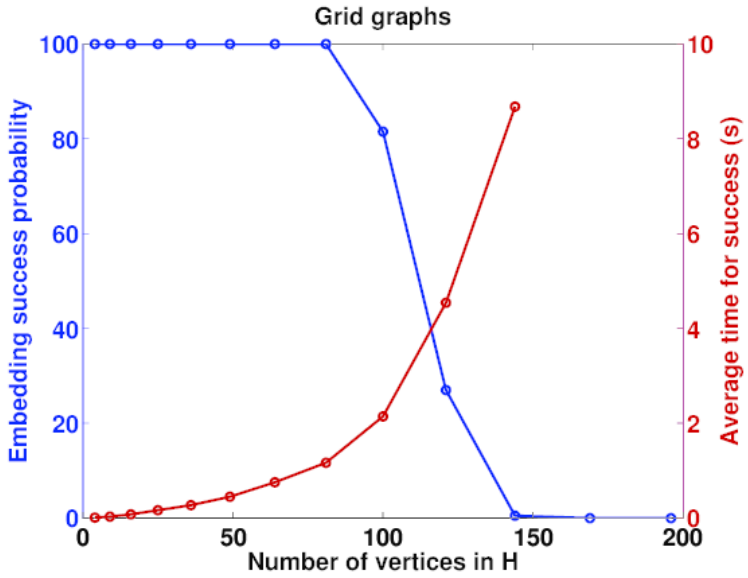
Analysis of run time

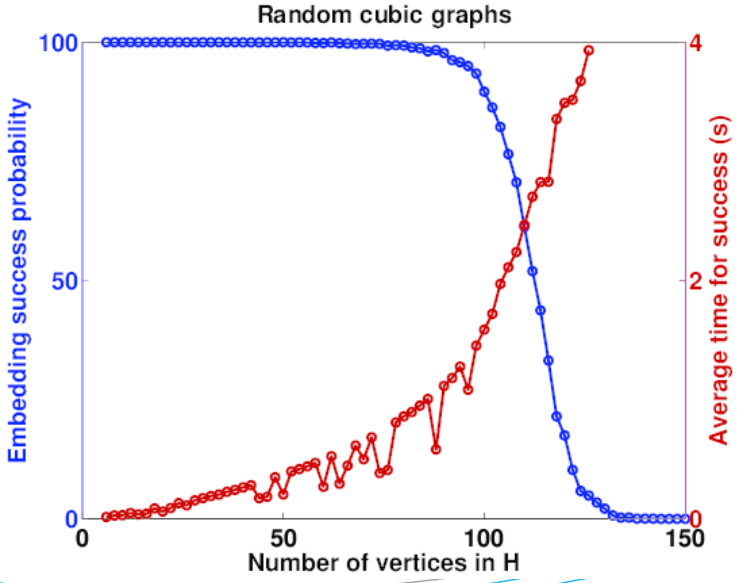
- For $H \to G$: each iteration takes $O(|E(Q)| * |V(G)| \log |V(G)|)$
- Algorithm terminates after $\leq |E(Q)| * |V(G)|$ iterations

- Measured number of iterations/success rate empirically
- "Easy" problem: embed K_N in C_8 for N = 1...33
  - provably largest embeddable complete graph
- "Hard" problem: embed $N \times N$ grid graph in C_8 for N = 1…16
  - again provably largest possible
- Random problem 2: random cubic graphs up to 40 variables
- Ran each problem 100 times (algorithm has several sources of randomness)

Embedding performance







D:wave
The Quantum Computing Company™