QAOA aproach to Battery Revenue Optimization Problem

Circuit and Optimizations

Alexander Pagonis

December 10, 2021

Abstract

- Problem definition
- Quantum approach
- Circuit construction
- Improve run-time
- Improve algorithm precision
- Simulate circuit on Qiskit
- Measure efficiency and cost

Battery Revenue Optimization Problem [1]

- Two markets M_1 and M_2 want to rent a battery for a time window of n days.
- At t_{th} day: market 1 offers $\lambda_1^{(t)}$ for the battery, but damages it by $c_1^{(t)}$
- At t_{th} day: market 2 offers $\lambda_2^{(t)}$ for the battery, but damages it by $c_2^{(t)}$
- Goal: What is the maximum obtainable revenue given the battery's endurance (C_{max})?

Battery Revenue Optimization: Importance

- Wide range of applications (Investments, Network packet fragmentation, etc)
- Exact solution: NP-Complete (Reduces easily to 0-1 Knapsack which is NP-Complete)
- Approximate solution (classically): in FPTAS [2] class (Fully Polynomial-Time Approximation Scheme), hard to implement, much restrictive
- Quantum approach: Elegant, Easy implementation, Versatile

Battery Revenue Optimization: Mathematical Formulation

- With z_t denoting our choice for the t_{th} day:

$$z_t = 0 \longrightarrow M_1$$
$$z_t = 1 \longrightarrow M_2$$

- We want to maximize the profit:

$$\underset{\vec{z}\in\{0,1\}^n}{\operatorname{argmax}} \left(\sum_{t=1}^{n} \left[(1-z_t)\lambda_1^{(t)} + z_t\lambda_2^{(t)} \right] \right)$$

- Subject to the constraint:

$$\sum_{t=1}^{n} \left[(1-z_t)c_1^{(t)} + z_t c_2^{(t)} \right] \leq C_{max}$$

Reduction to 0-1 Knapsack: Identify 4 cases

For every day only 4 cases exist:

- If $\lambda_1^{(t)} \geq \lambda_2^{(t)}$ and $c_1^{(t)} \leq c_2^{(t)}$, then choose $M1$
- If $\lambda_2^{(t)} \geq \lambda_1^{(t)}$ and $c_2^{(t)} \leq c_1^{(t)}$, then choose $M2$
- If $\lambda_2^{(t)} \geq \lambda_1^{(t)}$ and $c_2^{(t)} \geq c_1^{(t)}$, then revenue is at least $\lambda_1^{(t)}$ and damage is at least $c_1^{(t)}$.
- If $\lambda_1^{(t)} \geq \lambda_2^{(t)}$ and $c_1^{(t)} \geq c_2^{(t)}$, then revenue is at least $\lambda_2^{(t)}$ and damage is at least $c_2^{(t)}$.

Case 3 simplification

- Cases 1, 2 are *trivial* and thus excluded.
- Cases 3, 4 are *symmetric* so without loss of generality we assume we are always in case 3.
- Hence, the actual question is when do we prefer market M_2 instead of M_1?
- The reduced values are: $r_c = c_2^{(t)} - c_1^{(t)}$, $r_\lambda = \lambda_2^{(t)} - \lambda_1^{(t)}$
- The reduced C_{max} is: $C'_{max} = C_{max} - \sum_t c_1^{(t)}$

$$\sum_{t=1}^{n} \left[(1 - z_t)\lambda_1^{(t)} + z_t \lambda_2^{(t)} \right] \implies \cancelto{constant}{\sum_{t=1}^{n} \lambda_1^{(t)}} + \boxed{\sum_{t=1}^{n} z_t (\lambda_2^{(t)} - \lambda_1^{(t)})}$$

0-1 Knapsack Problem

- This is the equivalent to the *0-1 Knapsack Problem*:
- Given a collection of n items each with value v_t and weight w_t and a bag that holds W_{max} weight, what is the optimum choice of items?
- We want to maximize the profit: $\underset{z_i \in \{0,1\}^n}{\operatorname{argmax}} \left(\sum_{t=1}^{n} z_i v_t \right)$
- Subject to this constraint: $\sum_{t=1}^{n} z_i w_t \leq W_{max}$

Reduction to Relaxed 0-1 Knapsack

- Knapsack is NP-Complete
- But allowing all feasible solutions requires a non-NP approximate algorithm $\longrightarrow$ FPTAS class (still bad)
- This is the *Relaxed 0-1 Knapsack Problem*
- Our goal, then, is to maximize the algorithm's precision
- Classical Approximate solution (FPTAS) [2]:

$$O\left(n\log\left(\frac{1}{\epsilon}\right)+\frac{\left(\frac{1}{\epsilon}\right)^{\frac{9}{4}}}{2^{\Omega\left(\sqrt{\log\left(\frac{1}{\epsilon}\right)}\right)}}\right)$$

Adiabatic Quantum Approach?

- We need an quantum optimization algorithm that converges to solution
- First thought: Adiabatic Computing?
- Idea: Simulate the Hamiltonian: $\hat{H} = (1-t)\hat{H}_i + t\hat{H}_f$
- H_i Initial state $\longrightarrow$ Prepare problem
- H_f Final state $\longrightarrow$ Contains solution
- Let the time flow and then measure!
- But, Adiabatic Computing fails to solve Knapsack! [3]
- Although, its basic idea will help us in choosing parameters

Quantum Approximate Optimization Algorithm

- QAOA [1, 4]: Technique used for a variety of optimization problems
- Optimization Goal: Maximize objective function $f(\vec{z})$
- Corresponds to a repeated bipartite circuit
- First part "calculates" the function $f(\vec{z}) \longrightarrow$ operator C
- Second part mixes between choices $\vec{z} \longrightarrow$ operator B
- Idea: Apply parametric operators $C(\gamma)$ and $B(\beta)$ on $|\vec{z}\rangle$ for many (β, γ) pairs and it will converge to $\arg_{\vec{z}}[max\{f(\vec{z})\}]$

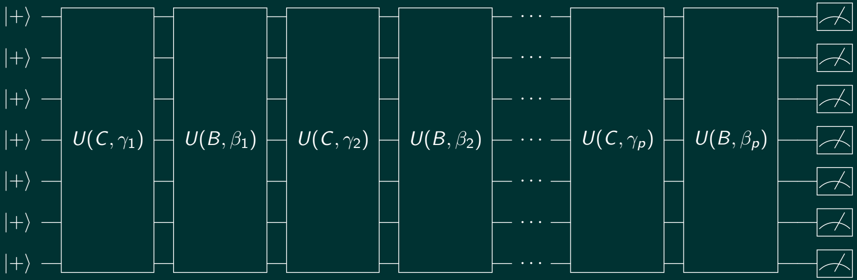
QAOA: Details

- Step-1: Calculate C operator such that: $C\left|\vec{z}\right\rangle = f(\vec{z})\left|\vec{z}\right\rangle$
- Step-2: Construct the corresponding unitary operator: $U(C,\gamma) = e^{-i\gamma C}$, $\gamma \in (0, 2\pi)$ with contribution γ
- Step-3: Use the (default) mixer operator $B = \sum_{t=1}^{n} \sigma_t^x$ where $\sigma_t^x = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ is applied on t_{th} qubit
- Step-4: Construct the unitary operator $U(C,\gamma) = e^{-i\beta B}$, $\beta \in (0, \pi)$ with contribution β
- Initialize Qubits with 50-50 probabilities: $\left|Init\right\rangle = \left|+\right\rangle^{\otimes n}$

QAOA: Expected results

- Trial and Error: C tries, B evaluates.
- We construct the state:
  $|\vec{\beta},\vec{\gamma}\rangle \equiv U(B,\beta_p)U(C,\gamma_p)\cdots U(B,\beta_1)U(C,\gamma_1)\,|+\rangle^{\otimes n}$
- Hence, the expected measurement is:
  $F_p(\vec{\beta},\vec{\gamma}) \equiv \langle\vec{\beta},\vec{\gamma}|\,C\,|\vec{\beta},\vec{\gamma}\rangle$
- **Quantum Adiabatic Theorem** ensures convergence to solution: $\lim_{p\to\infty}\left\{\max_{(\vec{\beta},\vec{\gamma})}\left[F_p(\vec{\beta},\vec{\gamma})\right]\right\} = \max_{\vec{z}\in\{0,1\}^n} C(\vec{z})$
  for "carefully selected" angles $\vec{\beta},\vec{\gamma}$

QAOA: Circuit Overview



State: $|\vec{\beta},\vec{\gamma}\rangle \equiv U(B,\beta_p)U(C,\gamma_p)\cdots U(B,\beta_1)U(C,\gamma_1)\,|+\rangle^{\otimes n}$

QAOA: Angles

The optimum parameters (β, γ) are not known. An extensive search for good angles would mean many measurements, and thus much time.
We choose an adiabatic approach [1, 5, 6] (as proposed above, even though adiabatic computing fails to solve Knapsack [3] and has a very difficult Hamiltonian to implement [5]):

$$\begin{aligned} \beta_k &= 1 - \frac{k}{p}, \quad \beta \overset{k \to p}{\to} 0 \\ \gamma_k &= \frac{k}{p}, \quad \gamma_k \overset{k \to p}{\to} 1 \end{aligned} \tag{1}$$

Architecture: Qubits

- Index qubits: n
- Maximum cost value (in binary representation):

$$d = \left\lceil \log_2 \left(\sum_t max(c_1^{(t)}, c_2^{(t)}) \right) \right\rceil$$

- Flag qubit (for constraint testing):

$$F = \left[cost(z) = \sum_{t=1}^{n} (1 - z_t) c_1^{(t)} + z_t c_2^{(t)} \leq C_{max} \right]_{0|1}$$

Architecture: QAOA Parameters

- Objective function $f(z) = return(z) + penalty(z)$
- $return(\vec{z}) = \sum_{t=1}^{n} \left[(1 - z_t)\lambda_1^{(t)} + z_t\lambda_2^{(t)} \right]$
- $penalty(\vec{z}) = \left\{ \begin{array}{ll} 0, & cost(z) \leq C_{max} \\ -a(cost(z) - C_{max}), & cost(z) > C_{max} \end{array} \right\}$
- As for angles (β, γ), we choose an adiabatic approach (even though adiabatic computing fails to solve Knapsack):

$$\begin{aligned} \beta_k &= 1 - \frac{k}{p}, \quad \beta \overset{k \to p}{\to} 0 \\ \gamma_k &= \frac{k}{p}, \quad \gamma_k \overset{k \to p}{\to} 1 \end{aligned} \tag{2}$$

State manipulation

- Initialization: $|\vec{z}\rangle = |+\rangle^{\otimes n}$
- Whole initial state: $|\vec{z}\rangle \otimes |\vec{0}\rangle \otimes |0\rangle$
- 1) Calculate cost: $|\vec{z}\rangle \otimes |cost(\vec{z})\rangle \otimes |0\rangle$
- 2) Check constraint: $|\vec{z}\rangle \otimes |cost(\vec{z})\rangle \otimes |cost(\vec{z}) \geq C_{max}\rangle$
- 3) Apply penalty: $|\vec{z_p}\rangle \otimes |cost(\vec{z_p})\rangle \otimes |F\rangle$
- 4) Reinitialize: $|\vec{z_p}\rangle \otimes |0\rangle \otimes |0\rangle$
- 5) Apply mix operator
- Repeat stages (1-5) p-times

C operator overview

- $f(z) = return(z) + penalty(z)$
- $U(C,\gamma)\,|\vec{z}\rangle = e^{-i\gamma f(z)}\,|\vec{z}\rangle = e^{-i\gamma.penalty(z)}e^{-i\gamma.return(z)}\,|\vec{z}\rangle$
- Return part:

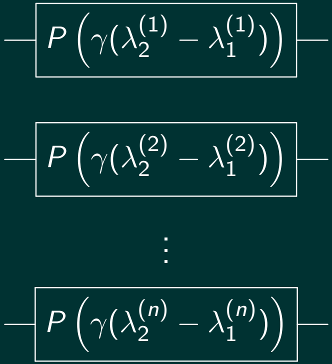
$$e^{-i\gamma.return(z)}\,|\vec{z}\rangle = \left(\prod_{t=1}^{n} e^{-i\gamma.return_t(z)}\right)|\vec{z}\rangle$$

$$= \cancel{e^{i\theta}} \bigotimes_{t=1}^{n} e^{-i\gamma z_t(\lambda_2^{(t)}-\lambda_1^{(t)})}\,|z_t\rangle \qquad (3)$$

$$\text{with } \theta = \sum_{t=1}^{n} \lambda_1^{(t)} = \text{ constant}$$

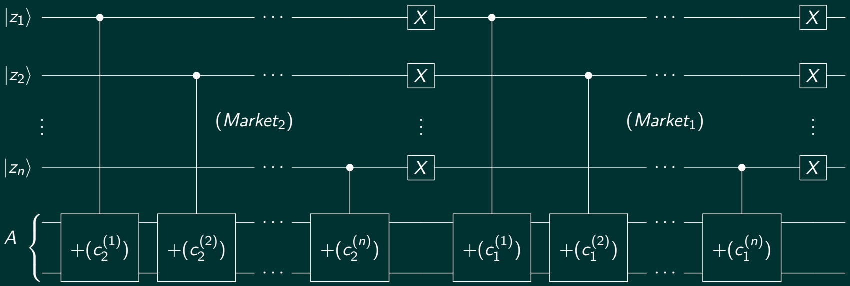
1) Return Part Circuit

- Return part: $e^{-i\gamma.return(z)}\,|\vec{z}\rangle = \bigotimes_{t=1}^{n} e^{-i\gamma z_t(\lambda_2^{(t)}-\lambda_1^{(t)})}\,|z_t\rangle$
- Return part circuit:


$P\left(\gamma(\lambda_2^{(1)} - \lambda_1^{(1)})\right)$
$P\left(\gamma(\lambda_2^{(2)} - \lambda_1^{(2)})\right)$
$\vdots$
$P\left(\gamma(\lambda_2^{(n)} - \lambda_1^{(n)})\right)$

2) Penalty Part: Cost calculation

Calculate $cost(z)$:

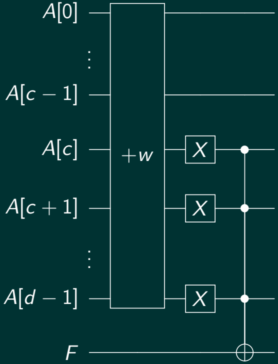


2) Penalty Part: Constraint Checking

Comparing with an arbitrary number is difficult. But, comparing with a power of 2 is an easy process. Adding on both sides leaves the difference invariant:

$$cost(z) \leq C_{max} \quad \overset{+w}{\Longleftrightarrow}$$
$$cost(z) + w \leq C_{max} + w = 2^c$$

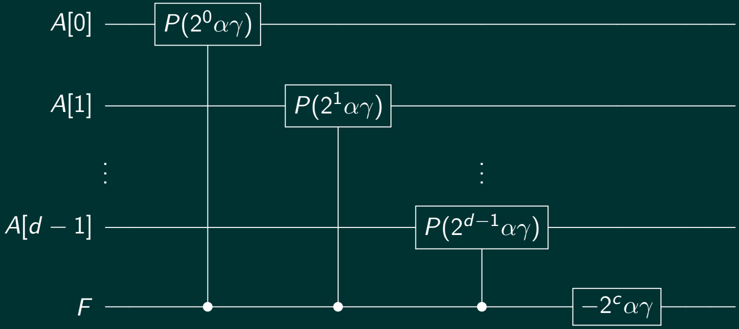
We only need to check the higher power qubits to set the flag



2) Penalty Part: Penalty Dephasing

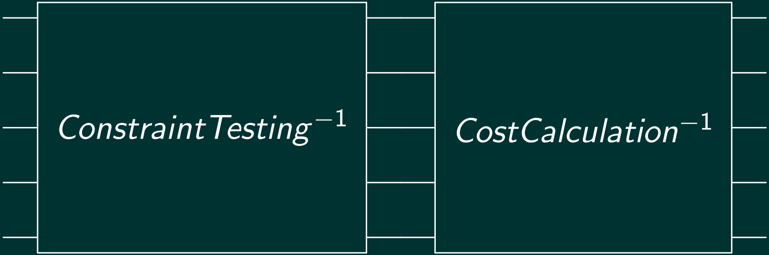
Penalty:

$$a(cost(\vec{z}) - C_{max}) = \sum_{j=0}^{d-1} 2^j aA[j] - 2^c a$$



2) Penalty Part: Reinitialization

Reverse process: Trace back $|cost(\vec{z_p})\rangle$ to $|\vec{z_p}\rangle$

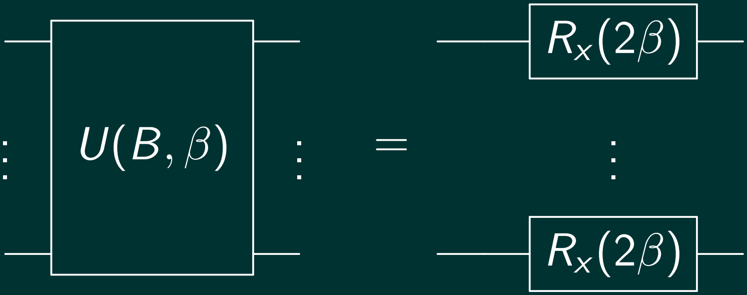


B operator overview

- Mixer Operator: flip qubits by some degree
- σ_t^x is $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ applied on t-th qubit.
- Mixing all qubits: $B = \sum_{t=1}^{n} \sigma_t^x$
- $B = \sigma_1^x \otimes I^{\otimes(n-1)} + I_1 \otimes \sigma_2^x \otimes I^{\otimes(n-2)} + I^{\otimes(n-1)} \otimes \sigma_n^x$
- $[\sigma_i^x \otimes I_j, I_i \otimes \sigma_j^x] = 0 \longrightarrow$ we can compute in parallel
- $U(B, \beta) = e^{-i\beta B} \longrightarrow R_x(2\beta)$ gate on every qubit

B Operator Circuit

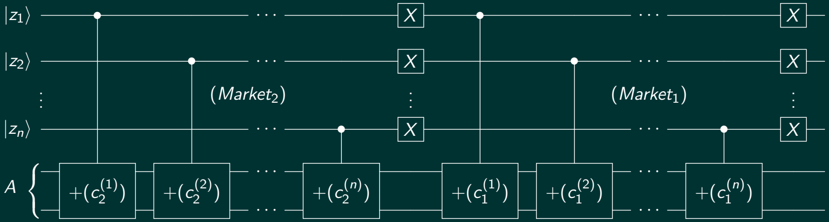
Mix qubits in parallel:
$U(B,\beta) = e^{-i\beta B} \longrightarrow R_x(2\beta))$



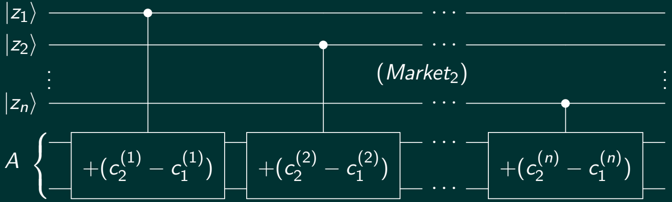
Optimization I: Reduction to 0-1 Knapsack

- In case 3 the actual choice is: when do we prefer market M_2 over M_1? So we get the reduction:
- $v_t = (\lambda_2^{(t)} - \lambda_1^{(t)})$
- $w_t = (c_2^{(t)} - c_1^{(t)})$
- $W_{max} = C_{max} - \sum_{t=1}^{n} c_1^{(t)}$
- 0-1 Knapsack formulation:
- Goal: $\underset{z_t \in \{0,1\}^n}{\text{argmax}} \sum_{t=1}^{n} z_t v_t$
- Constraint: $\sum_{t=1}^{n} z_t w_t \leq W_{max}$

Optimization I: Reduction to 0-1 Knapsack



$\downarrow$Reduction


$|z_1\rangle$
$|z_2\rangle$
$|z_n\rangle$
A
$+(c_2^{(1)} - c_1^{(1)})$
$+(c_2^{(2)} - c_1^{(2)})$
$(Market_2)$
$+(c_2^{(n)} - c_1^{(n)})$

Optimization I: Reduction to 0-1 Knapsack

Not only we get half the cost calculation circuit... we also reduce the qubits needed!
(New) maximum cost value (in binary representation):

$$d = \left\lceil \log_2 \left(\sum_t max(c_1^{(t)}, c_2^{(t)}) \right) \right\rceil \longrightarrow \left\lceil \log_2 \left(\sum_t \left(c_2^{(t)} - c_1^{(t)} \right) \right) \right\rceil$$

Optimization II: QFT Adders

Quantum (binary) adders come in many implementations:

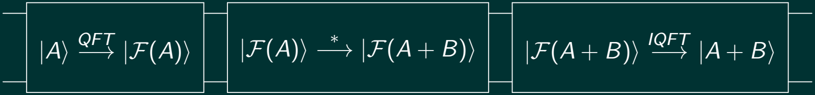
- Plain adder network [7]
- Ripple carry adder [8]
- QFT adder [9, 10]

QFT Adders, in our case, have many advantages. So we choose them.

Optimization II: QFT Adder overview

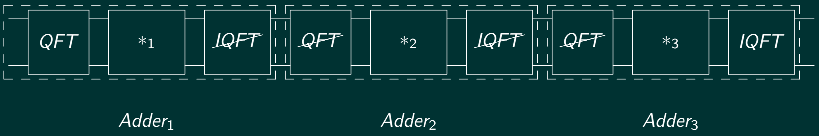
QFT Adder Idea: Add in phase space, where it's simpler.

- $|A\rangle \xrightarrow{QFT} |\mathcal{F}(\mathcal{A})\rangle$
- $|\mathcal{F}(\mathcal{A})\rangle \xrightarrow{phases} |\mathcal{F}(\mathcal{A}+\mathcal{B})\rangle$
- $|\mathcal{F}(\mathcal{A}+\mathcal{B})\rangle \xrightarrow{IQFT} |A+B\rangle$

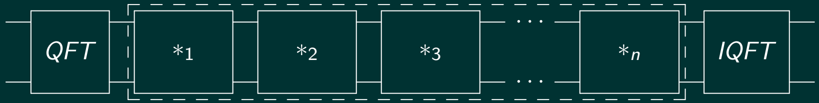

$|A\rangle \xrightarrow{QFT} |\mathcal{F}(A)\rangle$
$|\mathcal{F}(A)\rangle \xrightarrow{*} |\mathcal{F}(A+B)\rangle$
$|\mathcal{F}(A+B)\rangle \xrightarrow{IQFT} |A+B\rangle$

Optimization II: QFT Adder's main advantage

We have additions in series $\implies$ QFT and IQFT only once!



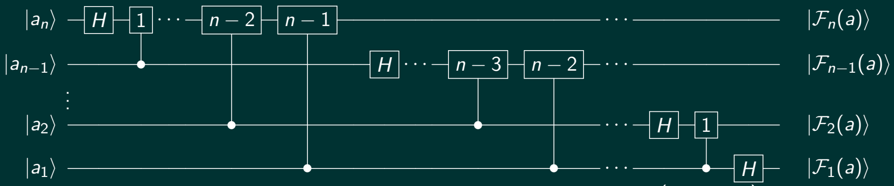
$\Downarrow$



All adders in phase space

Optimization II: QFT Adder circuits

QFT circuit implementation:

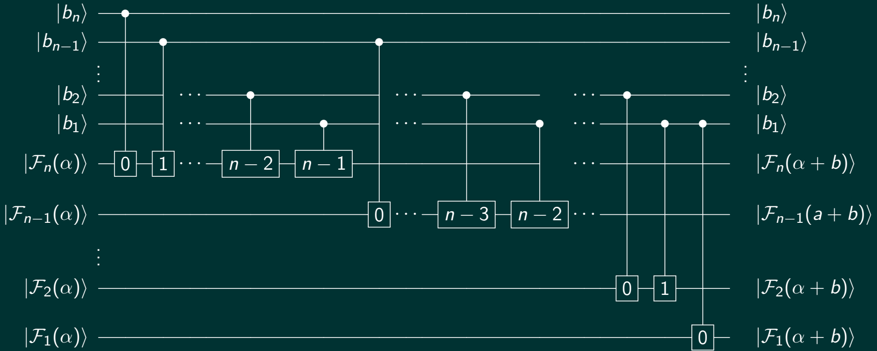


$|a_n\rangle$
$|a_{n-1}\rangle$
$|a_2\rangle$
$|a_1\rangle$
H
1
$n-2$
$n-1$
$n-3$
$|\mathcal{F}_n(a)\rangle$
$|\mathcal{F}_{n-1}(a)\rangle$
$|\mathcal{F}_2(a)\rangle$
$|\mathcal{F}_1(a)\rangle$

where $\boxed{k}$ is the gate corresponding to $P(\frac{\pi}{2^k}) = \begin{pmatrix} 1 & 0 \\ 0 & e^{\frac{i\pi}{2^k}} \end{pmatrix}$.

Optimization II: QFT Adder circuits

Addition in phase-space (circuit):

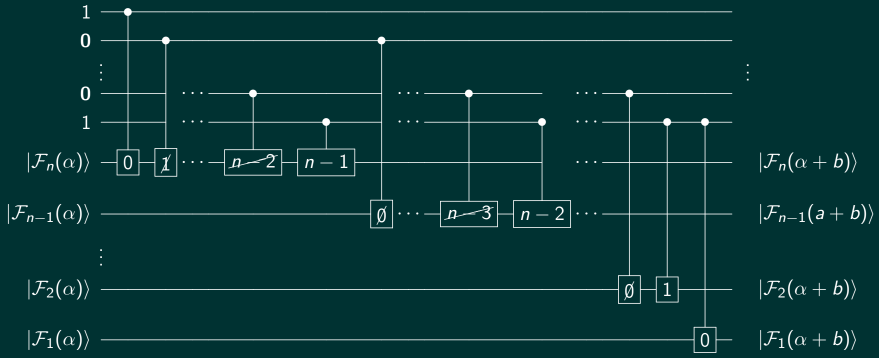


$|b_n\rangle$
$|b_{n-1}\rangle$
$|b_2\rangle$
$|b_1\rangle$
$|\mathcal{F}_n(\alpha)\rangle$
$|\mathcal{F}_{n-1}(\alpha)\rangle$
$|\mathcal{F}_2(\alpha)\rangle$
$|\mathcal{F}_1(\alpha)\rangle$
0
1
$n-2$
$n-1$
0
$n-3$
$n-2$
0
1
0
$|b_n\rangle$
$|b_{n-1}\rangle$
$|b_2\rangle$
$|b_1\rangle$
$|\mathcal{F}_n(\alpha+b)\rangle$
$|\mathcal{F}_{n-1}(a+b)\rangle$
$|\mathcal{F}_2(\alpha+b)\rangle$
$|\mathcal{F}_1(\alpha+b)\rangle$

where $\boxed{k}$ is the gate corresponding to $P(\frac{\pi}{2^k}) = \begin{pmatrix} 1 & 0 \\ 0 & e^{\frac{i\pi}{2^k}} \end{pmatrix}$.

Optimization II: QFT Adder circuits

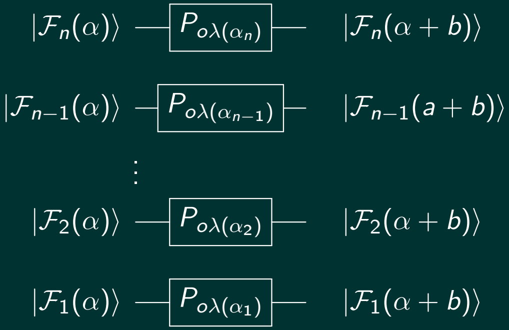
$|B\rangle$ qubits are classical bits, which we know.
When b_i is zero delete gate, when b_i is one keep gate.



1
0
0
1
$|\mathcal{F}_n(\alpha)\rangle$
$|\mathcal{F}_{n-1}(\alpha)\rangle$
$|\mathcal{F}_2(\alpha)\rangle$
$|\mathcal{F}_1(\alpha)\rangle$
0
$\cancel{1}$
$\cancel{n-2}$
$n-1$
$\cancel{0}$
$\cancel{n-3}$
$n-2$
$\cancel{0}$
1
0
$|\mathcal{F}_n(\alpha+b)\rangle$
$|\mathcal{F}_{n-1}(a+b)\rangle$
$|\mathcal{F}_2(\alpha+b)\rangle$
$|\mathcal{F}_1(\alpha+b)\rangle$

Optimization II: QFT Adder phase reduction

But for (uncontrolled) phase gates: $\boxed{P(\varphi)P(\psi) = P(\varphi+\psi)}$
So we reduce into one gate per qubit, containing the whole phase!



$|\mathcal{F}_n(\alpha)\rangle$
$P_{o\lambda(\alpha_n)}$
$|\mathcal{F}_n(\alpha+b)\rangle$
$|\mathcal{F}_{n-1}(\alpha)\rangle$
$P_{o\lambda(\alpha_{n-1})}$
$|\mathcal{F}_{n-1}(a+b)\rangle$
$\vdots$
$|\mathcal{F}_2(\alpha)\rangle$
$P_{o\lambda(\alpha_2)}$
$|\mathcal{F}_2(\alpha+b)\rangle$
$|\mathcal{F}_1(\alpha)\rangle$
$P_{o\lambda(\alpha_1)}$
$|\mathcal{F}_1(\alpha+b)\rangle$

Optimization II: QFT Adder approximation

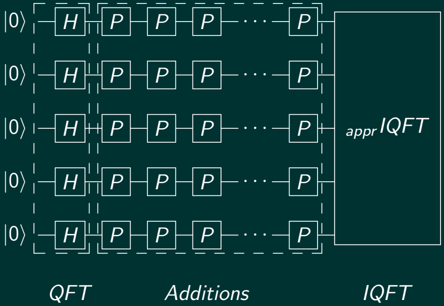
Phase gates $\boxed{k} \longrightarrow P(\frac{\pi}{2^k}) = \begin{pmatrix} 1 & 0 \\ 0 & e^{\frac{i\pi}{2^k}} \end{pmatrix}$ with big k can be ignored!

Approximate QFT circuit is viable for k down to: $k \approx \log_2(n)$ [9, 11].

So QFT circuit complexity reduces: $O(n^2) \longrightarrow \boxed{O(n \log_2 n)}$

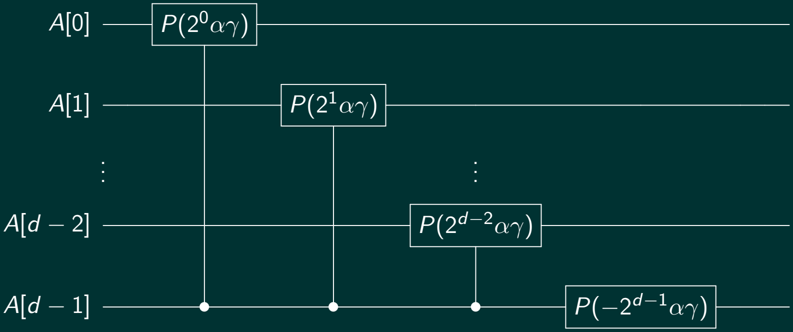
Optimization II: QFT Adder special case

In our case, QFT is applied into the state $|0\rangle$.
So the circuit is equivalent to hadamard gates:



Optimization III: Avoid Flag qubit

We added w into $cost(z)$ to compare with 2^c. But, we can add up to 2^d and avoid the Multi-NOT gate.
(Note: Multi-NOT gate was using $d - c - 3$ ancillary qubits!)
So we change penalty dephasing as well:



Optimization IV: Increase precision

Initial possibility distribution (50/50) is completely arbitrary!
We must find a more data-specific one [12].
Of course, that would change the mixer.
The default mixer (= X gate) has as its eigenstates:

$$\frac{|0\rangle + |1\rangle}{\sqrt{2}}, \quad \frac{|0\rangle - |1\rangle}{\sqrt{2}}$$

That's why we had X gates as mixer for 50-50 distribution

Optimization IV: Mixer and possibilities relation

A mixer must correspond to a possibility distribution:

$$|p_i\rangle := \sqrt{1-p_i}\,|0\rangle + \sqrt{p_i}\,|1\rangle$$
$$|p_i^{\perp}\rangle := -\sqrt{p_i}\,|0\rangle + \sqrt{1-p_i}\,|1\rangle$$
$$|p\rangle = |p_1\rangle \otimes |p_2\rangle \otimes \cdots |p_n\rangle\,.$$

having the eigenstates:

$$\mathbb{X}_{p_i}\,|p_i\rangle = -\,|p_i\rangle$$
$$\mathbb{X}_{p_i}\,|p_i^{\perp}\rangle = +\,|p_i\rangle$$

Optimization IV: Hourglass mixers

This mixer can be written as linear combination of X and Z gates:

$$\begin{aligned} ⧖_{p_i} &= -(1-2p_i)Z - 2\sqrt{p_i(1-p_i)}X \\ &= -\begin{pmatrix} 1-2p_i & 2\sqrt{p_i(1-p_i)} \\ 2\sqrt{p_i(1-p_i)} & 1-2p_i \end{pmatrix} \end{aligned}$$

(Note: Putting X on top of Z gives us the hourglass symbol)
This mixer is a generalization of the default mixer:

$$⧖_0 = -Z, \quad ⧖_{1/2} = -X$$

Optimization IV: Hourglass mixers

$$\mathbb{X}_{p_i} = -\begin{pmatrix} 1-2p_i & 2\sqrt{p_i(1-p_i)} \\ 2\sqrt{p_i(1-p_i)} & 1-2p_i \end{pmatrix} \overset{\varphi_{p_i}=2\sin^{-1}(\sqrt{p_i})}{\Longrightarrow}$$

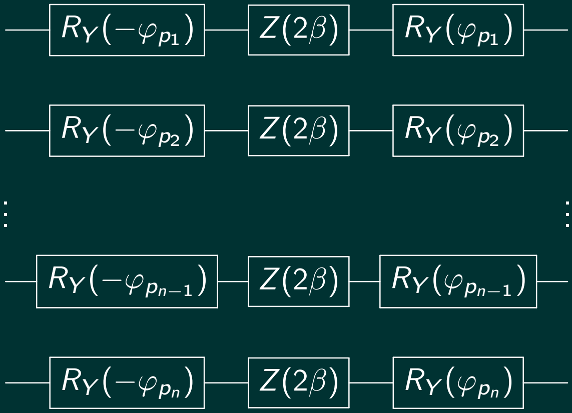
$$\mathbb{X}_{p_i} = -\begin{pmatrix} 1-2\sin^2(\frac{\varphi_{p_i}}{2}) & 2\cos(\frac{\varphi_{p_i}}{2})\sin(\frac{\varphi_{p_i}}{2}) \\ \\ 2\cos(\frac{\varphi_{p_i}}{2})\sin(\frac{\varphi_{p_i}}{2}) & -(1-2\sin^2(\frac{\varphi_{p_i}}{2})) \end{pmatrix} \Longrightarrow$$

$$\mathbb{X}_{p_i} = R_Y(\varphi_{p_i}) Z R_Y(\varphi_{p_i})^\dagger \Longrightarrow$$

$$e^{-i\beta \mathbb{X}_{p_i}} = R_Y(\varphi_{p_i}) e^{-i\beta Z} R_Y(\varphi_{p_i})^\dagger = \boxed{R_Y(\varphi_{p_i}) e^{-i\beta Z} R_Y(-\varphi_{p_i})}$$

Optimization IV: Hourglass mixer circuit

Hence, we construct the circuit:



Optimization IV: Possibility distributions

Now we must find some good possibility distribution.
One Idea: Constant Biased State: we exhaust C_{max}

$$\Pr([Q_i = 1]) = \frac{C_{max}}{\sum_t c_i}$$

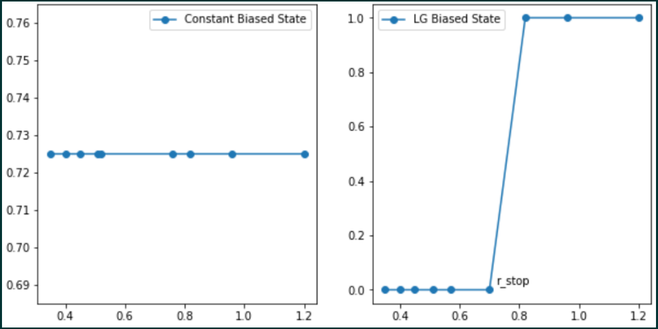
$$E[cost(z)] = \sum_{t=1}^{n} c_i \cdot p_i = \frac{\sum_{t=1}^{n} c_i \cdot C_{max}}{\sum_{t=1}^{n} c_i} = C_{max}$$

Optimization IV: Possibility distributions

Another approach: Mimic Lazy-Greedy algorithm.
Lazy-Greedy: Sort choices by the efficiency ratio $r_i = \frac{\lambda_i}{c_i}$ and chooce the most efficient ones up to C_{max} (With corresponding ratio r_{stop}).
It is easy and very greedy, unlike the constant approach.

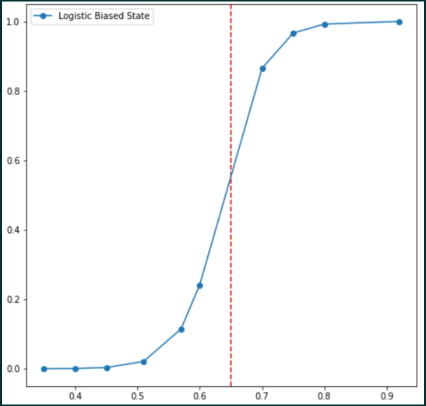
Optimization IV: Distribution combination

The two opposite approaches (constant and completely biased):



Optimization IV: Distribution combination

Combine the two approaches: The constant with the most greedy!



Optimization IV: Distribution combination

Using the Logistic function distribution:

$$p_i = \frac{1}{1 + Ce^{-k(r_i - r_{stop})}}$$

$$C = \frac{\sum c_i}{C_{max}} - 1$$

This logistic function is generalization of both distributions:

$$\lim_{k \to 0} p_i = \text{Constant Biased State}$$
$$\lim_{k \to \infty} p_i = \text{Lazy Greedy State}$$

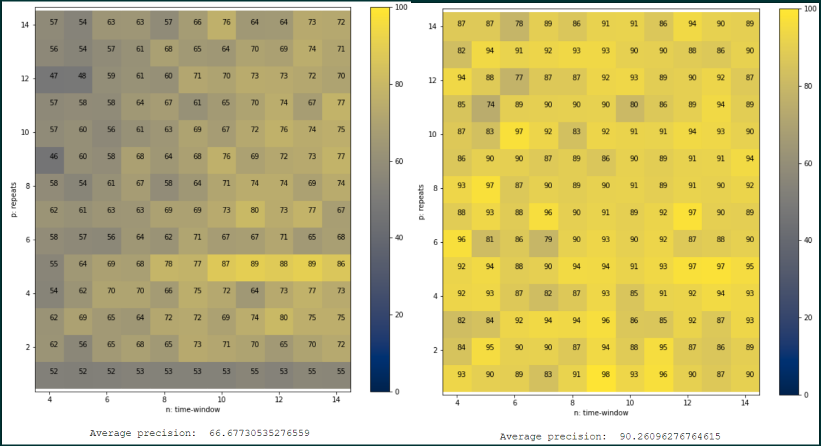
Analytics: Measure efficiency

Precision measure: $\frac{\text{Estimated returns}}{\text{optimum returns}} \in (0,1)$:

$$\frac{\sum_z \left[\left(R(z) - \sum_t \lambda_1^{(t)} \right) H(cost(z) \leq C_{max}) \right]}{N_{feasible} \cdot \left(\lambda_{opt} - \sum_t \lambda_1^{(t)} \right)} \tag{4}$$

where H(x) is the Heaviside step function.

Analytics: Precision comparison



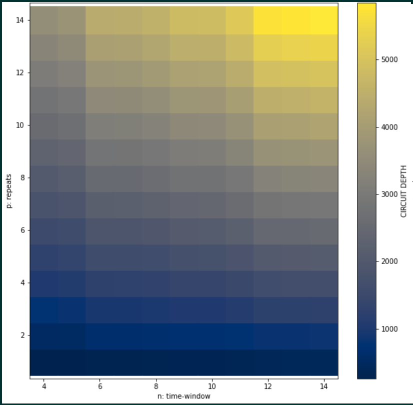
```
Average precision:  66.67730535276559
```

```
Average precision:  90.26096276764615
```

Figure: Precision for distributions $|+\rangle^{\otimes n}$ and **Logistic** ($k = 5$) [100%]

Analytics: Depth and Gates

We transpile the circuit into the basis gates: [rz,sx,cx]



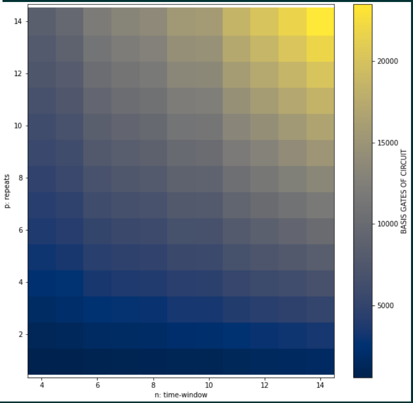


Figure: **Depth** and **basis gates** growing linearly for p and n

Epilogue

QAOA is easy to implement, versatile, efficient, elegant.
We need to improve probability distributions and angles (β, γ).
Also, new mixers considering qubit correlations (preferably cheap ones!)
Parallel additions [1]
Better Adders (?)

End

Thank you for your time!

References I

[1] Pierre Dupuy de la Grand'rive and Jean-Francois Hullo. "Knapsack problem variants of qaoa for battery revenue optimisation". In: *arXiv preprint arXiv:1908.02210* (2019).

[2] Ce Jin. "An improved FPTAS for 0-1 knapsack". In: *arXiv preprint arXiv:1904.09562* (2019).

[3] Lauren Pusey-Nazzaro et al. "Adiabatic quantum optimization fails to solve the knapsack problem". In: *arXiv preprint arXiv:2008.07456* (2020).

[4] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. "A quantum approximate optimization algorithm". In: *arXiv preprint arXiv:1411.4028* (2014).

References II

[5] Mark W Coffey. "Adiabatic quantum computing solution of the knapsack problem". In: *arXiv preprint arXiv:1701.05584* (2017).

[6] Stefan H Sack and Maksym Serbyn. "Quantum annealing initialization of the quantum approximate optimization algorithm". In: *arXiv preprint arXiv:2101.05742* (2021).

[7] Vlatko Vedral, Adriano Barenco, and Artur Ekert. "Quantum networks for elementary arithmetic operations". In: *Physical Review A* 54.1 (1996), p. 147.

[8] Steven A Cuccaro et al. "A new quantum ripple-carry addition circuit". In: *arXiv preprint quant-ph/0410184* (2004).

[9] Thomas G Draper. "Addition on a quantum computer". In: *arXiv preprint quant-ph/0008033* (2000).

References III

[10] Lidia Ruiz-Perez and Juan Carlos Garcia-Escartin. "Quantum arithmetic with the quantum Fourier transform". In: *Quantum Information Processing* 16.6 (2017), p. 152.

[11] Adriano Barenco et al. "Approximate quantum Fourier transform and decoherence". In: *Physical Review A* 54.1 (1996), p. 139.

[12] Wim van Dam et al. "Quantum Optimization Heuristics with an Application to Knapsack Problems". In: *arXiv preprint arXiv:2108.08805* (2021).