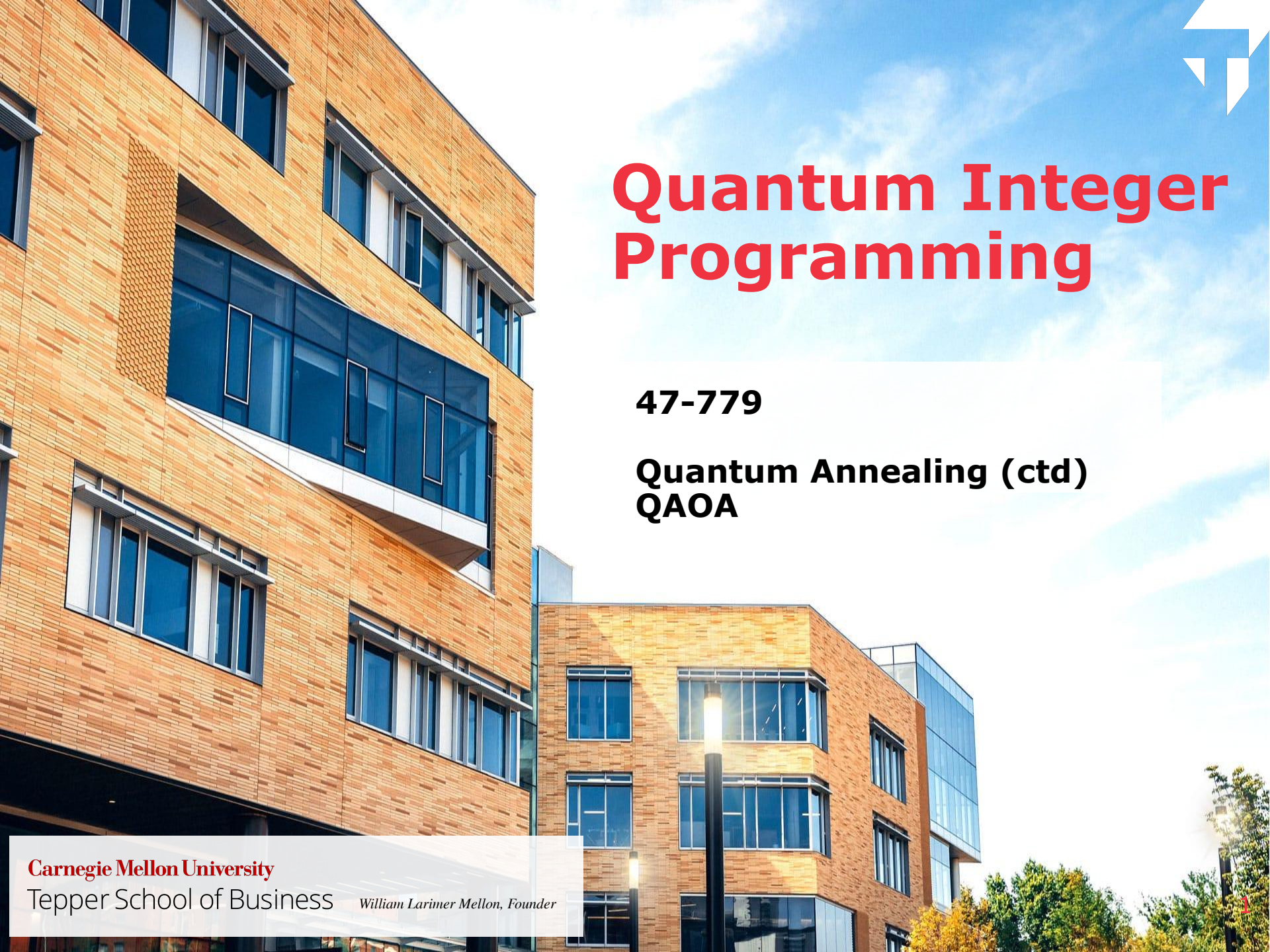


Quantum Integer Programming

47-779

Quantum Annealing (ctd)
QAOA

Carnegie Mellon University
Tepper School of Business
William Larimer Mellon, Founder

Agenda

- Minor Embedding
- Tuning Quantum Annealer
- Amazon Braket - Quantum Annealing
- How to do a Benchmarking Study
- Quantum Approximate Optimization Algorithm
- Amazon Braket - QAOA

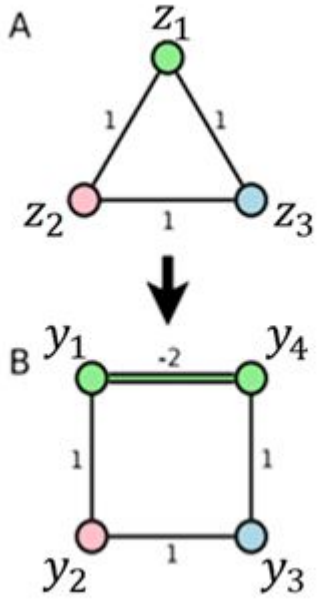
Minor Embedding - Example

$$\min_{z\in\{-1,+1\}^3} z_1z_2 + z_2z_3 + z_3z_1$$

$$\downarrow$$

$$\min_{y\in\{-1,+1\}^3} y_1y_2 + y_2y_3 + y_3y_4$$

$$\text{s.t. } y_1 = y_4$$



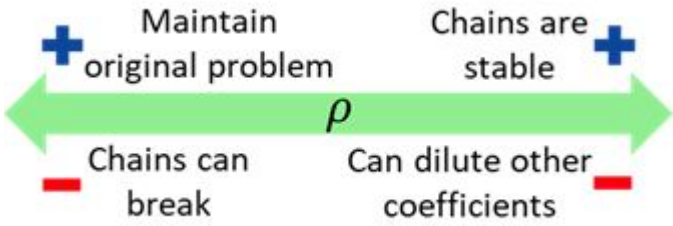
Penalizing the new constraints

$$\min_{y\in\{-1,+1\}^3} y_1y_2 + y_2y_3 + y_3y_4 + \rho(y_1 - y_4)^2$$

$$\min_{y\in\{-1,+1\}^3} y_1y_2 + y_2y_3 + y_3y_4 - \rho y_1y_4$$

ρ multiplier known as Chain strength

Solution: Make it a factor of the Q coefficients



[1] Nielsen, M.A., Chuang, I.L., 2010. Quantum computation and quantum information. Cambridge University Press.
[2] William Larimer Mellon, Founder. https://www.dwavesys.com/tutorials/background-reading-series/introduction-d-wave-quantum-hardware

Minor Embedding - Example

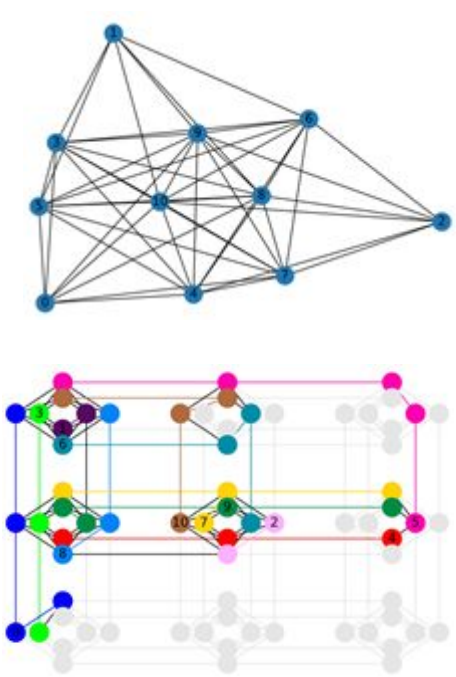
From our main example

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

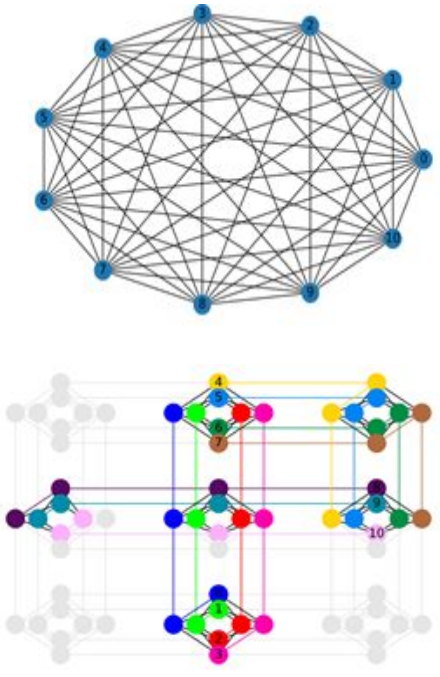
And embed it into a Chimera graph (subgraph of the Chip)

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



Best embedding in 1000 heuristic runs

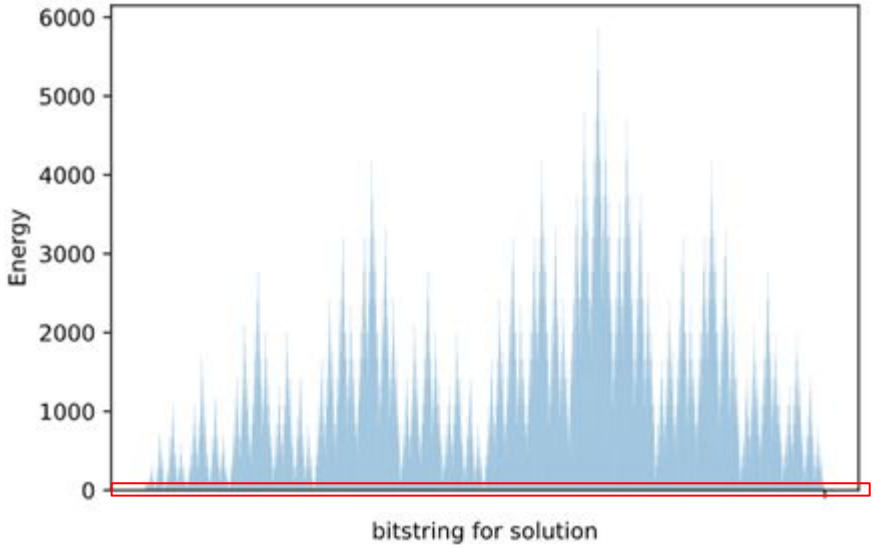


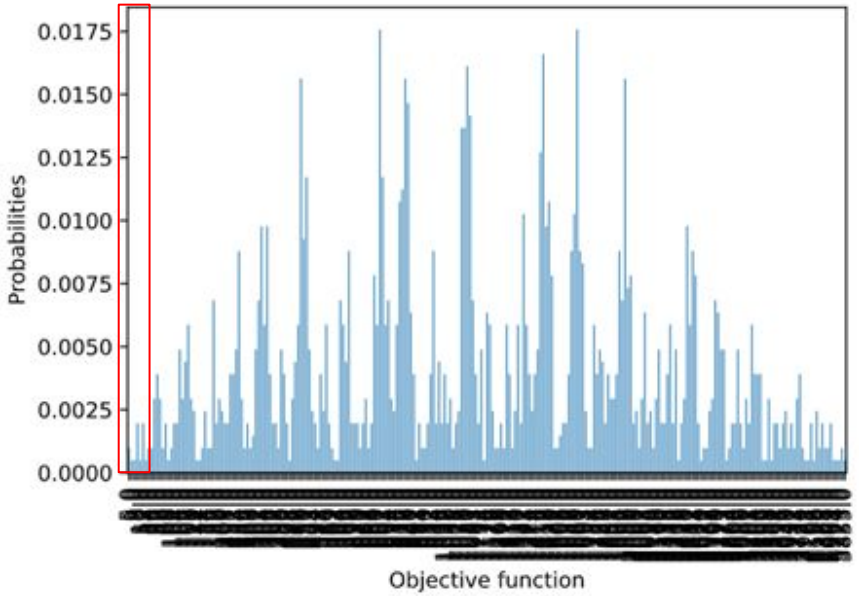
Full graph embedding

Solution via Enumeration

Before presenting the solution using QA let's return to our example

If we enumerate all solutions for the QUBO, we obtain the following profiles

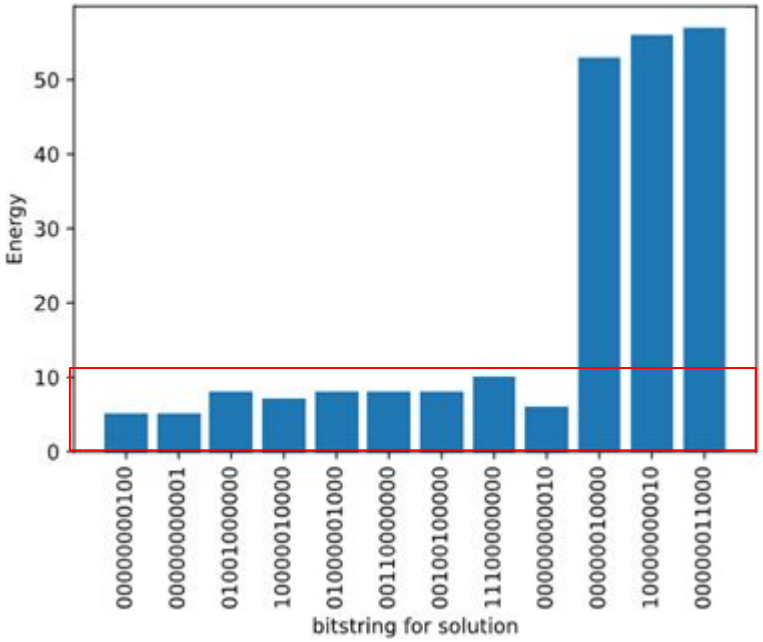


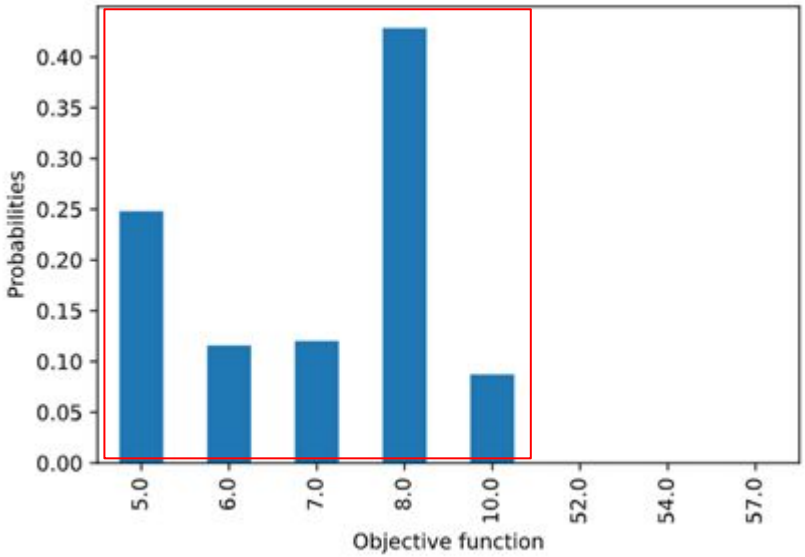


The "infeasible" solutions are heavily penalized and randomly sampling is not an option

Solution via Simulated Annealing

Using classical Simulated Annealing with the default parameters increases the probability of find the optimal solution from ½^11 to 0.25 and a feasible solution to almost 1

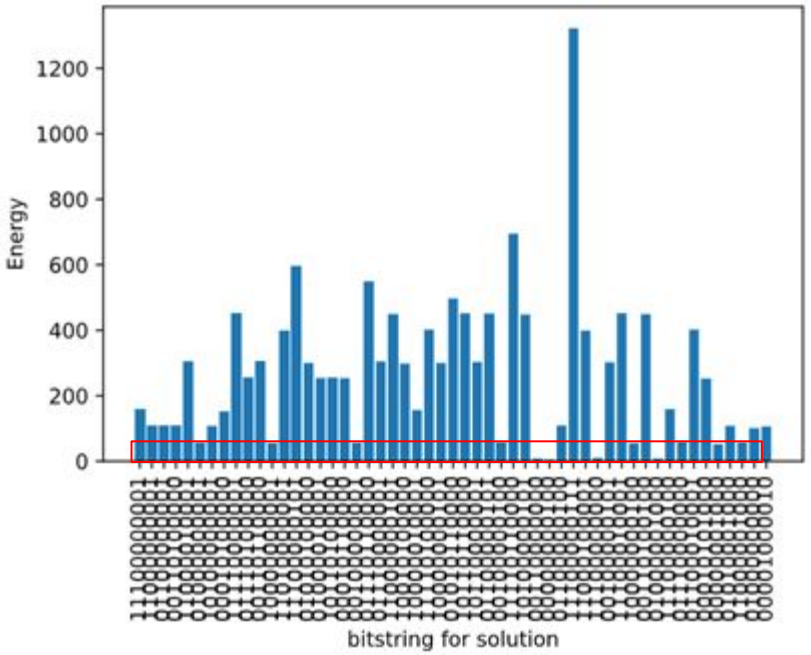


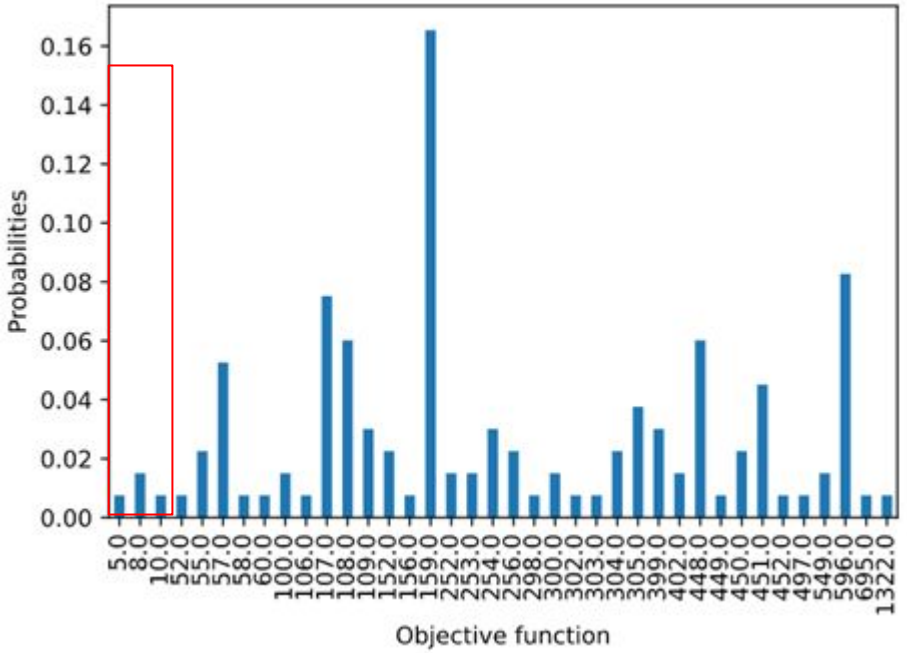


The “infeasible” solutions are heavily penalized and randomly sampling is not an option

Solution via Quantum Annealing

Using Quantum Annealing with the default parameters (annealing time, chain strength) results on probability of find the optimal solution of 5/10000 and feasible of 15/10000

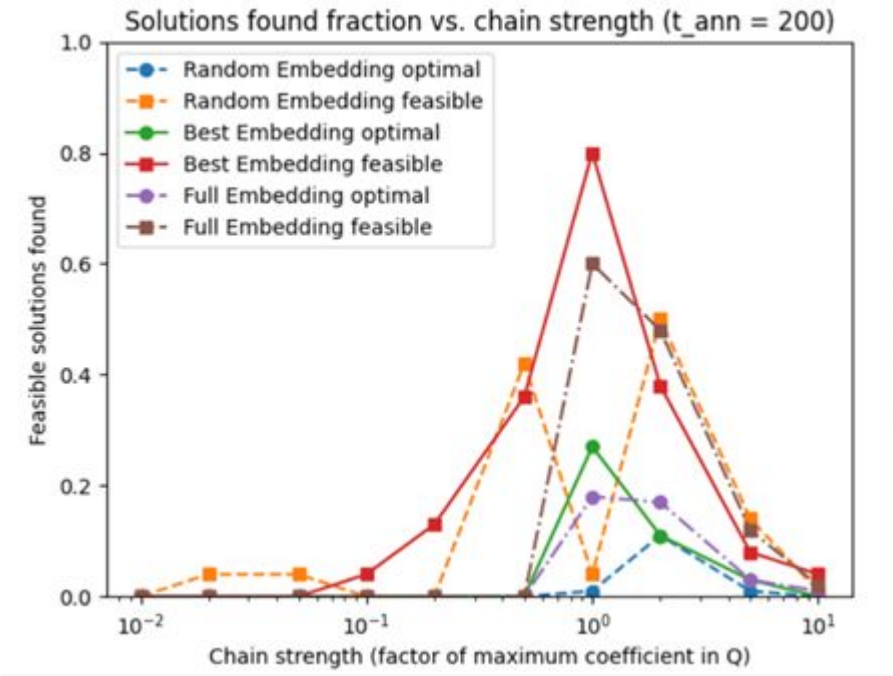


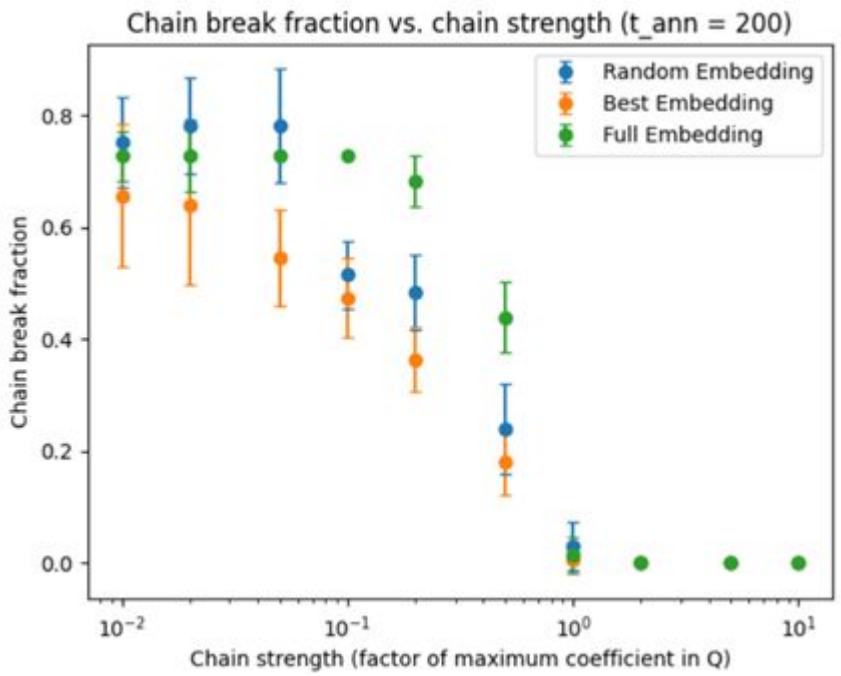


Quantum Annealing tuning t=200μs

In Quantum Annealing we analyzed two different factors, the chain strength and the annealing time.

Our main concern is maximizing the probability of success (feasible and optimal)

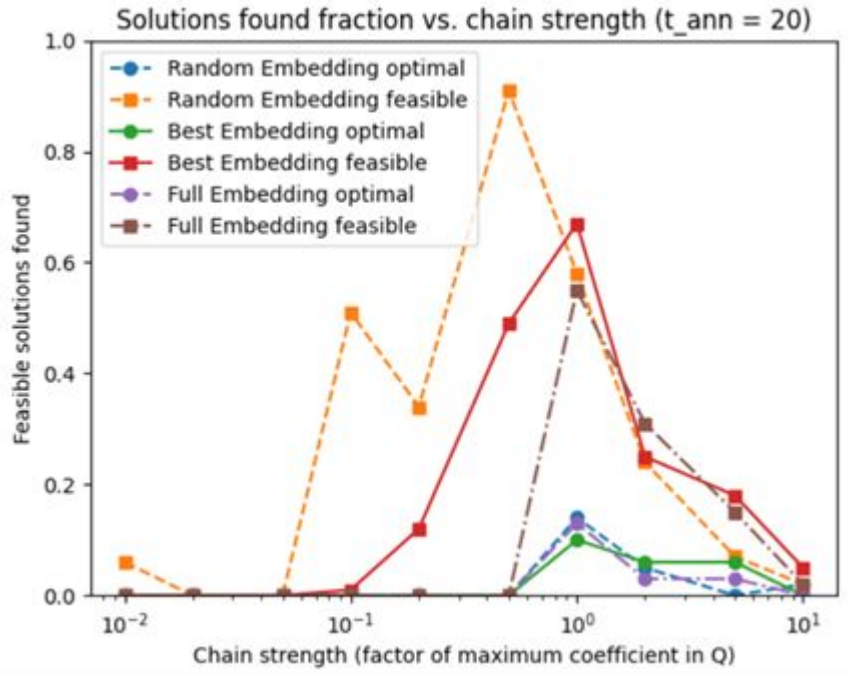


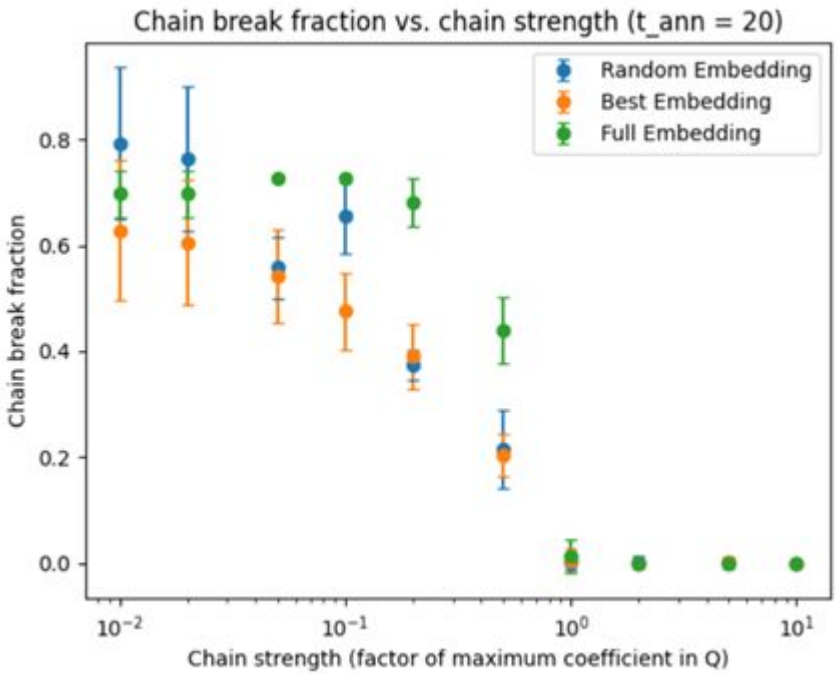


Quantum Annealing tuning t=20μs

In Quantum Annealing we analyzed two different factors, the chain strength and the annealing time.

Our main concern is maximizing the probability of success (feasible and optimal)

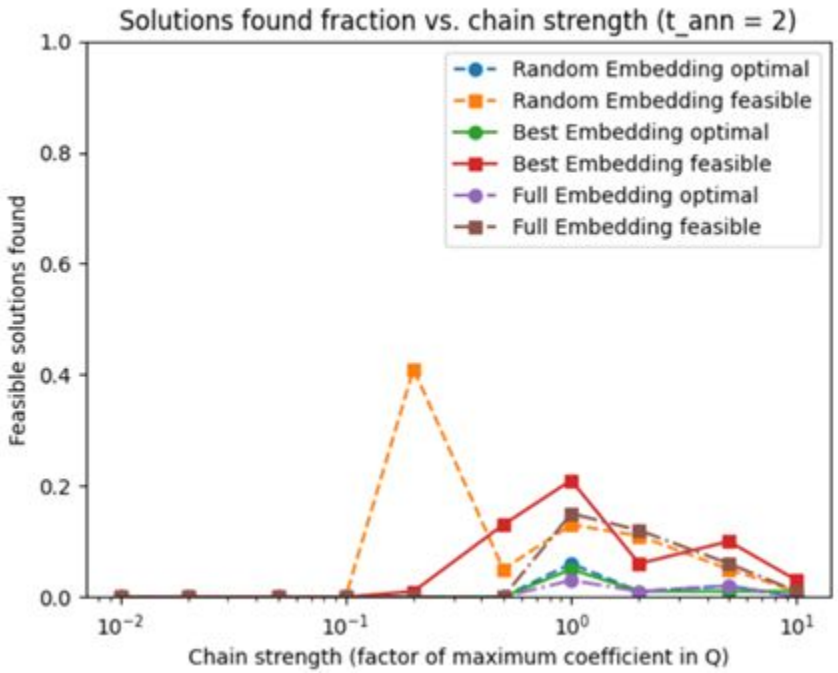


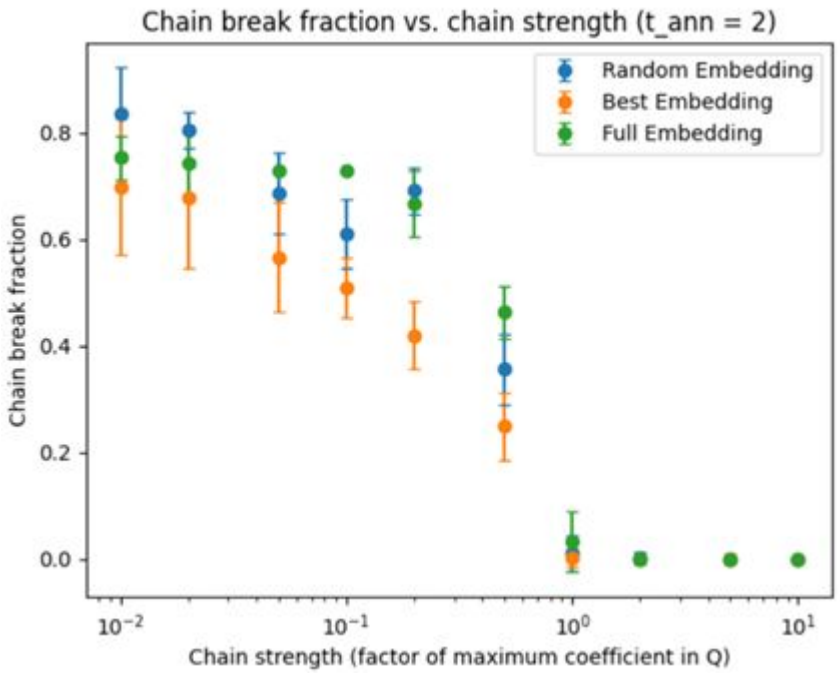


Quantum Annealing tuning t=2μs

In Quantum Annealing we analyzed two different factors, the chain strength and the annealing time.

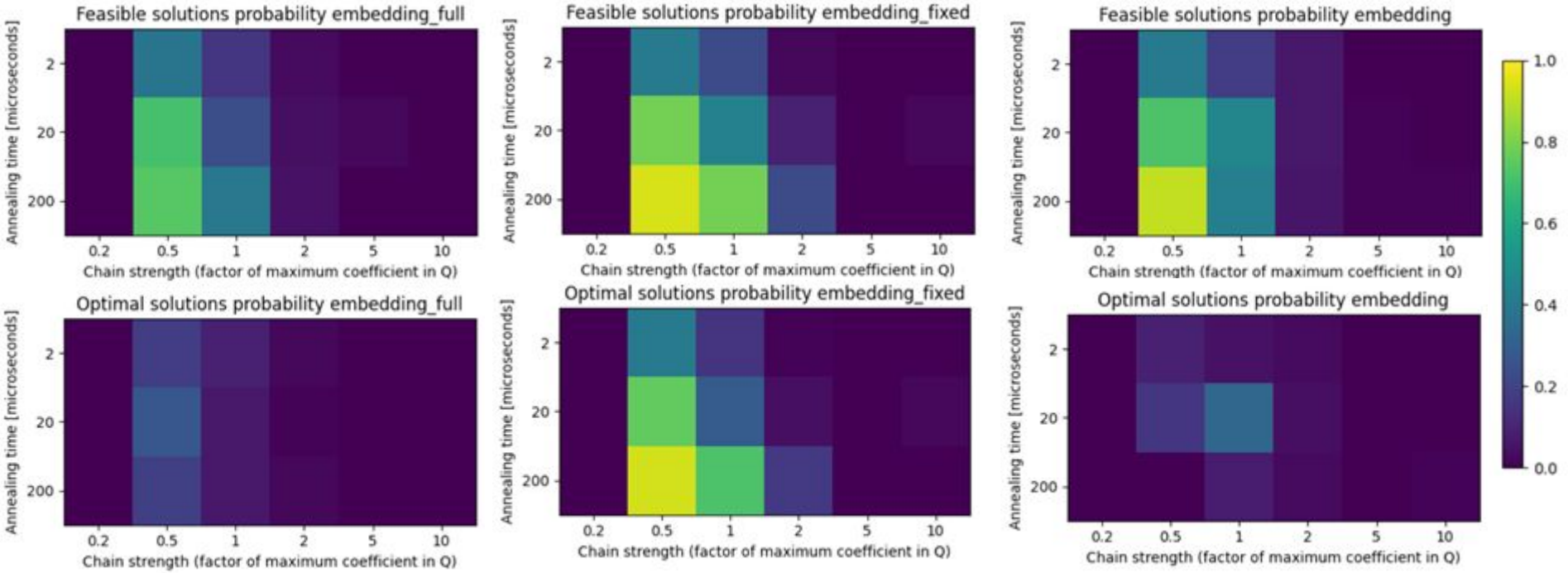
Our main concern is maximizing the probability of success (feasible and optimal)





Quantum Annealing tuning

Longer annealing times and chain strengths of the same order of magnitude as max(Q) are beneficial for this instance. The best embedding proved to be better than the random or full embeddings.



Amazon Braket for Quantum Annealing

Let’s go to Amazon Braket

https://console.aws.amazon.com/braket

Benchmarking and parameter setting

$$S(N) = \frac{C(N)}{Q(N)}$$

Speedup as a function of problem size

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

- **Provable Quantum Speedup (e.g. Grover)**
- **Strong Quantum Speedup (e.g. Shor)**
- **Quantum Speedup (potential, limited)**

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

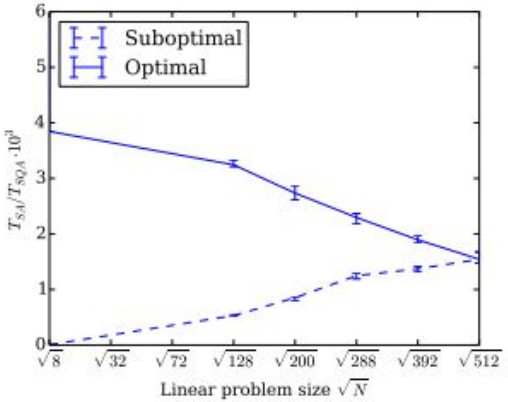
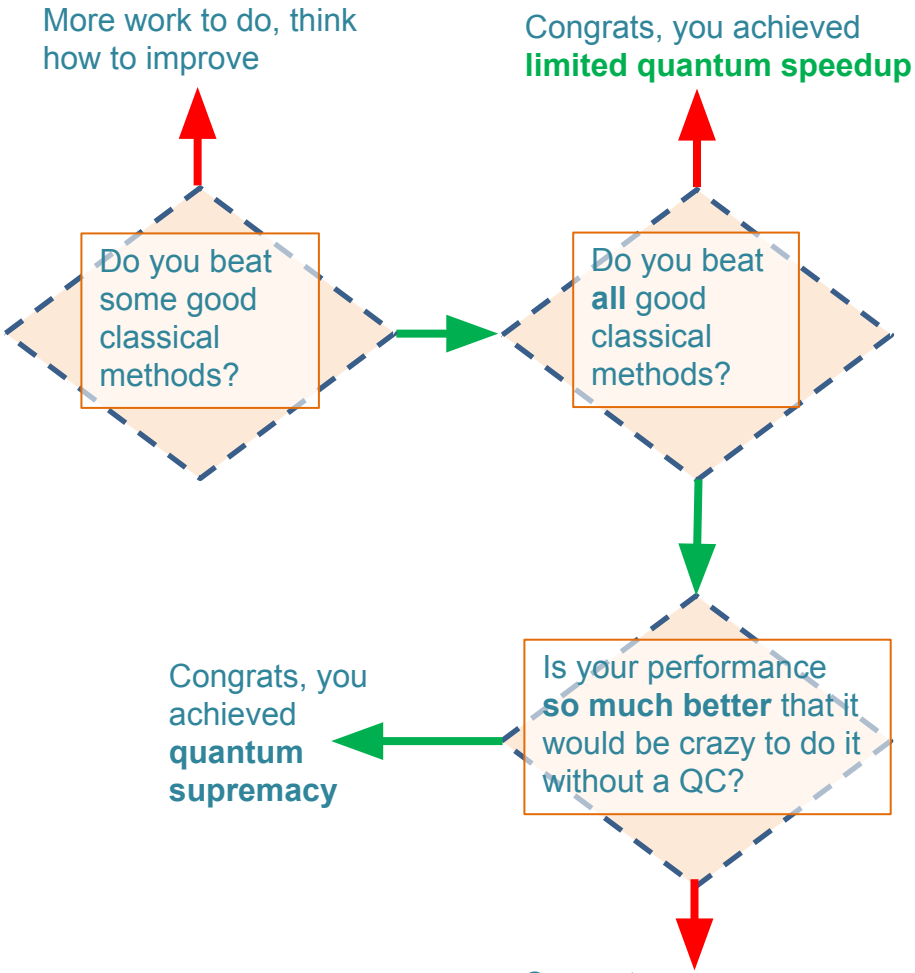


FIG. 2. **Pitfalls when detecting speedup.** Shown is the speedup of SQA over SA, defined as the ratio of median time to find a solution with 99% probability between SA and SQA. Two cases are presented: a) both SA and SQA run optimally (i.e., the ratio of the true scaling curves shown in Figure 1), and 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.

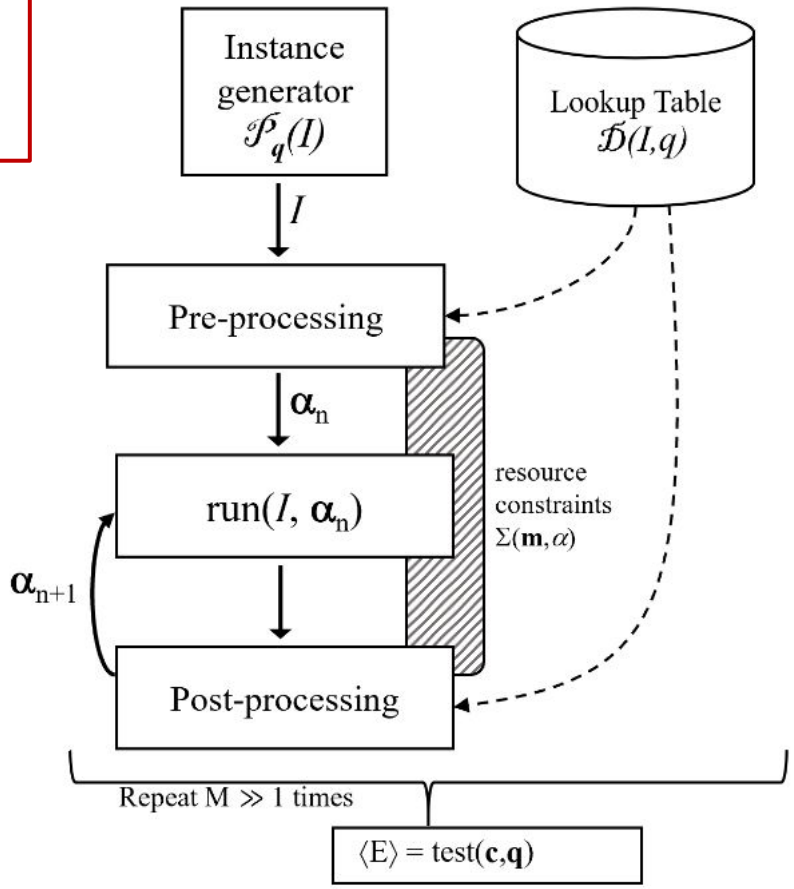
How to do a benchmark study

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)



How to do a benchmark study

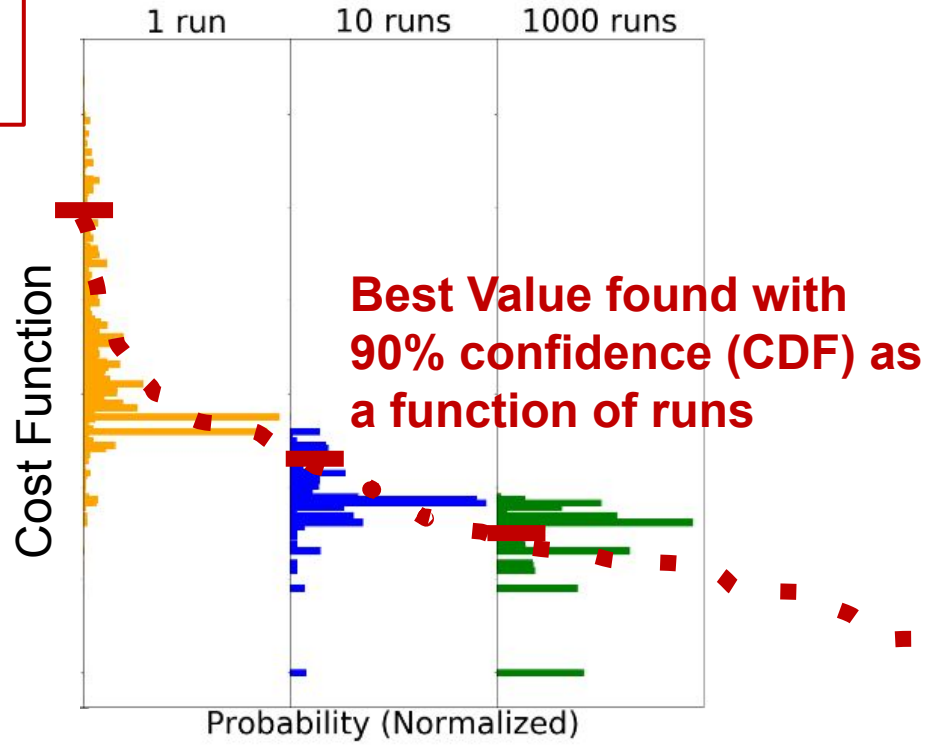
1. **The benchmarking question is: once I decide how to run, what is the quality of a solution that I can expect for a random new instance with a given confidence?**
2. performance collecting the statistics of distribution
3. If performance is not acceptable, use the distribution to choose new parameters (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)



How to do a benchmark study

1. **The benchmarking question is: once I decide how to run, what is the quality of a solution that I can expect for a random new instance with a given confidence?**
2. 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)

Expectation value of best found after N IID runs for one instance

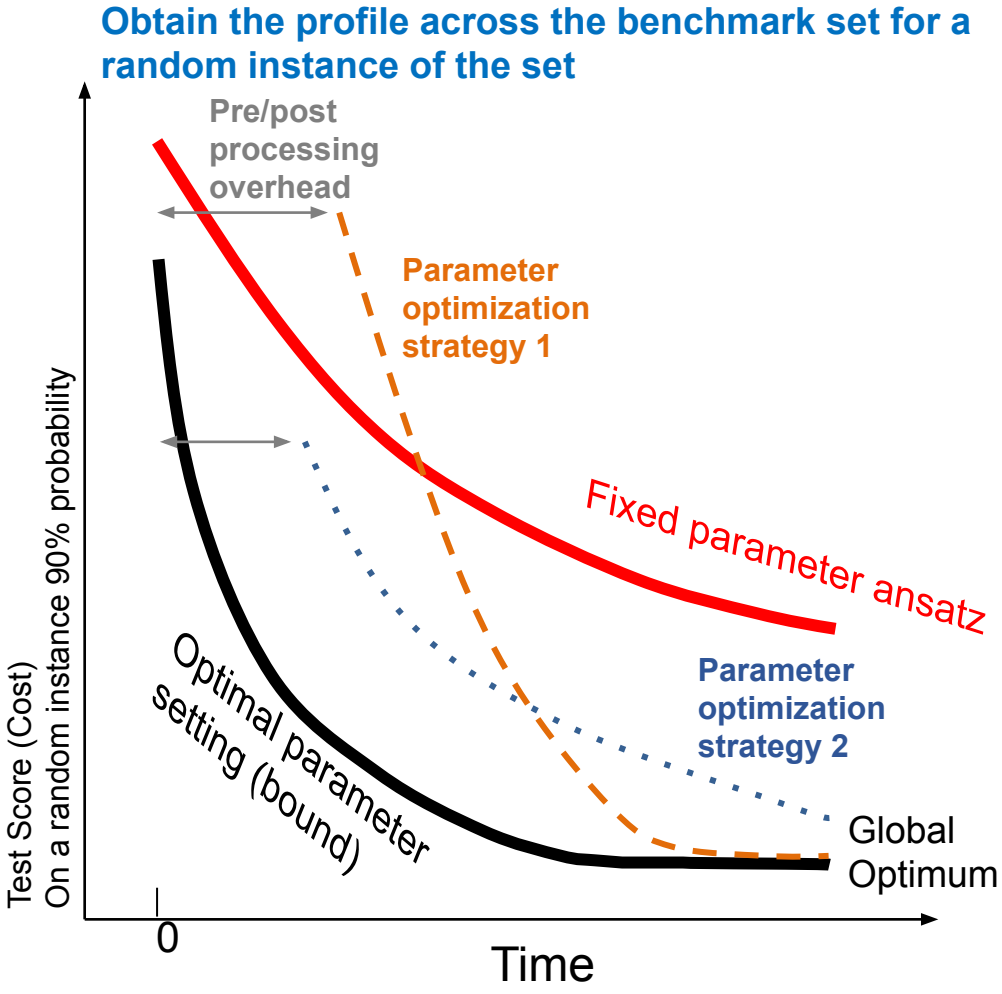


Best Value found with 90% confidence (CDF) as a function of runs

$$\mathbb{E}(Y_N) = \sum_{k=1}^{L} \left[\left(\sum_{r=k}^{L} p(x_r) \right)^N - \left(\sum_{r=k+1}^{L} p(x_r) \right)^N \right] x_k$$

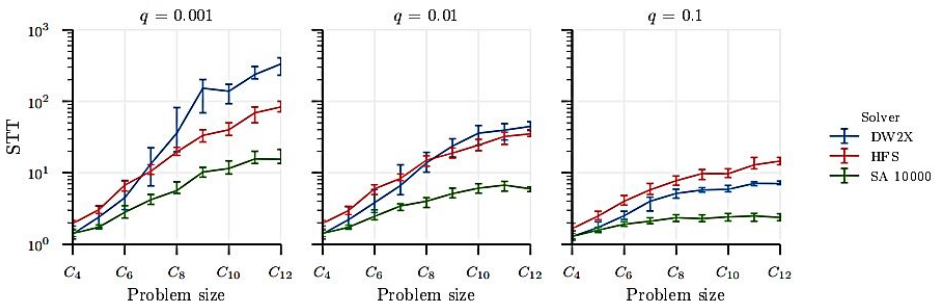
How to do a benchmark study

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)

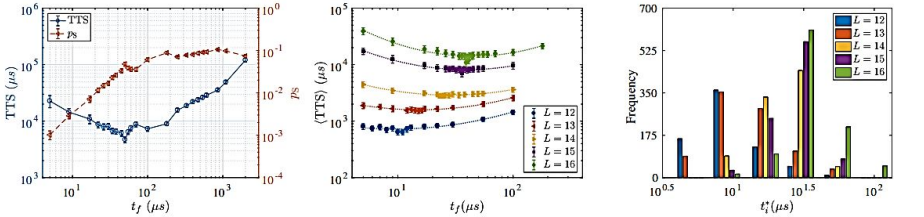


Example benchmarkings

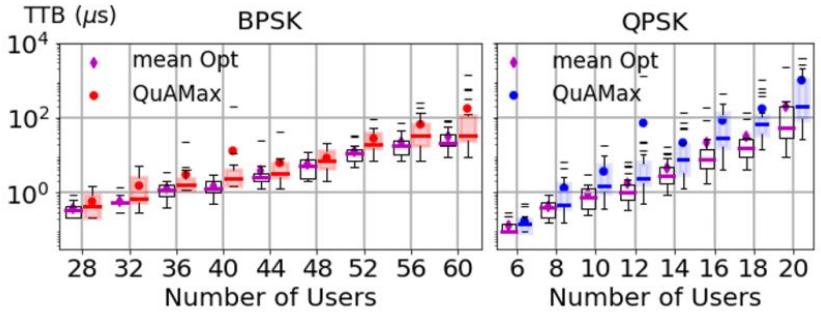
- Time-to-Bit-Error-Rate 10^{-6} (From "Benchmarking a quantum annealing processor with the time-to-target metric" https://arxiv.org/pdf/1508.05087.pdf)



- Time-to-solution (From "Demonstration of a scaling advantage for a quantum annealer over simulated annealing" https://arxiv.org/pdf/1508.05087.pdf)

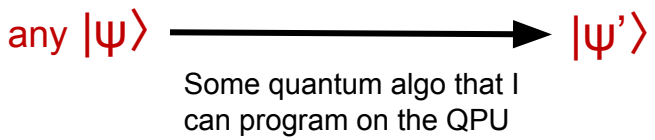


- Time-to-Bit-Error-Rate 10^{-6} (From "Leveraging Quantum Annealing for Large MIMO Processing in Centralized Radio Access Networks" https://arxiv.org/pdf/2001.04014.pdf)



Universal Quantum Computing

A quantum computer is UNIVERSAL if its instruction set allows the implementation of any algorithm allowed by quantum mechanics.



The time-evolution of the Ising model in a transverse field (Quantum Annealing as implemented in D-Wave) is **NOT** universal. However the general AQC procedure is universal (need more complex Hp and H_D).

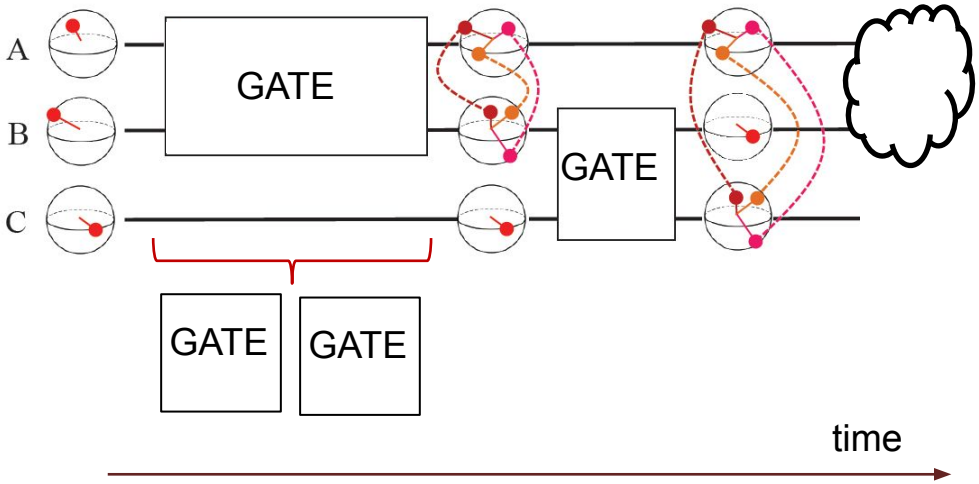
Why you might want a Universal Quantum Computer?

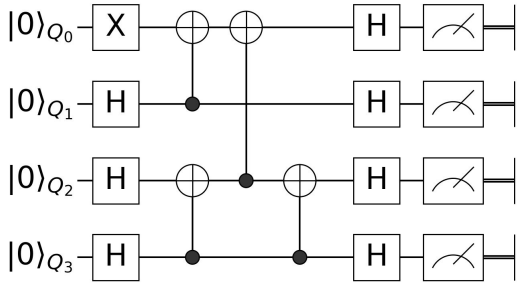
(1) Simulation of Quantum Systems
(2) Flexibility of implementation of multiple quantum algorithms (e.g. Grover/Shor)
(3) Exploit all the power of quantum mechanics
(4) Making sure that what you do is not classically simulatable efficiently

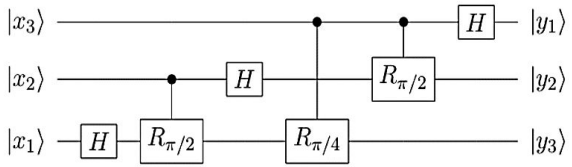
For quantum advantage in optimization heuristics, universality is not necessarily required (the final state we are searching is classical).

Gate-Model and Quantum Circuits

The gate-model is a simple way to break down the quantum coherent operations we use in quantum computing.







Reminder: every operation on a N qubit system is mathematically equivalent to multiplying a unitary matrix of $2^N x 2^N$ to a normalized vector. = you cannot keep track numerically of the amplitudes of large circuits.

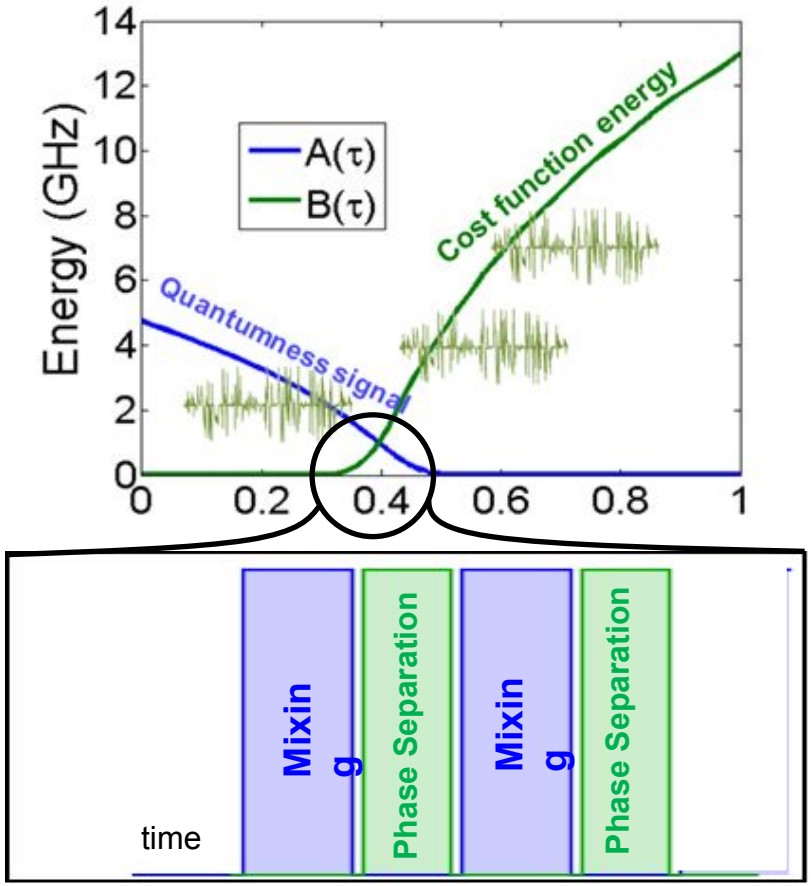
$U = U_{BC} U_{AB} |ABC\rangle$

$U_{AB} |ABC\rangle = \psi_{00}|00C\rangle + \psi_{01}|01C\rangle + \psi_{10}|10C\rangle + \psi_{11}|11C\rangle$

$U_{BC} U_{AB} |ABC\rangle = \psi_{00} U_{BC} |00C\rangle + \psi_{01} U_{BC} |01C\rangle + \psi_{10} U_{BC} |10C\rangle + \psi_{11} U_{BC} |11C\rangle$
$= \psi_{000}|000\rangle + \psi_{001}|001\rangle + \psi_{010}|010\rangle + \psi_{011}|011\rangle + \psi_{100}|100\rangle + \psi_{101}|101\rangle + \psi_{110}|110\rangle + \psi_{111}|111\rangle$

Quantum Approximate Optimization Algorithm

QUANTUM ANNEALING



MIT-CTP/4610

A Quantum Approximate Optimization Algorithm

Edward Farhi and Jeffrey Goldstone

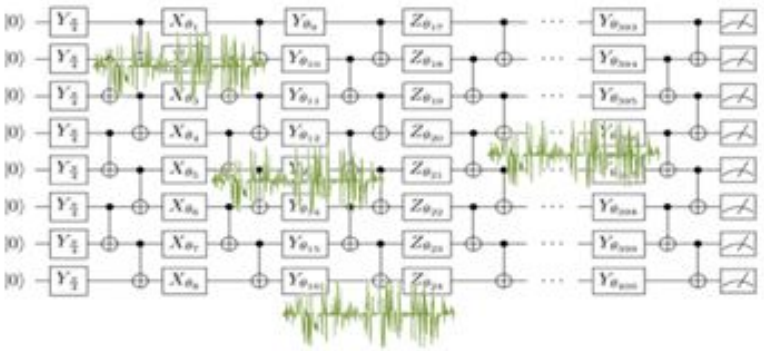
Center for Theoretical Physics

Massachusetts Institute of Technology

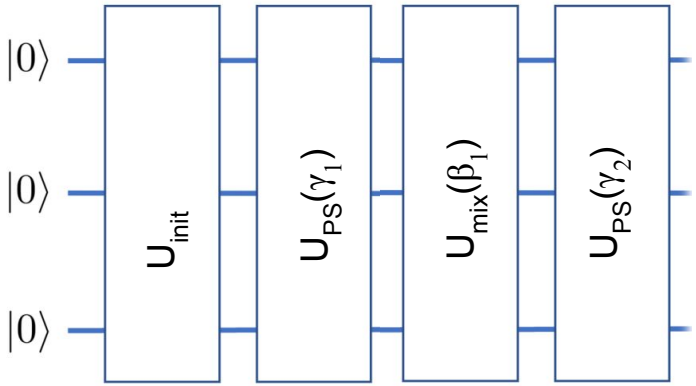
Cambridge, MA 02139

Sam Gutmann

- QAOA aims to implement (a)diabatic transitions coherent operations more flexibly than AQC (digitally).
- For infinite circuit this is at least as powerful as AQC.
- For finite circuit its power is unknown in general.



Quantum Approximate Optimization Algorithm: Example



mix the amplitudes by a transverse field rotation exp(iβX) on each qubit (arbitrary parameter)

$$|\psi\rangle_{mix(1)} = (2^{n/2})^{-1} \sum_s \beta_{1s}(\beta_1,\gamma_1) e^{i\gamma_{1s}(\beta_1,\gamma_1)} |s\rangle$$

Now if you measure, the probability of a bitstring depends both on γ and β in a non-linear way.

Initialization operator

Phase separation operator dependent on a parameter γ_1

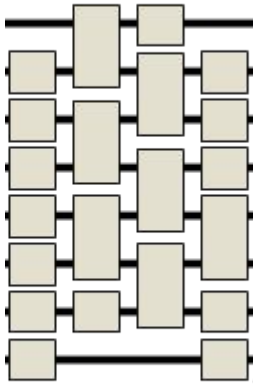
Exercise $R_y(-\pi/2)R_x(\pi)|0\rangle = (|0\rangle+|1\rangle)/\sqrt{2}$

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

$$\underbrace{exp(i\beta Z_1 Z_2)}\ |s_1 s_2\rangle = e^{i\beta s_1 s_2}\ |s_1 s_2\rangle$$

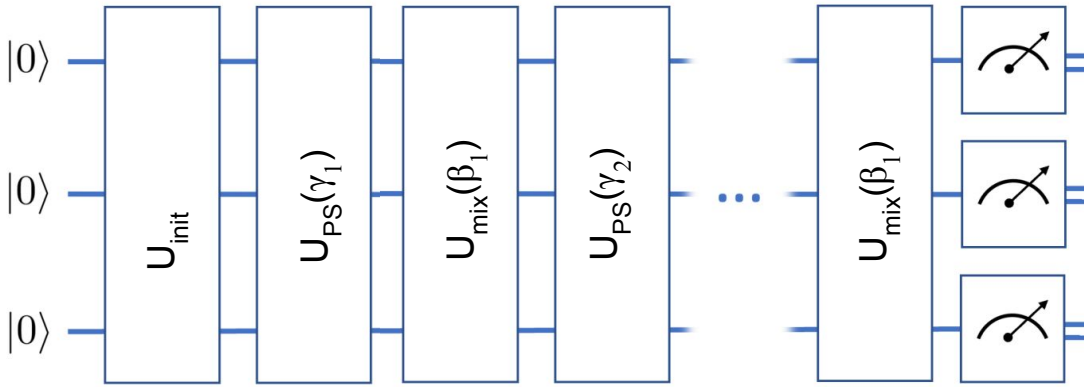
Logical 2-qubit gate representing the Ising interaction

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



You need to schedule the gates for every term of the objective function !

Quantum Approximate Optimization Algorithm: Example



$$|\psi\rangle_{qaoa(p)} = (2^{N/2})^{-1} \sum_s \mathbf{B_{2s}(\beta_1,\gamma_1,\beta_2,\gamma_2,\ldots\ \beta_p,\gamma_p)} e^{i\Gamma_{1s}(\beta_1,\gamma_1,\beta 2,\gamma 2,\ldots\ \beta_p,\gamma_p)} |s\rangle$$

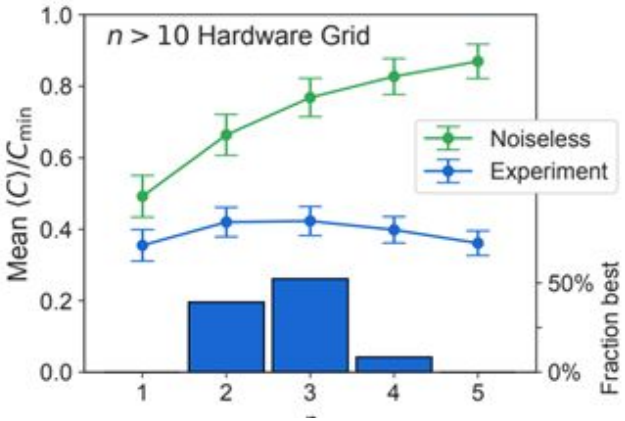
Now if you measure, the probability of a bitstring depends both on γ and β in a non-linear way.
It is exponentially difficult to predict or simulate the probability $|B_{2s*}(\beta_1,\gamma_1,\beta_2,\gamma_2,\ldots\ \beta_p,\gamma_p)|^2$ to find the optimal unknown solution s*

For p=∞ you can map this evolution to AQC; discrete becomes continuous; so you know how to do it. For finite p there is currently not a lot of guidance, big sector of research.
The search over the parameter space γ and β is done heuristically (e.g. Gradient descent)

Recent Results: QAOA

Quantum Approximate Optimization of Non-Planar Graph Problems on a Planar Superconducting Processor

Google AI Quantum and Collaborators*
(Dated: April 10, 2020)



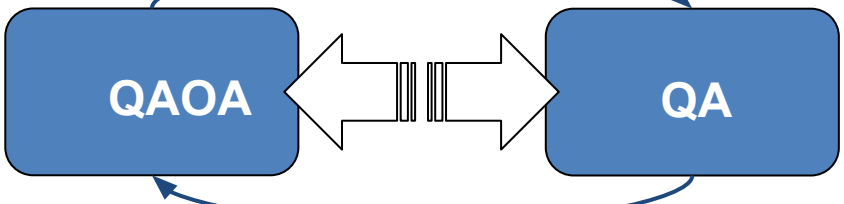
Reference	Date	Problem topology	$\Delta(G)$	n	p	Optimization
Otterbach *et al.* [22]	2017-12	Hardware	3	19	1	Yes
Qiang *et al.* [27]	2018-08	Hardware	1	2	1	No
Pagano *et al.* [26]	2019-06	Hardware[1] (system 1)	n	12, 20	1	Yes
		Hardware[1] (system 2)	n	20–40	1–$2^{(2)}$	No
Willsch *et al.* [23]	2019-07	Hardware	3	8	1	No
Abrams *et al.* [24]	2019-12	Ring	2	4	1	No
		Fully-connected	n			No
Bengtsson *et al.* [25]	2019-12	Hardware	1	2	1, 2	Yes
This work		Hardware	4	2–23	1–5	Yes
		3-regular	3	4–22	1–3	Yes
		Fully-connected	n	3–17	1–3	Yes

Optimizing Variational Quantum Algorithms Using Pontryagin's Minimum Principle

Zhi-Cheng Yang,[1] Armin Rahmani,[2,3] Alireza Shabani,[4] Hartmut Neven,[4] and Claudio Chamon[1]

Low depth mechanisms for quantum optimization

Jarrod R. McClean,[1,*] Matthew P. Harrigan,[1] Masoud Mohseni,[1] Nicholas C. Rubin,[1] Zhang Jiang,[1] Sergio Boixo,[1] Vadim N. Smelyanskiy,[1] Ryan Babbush,[1] and Hartmut Neven[1]
[1]*Google Research, 340 Main Street, Venice, CA 90291, USA*
(Dated: August 21, 2020)



Quantum Approximate Optimization Algorithm: Performance, Mechanism, and Implementation on Near-Term Devices

Leo Zhou,[1,*] Sheng-Tao Wang,[1,†] Soonwon Choi,[1,2] Hannes Pichler,[3,1] and Mikhail D. Lukin[1]

DARPA Optimization with Noisy Intermediate Scale Quantum systems (ONISQ)

From the Quantum Approximate Optimization Algorithm to a Quantum Alternating Operator Ansatz

Stuart Hadfield [1,2,3,*], Zhihui Wang [1,2], Bryan O'Gorman [1,4,5], Eleanor G. Rieffel [1], Davide Venturelli [1,2] and Rupak Biswas [1]

Amazon Braket for QAOA

Let’s go to Amazon Braket

https://console.aws.amazon.com/braket