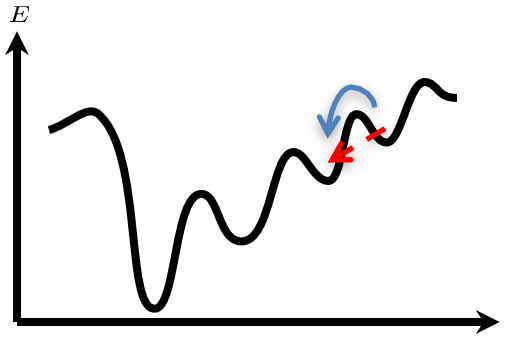
Overcoming limitations of quantum annealers

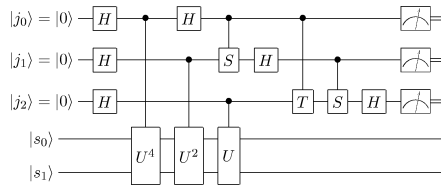
Federico Spedalieri
Information Sciences Institute
USC

International Workshop on Quantum Annealing and its Applications in Science and Industry (QuAASI'16) at Jülich Supercomputing Centre

Quantum annealers



- Quantum annealers are quantum computing devices based on the adiabatic model of QC



- Less general than universal quantum computing (like what is implemented by the circuit model)

- Designed to solve computationally hard problems know as combinatorial optimization (like the Traveling Salesman problem)

Combinatorial optimization

- Optimization of a function over a discrete space
- State space grows exponentially with problem size
- Classically intractable (best algorithms scale exponentially)
- Related to decision problems (SAT) and NP-hardness

Problem	Application
Traveling salesman	Logistics, vehicle routing
Minimum Steiner tree	Circuit layout, network design
Graph coloring	Scheduling, register allocation
MAX-CLIQUE	Social networks, bioinformatics
QUBO	Machine learning, software V&V
Integer Linear Programming	Natural language processing
Sub-graph isomorphism	Cheminformatics, drug discovery
Job shop scheduling	Manufacturing
Motion planning	Robotics
MAX-2SAT	Artificial intelligence

Ising and QUBO

- All these problems can be converted into one another with at most a polynomial penalty
- A faster algorithm for one problem can be used for another
- Two of these problems are of interest for quantum annealing

Ising minimize $E(\vec{s}) = \sum_i h_i s_i + \sum_{ij} J_{ij} s_i s_j \quad , \quad s_i \in \{-1, +1\}$

QUBO
(quadratic unconstrained binary optimization)

$$\min_{\mathbf{x}} \mathbf{x}^T Q \mathbf{x} \quad , \quad x_i \in \{0, 1\}$$

$$\mathbf{s} = 2\mathbf{x} - 1 \longrightarrow \begin{array}{rcl} h_i & = & \frac{1}{2} Q_{ii} + \frac{1}{4} \sum_{j=1}^{n} Q_{ij} \\ J_{ij} & = & \frac{1}{4} Q_{ij} \end{array}$$

Adiabatic quantum optimization

- Find the minimum of $E(\vec{s}) = \sum_i h_i s_i + \sum_{ij} J_{ij} s_i s_j \quad , \quad s_i \in \{-1, +1\}$

  Shown to be NP-hard (Barahona, 1982)

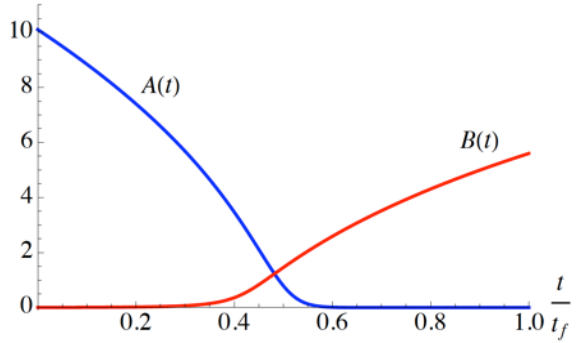
- Solve using AQO

  Find the ground state of the Ising Hamiltonian

$$H_{\text{Ising}} = \sum_i h_i \sigma_i^z + \sum_{ij} J_{ij} \sigma_i^z \sigma_j^z$$

  using adiabatic interpolation from the transverse field (Farhi et al., 2000)

$$H(t) = A(t) \sum_j \sigma_j^x + B(t) H_{\text{Ising}} \quad , \quad t \in [0, t_f]$$



- Programming means finding the appropriate $\{h_i, J_{ij}\}$

Physical implementation

- D-Wave processor: implement a programmable set of interacting magnetic elements (Ising)

- Idea: exploit quantum effects to produce the minimizing configuration more efficiently

- Quantum effects have been experimentally observed on the D-Wave processor

  - Single qubit tuneling
  - Multi-qubit tuneling
  - Entanglement

Solving problems with QA

- Great!
  - We have a quantum annealer (D-Wave processor)
  - It exhibits quantum effects
  - Just pick you favorite problem and solve it!
- Example:
  - Given a graph G=(V,E) and a set of n colors, can we color all vertices such that no edge connects two vertices with the same color?

Application: scheduling, register allocation

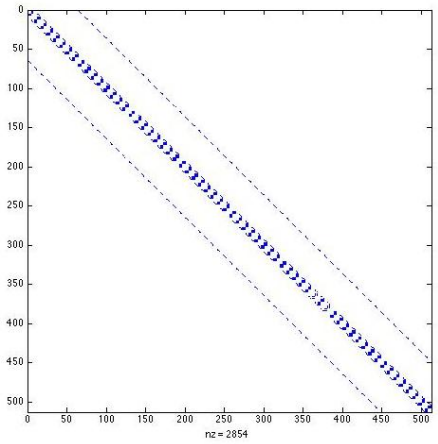
Variables $x_{v,i} = \begin{cases} 1 & \text{if vertex } v \text{ has color } i \\ 0 & \text{otherwise} \end{cases}$

$$H = A \sum_{v} \left(1 - \sum_{i=1}^{n} x_{v,i} \right)^2 + A \sum_{(uv) \in G} \sum_{i=1}^{n} x_{u,i}\, x_{v,i}$$

Solving problems with QA

- Suppose we have a graph with 10,000 vertices and the number of colors is 10
- The total number of variables is 100,000
- But the number of qubits in our QA is only about 1000!

- Furthermore, the QUBO matrix of the problem is fully connected
- But the couplings we can implement on the QA are restricted



DW2 adjacency matrix

Limitations of QA

- Limited size

- Restricted connectivity

$$J_{ij} = 0$$

If (i,j) not in underlying physical connectivity graph

- Limited parameter range and precision

$$h_i \in [-2, 2]$$
$$J_{ij} \in [-1, 1]$$

(for D-Wave processor)

- Intrinsic control errors

$$(h_i, J_{ij}) \longrightarrow (h_i + \Delta h_i, J_{ij} + \Delta J_{ij})$$

Limitations of QA

- We will discuss strategies to deal with size and connectivity limitations
  - Limited size: decompose large problems into smaller ones
  - Limited connectivity:
    - Create smaller, more connected effective graphs
    - Exploit sampling capabilities

Problem Decomposition

- We need to reduce a QUBO problem on N variables to several smaller ones
- Choose a set of n < N variables and fix the values of the remaining ones
- Solve the n variable QUBO problem

$$\sum_{i,j=1}^{N} J_{ij} x_i x_j \longrightarrow \sum_{i,j \in S} J_{ij} x_i x_j + \sum_{i \in S} b_i x_i$$

Terms involving fixed variables become linear terms

- Repeat by choosing a different set of n variables until some exit criteria are met

- There is no guarantee that decomposition will find the best answer
- How many and how the variables are chosen will impact the solution

Decomposition approaches

- We will discuss two possible approaches
  - Cluster based: choose variables according to the strength of their coupling
  - Backbone based: choose variables according to their contribution to the objective function

Cluster based decomposition

- Pick a random variable
- Find the variable that has the stronger coupling to it and add it to the cluster
- Of all the variables coupled to the cluster, pick the variable "j" that maximizes

$$\sum_{i \in \text{Cluster}} |J_{ij}|$$

- Repeat until the desired cluster size is reached
- Solve reduced QUBO on cluster variables
- Repeat for different cluster

Backbone based decomposition

- Certain variables seem to have a preferred value on most good solutions, and are referred to as the "backbone" of the solution
- Identifying some of those variables allows us to reduce the problem size, since they will have a fixed value for good solution

Procedure

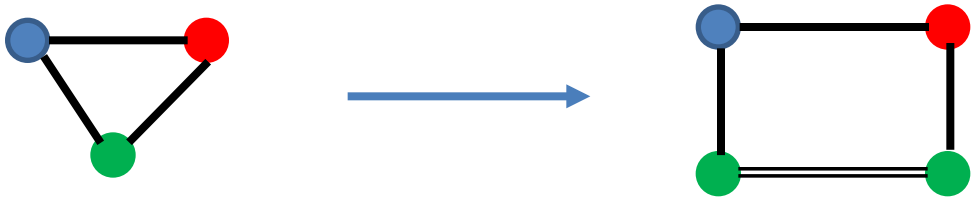
- To start, pick a random configuration as a candidate solution
- Order the variables according to how much they change the objective function if flipped from their current values
- Choose the n variables that either reduce it or increase it the least
- Variables that increase the objective by a large amount when flipped will remain with their assigned value and will not be optimized over

Embedding approaches

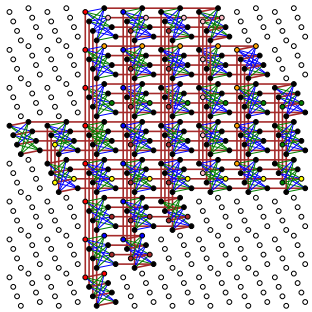
- Once the problem is reduced we still need to embed it on the processor
  - Canonical minor embedding: can embed any problem up to a certain number of variables (45 in DW2X)
  - Specialized minor embedding: can embed some larger problems but requires computational effort
  - Iterative sampling: can tackle much larger problems but optimality is not guaranteed
- The embedding approach chosen will constraint the size of the subproblems that can be considered

Minor embedding

- The first two approaches are based on the minor embedding technique



- The canonical approach requires no precomputing but puts the stronger limit on the size of the subproblems (Sqrt(N))
- The specialized approach may be able to solve larger subproblems, but non trivial precomputing may be required



Embedding of 32 node, fully connected graph

Beyond minor embedding

- If minor embedding is hard to find, use approximate embedding

  Approximate embedding: map only a subset of logical couplers to physical couplers

  - Use some criteria to choose mapping, like giving priority to strongest couplers (requires pre-computing effort)
  - Project into Chimera (cross fingers, hope for the best)

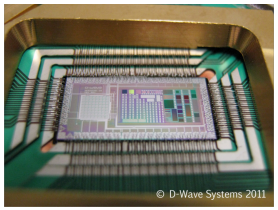
Both of these approaches can be improved by exploiting the sampling capabilities of the D-Wave device

Iterative sampling

- Replace optimization by sampling

$$\min_{\mathbf{s}} G(\mathbf{s}) \quad \rightarrow \quad P_G(\mathbf{s}) = \frac{e^{-\beta G(\mathbf{s})}}{Z}$$

- Consider the D-Wave processor as a sampler with knobs

$$(h_i, J_{ij}) \longrightarrow$$  $$\longrightarrow \quad P_C(\mathbf{s}\,; h_i, J_{ij})$$

- Find (h_i, J_{ij}) such that $P_C(\mathbf{s}\,; h_i, J_{ij})$ is "close" to $P_G = \frac{e^{-\beta G(\mathbf{s})}}{Z}$

Parameterized approximate sampling

- Use relative entropy as a measure of closeness

$$\underset{(h_i, J_{ij})}{\text{Minimize}} \quad D(P_C||P_G) = \sum_{\mathbf{s}} P_C(\mathbf{s}\,; h_i, J_{ij}) \log\left(\frac{P_C(\mathbf{s}\,; h_i, J_{ij})}{P_G}\right)$$

- In order to proceed, we'll make the following assumption

$$P_C(\mathbf{s}\,; h_i, J_{ij}) \simeq \frac{1}{Z} \exp\left(-\beta(\sum_i h_i s_i + \sum_{ij} J_{ij} s_i\, s_j)\right)$$

Not completely accurate, not terribly inaccurate

Parameterized approximate sampling

- Compute (approximate) gradient of relative entropy

$$\nabla_{\{h_i,J_{ij}\}} D(P_C||P_G) = -\beta \begin{cases} \langle s_i \log\left(\frac{P_C}{e^{-\beta G(\mathbf{s})}}\right)\rangle_C - \langle s_i\rangle_C \langle \log\left(\frac{P_C}{e^{-\beta G(\mathbf{s})}}\right)\rangle_C \\ \langle s_i s_j \log\left(\frac{P_C}{e^{-\beta G(\mathbf{s})}}\right)\rangle_C - \langle s_i s_j\rangle_C \langle \log\left(\frac{P_C}{e^{-\beta G(\mathbf{s})}}\right)\rangle_C \end{cases}$$

$\langle \cdots \rangle_C \longrightarrow$ expected value on "Boltzmann" Chimera distribution

- Approximate $\langle \cdots \rangle_C$ by sample average (processor's output)

- Update values $(h_i, J_{ij}) \longrightarrow (h_i, J_{ij}) + \alpha \nabla_{\{h_i,J_{ij}\}} D(P_C||P_G)$

Iterative approximate sampling

- The algorithm goes as follows

1. Choose an initial embedding (i.e., map variables to physical qubits)
2. Project the original problem into the underlying Chimera graph
3. Run the annealer to generate N samples
4. Compute the approximate relative entropy gradient
5. Update couplers and local fields
6. Go to 3 (until some stopping criteria is reached)

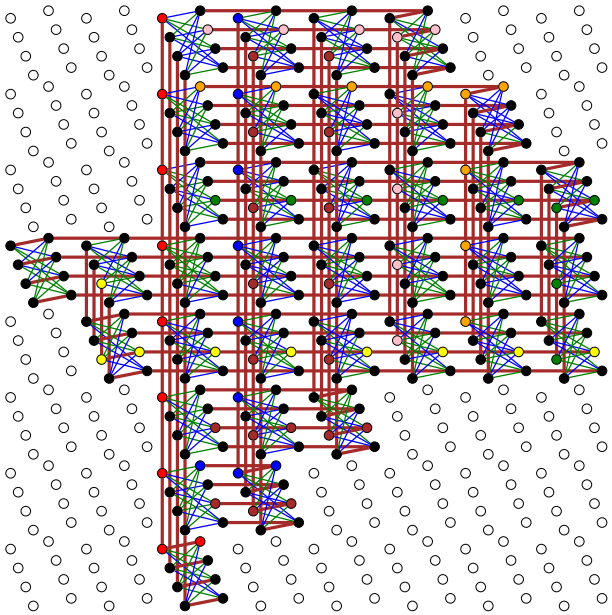
- Parameters are updated respecting the box constraint

$$h_i \in [-2, 2]$$
$$J_{ij} \in [-1, 1]$$

Complete graph with N=32

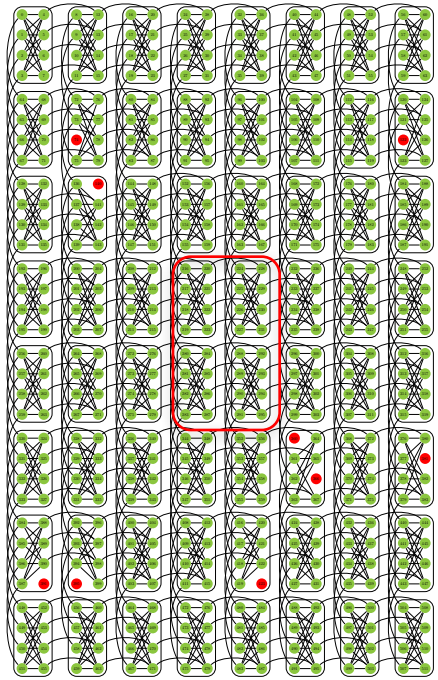
- Compare minor embedding of N=32 complete graph with iterative sampling approach

Minor embedding



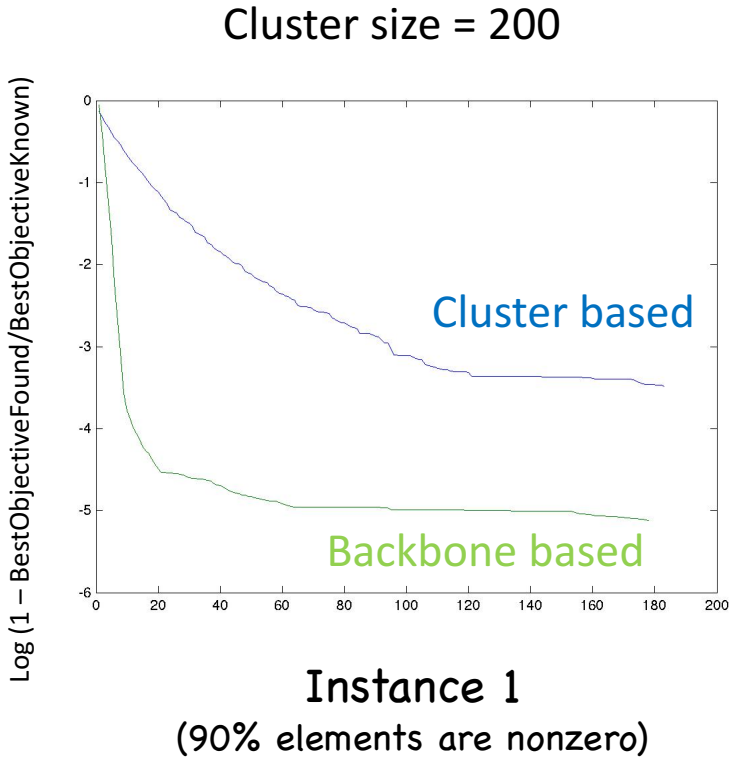
VS

Iterative sampling



Decomposition + embedding

- Some preliminary results (ongoing research)
- We considered a set of 10 QUBO instances with 2500 variables designed by Beasley (OR-library, http://people.brunel.ac.uk/~mastjjb/jeb/info.html)
- We solved them using the two decomposition approaches and then applying the iterative sampling
- The iterative sampling was run on DW2X
- We compared our results against the best known solutions as a function of the number of calls to the DW2X processor



Summary

- Quantum annealers are designed to solve combinatorial optimization problems
- They have limitations on size, connectivity, parameter range and precision
- To solve large problems we need to break them into pieces
- The way those pieces are chose can impact performance
- Even when small in size, problems my not fit due to connectivity constraints
- We can embed general general graphs if we pay a quadratic penalty in the number of qubits required
- Sampling capabilities can also be exploited to approximate more connected problems
- QA seems to be good at providing "good" solutions fast (even if it not provides the best solution)