18-819F: Introduction to Quantum Computing
47-779/47-785: Quantum Integer Programming & Quantum Machine Learning

Quantum Annealing, Quantum-Inspired Heuristics, Benchmarking, and Parameter setting

Lecture 14

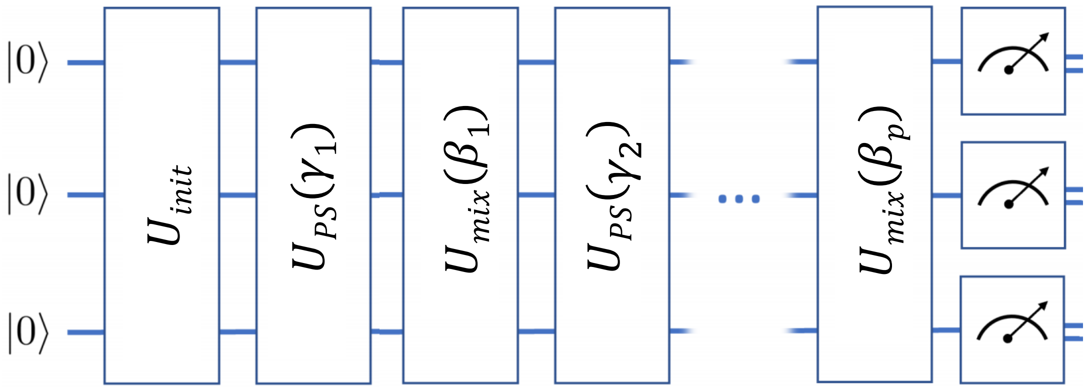
2021.10.21

Electrical & Computer ENGINEERING

TEPPER

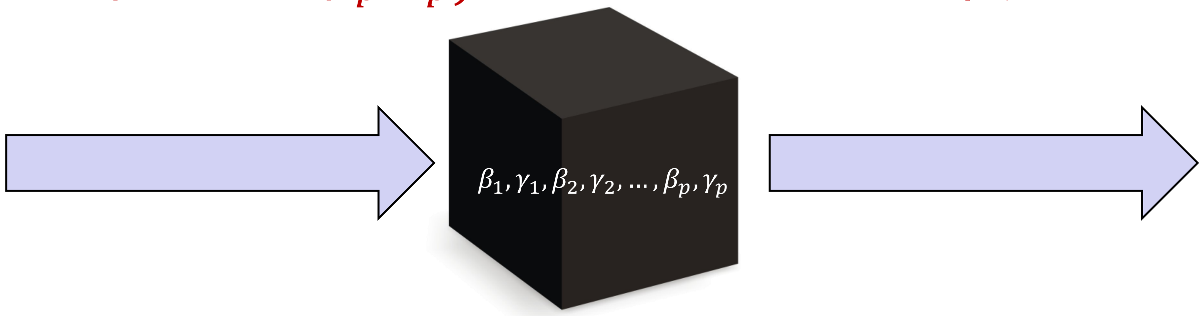
USRA Universities Space Research Association

Quantum Approximate Optimization Algorithm: RECAP



Now if you measure, the probability of a bitstring depends both on γ and β in a non-linear way.
It is exponentially difficult to predict or simulate the probability $\left|\mathrm{B}_{2s}(\beta_1, \gamma_1, \beta_2, \gamma_2, \dots, \beta_p, \gamma_p)\right|^2$ to find the optimal unknown solution s^*

$$|\psi\rangle_{\mathrm{QAOA}(p)}$$
$$= \left(2^{N/2}\right)^{-1} \sum_s \mathrm{B}_{2s}(\beta_1, \gamma_1, \beta_2, \gamma_2, \dots, \beta_p, \gamma_p) e^{i\Gamma_{1s}(\beta_1, \gamma_1, \beta_2, \gamma_2, \dots, \beta_p, \gamma_p)} |s\rangle$$

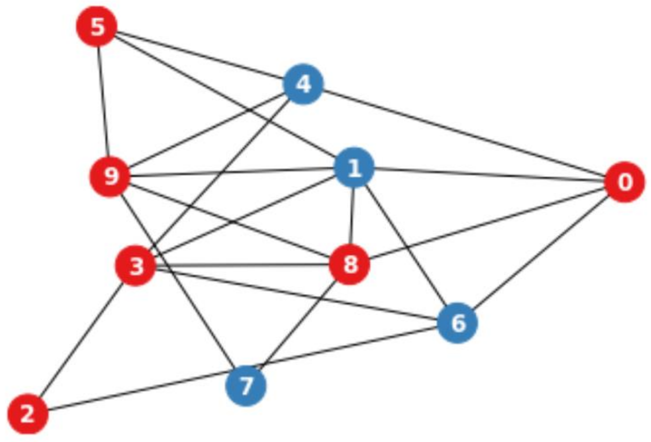


Quantum Approximate Optimization Algorithm: AWS Braket Excercise

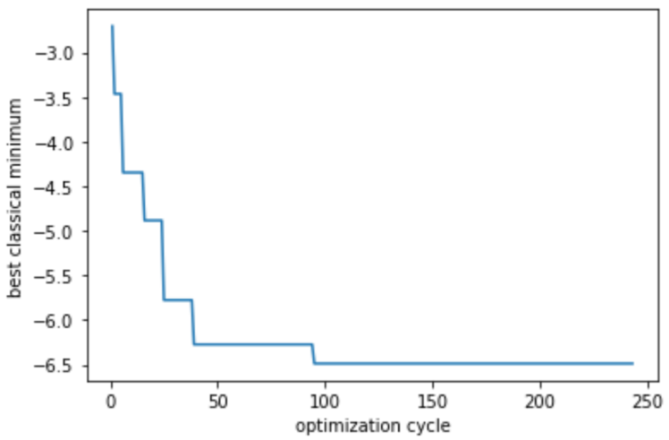
```
Circuit depth hyperparameter: 3
Problem size: 10
Starting the training.
====================================================================
OPTIMIZATION for circuit depth p=3
Param "verbose" set to False. Will not print intermediate steps.
====================================================================
Optimization terminated successfully.
         Current function value: -0.937169
         Iterations: 2
         Function evaluations: 243
Final average energy (cost): -0.937169150437591
Final angles: [4.88261413 2.60870766 2.26309012 1.6607821  1.28592103 2.39641137]
Training complete.
Code execution time [sec]: 4.882274389266968
Optimal energy: -6.486032631497276
Optimal classical bitstring: [-1  1 -1 -1  1 -1  1  1 -1 -1]
```

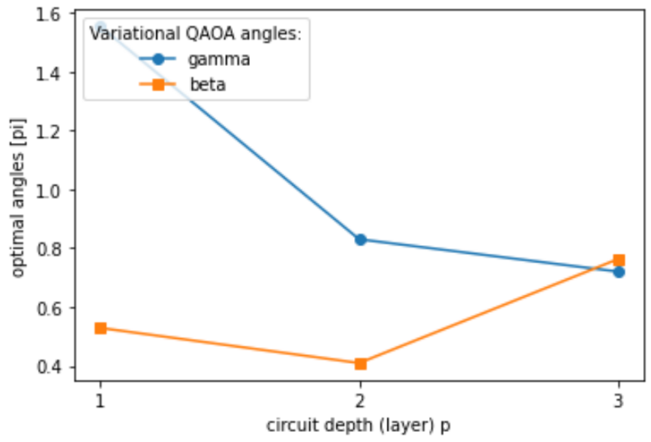
Training of $p = 3$ circuit using either quantum simulator or actual quantum hardware

Minimal energy found with QAOA: -6.486032631497276



Optimization on graph with n=10 vertices, m=20 edges, optimized with Powell and 10 shots per call; seed=42.



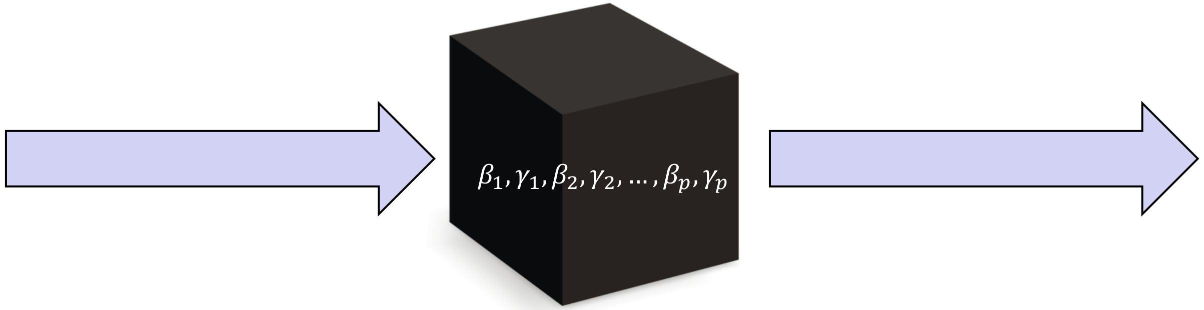


Electrical & Computer ENGINEERING

TEPPER

USRA Universities Space Research Association

Stochastic Optimization, Parameters and Statistics of Ising Solvers



More work to do, think how to improve

NO

Do you beat some good classical methods?

YES

Congrats, you achieved **limited quantum speedup**

Do you beat **all** good classical methods?

YES

Congrats, you achieved **quantum advantage**

Is your performance **so much better** that it would be crazy do to it without a QC?

YES

Congrats, you achieved **quantum supremacy**

1. Set up the quantum algorithm on the QPU with some initial parameters
2. Run it a number of times and process the performance collecting the statistics of distribution
3. If performance is not acceptable, use the distribution to choose new parameters (might involve processing)
   → Repeat 1-3 until satisfaction
4. Process final result and measure resource used (time, energy)
   → Repeat 1-4 for many benchmarking instances and collect distribution of performance.
5. Compare against best classical method on available hardware (time, energy)

Quantum and Quantum-Inspired Ising Solvers Examples

1. Quantum Annealing: D-Wave Systems
2. Coherent Ising Machines and Bifurcation Machines

The Quantum Adiabatic Algorithm for Ising Machines

(1) Map objective function into energy of a quantum Ising system

$$H_p = \sum_{ij} J_{ij} Z_i \otimes Z_j + \sum_i h_i Z_i$$

(2) Start from easy problem to solve with known solution

$$H_D = \Gamma \sum_i X_i \quad \text{(transverse field)}$$

$$|\Psi\rangle_{Nqubits} = \frac{1}{\sqrt{2^N}} \sum_{n=1}^{2^N} |solution(n)\rangle$$

(3) Do any Schrödinger evolution (no measurement! No noise!) that changes the energy states «sufficiently slow».

How slow? It depends on the problem, on H_D and on the <u>Annealing Schedule</u>
No way to predict efficiently. Try!

$$H|s_1 s_2 \ldots s_N\rangle = EN|s_1 s_2 \ldots s_N\rangle$$

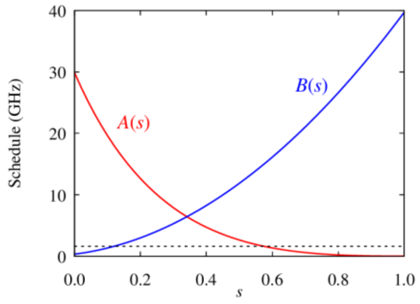
$$\exp(iH)\,|s_1 s_2 \ldots s_N\rangle = e^{iE_N}|s_1 s_2 \ldots s_N\rangle$$

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

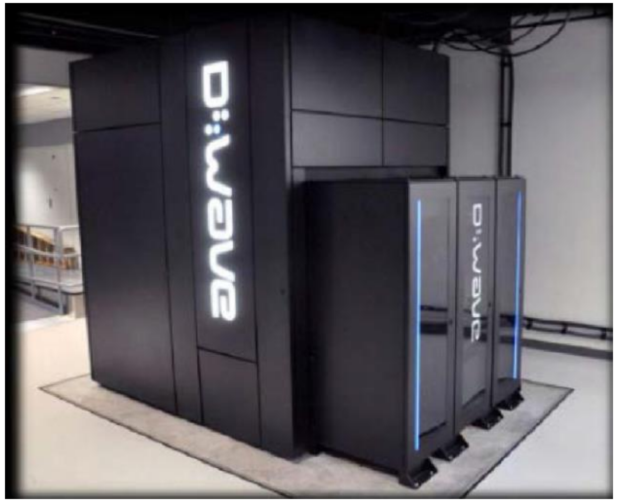
$$R_x(\pi/2)|0\rangle = |1\rangle$$
$$R_x(\pi/2)|1\rangle = |0\rangle$$

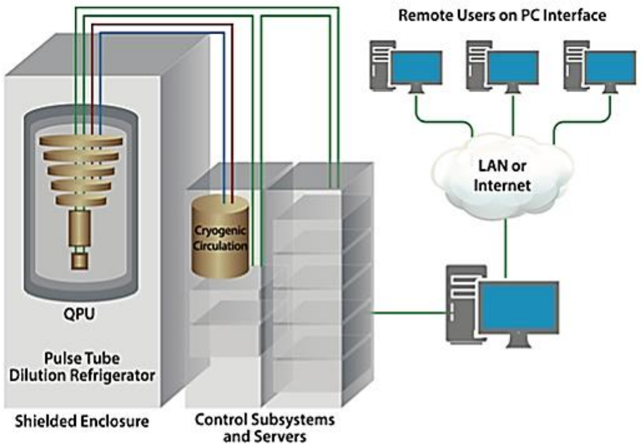
If this field is always on and constant the minimum energy state is the **all-superposed state**

$H = A(t)\,H_D + B(t)\,H_P$



Quantum annealing à la D-Wave

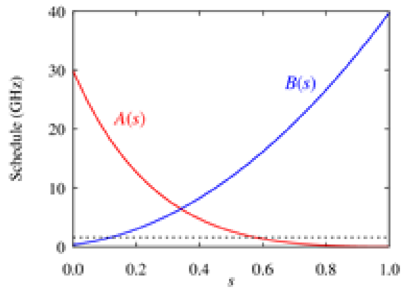


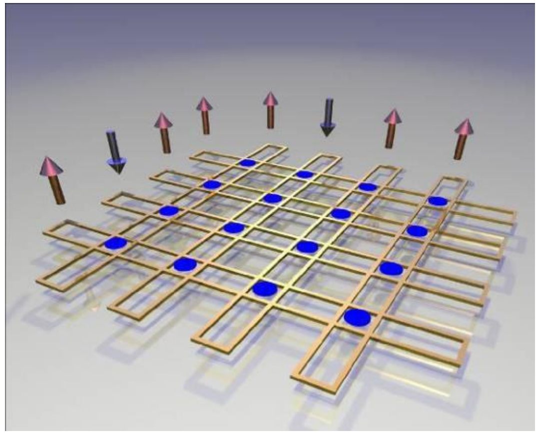


J_{ij} and h_i have maximum value and fluctuating intrinsic control errors:

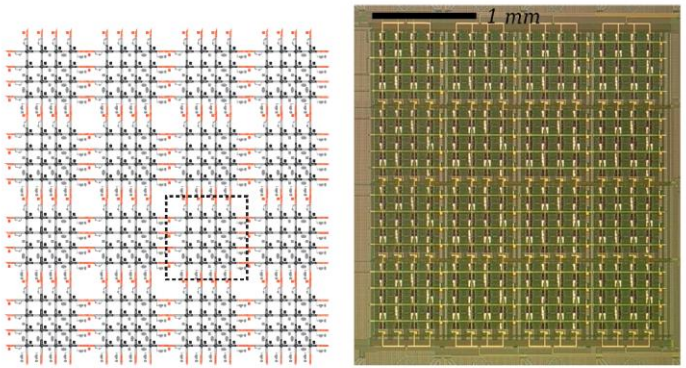
$$J_{ij} + \delta J$$

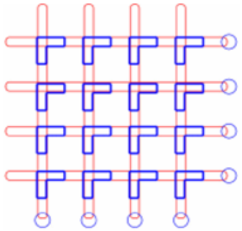
$$h_i + \delta h$$

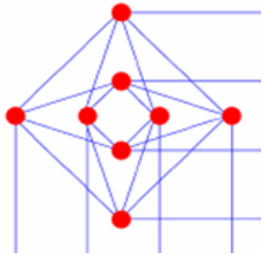


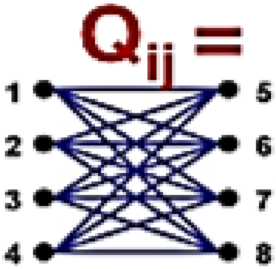


A 128-qubit chip composed of a 4 × 4 array of eight-qubit unit cells.

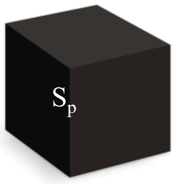


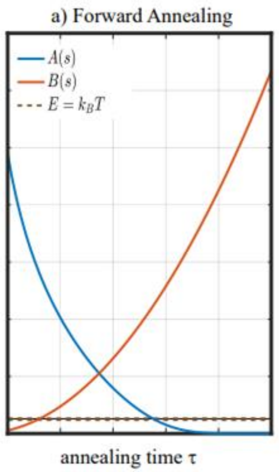




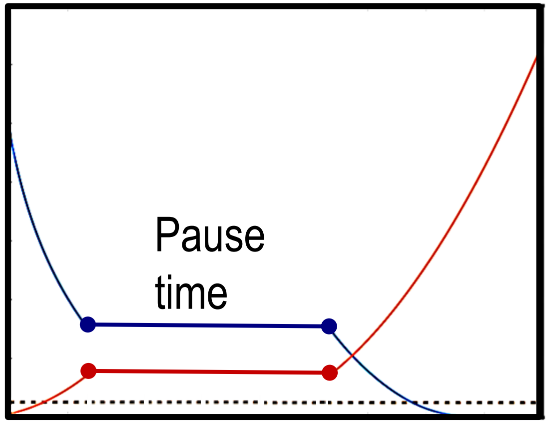


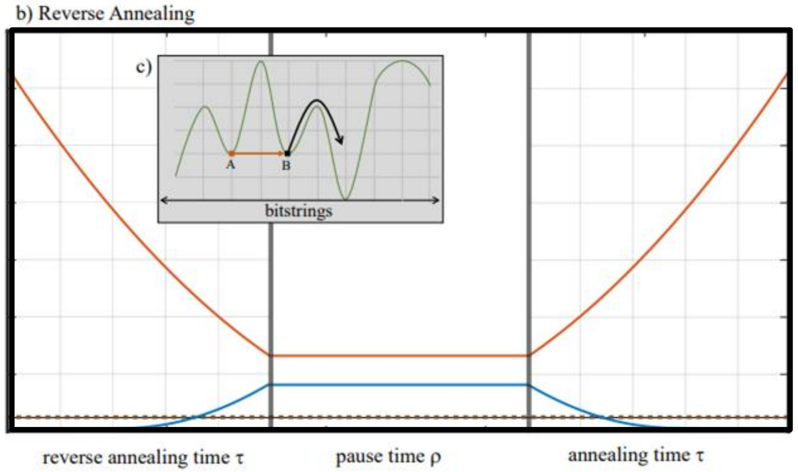
Annealing Schedule Parameters





Time for annealing
(if AQC controls performance)





First pause time

reversal time

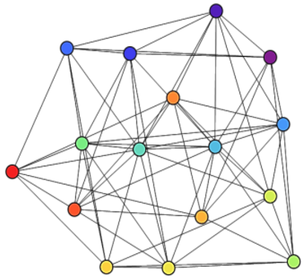
These are all parameters that influence performance.

Only for elegant problems they can be derived ab-initio. In the real world you have some physics guidance for best guess then you use a heuristics to find them

Minor Embedding

Topological Embedding

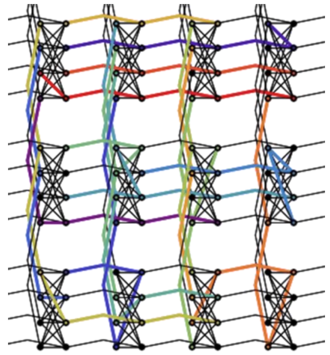
(n_H hardware qubits)

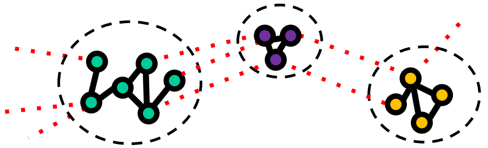


$$\varepsilon(i)\colon \{1,\dots,n_L\} \rightarrow 2^{\{1,\dots,n_P\}}$$

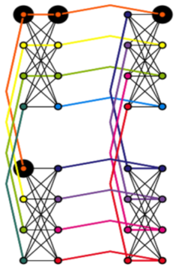
Assign "colors" to connected sets of qubits

(n_P logical bits)

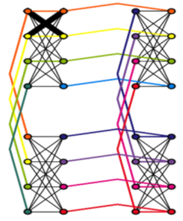




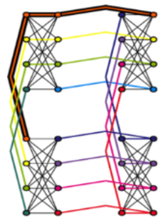
Parameter Setting



$$\sum_{j\in\varepsilon(i)} h'_j = h_i$$

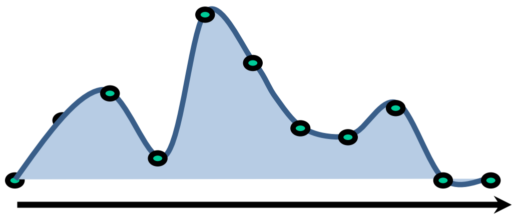


$$\sum_{j_1\in\varepsilon(i_1)}\sum_{j_2\in\varepsilon(i_2)} J'_{j_1j_2} = J_{i_1i_2}$$

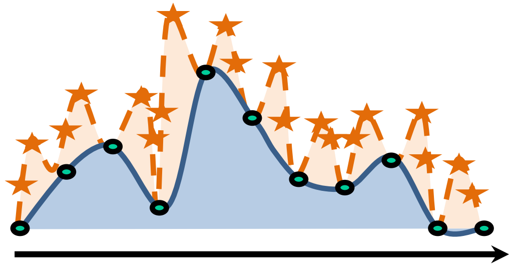


$$J'_{j_1j_2} < |h_i| - \sum_{k=1}^{n} |J_{ij}|$$

Energy Landscape Before embedding

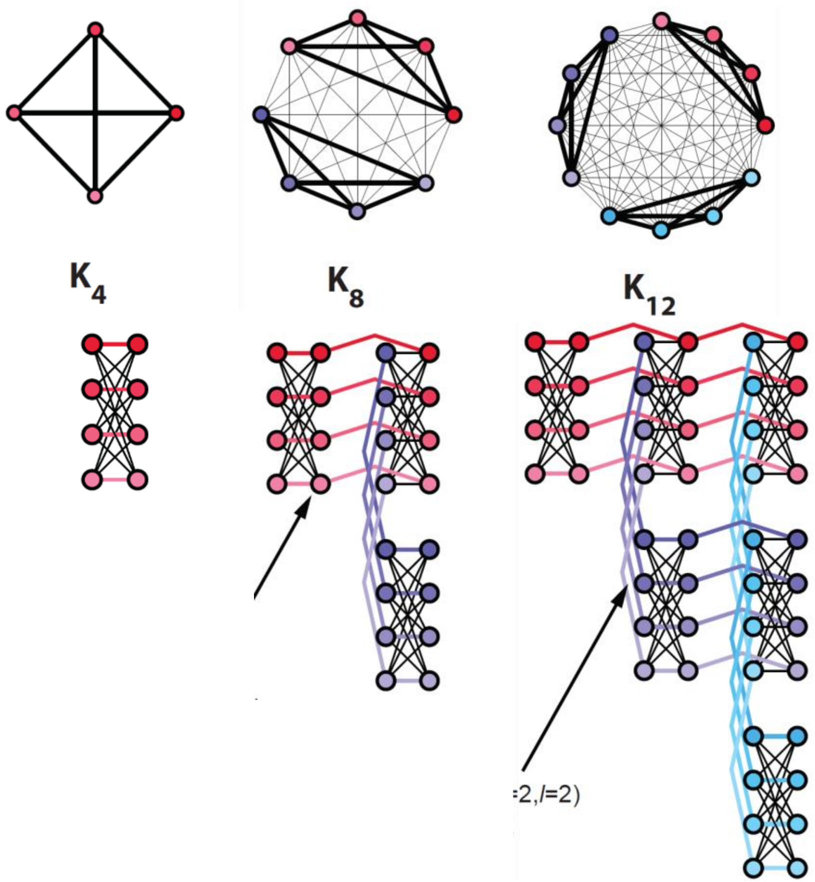


Energy Landscape After embedding

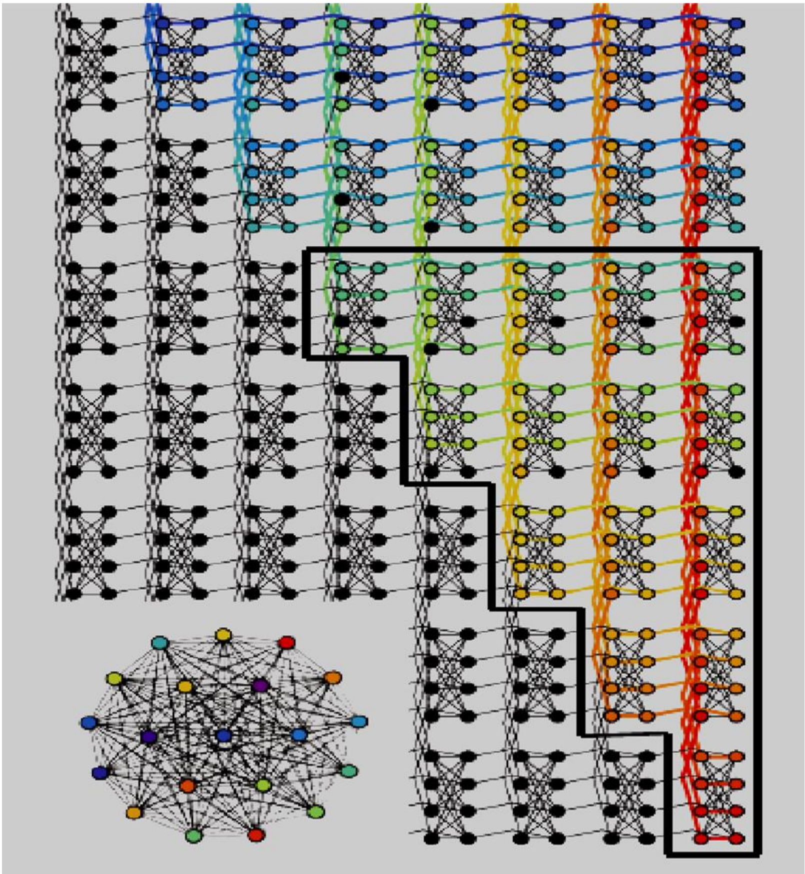


Minor Embedding of a fully connected graph

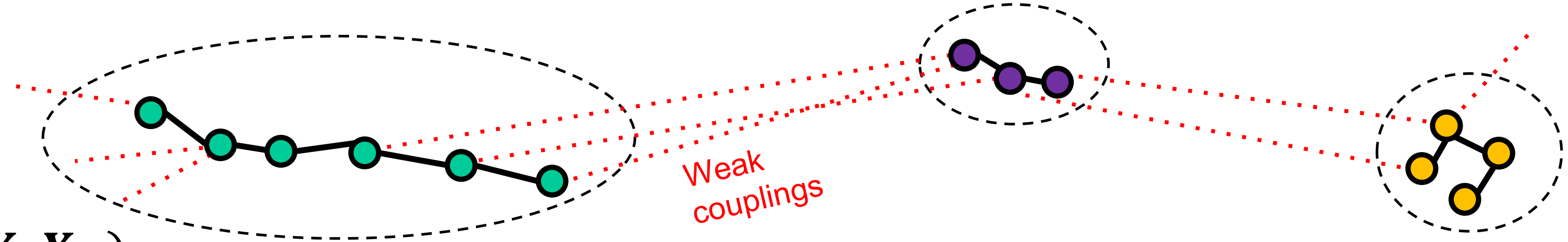
Systematic Rule for Embedding



Quadratic overhead



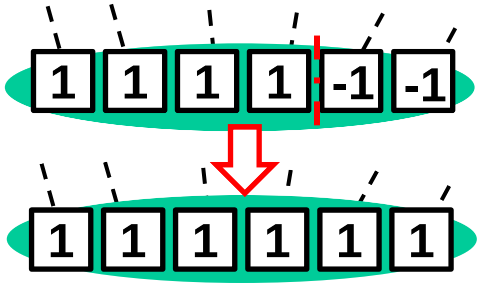
Unembedding



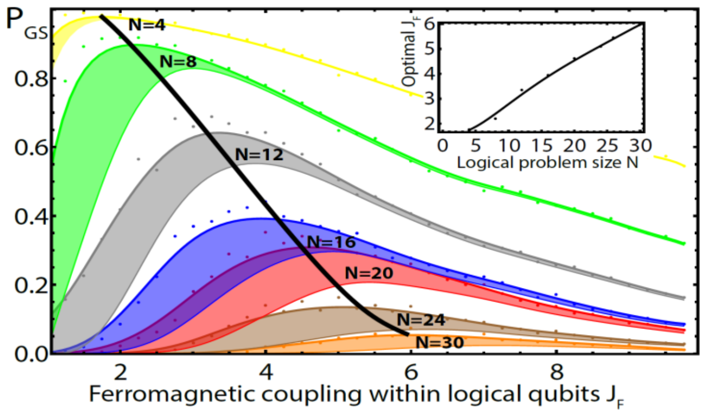
Ferromagnetic Coupling

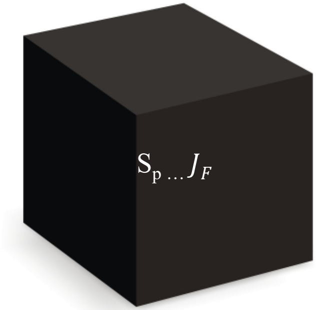
$$f(S_1, S_2) = -JF\, s_1\, s_2$$
$$f(X_1, X_2) = -4JF(-X_1 - X_2 + X_1X_2)$$

Majority Voting



What is the correct J_F ?





Not too large, not too small. Trial and error.
(See Venturelli et al. https://journals.aps.org/prx/abstract/10.1103/PhysRevX.5.031040)

Check the AWS Exercises

Let's go to the AWS Braket https://console.aws.amazon.com/braket Quantum Annealing tutorials from Quiz III

quantum_annealing/Dwave_Anatomy.ipynb

(also found in GitHub) but if you want to execute it open it in AWS.

- Great background information on
  - Quantum Annealing
  - Embedding
  - Ising Model / MAXCUT / QUBO

Anatomy of quantum annealing with D-Wave on Amazon Braket

This tutorial notebook dives deep into the **anatomy of quantum annealing** with D-Wave on Amazon Braket.

First, the concept of quantum annealing as used by D-Wave is introduced to show how it probabilistically finds the (approximate) optimum to some optimization problem.

The next section introduces the structures of the D-Wave QPUs and the concept of **embedding**. Amazon Braket provides two D-Wave devices, 2000Q and Advantage. The 2000Q device has the **Chimera** topology, while the Advantage device has the **Pegasus** topology. Running a problem on a particular D-Wave device requires to map the original source graph onto the targe graph. This mapping is called embedding.

Finally, an example QUBO problem is solved using both the classical annealers and QPU to demonstrate the sampling process and a breakdown of the QPU access time.

Background: quantum annealing

Introduction: On a high level, quantum annealing (QA) is a specific approach to quantum computing, as opposed to the common gate-based model. Quantum annealers are specific-purpose machines designed to solve certain problems belonging to the class of Quadratic Unconstrained Optimization (QUBO) problems. The QUBO model unifies a rich variety of NP-hard combinatorial optimization problems, such as Quadratic Assignment Problems, Capital Budgeting Problems, Task Allocation Problems and Maximum-Cut Problems, just to name a few [1]. Since quantum annealers do not have to meet the strict engineering requirements that universal gate-based machines have to meet, already today this technology features ~ 5000 superconducting qubits, compared to less than 100 qubits on gate-model quantum computers. Amazon Braket offers access to the superconducting quantum annealers provided by D-Wave Systems that can be programmed using the high-level, open source tool suite called `Ocean`.

Adiabatic quantum computing: The paradigm of QA is closely related to *adiabatic quantum computing* (with only subtle differences discussed later) [2]. In essence, adiabatic quantum computing makes use of an adiabatic process where parameters are changed sufficiently slow for the system to adapt to the new parameter configuration quasi-instantaneously. For example, in a quantum mechanical system, some Hamiltonian starts from H_0 and slowly changes to some other Hamiltonian H_1, with (for example) a linear ramp (also called schedule):

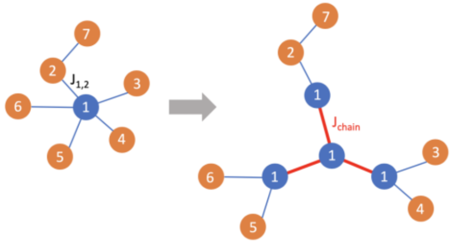
$$H(t) = (1\text{Unknown character}\text{Unknown character}\text{Unknown character}t)H_0 + tH_1,$$

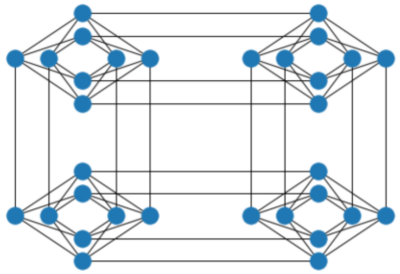
where $t \in [0, 1]$ on some time scale. Accordingly, the system's final dynamics at $t = 1$ is governed by H_1 while initially it is determined by H_0. If the change in the time-dependent Hamiltonian $H(t)$ is sufficiently slow, the resulting dynamics are very simple (according to the adiabatic theorem): if the system starts out in an eigenstate of H_0, the system remains in an instantaneous eigenstate throughout the evolution. Specifically, if the system started in the ground state (the eigenstate with minimal energy), the system stays in the ground state, if the condition of adiabaticity (that is related to energy difference between the ground state and the first excited state called the **gap**) is satisfied. This means, to solve a (unknown) ground state of a Hamiltonian for a (hard-to-solve) problem, one can start from an easy-to-solve Hamiltonian with a known ground state.

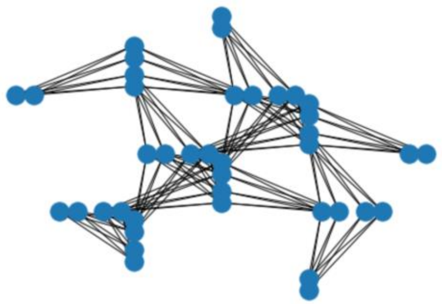
Quantum Annealing: In practice, it is very difficult to fulfill the adiabaticity conditions, because of undesired noise. Therefore, quantum annealers forego the stringent, theoretical adiabaticity conditions and heuristically repeat the annealing procedure many times, thereby collecting a number of samples from which the configuration with the lowest energy can be selected as the optimal solution. However, there is no strict guarantee to find the true ground state.

Typically the Hamiltonian H_0 has the form: $H_0 = \sum_i \sigma_i^x$; with well-known ground state (the equal superposition of all bitstrings). And H_1 is described by the canonical Ising model

$$H_{\text{Ising}} = \sum_{i,\,j} J_{i,\,j}\sigma_i^z\sigma_j^z.$$







Other exercises on DWave

In AWS Braket https://console.aws.amazon.com/braket there are other interesting Quantum Annealing tutorials

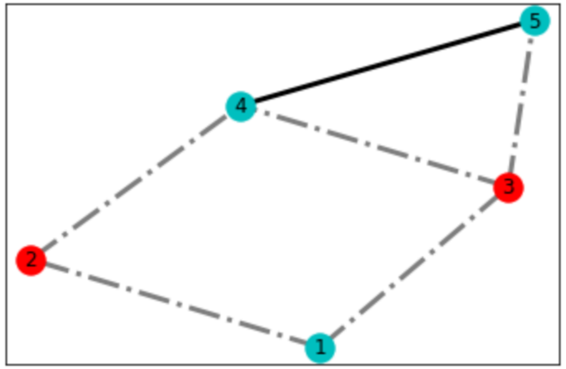
quantum_annealing/*

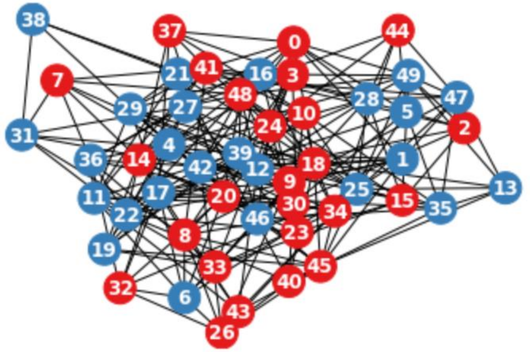
(also found in [GitHub](#)) but if you want to execute it open it in AWS.

- Classical combinatorial problems: MAXCUT, Graph partitioning, Min vertex cover, Traveling Salesman Person
- Other cases: Factoring, Structural Imbalance

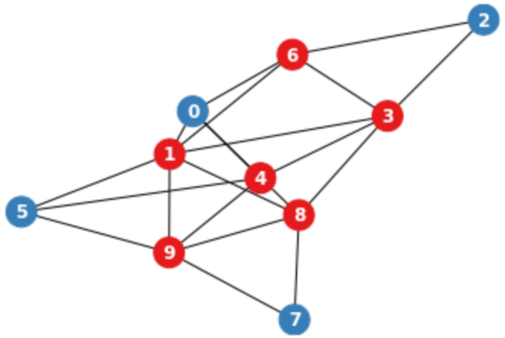
Check (and run) them all

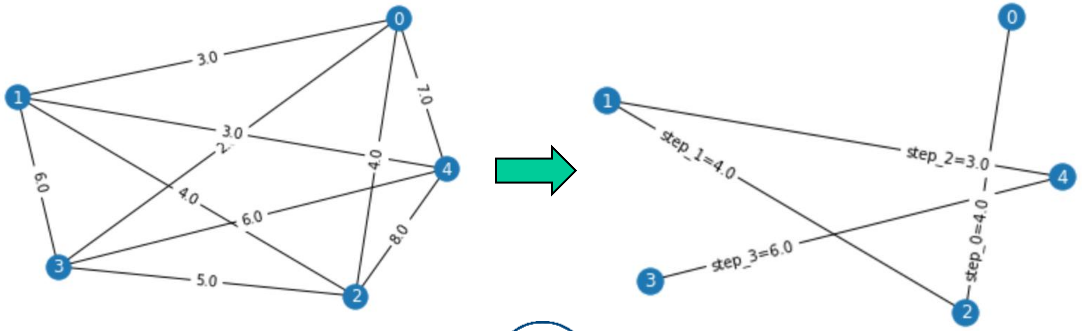
```
Your plot is saved to maxcut_plot.png
```





```
Result to MVC problem: [1, 3, 4, 6, 8, 9]
Size of the vertex cover: 6
```



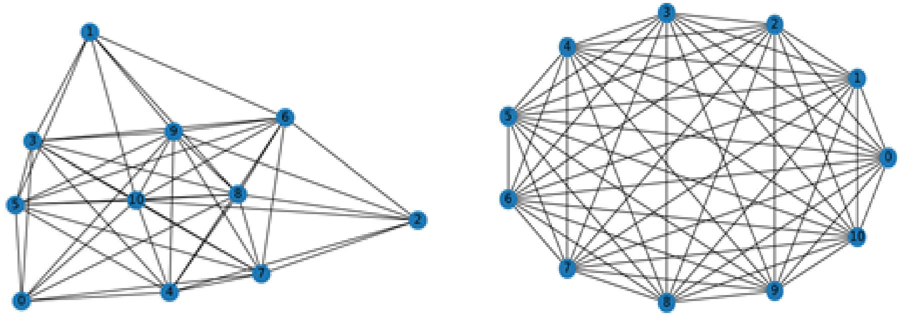


Minor Embedding - Example

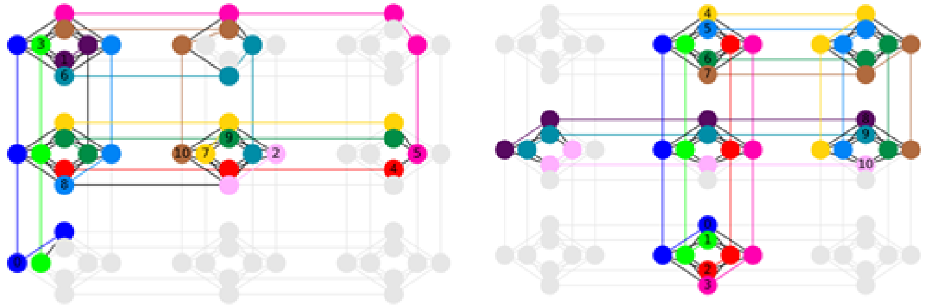
https://colab.research.google.com/github/bernalde/QuIPML/blob/master/notebooks/Notebook%207%20-%20DWave.ipynb

From our main example

$$
\begin{bmatrix}
-46. & 0. & 0. & 48. & 48. & 48. & 0. & 48. & 48. & 48. & 48. \\
0. & -44. & 0. & 48. & 0. & 48. & 48. & 0. & 48. & 48. & 48. \\
0. & 0. & -44. & 0. & 48. & 0. & 48. & 48. & 48. & 48. & 48. \\
48. & 48. & 0. & -92. & 48. & 96. & 48. & 48. & 96. & 96. & 96. \\
48. & 0. & 48. & 48. & -92. & 48. & 48. & 96. & 96. & 96. & 96. \\
48. & 48. & 0. & 96. & 48. & -92. & 48. & 48. & 96. & 96. & 96. \\
0. & 48. & 48. & 48. & 48. & 48. & -91. & 48. & 96. & 96. & 96. \\
48. & 0. & 48. & 48. & 96. & 48. & 48. & -92. & 96. & 96. & 96. \\
48. & 48. & 48. & 96. & 96. & 96. & 96. & 96. & -139. & 144. & 144. \\
48. & 48. & 48. & 96. & 96. & 96. & 96. & 96. & 144. & -138. & 144. \\
48. & 48. & 48. & 96. & 96. & 96. & 96. & 96. & 144. & 144. & -139.
\end{bmatrix}
$$



And embed it into a Chimera graph (subgraph of the Chip)



Best embedding in 1000 heuristic runs

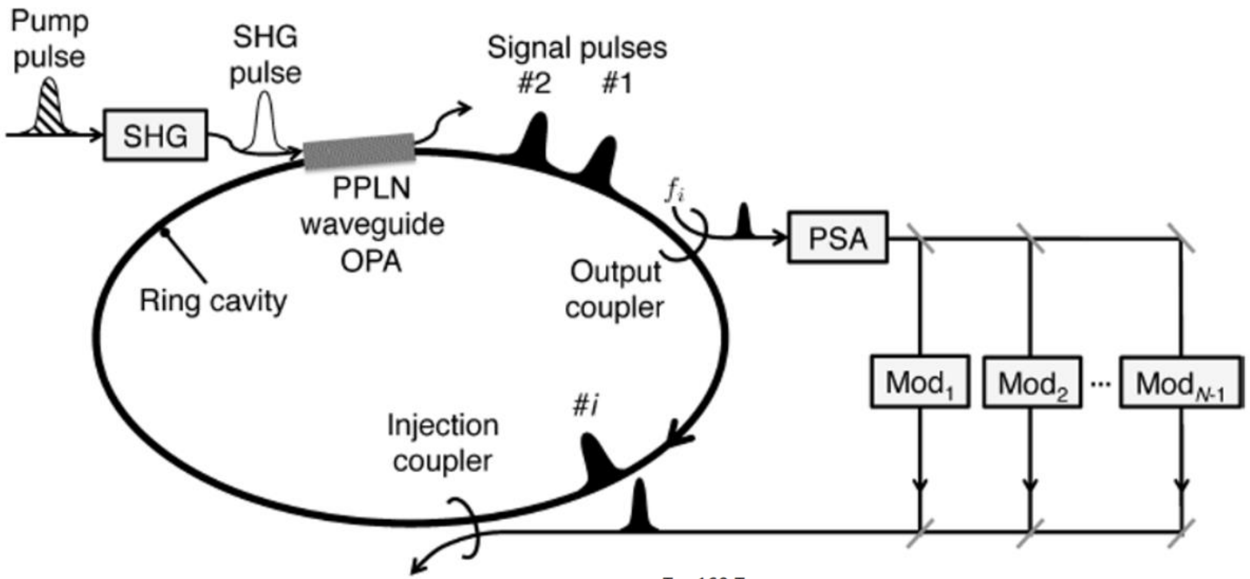
Full graph embedding

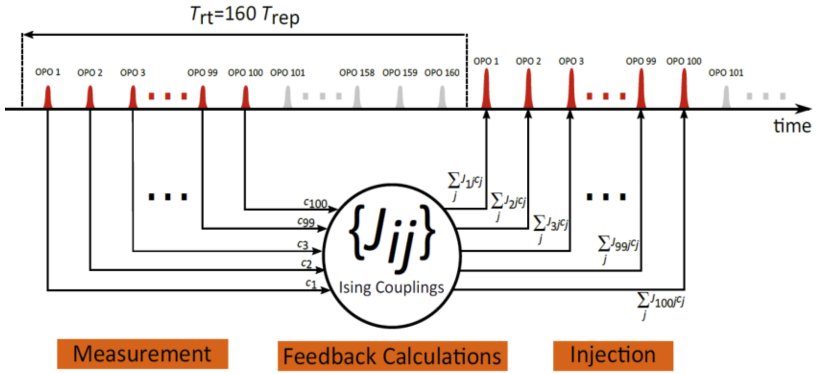
Notice that we need to "duplicate" certain variables into several qubits

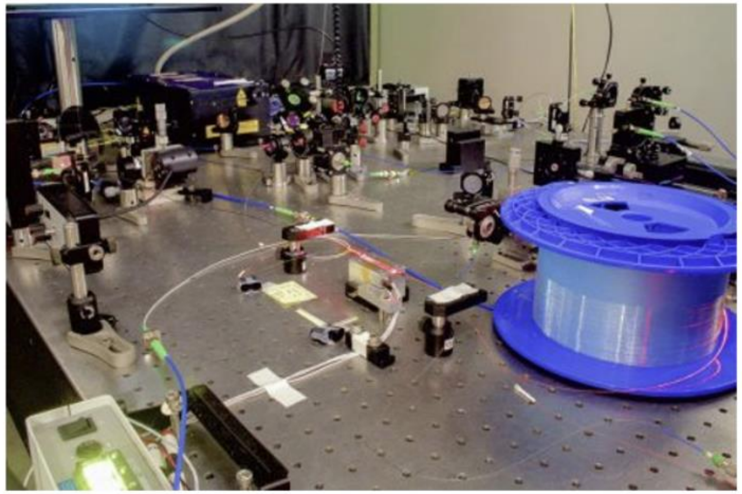
This step is non-trivial:

Either use heuristic methods or solve highly constrained problem

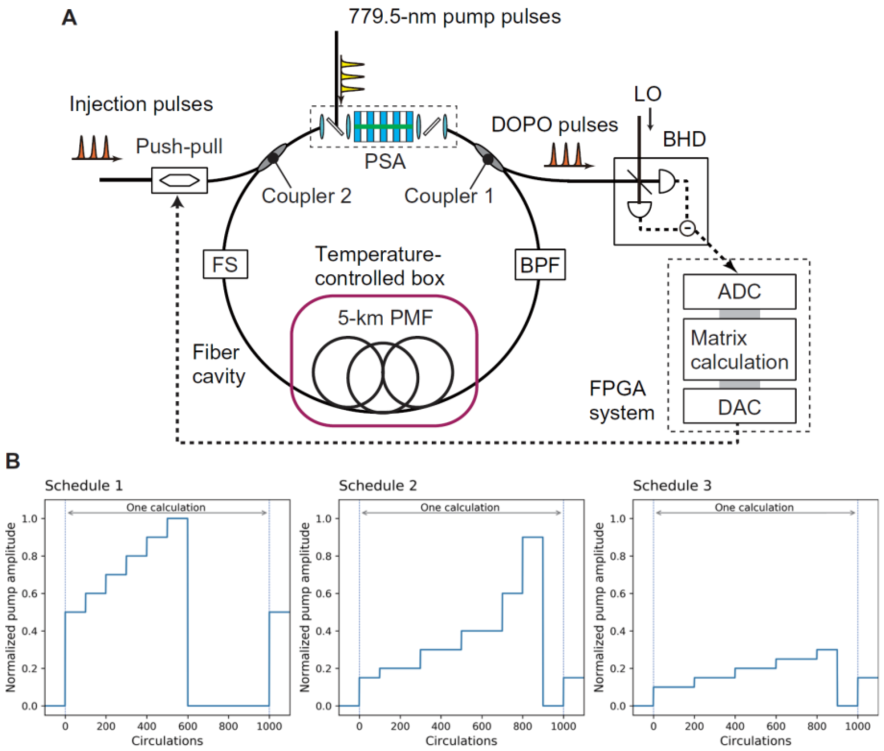
Coherent Ising Machines







NEWS: 100,000 Spins



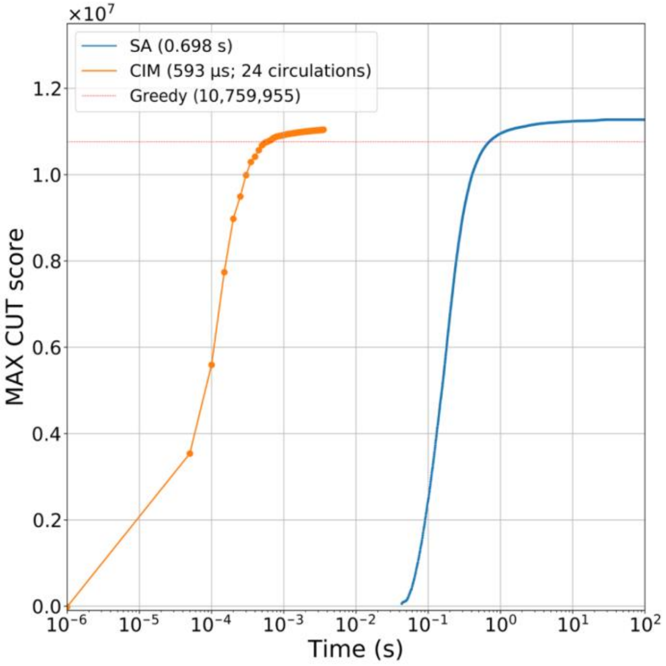


Fig. 3. MAX CUT score as a function of computation time obtained with the CIM (orange line) and SA (blue line). The data points exhibit the scores evaluated at the intermediate steps in the CIM and SA computation. The dotted line denotes the score obtained with SG (10,759,955).

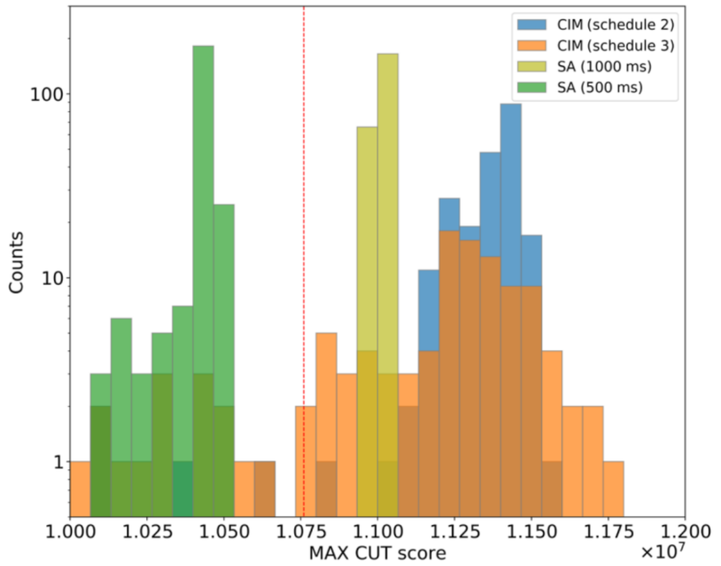


Fig. 6. Histograms of MAX CUT score with CIM and SA. The vertical dashed line shows the SG score (10,759,955).

SCIENCE ADVANCES | RESEARCH ARTICLE

COMPUTER SCIENCE

100,000-spin coherent Ising machine

Toshimori Honjo[1]*, Tomohiro Sonobe[2], Kensuke Inaba[1], Takahiro Inagaki[1], Takuya Ikuta[1], Yasuhiro Yamada[1], Takushi Kazama[3], Koji Enbutsu[3], Takeshi Umeki[3], Ryoichi Kasahara[3], Ken-ichi Kawarabayashi[2], Hiroki Takesue[1]*

Coherent Ising Machines: Stochastic Differential Equations

Describing the system with zero quantum noise, and neglecting the out-of-phase component of the signal

$$\boxed{\dot{x}_i = (p-1)x_i - x_i^3 + \epsilon \sum_j J_{ij} x_j}$$

$$(\dot{x}_i = \partial V / \partial x_j)$$

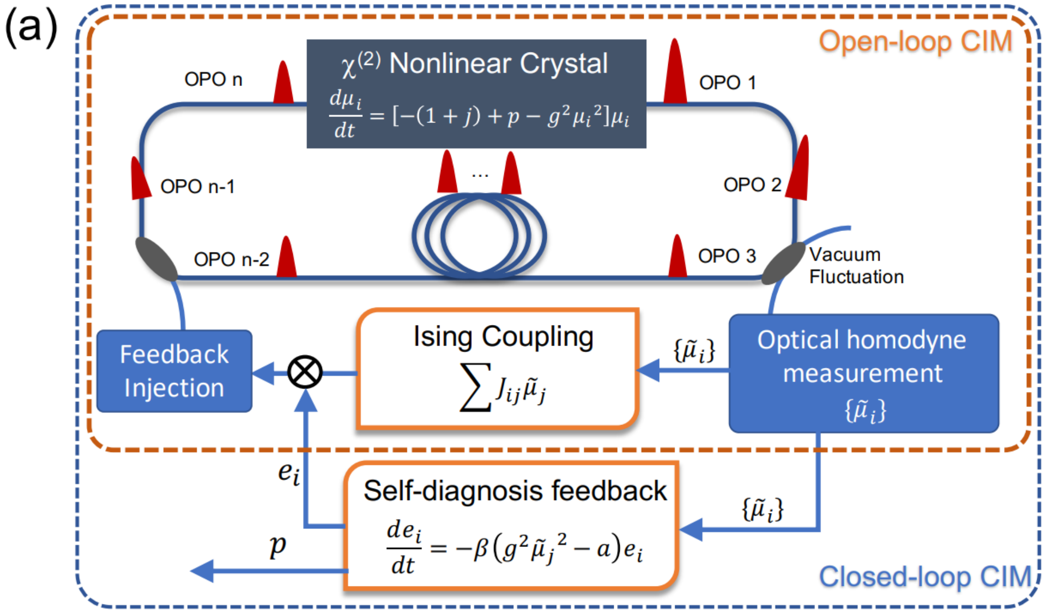
$$V := \sum_i \left(\frac{(p-1)x_i^2}{2} + \frac{x_i^4}{4} \right) - \epsilon \sum_{ij} J_{ij} x_i x_j$$

$s_i = sign(x_i)$ is the bit variable

Chaotic-Amplitude-Control (CAC) *or* Amplitude Heterogeinity Correction *or* Error-variable Feedback

$$\dot{x}_i = (p-1)x_i - x_i^3 + e_i \epsilon \textstyle\sum_j J_{ij} x_j,$$
$$\dot{e}_i = -\beta\left(x_i^2 - a\right)e_i,$$

Various architectures



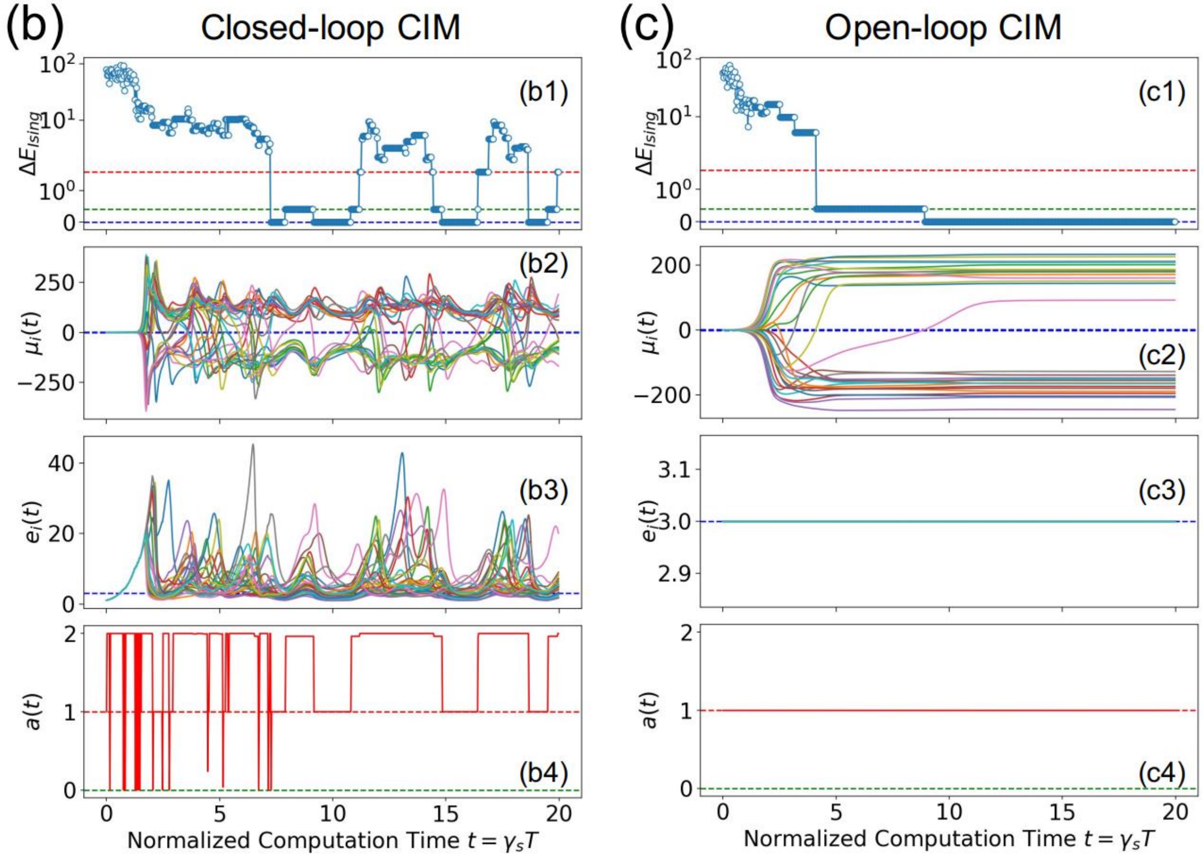
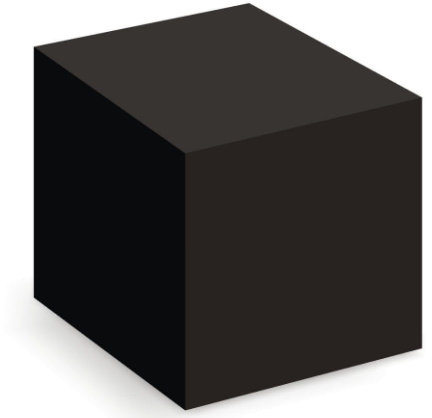


FIG. 1: (a) Schematic diagram of the measurement-feedback coupling CIMs with and without the self-diagnosis and dynamic feedback control (closed-loop and open-loop CIMs) indicated using dashed blue and orange lines, respectively. (b) and (c) Dynamical behaviour of the closed-loop and open-loop CIMs, respectively. (b1) and (c1) Inferred Ising energy (the dashed horizontal lines are the lowest three Ising eigen-energies). (b2) and (c2) Mean-field amplitude $\mu_i(t)$. (b3) and (c3) Feedback-field amplitude $e_i(t)$. (b4) Target squared amplitude $a(t)$. (c4) Pump rate $p(t)$.

https://arxiv.org/abs/2105.03528

Coherent Ising Machines: Stochastic Differential Equations



Chaotic-Amplitude-Control (CAC)
or Amplitude Heterogeinity Correction
or Error-variable Feedback

$$\dot{x}_i = (p-1)x_i - x_i^3 + e_i\epsilon \textstyle\sum_j J_{ij}x_j,$$

$$\dot{e}_i = -\beta\left(x_i^2 - a\right)e_i,$$

In our simulation, the x_i variables are restricted to the range $[-\frac{3}{2}\sqrt{\alpha}, \frac{3}{2}\sqrt{\alpha}]$ at each time step. The parameters p and α are modulated linearly from their starting to ending values during the T_r time steps and are kept at the final value for an additional T_p time steps. The initial value x_i is set to a random value chosen from a zero-mean Gaussian distribution with a standard deviation of 10^{-4} and $e_i = 1$. Furthermore, 3200 trajectories are computed per instance to evaluate TTS. The actual parameters used for simulation are listed below:

N step	3200
ΔT	0.125
T_r	2880
T_p	320
p	-1.0 $\rightarrow$ 1.0
α	1.0 $\rightarrow$ 2.5
β	0.8

See for instance:

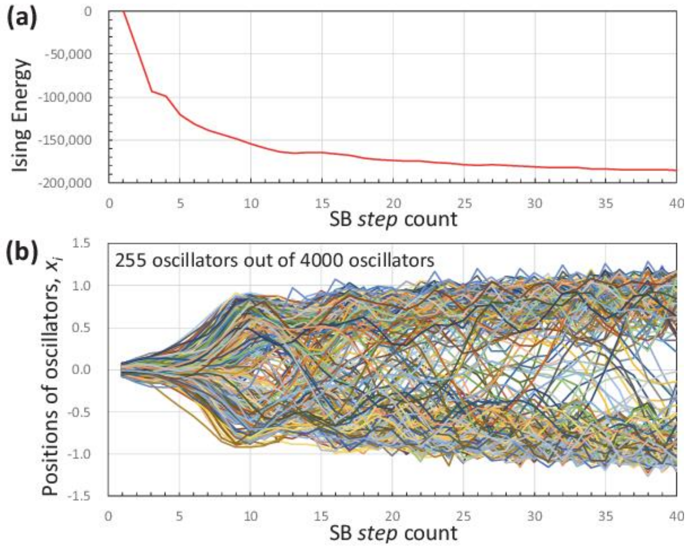
https://arxiv.org/pdf/2108.07369.pdf

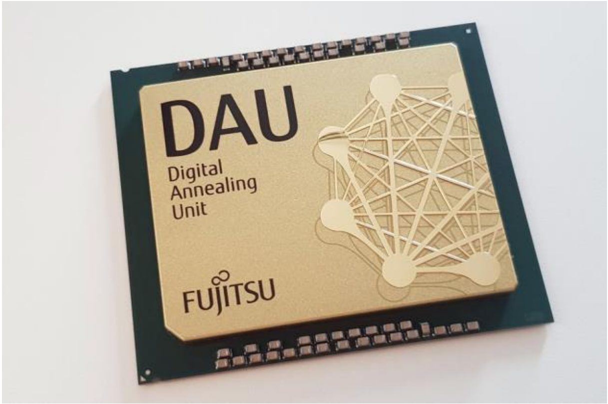
Oscillation Based Machines, Bifurcation Machines, MemComputers etc.

The Fujitsu Digital Annealer

The Kuramoto Model describes synchronization of oscillators

The Toshiba Simulated Bifurcation Machine, Memcomputing





https://ieeexplore.ieee.org/document/8892209

Time-to-Solution and variants

- Fix parameters, run many times (num$_{\text{trials}}$) – estimate cost $PDF\ f(x_i)$
- Define a success test: OK iff $cost\ <\ target$
- Estimation of success probability $P_{success}\ =\ num_{OK}\ /\ num_{trials}\ =\ \int_{x<x_{OK}} dxf(x)\ =\ CDF(x_{OK})$
- Probability of succeeding at least once in R attempts: $\mathcal{P}(R)\ =\ 1-(1-Psuccess)^R$
- Invert to find R required to achieve success with at least probability $\mathcal{P}$

$$\langle \boldsymbol{TTS} \rangle\ =\ \log(\mathbf{1}-\mathcal{P})/\log(\mathbf{1}-\boldsymbol{Psuccess})$$

Academic Benchmarking
Standard: **median⟨TTS⟩(N)**


REPORT
Defining and detecting quantum speedup
Troels F. Rønnow[1], Zhihui Wang[2,3], Joshua Job[3,4], Sergio Boixo[5,6], Sergei V. Isakov[7], David Wecker[8], John M. Martinis[9], Dan...
+ See all authors and affiliations
Science 25 Jul 2014:
Vol. 345, Issue 6195, pp. 420-424
DOI: 10.1126/science.1252319

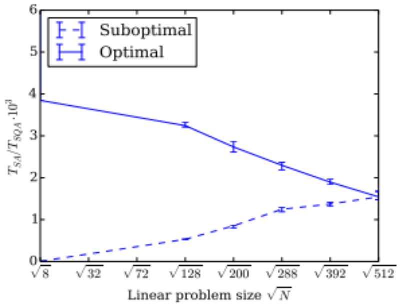
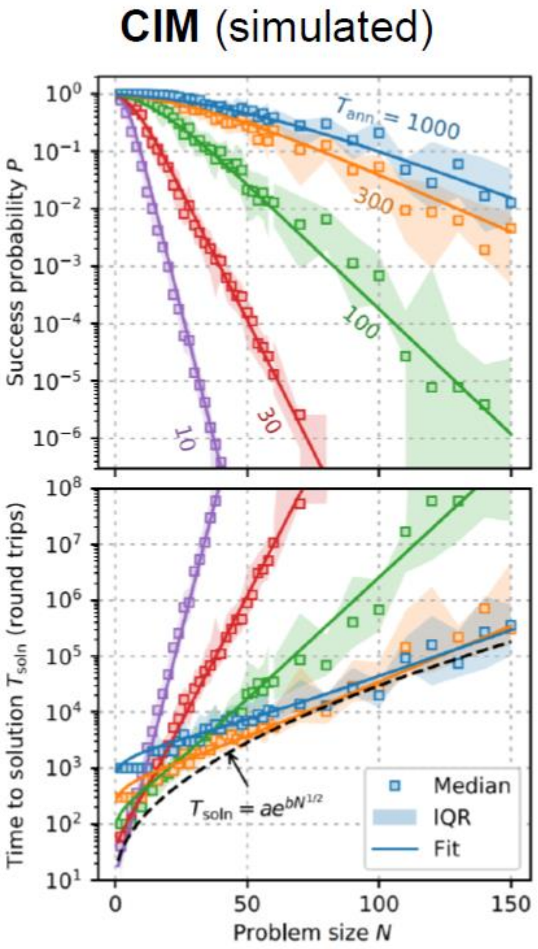


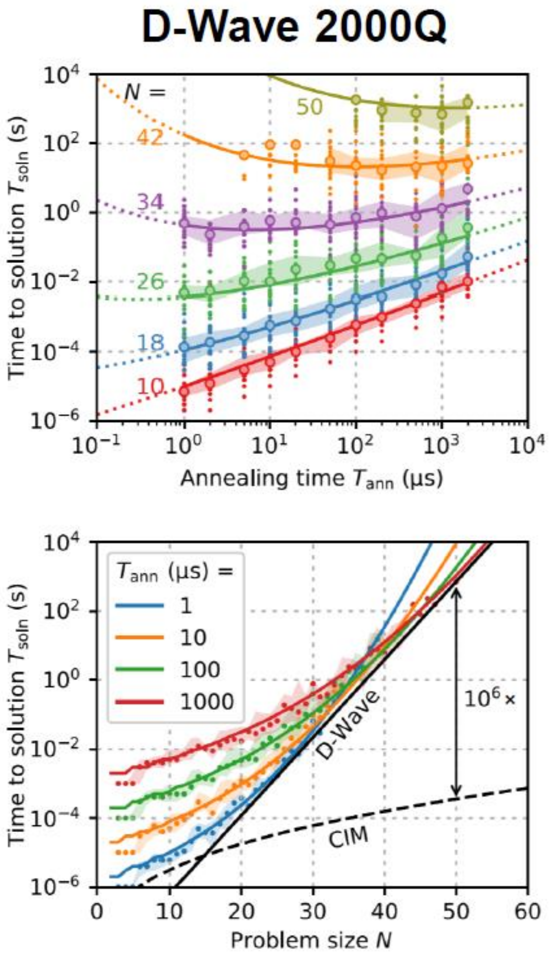
FIG. 2. **Pitfalls when detecting speedup.** Shown is the speedup of SQA over SA, defined as the ratio of median time to find a solution with 99% probability between SA and SQA. Two cases are presented: a) both SA and SQA run optimally (i.e., the ratio of the true scaling curves shown in Figure 1), and there is no asymptotic speedup (solid line). b) SQA is run suboptimally at small sizes by choosing a fixed large annealing time t_a = 10000 MCS (dashed line). The apparent speedup is, however, due to suboptimal performance on small sizes and not indicative of the true asymptotic behavior given by the solid line, which displays a slowdown of SQA compared to SA.

If the parameters are suboptimal you can be fooled into thinking the complexity decreases with problem size!

Example 1: CIM vs D-Wave (2020)

https://www.science.org/doi/10.1126/sciadv.aau0823





Example 2: Sim-CIM vs Adiabatic QAOA

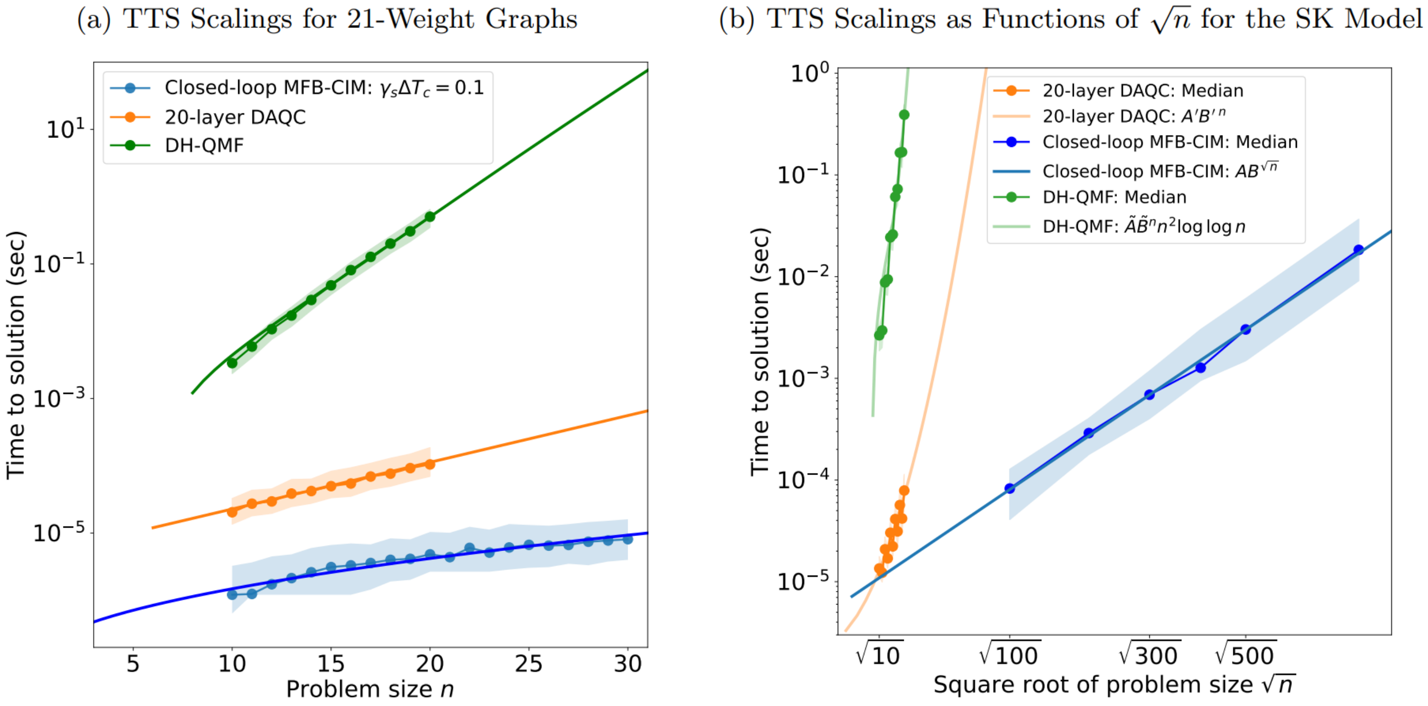
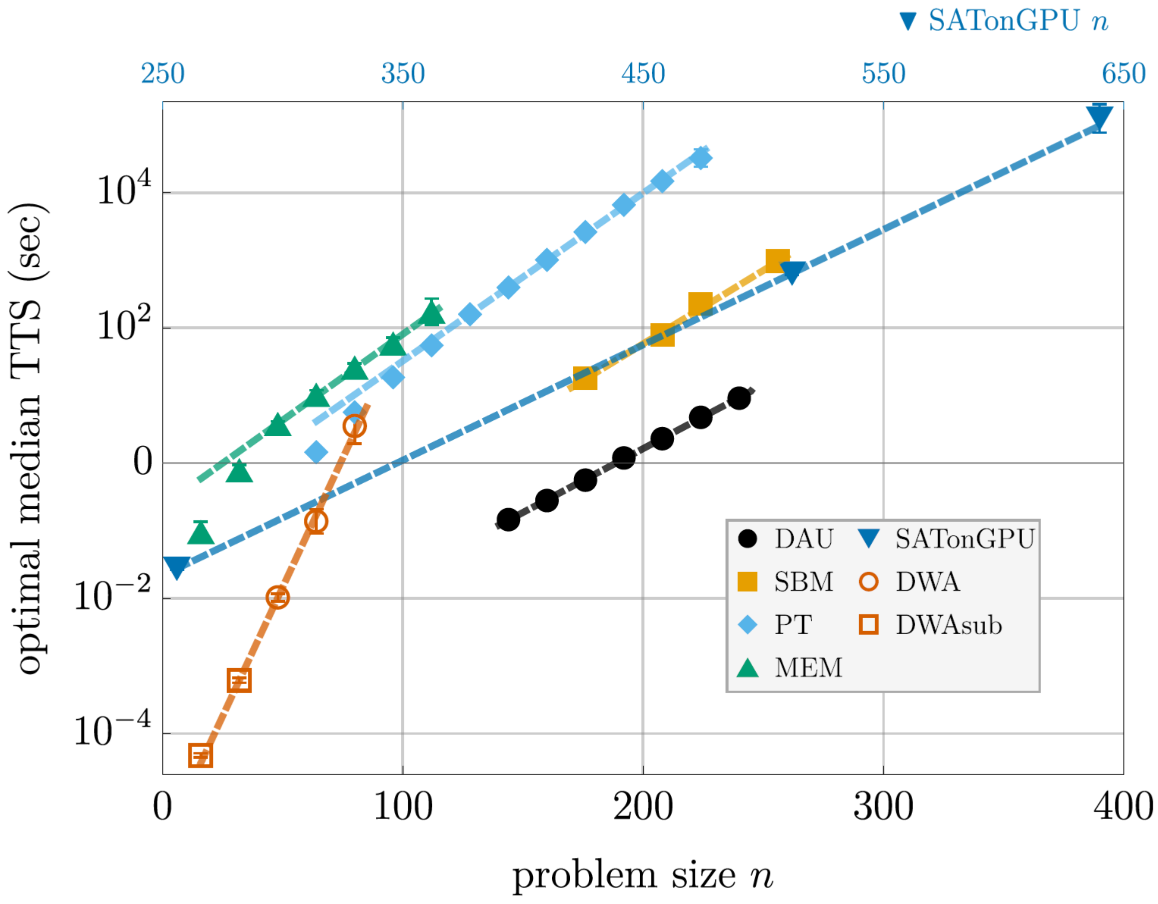


FIG. 14: Comparison of the time-to-solution (TTS) scalings for the MFB-CIM, DAQC, and DH-QMF in solving MaxCut. (a) Wall-clock time of a closed-loop CIM with a low-finesse cavity ($\gamma_s \Delta T_c = 0.1$), DAQC with an optimum number of layers ($p = 20$), and DH-QMF with an a priori known number of optimum iterations versus problem size n. (b) TTS of the closed-loop CIM on the fully connected SK model for problem sizes from $n = 100$ to $n = 800$, in steps of 100. For each problem size, the minimum TTS with respect to the optimization over $t_{\max}$ is plotted. In comparison, the SK model TTSs are shown for 20-layer DAQC and DH-QMF for problem sizes ranging from $n = 10$ to $n = 20$. The straight, lighter-blue line (a linear regression) for the CIM demonstrates a scaling according to $AB^{\sqrt{n}}$. The lighter-orange and lighter-green best-fit curves for DAQC and DH-QMF are extrapolated to larger problem instances, illustrating a scaling that is exponential in n rather than in $\sqrt{n}$. In both figures, the shaded regions show the IQRs.

https://arxiv.org/abs/2105.03528

Example 3: D-Wave vs Digital Annealer and Parallel Tempering



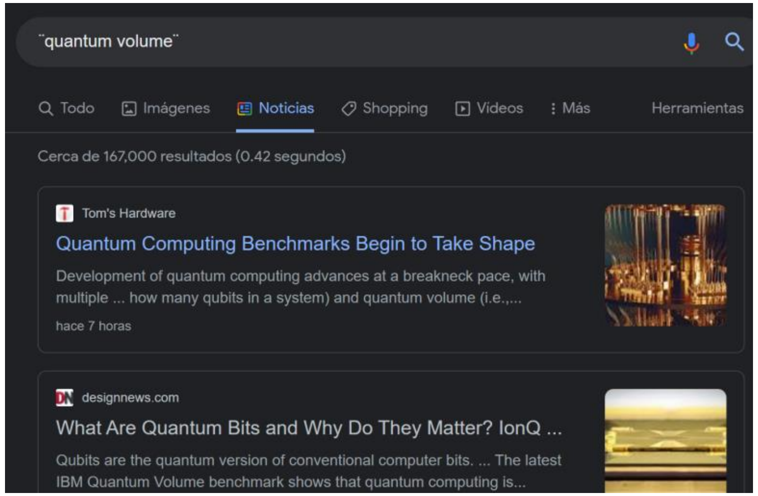
https://arxiv.org/abs/2103.08464

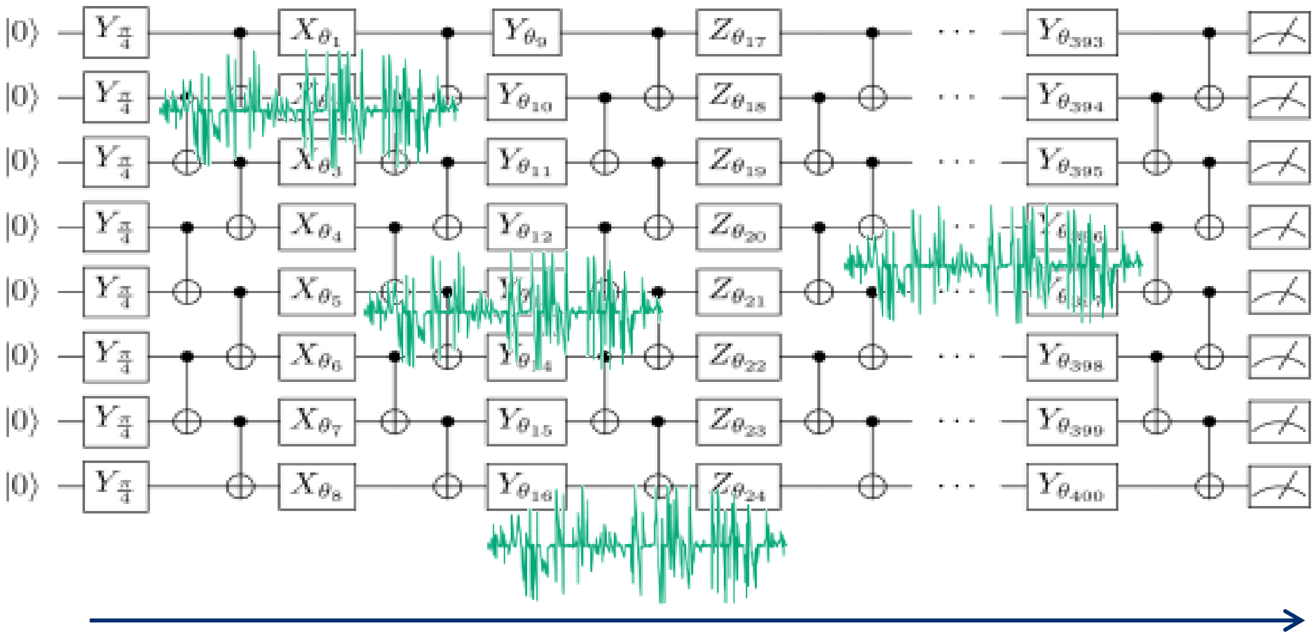
FIG. 2. Median optTTS for different solvers for the quadratic 3R3X instances at different problem sizes n. The error bars correspond to 2σ confidence intervals calculated using a Bayesian-bootstrap. Each solver is represented with a different marker and color, as denoted by the legend. DAU is Fujitsu's Digital Annealer Unit, run in parallel mode on 25 April 2020. SBM is Toshiba's Simulated Bifurcation Machine, accessed via Amazon Web Services on 20 August 2020. PT is our implementation of parallel tempering. MEM is the Virtual MemComputing Machine. SATonGPU is the data from Fig. 5 of Bernaschi *et al.* [37], after converting their native three-body 3R3X results in $n/2$ variables to n two-body variables by simplying doubling their reported n values. Note that the SATonGPU results are plotted on a separate, shifted horizontal axis (top, blue), as this solver reached significantly larger problem sizes than the other solvers; its optTTS is smaller by at least two orders of magnitude than the rest. DWA is the D-Wave Advantage1.1 device accessed via LEAP on 31 October 2020. DWAsub are suboptimal points in which the optimal runtime is below $1\mu s$, the lowest runtime possible on the Advantage1.1 device. The lines correspond to the exponential fits [Eq. (3)] of the data, with the coefficients given in Table II. The DWAsub points were not used in computing the DWA scaling exponent reported in Table II.

Quantum Volume and variants

How large and how long are the programs that a quantum processor can run reliably today?

https://quantum-journal.org/papers/q-2020-11-15-362/





Fidelity of two pure quantum states is a distance metric that could be defined as the overlap/transition probability:

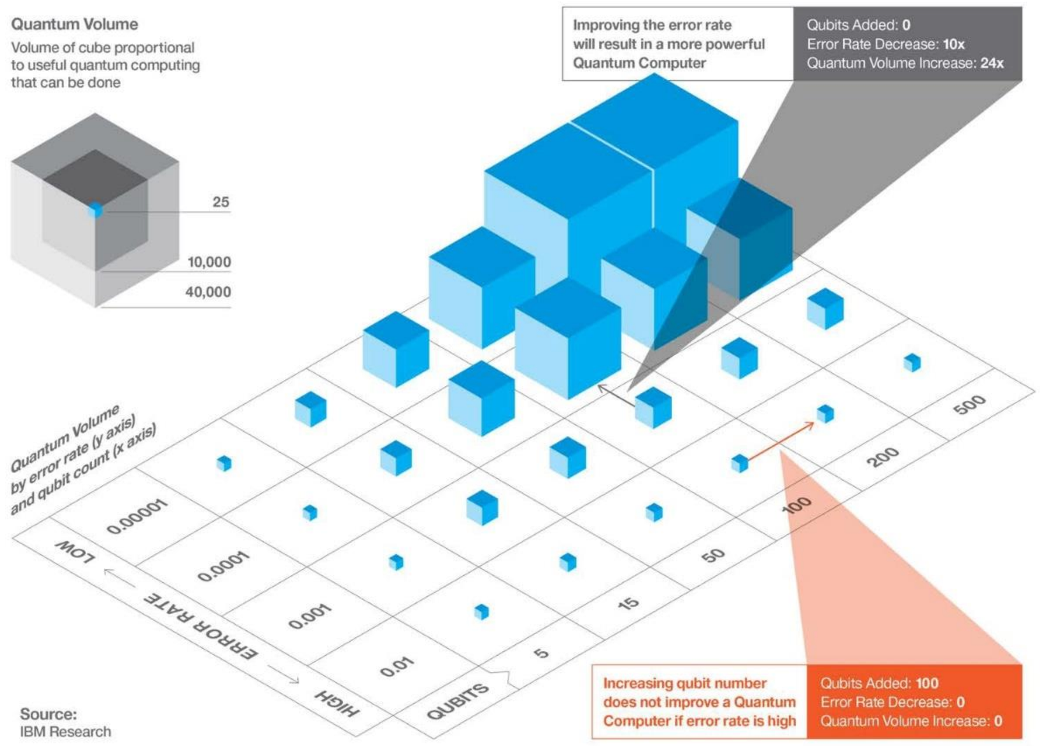
$$F(\psi, \phi) = |\langle \psi | \phi \rangle|^2$$

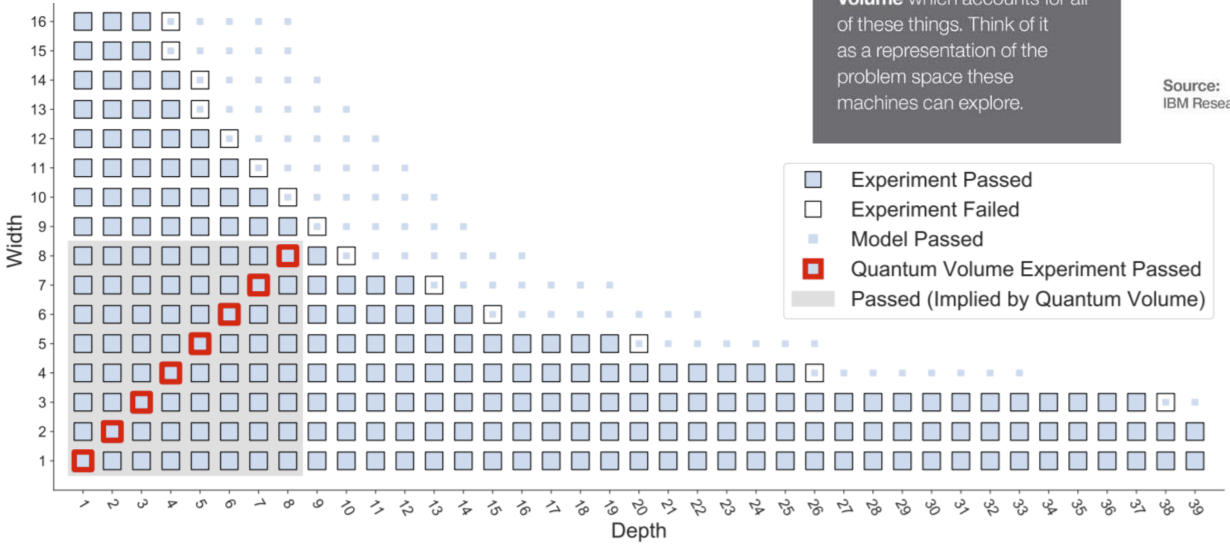
Can be generalized for noisy quantum states.

Quantum Volume and variants

A Quantum Computer's power depends on more than just adding qubits

If we want to use quantum computers to solve real problems, they will need to explore a large space of quantum states. The number of qubits is important, but so is the error rate. In practical devices, the effective error rate depends on the accuracy of each operation, but also on how many operations it takes to solve a particular problem as well as how the processor performs these operations. Here we introduce a quantity called **Quantum Volume** which accounts for all of these things. Think of it as a representation of the problem space these machines can explore.





Approximation Ratio as a function of time

Approximation Ratio

The probability that R variables are all less than x is $F(x)^R$

The PDF for the maximum of R iid runs is $= dF/dE = R\, F_E(x)^{R-1} P_E(x) E(x)$

For a discrete distribution: $$\mathbb{E}(Y_N) = \sum_{k=1}^{L} \left[\left(\sum_{r=k}^{L} p(x_r) \right)^N - \left(\sum_{r=k+1}^{L} p(x_r) \right)^N \right] x_k$$ https://arxiv.org/pdf/2001.04014.pdf

Can be obtained by bootstrapping!

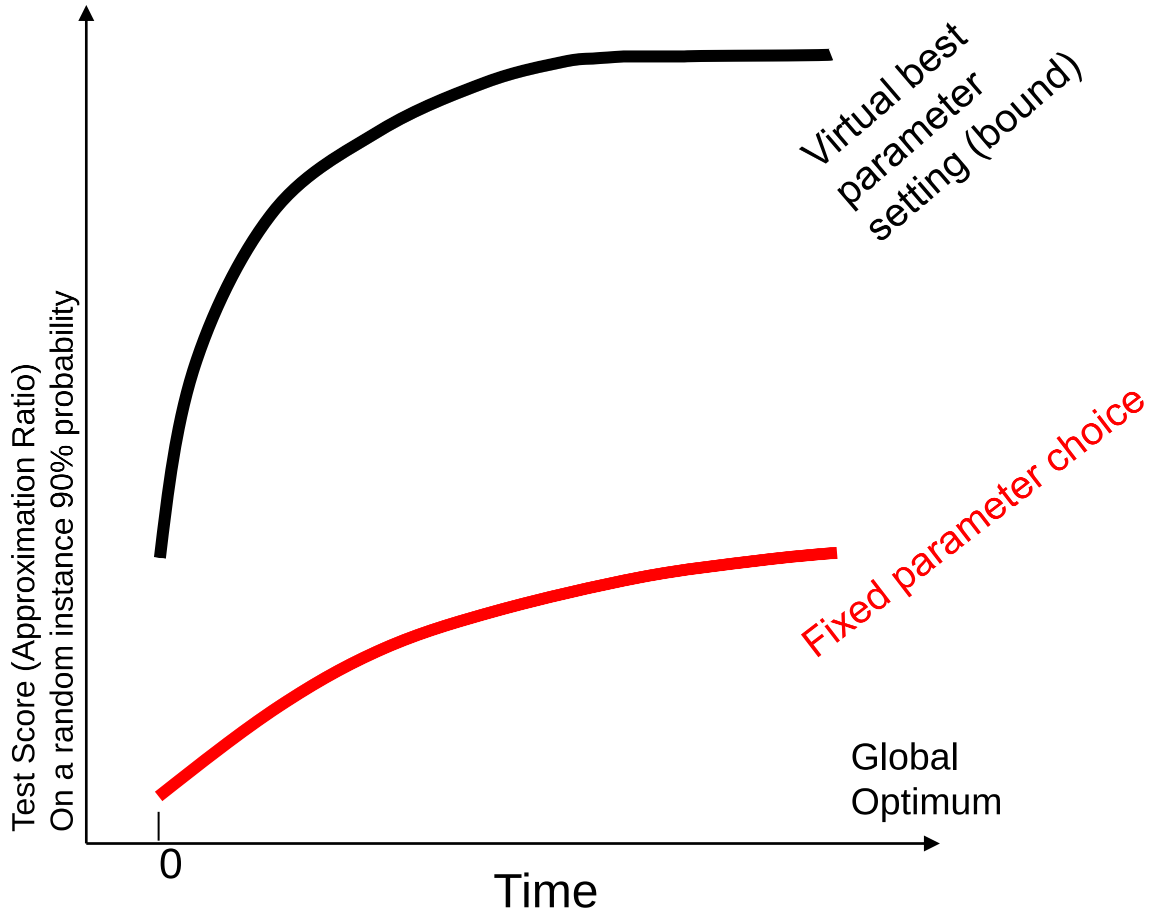
Generate $N >> 1$ runs. Select randomly $R << N$. Compute the max of R. Repeat and average.

How to evaluate the parameter setting

In the real world what you care about is «speedup at application scale» for your problem of interest.

R runs at fixed parameters

See exercise on example of Simulated annealing.



Resources to study

Comparison D-Wave, CIM
Comparison CIM, QA-QAOA
Comparison FujitsuDA, PT, D-Wave

(see slides online for links)

MIMO wireless with D-Wave
and with CIM dynamics

Let's go to Colab

https://colab.research.google.com/github/bernalde/QuIPML/blob/master/notebooks/Notebook%204%20-%20Benchmarking.ipynb