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

Quantum Approximate (Alternating) Optimization Algorithm (Ansatz)

Lecture 12

2021.10.12

Quiz III

Follow Lecture X in order to create an IBM Qiskit, DWave Leap, and Amazon Web Services account.

Access Amazon Braket on AWS and execute the QAOA example that we will show at the end of the lecture today.

Update a PDF with a proof that you have created each account and that you have executed the AWS Braket QAOA tutorial.

Agenda

- A Quantum Optimization Algorithm
- Quantum Adiabatic Algorithm
- Adiabatic Quantum Computing
- Quantum Approximate Optimization Algorithm (QAOA)
- QAOA for Constrained Optimization Problems
- Quantum Alternating Optimization Ansatz
- QAOA in the Real World
- Amazon Braket Exercise

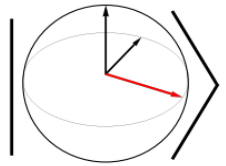
A Quantum Optimization Algorithm

1) Map a QUBO Objective function into Ising form and assign the logical identity of each spin variable to a qubit in the processor.

$$x_i = (si + 1)/2 \rightarrow |xi\rangle$$

2) Apply single-qubit rotations to every qubit to put the state of the QPU in superposition of all possible solutions of the optimization problem (Hadarmard gates)

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



(3) Apply two level gates and single qubits rotations to change the state, having some smart idea on how to increase the value of $|\Psi_{n=target}|^2$

Algorithms are difficult to design because you are doing matrix multiplication with matrices of dimensions $2^N \times 2^N$ – nature does it for you! you don't need to do it but good luck simulating it

(4) Measure the state, read the qubits (they are a single bitstring after measurement) and hope to find the target(s).

(5) Repeat the procedure many times and keep the best result.

The Quantum Adiabatic Algorithm

AQC is based on a property of the time-dependent Schrödinger equation – the «adiabatic theorem».

Einstein's "Adiabaten hypothese": "If a system be affected in a reversible adiabatic way, allowed motions are transformed into allowed motions" (Einstein, 1914).

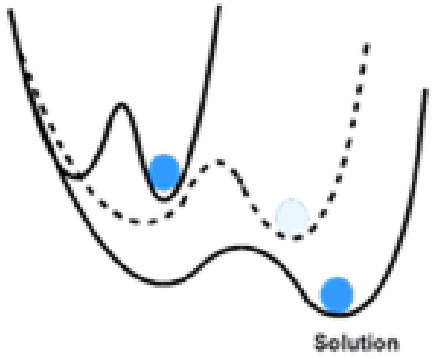
(1) Switch on a quantum interaction in your system
(2) Take the spectrum of possible energies of your quantum system as a function of the degrees of freedom and set the state to a well-defined energy (not metastable states) which is ranked n^{th} in order of magnitude (e.g., the second smallest)
(3) Do any Schrödinger evolution (no measurement! no noise!) that changes the energy states «sufficiently slow».
(4) Measure the energy of the state. You will find with 100% probability that the energy is ranked also n^{th}

Adiabatic evolution (e.g., Slow Schrödinger) preserves the energy ranking of your system. The smallest energy state (ground state) also maps into the ground state at the end.

IDEA: map objective function into energy. Start from easy problem to solve with known solution and modify slowly to difficult. Measure unknown solution

Albash, Lidar
Rev. Mod. Phys. 90, 015002 (2018)
https://arxiv.org/abs/1611.04471

- Apolloni 1989
- Finnila 1994
- Nishimori 1998
- Brooke 1999
- Fahri 2001



Adiabatic evolution

Solving ISING/QUBOs using Quantum Computing – How?

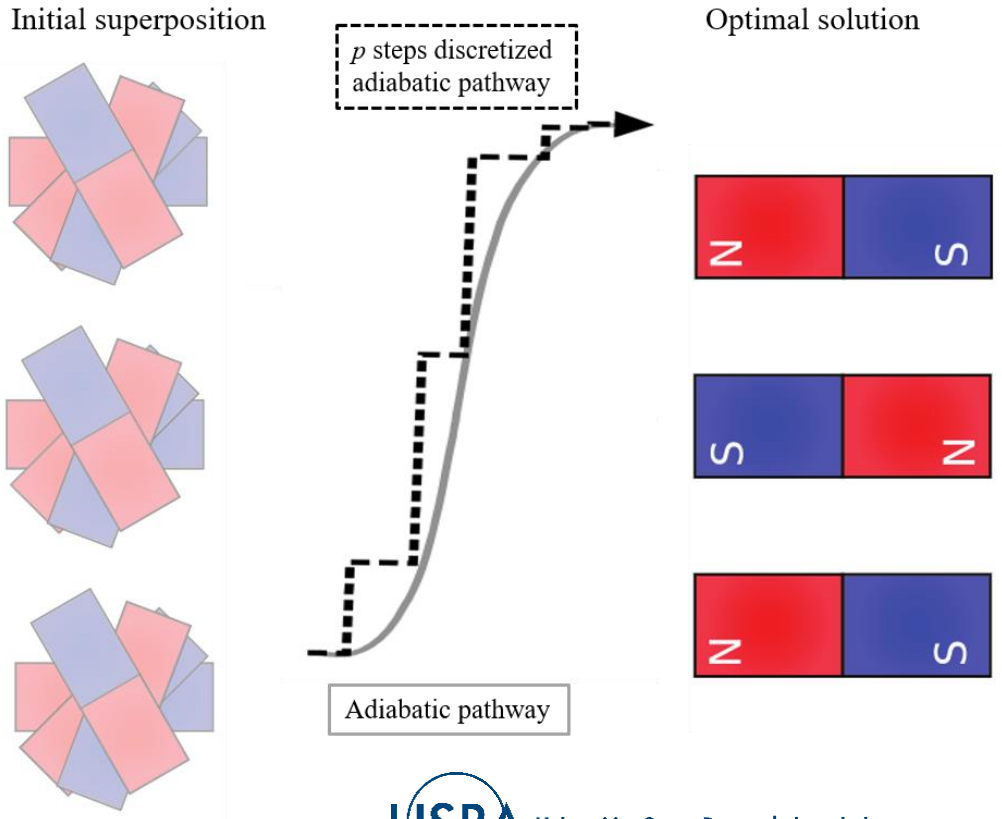
Adiabatic Quantum Computing

1. Write objective function into energy of a Quantum System (ISING=QUBO⊂MINLP).
2. Start from easy problem to solve with known solution and modify slowly to difficult.
3. Measure unknown solution

- Property of time-dependent Schroedinger equation – the «adiabatic theorem».

Using different models of Quantum Computers

- Gate-based computers
  - For solving QUBOs, we can use algorithms like:
    - Quantum Approximate Optimization Ansatz (QAOA)
    - Variational Quantum Eigensolver (VQE)
  - For optimization, algorithms can be understood as discretized adiabatic computation
  - IBM/Google quantum computers are gate-based
- Quantum annealers
  - They run a single quantum algorithm, quantum annealing
  - Finite temperature implementation of adiabatic quantum evolution
  - Analog computation
  - D-Wave quantum annealer is the best-known example



Adiabatic Quantum Computation

Encoding solution of a problem as the ground state of a quantum Hamiltonian.
Initially proposed as an algorithm that simulated quantum fluctuation and tunneling

- Contrary to thermal fluctuations in Simulated Annealing

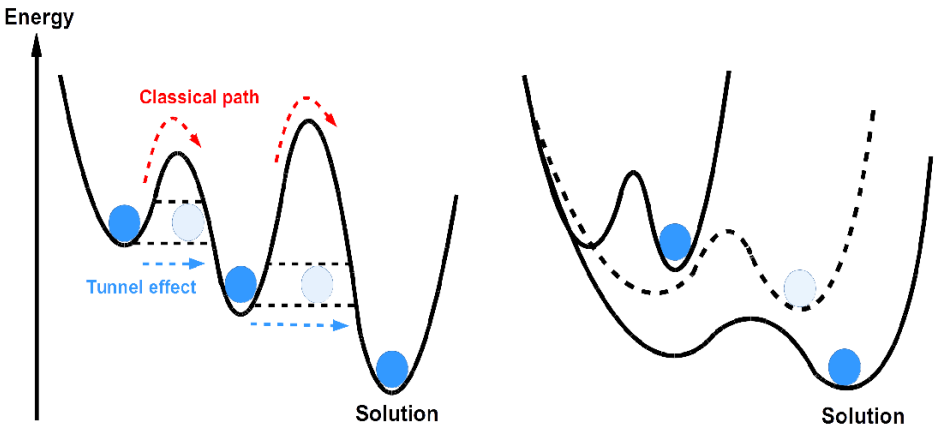
Problem Hamiltonian H_C

From an objective function we can construct a Hamiltonian on n qubits by replacing $\mathbf{x} \in \{0,1\}^n$ with $\mathbf{z} \in \{-1,+1\}^n$ obtaining a polynomial

$$f(\mathbf{z}) = \sum_{\mathcal{C} \subset \{1,\dots,n\}} \alpha_{\mathcal{C}} \prod_{j \in \mathcal{C}} z_j$$

We replace the Pauli z operator σ_i^z for each z_i variable leading to

$$H_C = \sum_{\mathcal{C} \subset \{1,\dots,n\}} \alpha_{\mathcal{C}} \bigotimes_{j \in \mathcal{C}} \sigma_j^z$$



Quantum Tunnelling Adiabatic evolution

$$\begin{bmatrix} 3/4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1/4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1/4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1/4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1/4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1/4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1/4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3/4 \end{bmatrix}$$

Quantum Approximate Optimization Algorithm

QAOA Tutorial Outline

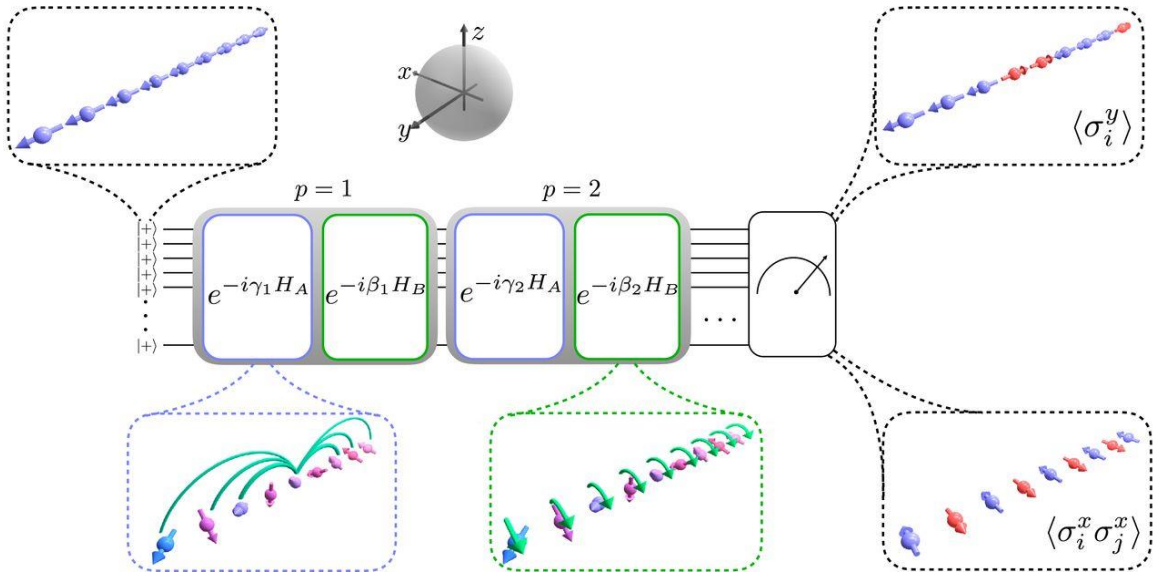
- Quantum Approximate Optimization Algorithm: review and status
- The «Quantum Alternating Operator Ansatz»
  - Mixing Operators
  - Examples
- Compiling and Executing
  - The gate synthesis problem
  - Review of compilation methods
  - Compiling framework in nearest-neighbor architectures

READING LIST

- **Quantum Approximate Optimization with Hard and Soft Constraints.** Hadfield, S., Wang, Z., Rieffel, E. G., O'Gorman, B., Venturelli, D., & Biswas, R. (2017, November). In *Proceedings of the Second International Workshop on Post Moores Era Supercomputing* (pp. 15-21). ACM.
- **From the quantum approximate optimization algorithm to a quantum alternating operator Ansatz** Hadfield, S, Z. Wang, B. O'Gorman, E. G. Rieffel, D. Venturelli, and R. Biswas. *arXiv preprint arXiv:1709.03489 (2017)*. Algorithms (2019).
  - **Best Paper Award MDPI Algorithms Journal**

Quantum Approximate Optimization Algorithm

- Gate-based quantum algorithm for QUBO optimization
- Iteratively alternates p times between applying two sets of operators: Mixing and Phase Shifting/Driving
  - Induce entanglement and the objective function
- Requires as many qubits as the size of the problem
- Requires polynomially many gates compared to the problem size
- Is an approximation algorithm:
  - One can theoretically prove that solution to any problem within a certain class using this algorithm will always be in a range (approximation ratio) of the true optimal



Quantum approximate optimization of the long-range Ising model with a trapped-ion quantum simulator
Guido Pagano, Aniruddha Bapat, Patrick Becker, Katherine S. Collins, Arinjoy De, Paul W. Hess, Harvey B. Kaplan, Antonis Kyprianidis, Wen Lin Tan, Christopher Baldwin, Lucas T. Brady, Abhinav Deshpande, Fangli Liu, Stephen Jordan, Alexey V. Gorshkov, Christopher Monroe
Proceedings of the National Academy of Sciences Oct 2020, 117 (41) 25396-25401; DOI: 10.1073/pnas.2006373117

- For MAXCUT of regular 3-degree graphs QAOA with p=1 has approximation ratio of 0.6942 vs. 2/3 of random guessing.
- For a satisfiability problem E3Lin2, QAOA with p=1 gave the best approximation ratio at the point.

Origins of the QAOA

MIT-CTP/4610

A Quantum Approximate Optimization Algorithm

Edward Farhi and Jeffrey Goldstone
Center for Theoretical Physics
Massachusetts Institute of Technology
Cambridge, MA 02139

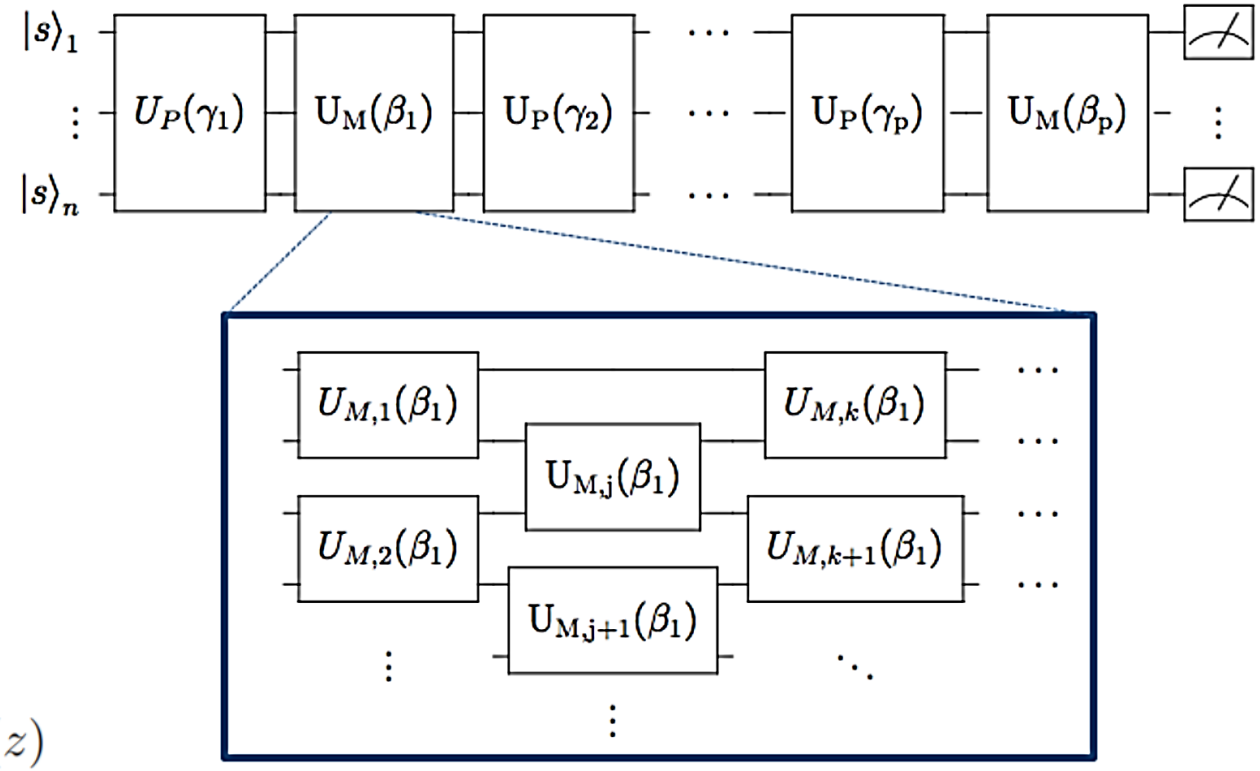
Sam Gutmann

Abstract

We introduce a quantum algorithm that produces approximate solutions for combinatorial optimization problems. The algorithm depends on an integer $p \geq 1$ and the quality of the approximation improves as p is increased. The quantum circuit that implements the algorithm consists of unitary gates whose locality is at most the locality of the objective function whose optimum is sought. The depth of the circuit grows linearly with p times (at worst) the number of constraints. If p is fixed, that is, independent of the input size, the algorithm makes use of efficient classical preprocessing. If p grows with the input size a different strategy is proposed. We study the algorithm as applied to MaxCut on regular graphs and analyze its performance on 2-regular and 3-regular graphs for fixed p. For $p = 1$, on 3-regular graphs the quantum algorithm always finds a cut that is at least 0.6924 times the size of the optimal cut.

$$|\beta,\gamma\rangle = Q_p(\beta,\gamma)\,|s\rangle$$

$$Q_p(\beta,\gamma) = U_{\mathrm{M}}(\beta_p)U_{\mathrm{P}}(\gamma_p)\cdots U_{\mathrm{M}}(\beta_1)U_{\mathrm{P}}(\gamma_1)$$



$$F_p(\gamma,\beta) = \langle\gamma,\beta|\,C\,|\gamma,\beta\rangle$$

$$M_p = \max_{\gamma,\beta} F_p(\gamma,\beta)$$

$$M_p \geq M_{p-1}$$

$$\lim_{p\to\infty} M_p = \max_z C(z)$$

QAOA

1. Design a binary optimization classical Hamiltonian **("phase separation")**
2. Design a unitary operator that can connect and allow jumps between different states **("mixing")**
3. Prepare a QAOA state for some parameters

$$|\boldsymbol{\beta}, \boldsymbol{\gamma}\rangle = Q_p(\boldsymbol{\beta}, \boldsymbol{\gamma}) \, |s\rangle$$

$$Q_p(\boldsymbol{\beta}, \boldsymbol{\gamma}) = U_{\mathrm{M}}(\beta_p) U_{\mathrm{P}}(\gamma_p) \cdots U_{\mathrm{M}}(\beta_1) U_{\mathrm{P}}(\gamma_1)$$

4. Measure the state in the computational value and compute the exp. value of C(z)

$$F_p(\boldsymbol{\gamma}, \boldsymbol{\beta}) = \langle \boldsymbol{\gamma}, \boldsymbol{\beta}| \, C \, |\boldsymbol{\gamma}, \boldsymbol{\beta}\rangle \qquad M_p \geq M_{p-1}$$

$$M_p = \max_{\boldsymbol{\gamma}, \boldsymbol{\beta}} F_p(\boldsymbol{\gamma}, \boldsymbol{\beta}) \qquad \lim_{p \to \infty} M_p = \max_z C(z)$$

5. Change the parameters if they are not proven optimal and repeat 3-4

Vanilla QAOA

$$O_{\mathrm{QUBO}}(q) = \sum_{i=1}^{N} a_i q_i + \sum_{i=1}^{N} \sum_{j=i+1}^{N} b_{ij} q_i q_j$$

Associate one qubit to each q_i

Initialize the registers in a superposition of all possible bitstring

$$|\psi\rangle_{in} = \left(2^{N/2}\right)^{-1} \sum_s |s\rangle$$

Assign to each superposed solution a phase proportional (arbitrary parameter γ_1) to its objective function value

$$|\psi\rangle_{ps(1)} = \left(2^{N/2}\right)^{-1} \sum_s e^{i\gamma_1 E_s} |s\rangle$$

Mix the amplitudes by a transverse field rotation $\exp(i\beta X)$ on each qubit (arbitrary parameter)

$$|\psi\rangle_{mix(1)} = \left(2^{N/2}\right)^{-1} \sum_s \mathrm{B}_{1s}(\beta_1, \gamma_1) e^{i\Gamma_{1s}(\beta_1, \gamma_1)} |s\rangle$$

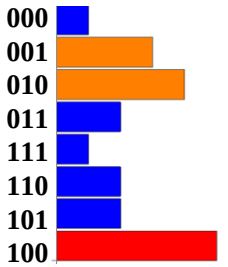
Phase separate again with new γ_2

$$|\psi\rangle_{ps(2)} = \left(2^{N/2}\right)^{-1} \sum_s \mathrm{B}_{1s}(\beta_1, \gamma_1) e^{i(\Gamma_{1s}(\beta_1, \gamma_1) + \gamma_2 E_s)} |s\rangle$$

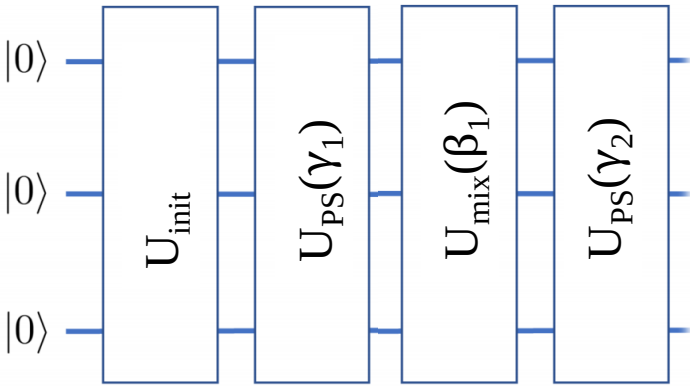
$$|\psi\rangle_{mix(2)} = \left(2^{N/2}\right)^{-1} \sum_s \mathrm{B}_{2s}(\beta_1, \gamma_1, \beta_2, \gamma_2) e^{i(\Gamma_{2s}(\beta_1, \gamma_1, \beta_2, \gamma_2))} |s\rangle$$

After having repeated the algorithm p times do measure in the computational base the expectation value of the objective function

$$\langle\psi|\mathrm{O}|\psi\rangle_{out} = \sum_s \mathrm{O}_s |\mathrm{B}_{ps}|^2$$



Quantum Approximate Optimization Algorithm: Example



Initialization operator

Hadamard Gates

$$|\psi\rangle_{in} = \frac{1}{\sqrt{2^N}} \sum_{n=1}^{2^N} |\text{solution(n)}\rangle$$
$$= \left(2^{N}/_{2}\right)^{-1} \sum_{s} |s\rangle$$

Mix the amplitudes by a transverse field rotation $\exp(i\beta X)$ on each qubit (arbitrary parameter)

$$|\psi\rangle_{mix(1)}$$
$$= \left(2^{N}/_{2}\right)^{-1} \sum_{s} \mathrm{B}_{1s}(\beta_1, \gamma_1) e^{i\Gamma_{1s}(\beta_1,\gamma_1)} |s\rangle$$

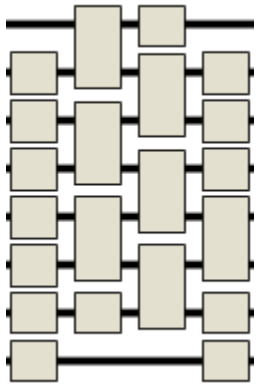
Now if you measure, the probability of a bitstring depends both on γ and β in a non-linear way.

Phase separation operator dependent on a parameter γ_1

$$\underbrace{\exp(i\beta Z_1 Z_2)}|s_1 s_2\rangle = e^{i\gamma_1 s_1 s_2}|s_1 s_2\rangle$$

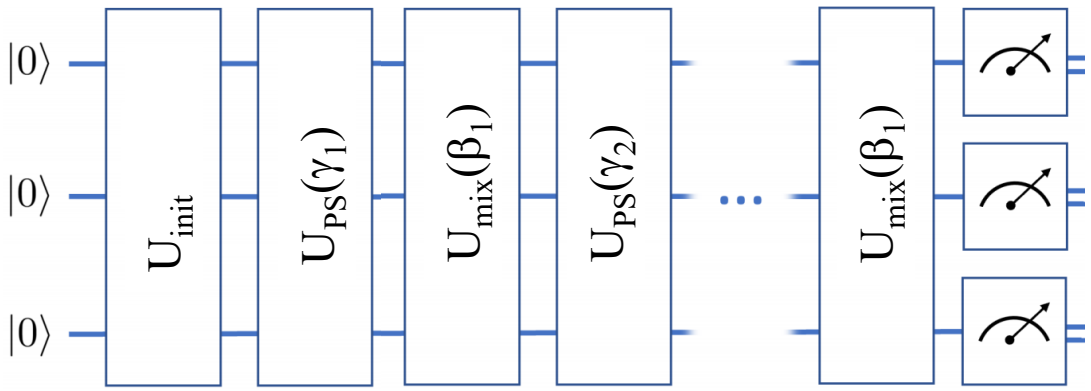
Logical 2-qubit gate representing the Ising interaction

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



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

Quantum Approximate Optimization Algorithm: Example



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 $\left|\mathrm{B}_{2s}(\beta_1,\gamma_1,\beta_2,\gamma_2,\ldots,\beta_p,\gamma_p)\right|^2$ to find the optimal unknown solution s^*

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

For $p \to \infty$ 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)

QAOA for Constrained Optimization Problems

QAOA for Constrained Combinatorial Optimization

** Stay in the computational subspace!*

Associate one qubit to each q_i

Initialize the registers with a candidate solution found through genetic algorithm or greedy search

$|\psi\rangle_{in}=|011010101100\rangle$

Mix the system by generating a superposition of the initial solution with all possible others (arbitrary parameter)

$|\psi\rangle_{mix}=\alpha|011010101100\rangle+\sum_k \beta_k|\phi_k\rangle$

Assign to each superposed solution a phase proportional (arbitrary parameter) to its objective function value

$|\psi\rangle_{mix}=\alpha e^{i\gamma E_{in}}|011010101100\rangle + \sum_k \beta_k e^{i\gamma E_k}|\phi_k\rangle$

Add phases again with new γ

Mix with new β^*

DIFFICULT

Only the logical subspace

VS

$|\psi\rangle_{mix}=\sum_s \beta_s|s\rangle$

All 2^N bitstrings

The Problem of Hard Constraints

$$O_{\text{QUBO}}(q) = \sum_{i=1}^{N} a_i q_i + \sum_{i=1}^{N} \sum_{j=i+1}^{N} b_{ij} q_i q_j$$

$$\sum_i q_i = k \xrightarrow{\text{Penalty Function}} C\left(\sum_i q_i - k\right)^2$$

Difficult to scale, does not guarantee results, hardness is large softness

Possible solution for these constraints: *XY*-Mixers.

What you would want is to start from a classical bitstring, and then be able to "mix it" coherently in the subspace where the constraint is satisfied

Enforcing the same number of bits=1 is the same as doing two spin-flips

$$XY = s^+ s^- + s^- s^+$$

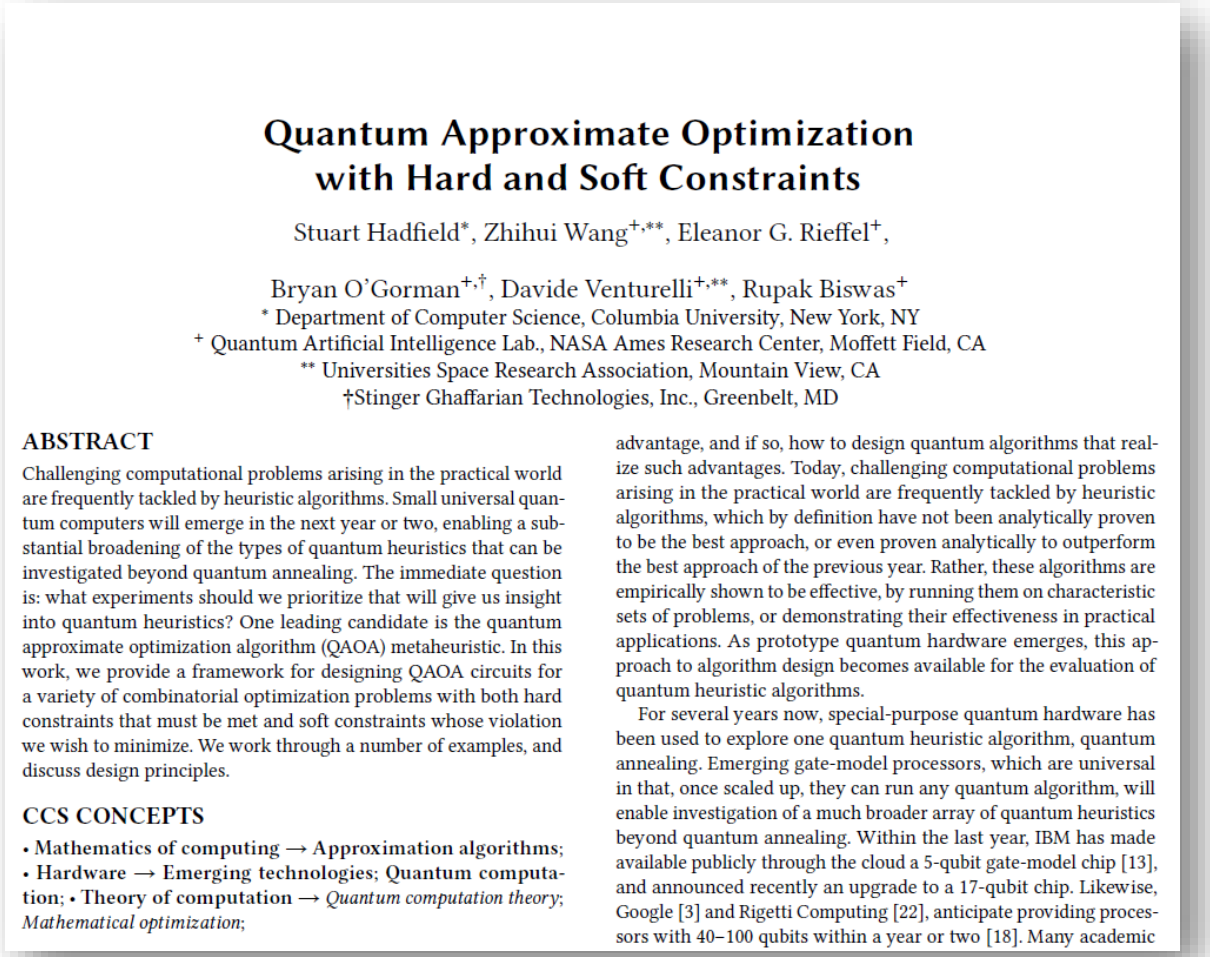
$$|001\rangle \xrightarrow{XY(2,3)} a|001\rangle + b|010\rangle \xrightarrow{XY(2,3)} a'|001\rangle + b'|010\rangle + c'|100\rangle$$

$$\mathrm{XY}(\theta) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\frac{\theta}{2} & i\sin\frac{\theta}{2} & 0 \\ 0 & i\sin\frac{\theta}{2} & \cos\frac{\theta}{2} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

QAOA Applications

- Maximum Cut
- Max-SAT, Min-SAT, NAE-SAT
- Set Splitting
- MaxE3LIN2
- Max-ColorableSubgraph
- Graph Partitioning
- Maximum Bisection
- Max Vertex k-Cover
- MaxIndependentSet
- MaxClique
- MinVertexCover
- MaxSetPacking
- MinSetCover
- TSP
- SMS with various metrics and constraints
- …

Objective Function: Soft Constraints
Feasible States: Hard Constraints



Quantum Approximate Optimization with Hard and Soft Constraints

Stuart Hadfield*, Zhihui Wang+,**, Eleanor G. Rieffel+,
Bryan O'Gorman+,†, Davide Venturelli+,**, Rupak Biswas+
* Department of Computer Science, Columbia University, New York, NY
+ Quantum Artificial Intelligence Lab., NASA Ames Research Center, Moffett Field, CA
** Universities Space Research Association, Mountain View, CA
†Stinger Ghaffarian Technologies, Inc., Greenbelt, MD

ABSTRACT

Challenging computational problems arising in the practical world are frequently tackled by heuristic algorithms. Small universal quantum computers will emerge in the next year or two, enabling a substantial broadening of the types of quantum heuristics that can be investigated beyond quantum annealing. The immediate question is: what experiments should we prioritize that will give us insight into quantum heuristics? One leading candidate is the quantum approximate optimization algorithm (QAOA) metaheuristic. In this work, we provide a framework for designing QAOA circuits for a variety of combinatorial optimization problems with both hard constraints that must be met and soft constraints whose violation we wish to minimize. We work through a number of examples, and discuss design principles.

CCS CONCEPTS

• **Mathematics of computing** → **Approximation algorithms**; • **Hardware** → **Emerging technologies**; **Quantum computation**; • **Theory of computation** → *Quantum computation theory*; *Mathematical optimization*;

advantage, and if so, how to design quantum algorithms that realize such advantages. Today, challenging computational problems arising in the practical world are frequently tackled by heuristic algorithms, which by definition have not been analytically proven to be the best approach, or even proven analytically to outperform the best approach of the previous year. Rather, these algorithms are empirically shown to be effective, by running them on characteristic sets of problems, or demonstrating their effectiveness in practical applications. As prototype quantum hardware emerges, this approach to algorithm design becomes available for the evaluation of quantum heuristic algorithms.

For several years now, special-purpose quantum hardware has been used to explore one quantum heuristic algorithm, quantum annealing. Emerging gate-model processors, which are universal in that, once scaled up, they can run any quantum algorithm, will enable investigation of a much broader array of quantum heuristics beyond quantum annealing. Within the last year, IBM has made available publicly through the cloud a 5-qubit gate-model chip [13], and announced recently an upgrade to a 17-qubit chip. Likewise, Google [3] and Rigetti Computing [22], anticipate providing processors with 40–100 qubits within a year or two [18]. Many academic

Alternating Operator Ansatz

$$Q_p(\boldsymbol{\beta},\boldsymbol{\gamma}) = U_{\mathrm{M}}(\beta_p)U_{\mathrm{P}}(\gamma_p)\cdots U_{\mathrm{M}}(\beta_1)U_{\mathrm{P}}(\gamma_1)$$

$$|\boldsymbol{\beta},\boldsymbol{\gamma}\rangle = Q_p(\boldsymbol{\beta},\boldsymbol{\gamma})\,|s\rangle$$

$e^{-i\beta H_{\mathrm{M}}}$

$e^{-i\gamma H_{\mathrm{P}}}$

Some initial state respecting:

- It is a superposition of several solutions in the feasible subspace
- It can be prepared efficiently

Some unitary respecting:

- Preserve the feasible subspace
- Provide all-to-all nonzero transitions between all feasible states
- Non-necessarily time evolution of a local Hamiltonian

Some unitary respecting:

- Is diagonal in the computational basis
- The spectrum of H_P encodes the objective function

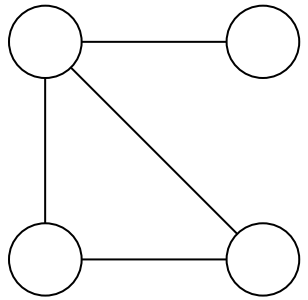
$$H_f\,|\mathbf{x}\rangle = f(\mathbf{x})\,|\mathbf{x}\rangle$$

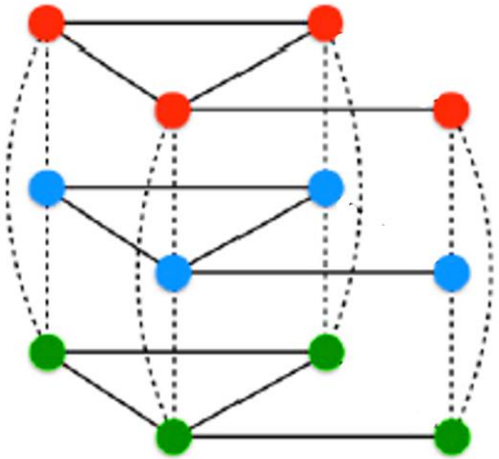
Alternating Operator Ansatz

Node u is colored by c

$X_{u,c}=1$

$$m - \sum_{\{u,v\}\in E} \sum_{a=1}^{k} x_{u,a}x_{v,a}$$





Phase Separator (QUBO objective function)

$$H'_{\mathrm{P}} = \frac{4-\kappa}{4} mI + \frac{1}{4} \sum_{\{u,v\}\in E} \sum_{a=1}^{\kappa} (Z_{u,a} + Z_{v,a} - Z_{u,a}Z_{v,a})$$

Initial state:

$$|W\rangle_v = \frac{1}{\sqrt{k}} (|100\cdots0\rangle + |010\cdots0\rangle + |0\cdots01\rangle)$$

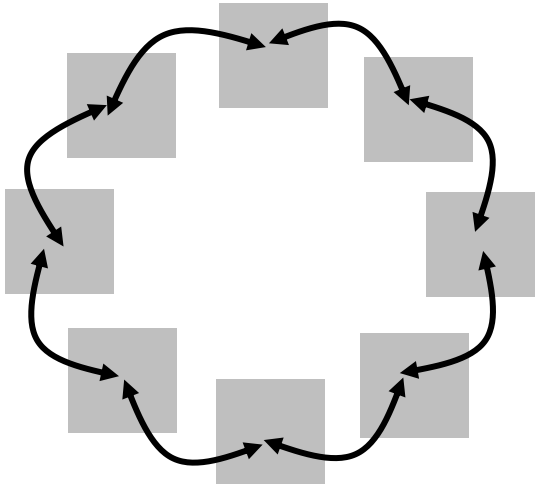
- *Babbush (2017)*
- *Verstraete (2009)*
- *Wang (2009)*
- *Childs (2002)*
- …

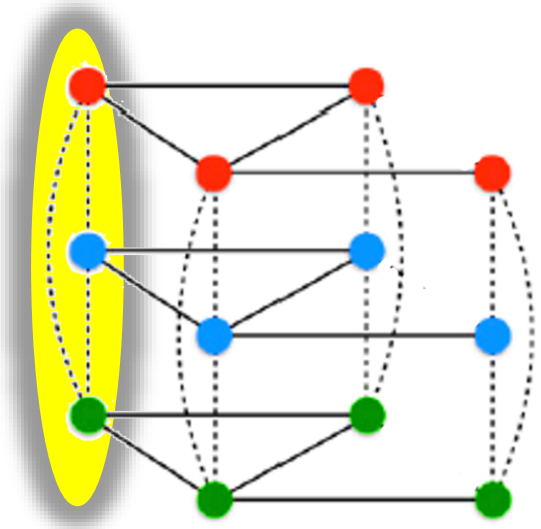
Engineering Mixing Operators

$$H_{\text{ring}}^{(\text{enc})} = \sum_{a}^{d} (X_a X_{a+1} + Y_a Y_{a+1})$$

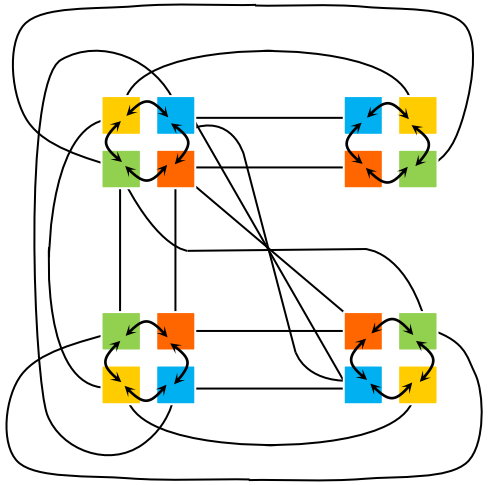
$\exp(iH_{\text{ring}})$ is difficult to implement

Respects the Hamming Weight constraint



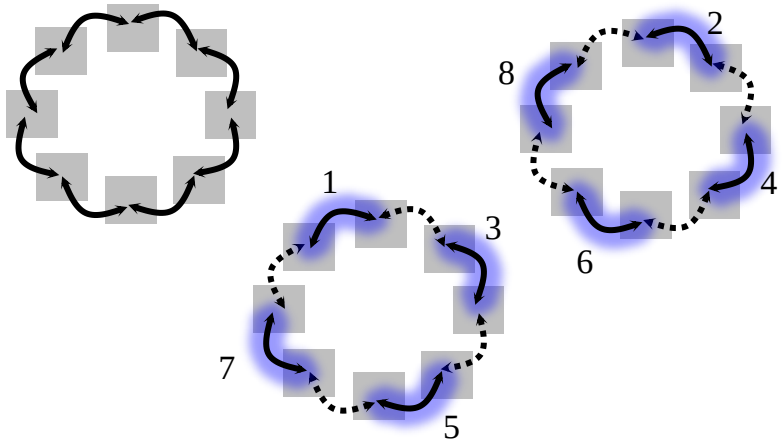


3-coloring



4-coloring

Desing Freedom and Tradeoffs



$$H_{\text{ring}}^{(\text{enc})} = \sum_{a}^{d} (X_a X_{a+1} + Y_a Y_{a+1})$$

Respects the Hamming Weight constraint

$\exp(iH_{\text{ring}})$ is difficult to implement

$$U_{\text{M}} = \prod_{v=1}^{n} U_{v,\text{parity}}^{(\text{enc})}$$

$\prod_{a \in \text{parity}} \text{Exp}(iX_aX_{a+1}+Y_aY_{a+1})$

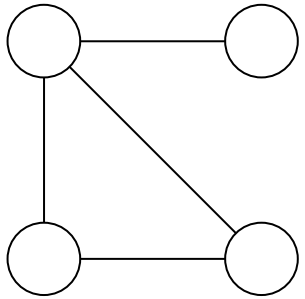
$U_M=[U_1U_3U_5U_7]\,[U_2U_4U_6U_8]\,[U_1U_3U_5U_7]\,[U_2U_4U_6U_8]\ldots$

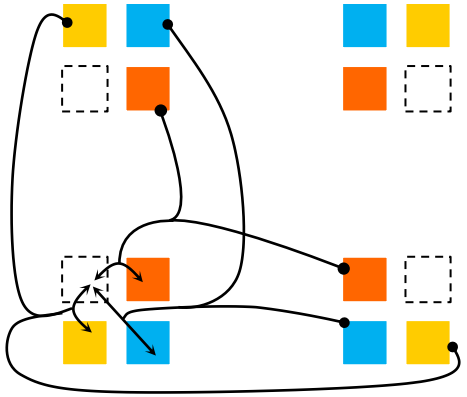
This couples only distance 2;
has to be repeated k/2 times

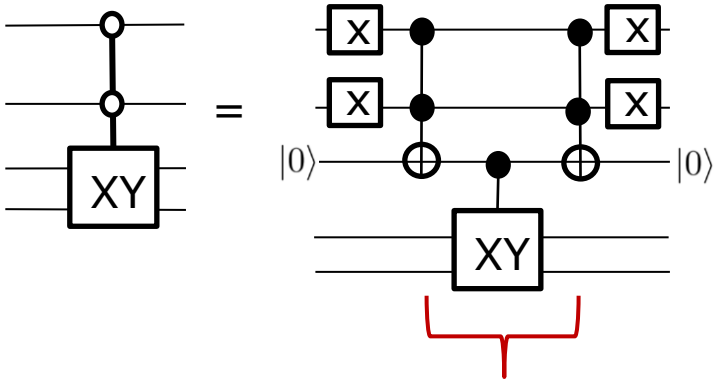
All these 2-qubit $k^2/2$ gates need to be scheduled

Other Mixers (controlled XY)

Finding the largest induced subgraph colorable by k colors







Still needs to be compiled to 2 qubit gates

All these gates need to be scheduled

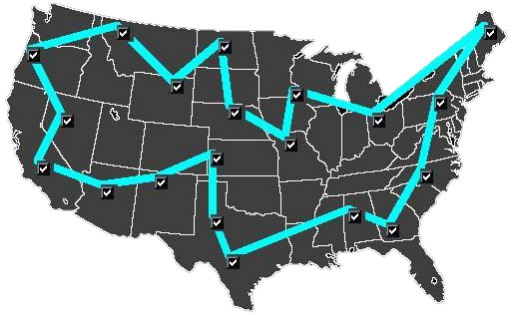
Node u is colored by c or uncolored (c=0)

$X_{u,c}=1$

$$C = m - \sum_v x_{v,0} \rightarrow H_C = \frac{1}{2}\sum_v Z_{v,0}$$

Mixers Navigation&Scheduling

In **traveling salesman** encoding



X_{vj}=1 if city v is visited as j^{th}

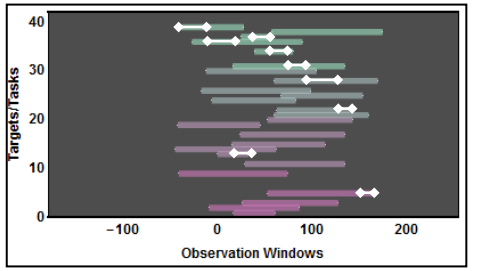
$$\sum_{\{u,v\}\in E} d_{u,v} \sum_{j=1}^{n} \left(x_{u,j}x_{v,j+1} + x_{v,j}x_{u,j+1}\right)$$

$$H^{(\mathrm{enc})}_{\mathrm{PS},\{i,j\},\{u,v\}} = S^{+}_{u,i}S^{+}_{v,j}S^{-}_{u,j}S^{-}_{v,i} + S^{-}_{u,i}S^{-}_{v,j}S^{+}_{u,j}S^{+}_{v,i},$$

(partitioned using edge coloring and parity ≈$(n-1)n^2/4$ mixers)

(needs to be repeated n(n-1)/2 times for all-to-all)

In **single machine scheduling**



X_{jt}=1 if job j starts at time t

$$C = \sum_{j} w_j \sum_{(d_j - p_j) < t < h} x_{j,t}(t + p_j - d_j)$$

$$H^{(\mathrm{enc})}_{\mathrm{TS},t,\{i,j\}} = S^{+}_{i,t+p_j}S^{+}_{j,t}S^{-}_{i,t}S^{-}_{j,t+p_i} + S^{+}_{i,t}S^{+}_{j,t+p_i}S^{-}_{i,t+p_j}S^{-}_{j,t}$$

(But if we add release dates then we need controls on the no-overlap constraint)

Zoology of Ansatze

(See Hadfield et al 2018 – «Quantum Alternating Operator Ansatz»)

Bitflip mixers
- Maximum Cut
- Max-SAT, Min-SAT, NAE-SAT
- Set Splitting
- MaxE3LIN2

...

XY mixers
- Max-ColorableSubgraph
- Graph Partitioning
- Maximum Bisection
- Max Vertex k-Cover

...

Controlled Bitflip mixers
- MaxIndependentSet
- MaxClique
- MinVertexCover
- MaxSetPacking
- MinSetCover

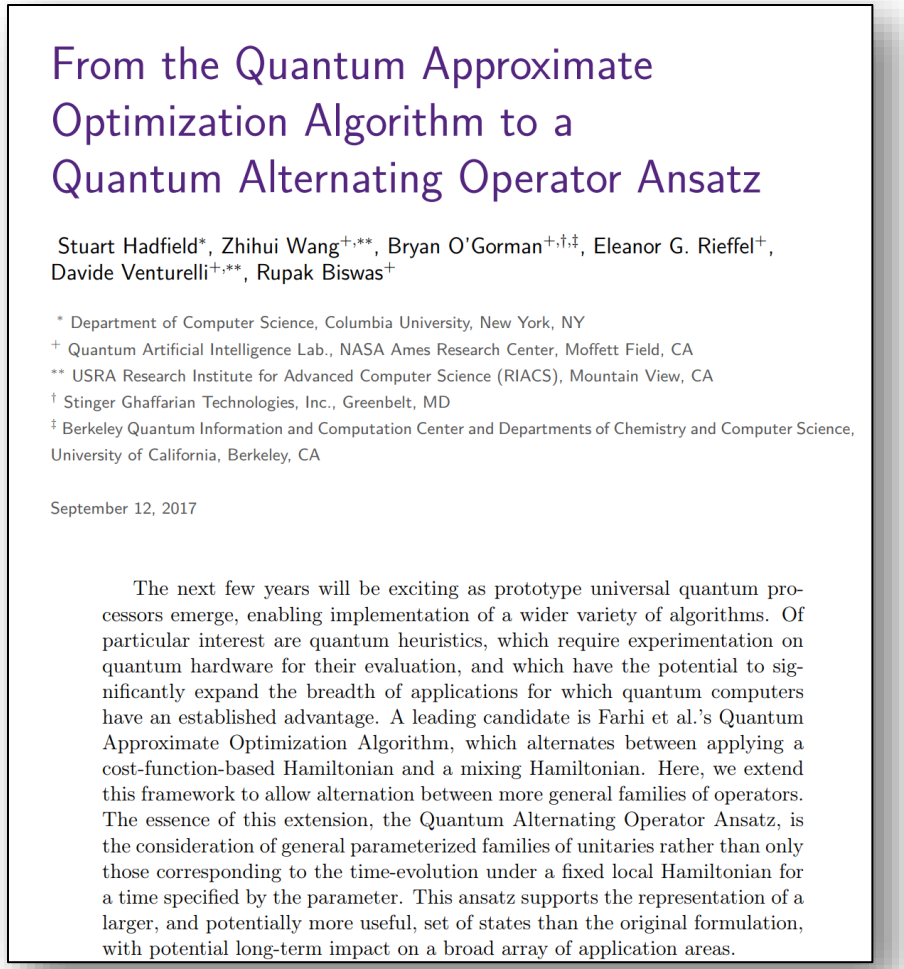
...

Controlled XY mixers
- Max-k-ColorableInducedSubgraph
- MinGraphColoring
- MinCliqueCover

...

Permutation mixers
- TSP
- SMS with various metrics and constraints

...



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

Stuart Hadfield*, Zhihui Wang+,**, Bryan O'Gorman+,†,‡, Eleanor G. Rieffel+, Davide Venturelli+,**, Rupak Biswas+

* Department of Computer Science, Columbia University, New York, NY
+ Quantum Artificial Intelligence Lab., NASA Ames Research Center, Moffett Field, CA
** USRA Research Institute for Advanced Computer Science (RIACS), Mountain View, CA
† Stinger Ghaffarian Technologies, Inc., Greenbelt, MD
‡ Berkeley Quantum Information and Computation Center and Departments of Chemistry and Computer Science, University of California, Berkeley, CA

September 12, 2017

The next few years will be exciting as prototype universal quantum processors emerge, enabling implementation of a wider variety of algorithms. Of particular interest are quantum heuristics, which require experimentation on quantum hardware for their evaluation, and which have the potential to significantly expand the breadth of applications for which quantum computers have an established advantage. A leading candidate is Farhi et al.'s Quantum Approximate Optimization Algorithm, which alternates between applying a cost-function-based Hamiltonian and a mixing Hamiltonian. Here, we extend this framework to allow alternation between more general families of operators. The essence of this extension, the Quantum Alternating Operator Ansatz, is the consideration of general parameterized families of unitaries rather than only those corresponding to the time-evolution under a fixed local Hamiltonian for a time specified by the parameter. This ansatz supports the representation of a larger, and potentially more useful, set of states than the original formulation, with potential long-term impact on a broad array of application areas.

The Flexible Design of NISQ Quantum Optimization Algorithms

Vanilla QAOA (Fahri 2014) and the QAOAnsatz (Hadfield 2017) were just the start of the field of modern Quantum Optimization Approaches

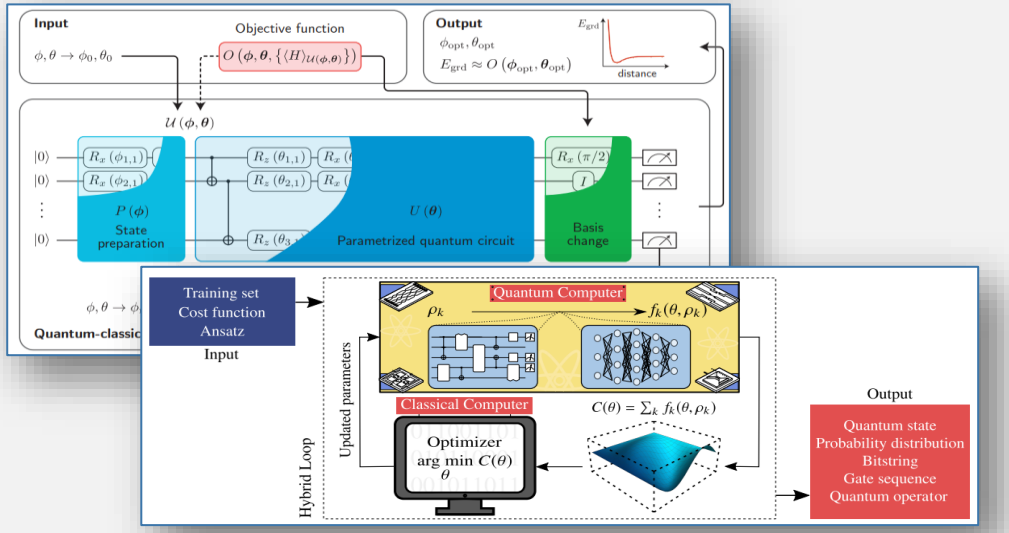
Variations:

- **Incomplete/Approximate:** e.g. mixing of a limited number of variables randomly selected.
- **Adaptive:** e.g. changing the circuit at runtime based on parameter exploration.
- **Unstructured:** e.g. the cost function could be evaluated only by classical hardware and is not in the ansatz, like learning in a neural network.
- **Overparametrized:** e.g. some gates might have offset angles
- **Digital-Analog:** i.e. global pulsing techniques that generate multi-qubit long range interactions.

Recent Review Articles:

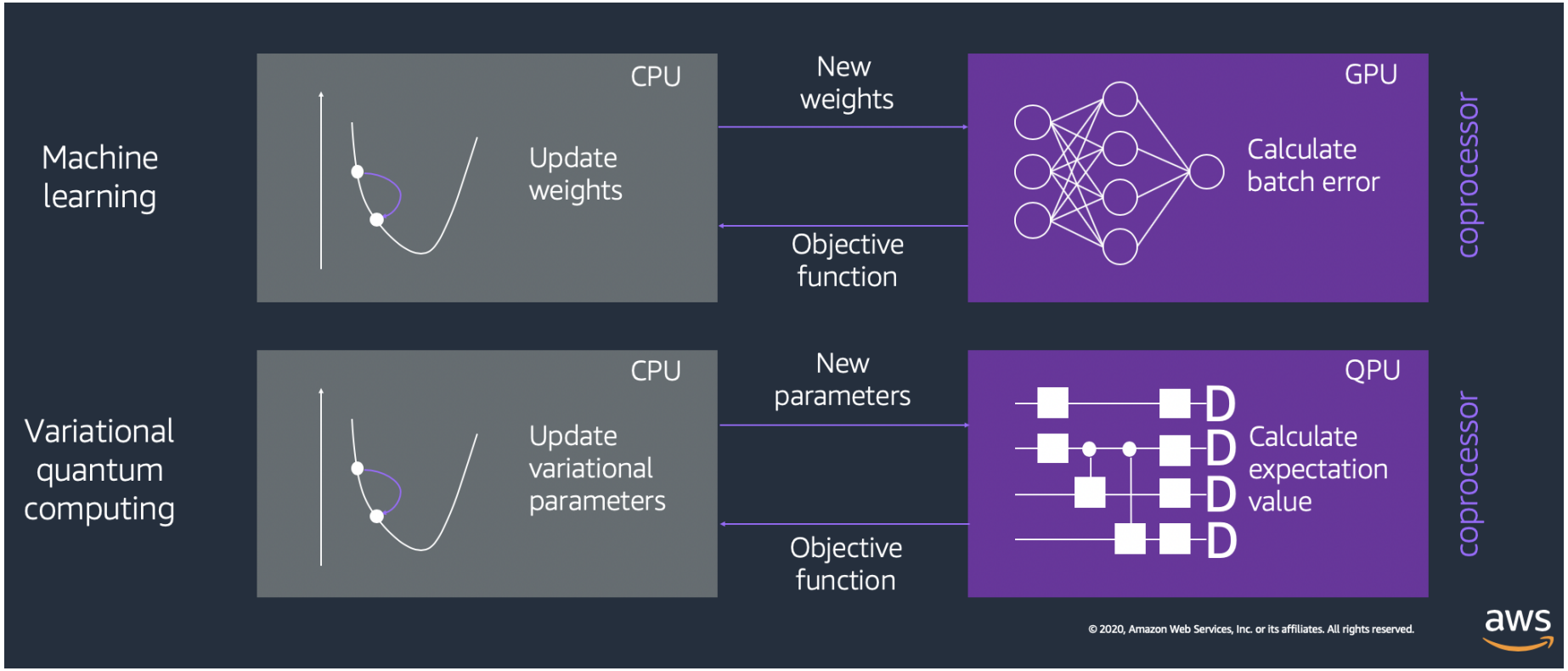
Noisy intermediate-scale quantum (NISQ) algorithms
Bharti et al. (Jan 2021) – arXiv:2101.08448

Variational Quantum Algorithms
Cerezo et al. (Dec 2020) – arXiv:2012.09265



Variational Quantum Computing – AWS view

Near-term quantum computers will be used as co-processors



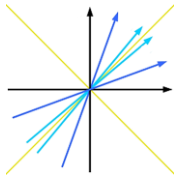
QAOA in the "Real World"

The Gate Synthesis Problem

Barenco et al. (1995)
Kraus, Cirac (2001)
Vatan, Williams (2003)

Quantum Circuits can be composed by single and two-qubit gates of universal set*
CNOT, $R_y(q)$ and $R_z(a)$

Each single qubit gate can be decomposed by single qubit rotations.
U1= $R_z(a)$ $R_y(b)$ $R_z(g)$ e^{if}

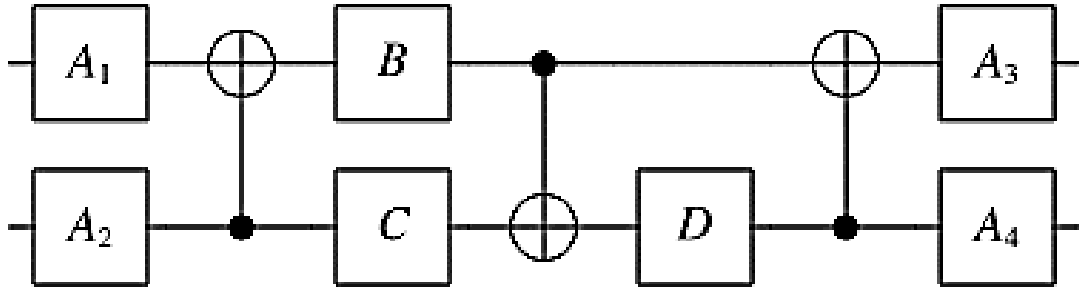


Each two qubit gate is reversible and it is representable by a Unitary Matrix.

R_Z gates can be «virtually» compiled.
(McKay 2017 and refs)

* active research to natively support multi-qubit gates

$U \equiv$



Maximum number of elementary 1-qubit gates: **15**
Maximum number of CNOTs: **3**
Maximum depth assuming R_Y, R_Z and simplifications: **11**

SWAP-Compilation (review)

Performance of algorithms in NISQ will depend on aspects such as gate fidelities, parallelization, idle time, crosstalks..

Different Metrics to optimize correlate to final performance:

- Total Quantum Factor
- Quantum Volume
- Number of Two-Qubit Gates
- **Makespan**

Guerreschi and Park (2018). Two-step approach to scheduling quantum circuits. *arXiv preprint arXiv:1708.00023*.

Khatri, Sumeet, et al. "Quantum assisted quantum compiling." *arXiv preprint arXiv:1807.00800* (2018).

Li, G., Ding, Y., & Xie, Y. (2018). Tackling the Qubit Mapping Problem for NISQ-Era Quantum Devices. *arXiv preprint arXiv:1809.02573 (2018)*.

Oddi, Angelo, and Riccardo Rasconi. "Greedy Randomized Search for Scalable Compilation of Quantum Circuits." *International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research*. Springer, Cham, (2018.

…

Example: MaxCut

$S_i=\pm 1$ Defines the cut

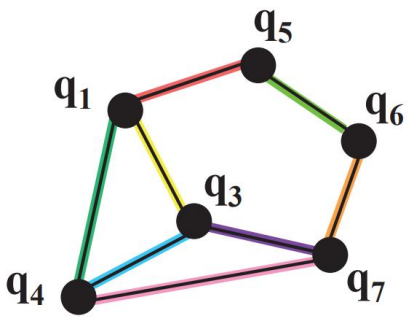
$$U = \frac{1}{2} \sum_{(i,j)\in E} (1 - s_i s_j)$$ Counts the edges in the cut

$\sum_i X_i$ Mixes the two partitions

$U_{PS}=\Pi_{<jk>}Exp(ibZ_jZ_k)$

$U_M=\Pi_j Exp(igX_j)$

Interaction graph obtained from quadratic objective function (MAXCUT)

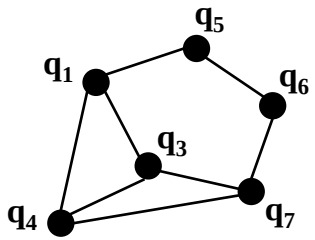


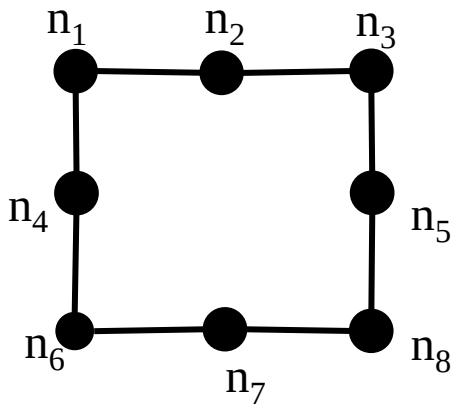
PS1	*MX*	*PS2*
P-S(q_1, q_4)	MIX(q_1)	P-S(q_1, q_4)
P-S(q_1, q_3)	MIX(q_3)	P-S(q_1, q_3)
P-S(q_3, q_4)	MIX(q_4)	P-S(q_3, q_4)
P-S(q_3, q_7)	MIX(q_5)	P-S(q_3, q_7)
P-S(q_4, q_7)	MIX(q_6)	P-S(q_4, q_7)
P-S(q_6, q_7)	MIX(q_7)	P-S(q_6, q_7)
P-S(q_5, q_6)		P-S(q_5, q_6)
P-S(q_1, q_5)		P-S(q_1, q_5)

- Every edge is a gate that needs to be executed (in arbitrary order)
- The same graph has to be executed multiple times (*p rounds*).
- Every qubit has to complete all the gates of round *p* before being involved in *p+1*

Circuit Execution Schedule

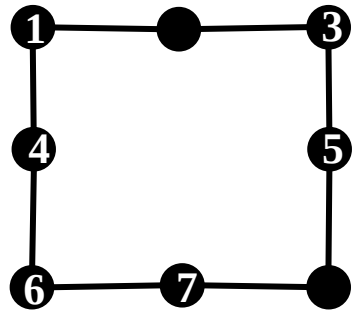
Interaction graph obtained from quadratic objective function (MAXCUT)





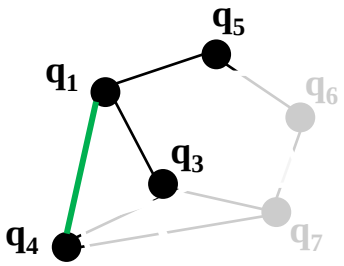
Initial assignment

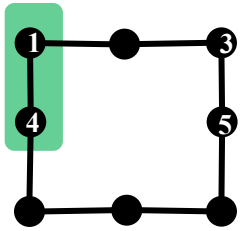
$\mathbf{q_i \rightarrow n_i}$



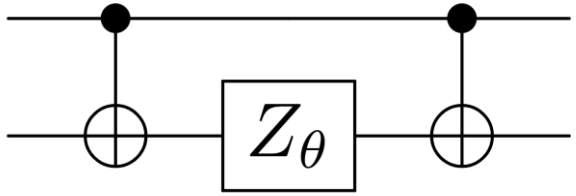
- Every edge is a gate that needs to be executed (in arbitrary order)
- The same graph has to be executed multiple times (*p rounds*).
- Every qubit has to complete all the gates of round *p* before being involved in *p+1*

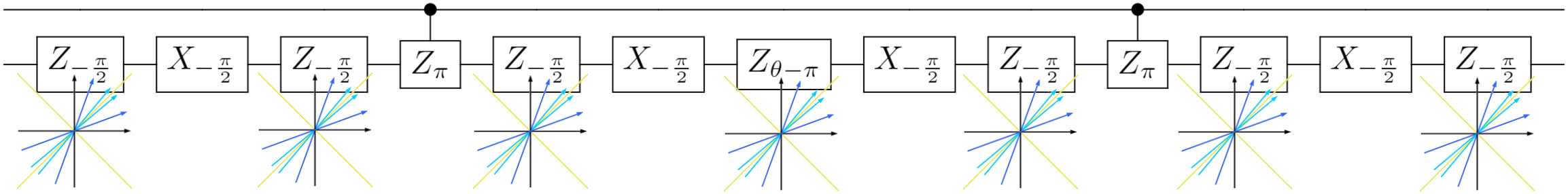
Circuit Execution Schedule

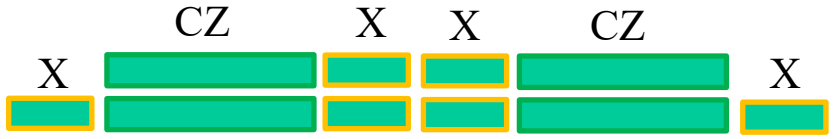




$\mathrm{Exp}(i\theta Z_1 Z_4)$ ZZ-Evolution Gate

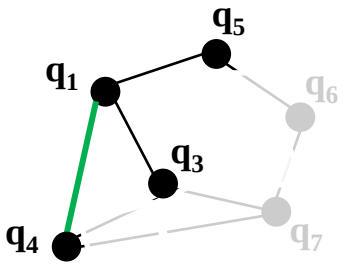


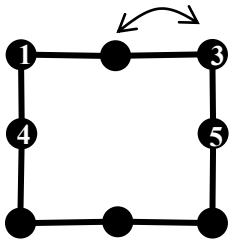




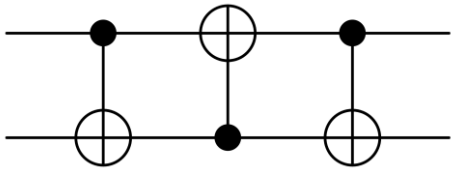
Duration $2\tau_2+4\tau_1$

Circuit Execution Schedule

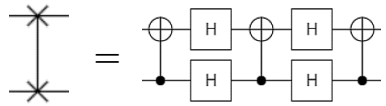


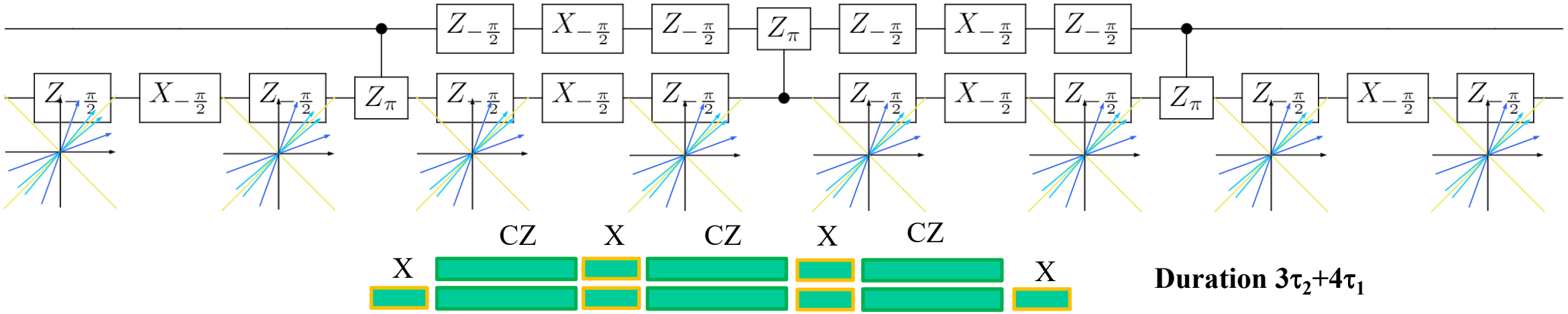


SWAP $|\Psi\rangle_3 \leftrightarrow |0\rangle$



From Unidirectional CNOTs to SWAP



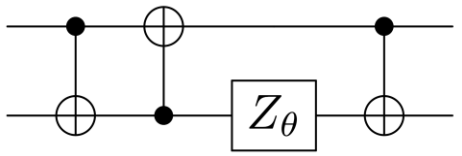


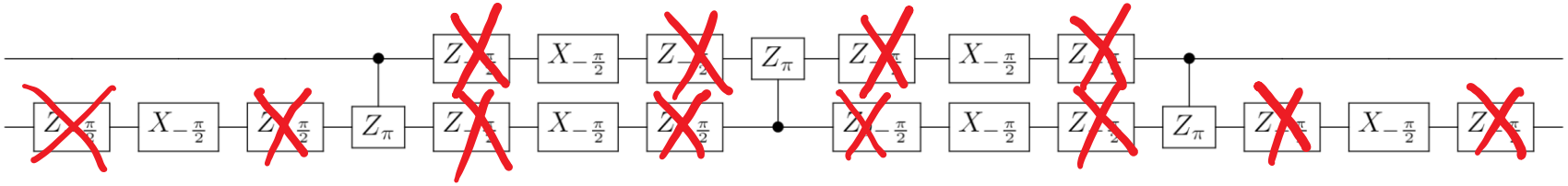
Duration $3\tau_2+4\tau_1$

Circuit Execution Schedule

SWAPS can also be inserted as part of the UZZ interaction without the need to be sequential.

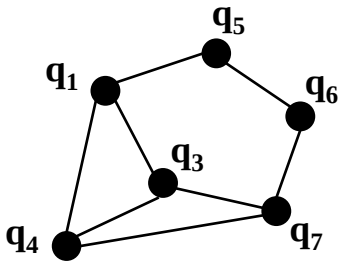
SWAP+ZZ-Evolution Gate

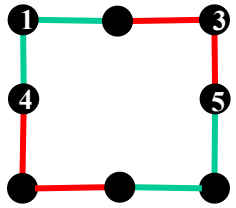




Duration $3\tau_2+4\tau_1$ (same as SWAP)

Circuit Execution Schedule

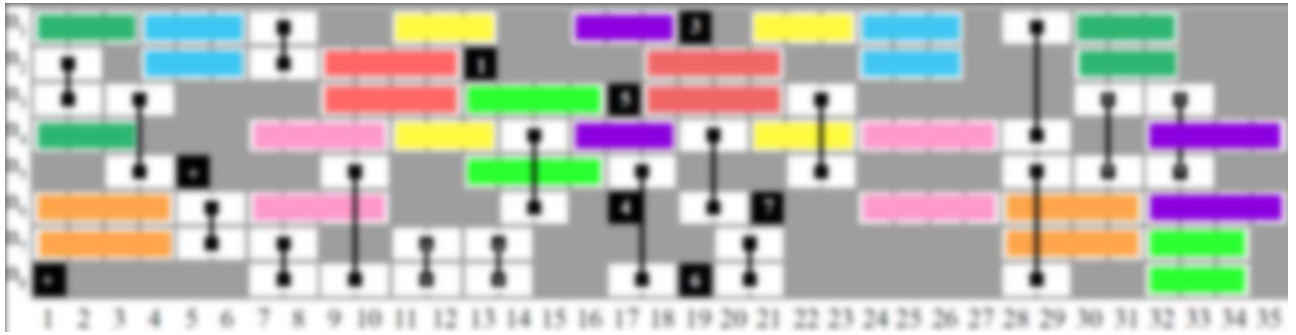




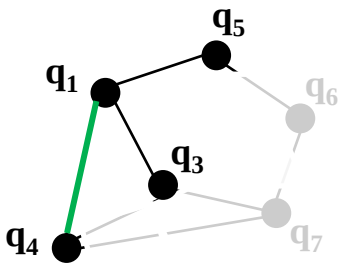
fast P-S (τ=3)
slow P-S (τ=4)
Swap (τ=2)

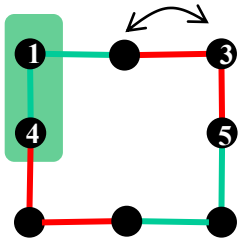
Benchmark presented at ICAPS17

Objective: finding the makespan-minimizing Gantt Schedule for p=1, p=2, N=8, N=21

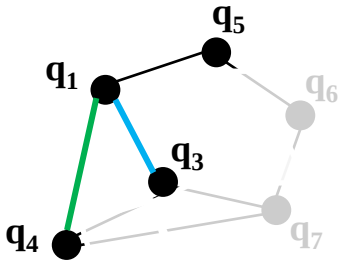


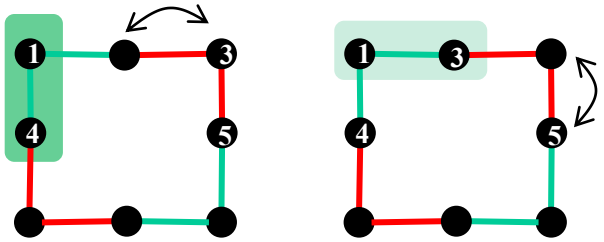
Circuit Execution Schedule



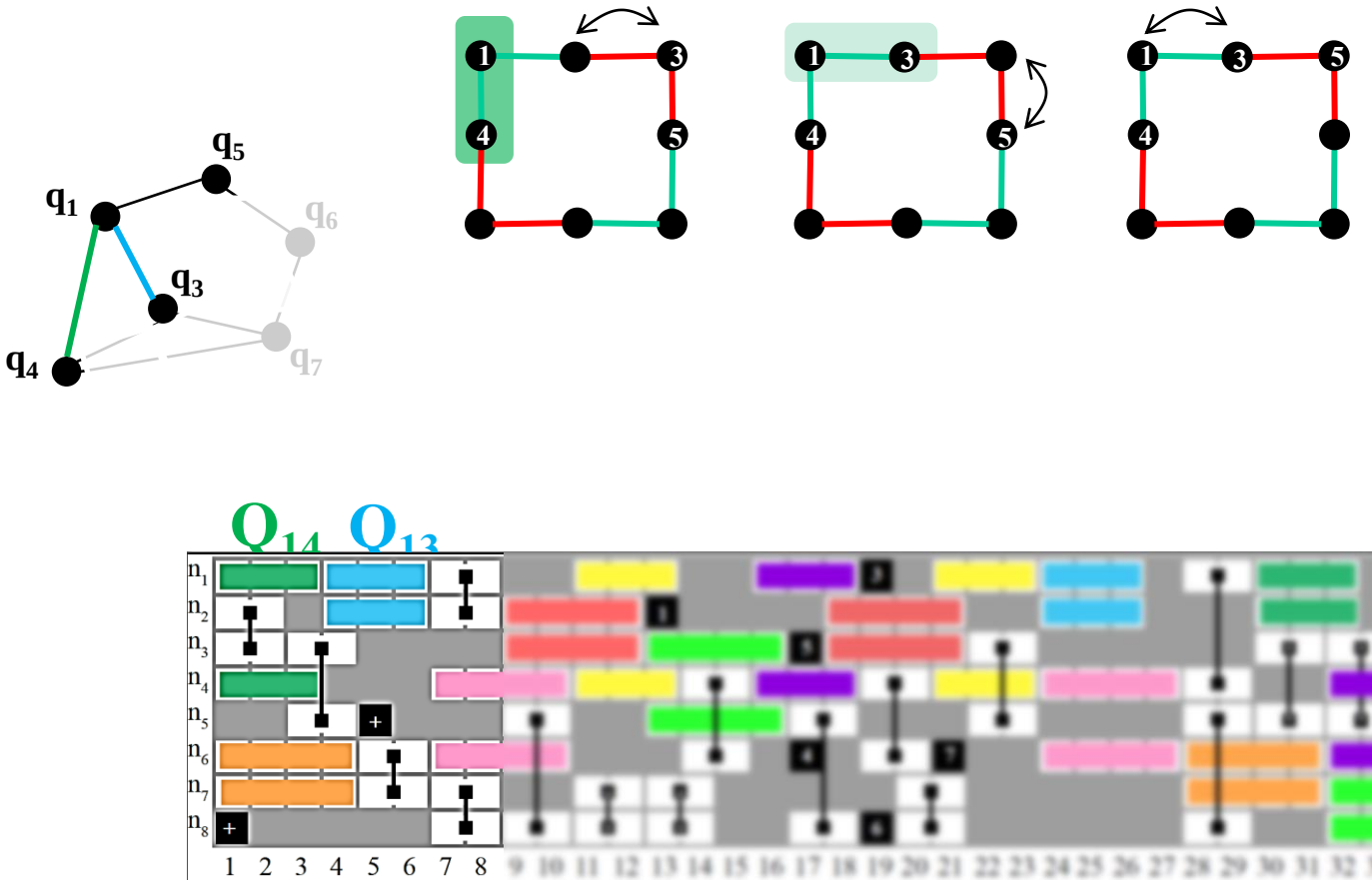


Circuit Execution Schedule

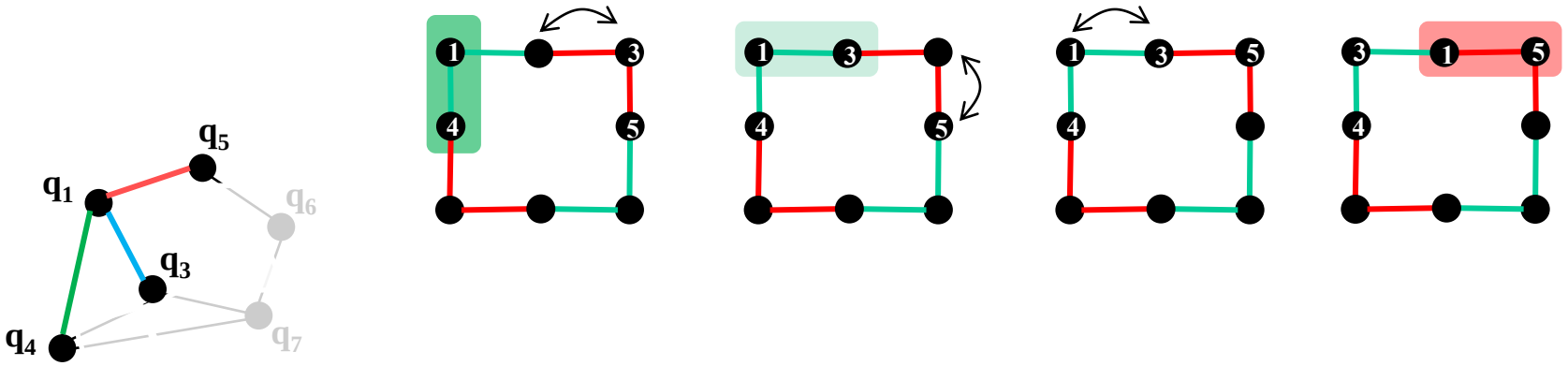


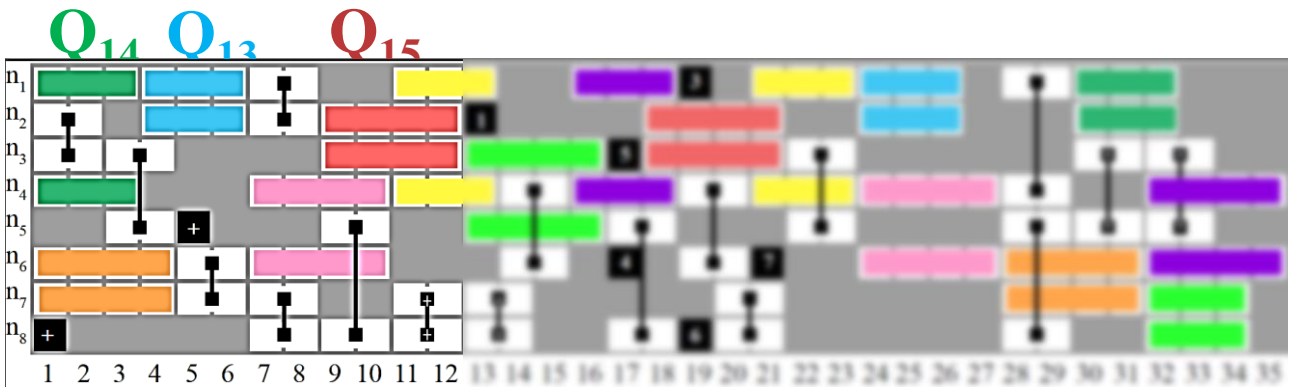


Circuit Execution Schedule

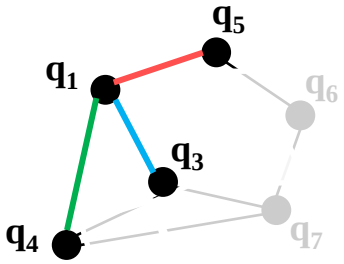


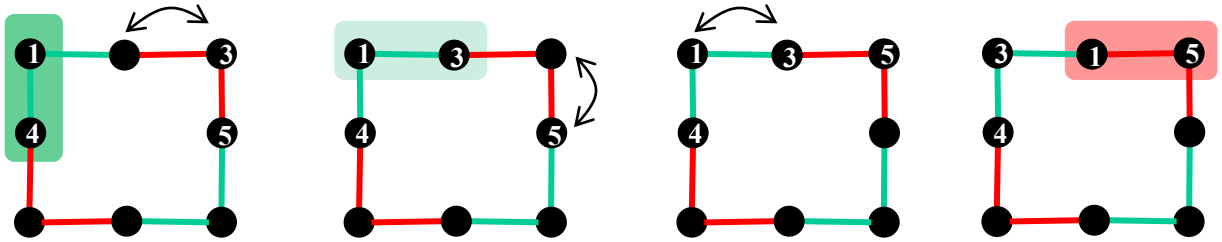
Circuit Execution Schedule



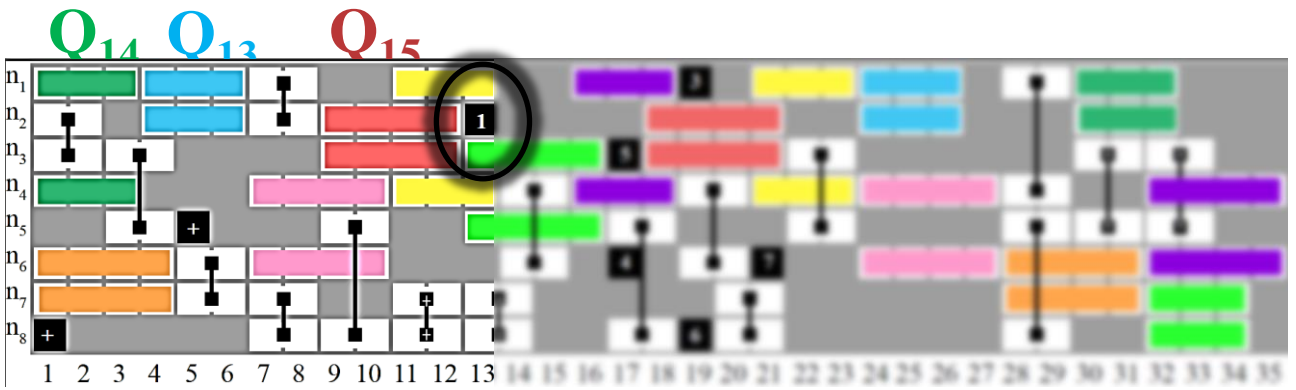


Circuit Execution Schedule

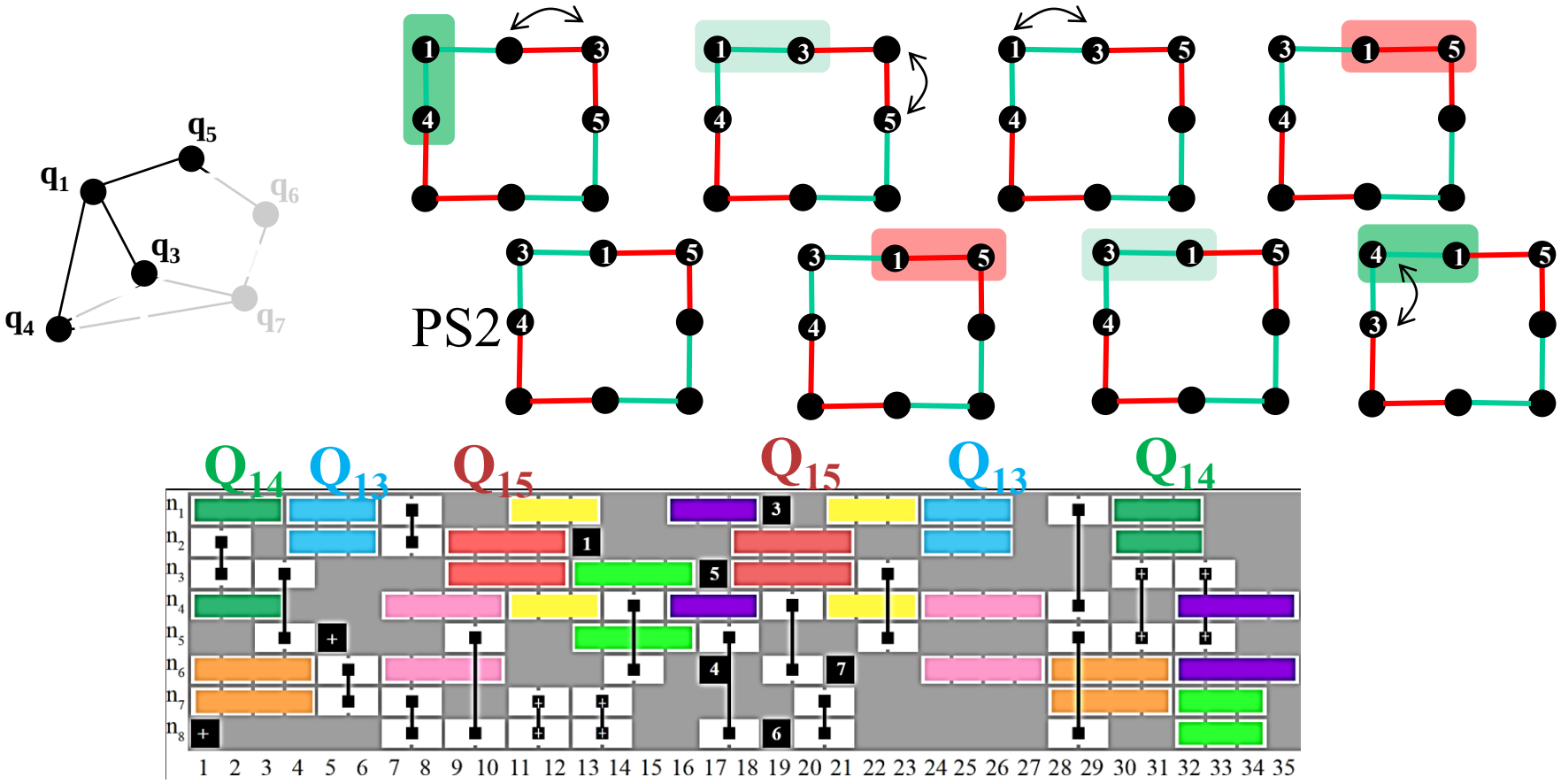




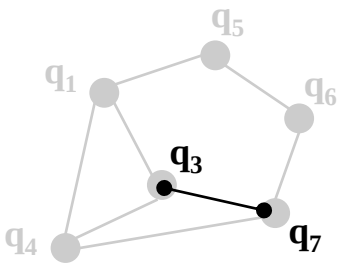
All actions of round 1 are completed – qubit can be mixed.
Qubit 1 can start participating to round 2.

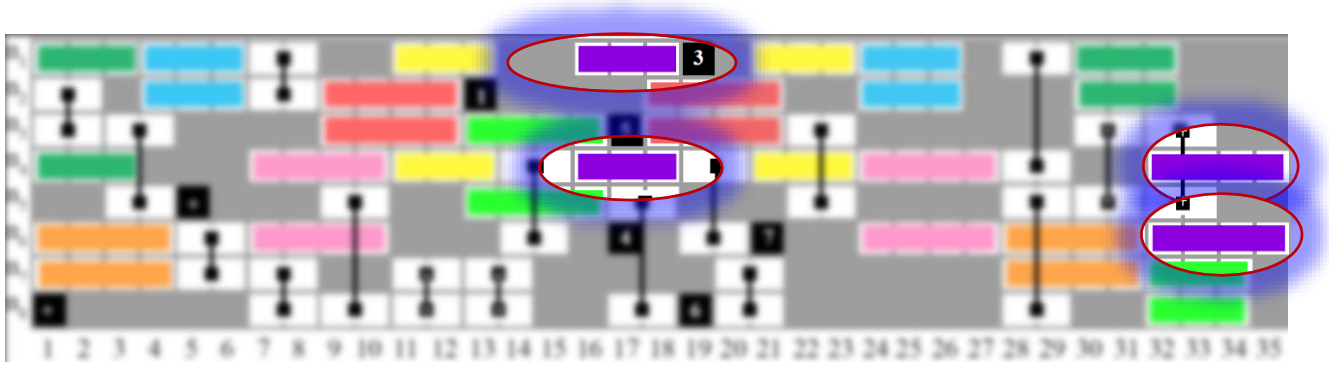


Circuit Execution Schedule

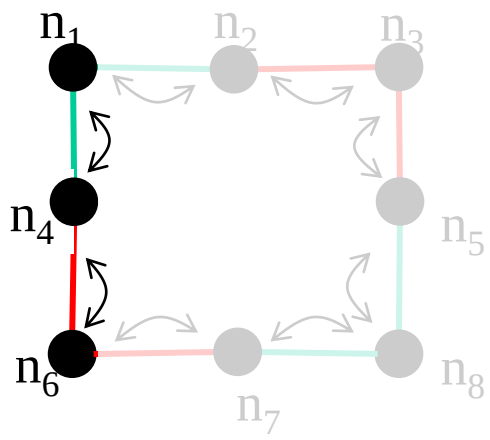


Circuit Execution Schedule





P-S(3,7) is fast on n_1, n_4
P-S(3,7) is slow on n_4, n_6



How to obtain these schedules efficiently?

Classical planning software is useful, and this is an active research field.

Amazon Braket for QAOA

Let’s go to Amazon Braket

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