QAOA aproach to Battery Revenue Optimization Problem

Circuit and Optimizations

alekospagon
alekospagon@gmail.com

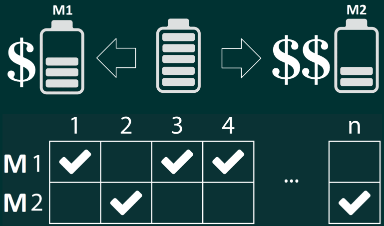
February 22, 2022

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 renters want to use a battery for n days
- They offer $\lambda_1^{(t)}$ and $\lambda_2^{(t)}$ for the t^{th} day
- But they damage the battery for $c_1^{(t)}$ and $c_2^{(t)}$, respectively.
- Which choice maximizes the profit while it prevents the battery's destruction?



Battery Revenue Optimization: Importance

- Wide range of applications (Investments, Network packet fragmentation, etc)
- Exact solution: NP-Complete
- Approximate solution (classically): in FPTAS class [2]

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:

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

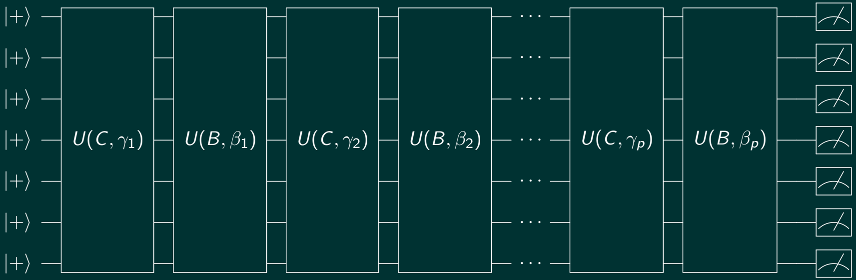
- Subject to the **constraint**:

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

Quantum Approximate Optimization Algorithm

- QAOA [1, 3]: Approximation Scheme
- Goal: Maximize function $f(\vec{z})$
- Calculate C operator such that: $\boxed{C\,|\vec{z}\rangle = f(\vec{z})\,|\vec{z}\rangle}$
- We construct the state:
  $|\vec{\beta},\vec{\gamma}\rangle \equiv U(B,\beta_p)U(C,\gamma_p)\cdots U(B,\beta_1)U(C,\gamma_1)\,|+\rangle^{\otimes n}$
- **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: $|\vec{\beta},\vec{\gamma}\rangle \equiv U(B,\beta_p)U(C,\gamma_p)\cdots U(B,\beta_1)U(C,\gamma_1)\,|+\rangle^{\otimes n}$

Architecture: Qubits

- Index qubits: n
- Cost Value Qubits:

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

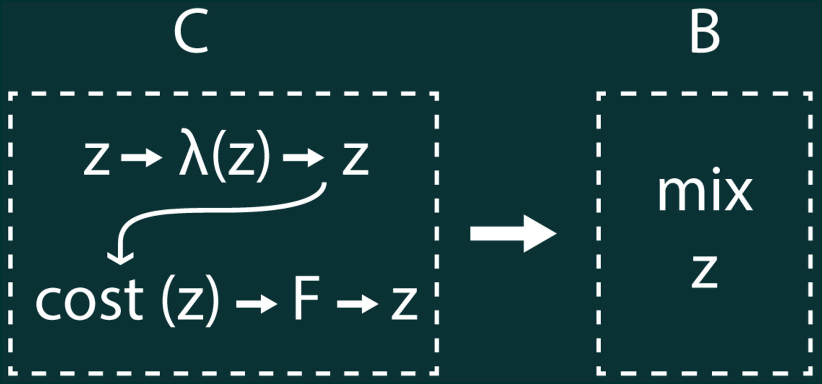
- Flag qubit (for constraint testing): F

Architecture: QAOA Parameters

- Objective function $\boxed{f(z) = return(z) + penalty(z)}$
- $return(\vec{z}) = \sum_{t=1}^{n} \left[(1 - z_t)\lambda_1^{(t)} + z_t \lambda_2^{(t)} \right]$
- $penalty(\vec{z}) = \left\{ \begin{array}{ll} 0, & cost(z) \leq C_{max} \\ -a(cost(z) - C_{max}), & cost(z) > C_{max} \end{array} \right\}$

State manipulation

Initialize and then repeat this 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:

$$-\boxed{P\left(\gamma(\lambda_2^{(1)} - \lambda_1^{(1)})\right)}-$$

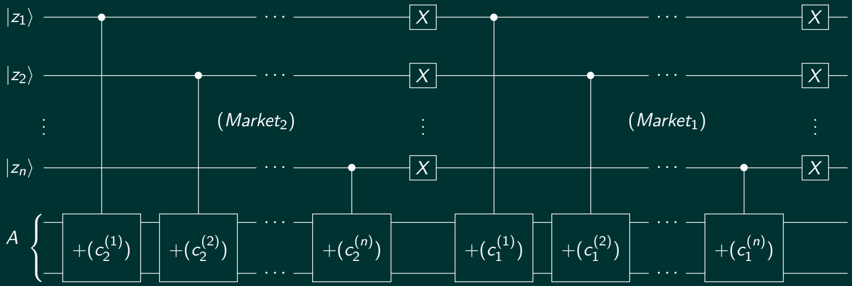
$$-\boxed{P\left(\gamma(\lambda_2^{(2)} - \lambda_1^{(2)})\right)}-$$

$$\vdots$$

$$-\boxed{P\left(\gamma(\lambda_2^{(n)} - \lambda_1^{(n)})\right)}-$$

2) Penalty Part: Cost calculation

Calculate $cost(z)$:

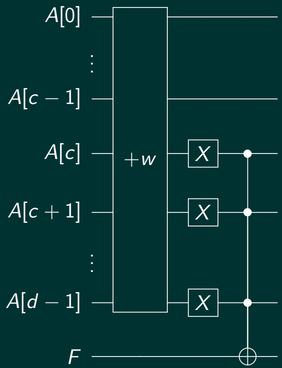


2) Penalty Part: Constraint Checking

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:

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

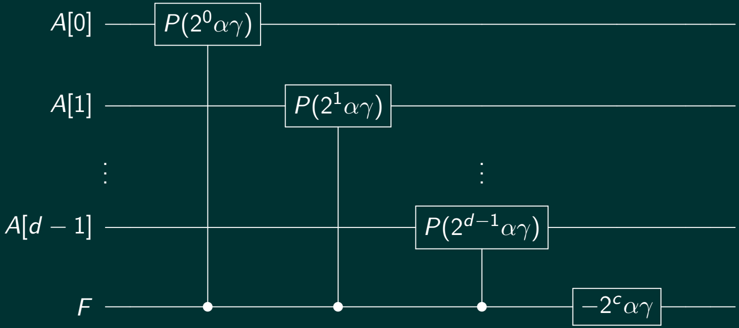
We only need to check the higher power qubits to set the flag



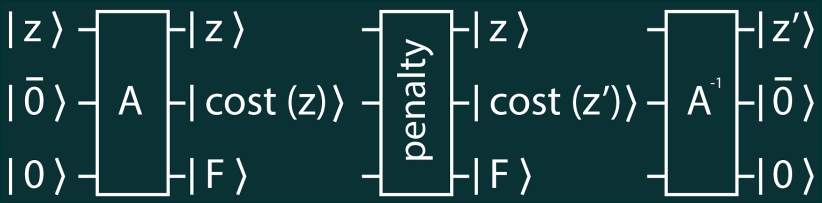
2) Penalty Part: Penalty Dephasing

Penalty:

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



2) Penalty Part: Reinitialization

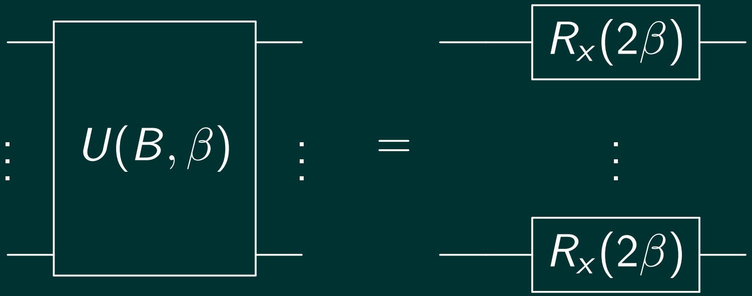


B operator overview

- 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: $B = \sum_{t=1}^{n} \sigma_t^x$
- $U(B, \beta) = e^{-i\beta B} \longrightarrow R_x(2\beta)$ gate on every qubit

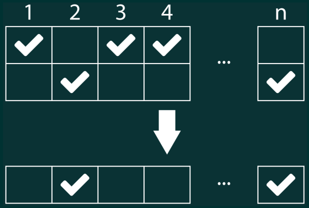
B Operator Circuit

Mix qubits in parallel:



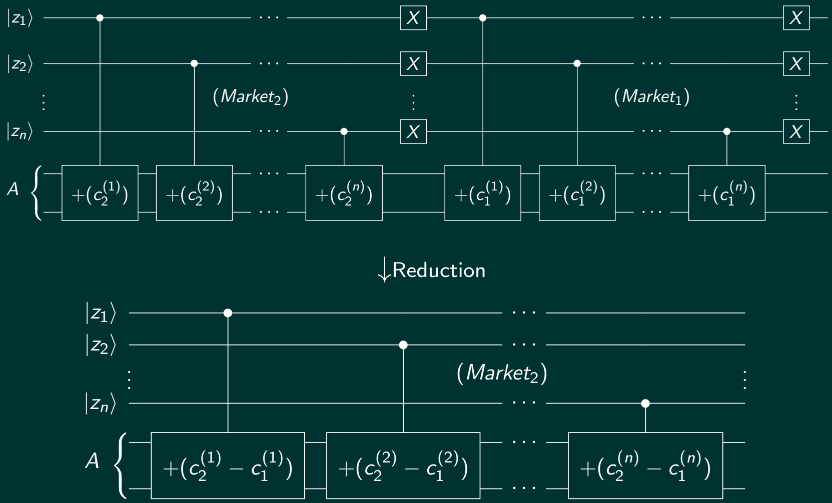
Optimization I: Reduction to 0-1 Knapsack

- Actual goal: when do we prefer market M_2 over M_1? So we get the reduction:
- $v_t = (\lambda_2^{(t)} - \lambda_1^{(t)})$
- $w_t = (c_2^{(t)} - c_1^{(t)})$
- $W_{max} = C_{max} - \sum_{t=1}^{n} c_1^{(t)}$



- Goal: $\underset{z_t \in \{0,1\}^n}{\operatorname{argmax}} \sum_{t=1}^{n} z_t v_t$
- Constraint: $\sum_{t=1}^{n} z_t w_t \leq W_{max}$

Optimization I: Reduction to 0-1 Knapsack



Optimization II: QFT Adders

Quantum (binary) adders come in many implementations:

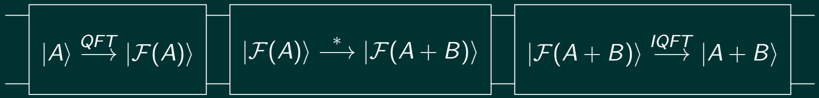
- Plain adder network [4]
- Ripple carry adder [5]
- QFT adder [6, 7]

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

Optimization II: QFT Adder overview

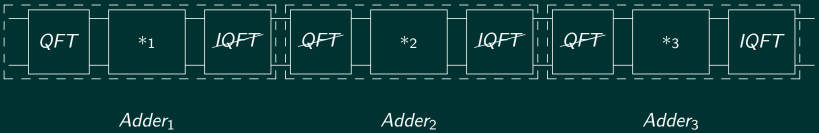
QFT Adder Idea: Add in phase space, where it's simpler.

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

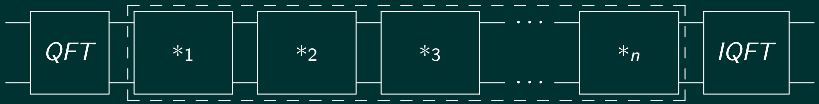


Optimization II: QFT Adder's main advantage

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



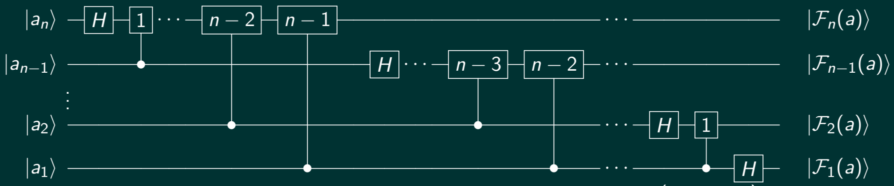
$\Downarrow$



All adders in phase space

Optimization II: QFT Adder circuits

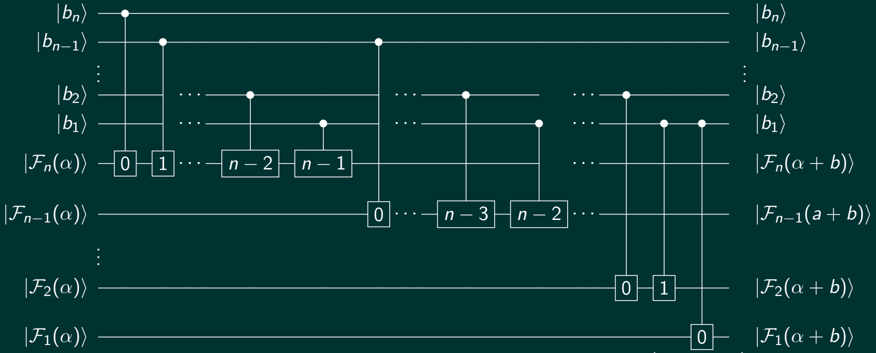
QFT circuit implementation:



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

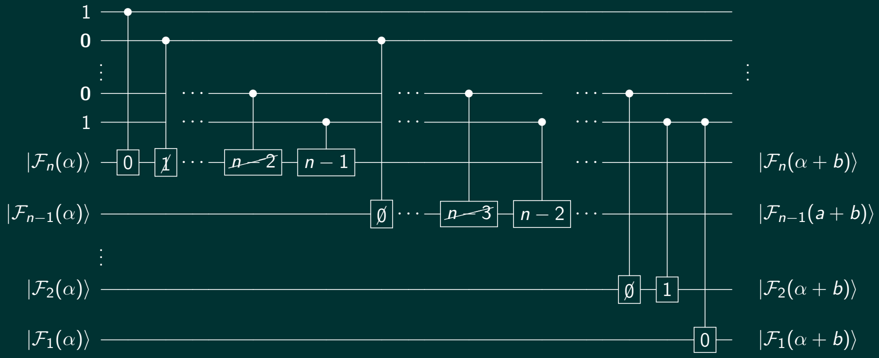
Addition in phase-space (circuit):



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.



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!

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

Optimization II: QFT Adder approximation

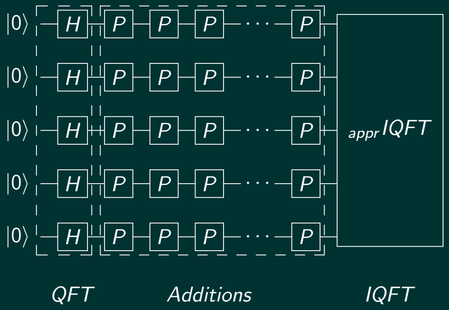
Phase gates $\boxed{k} \longrightarrow P(\frac{\pi}{2^k}) = \begin{pmatrix} 1 & 0 \\ 0 & e^{\frac{i\pi}{2^k}} \end{pmatrix}$ with big k can be ignored!

Approximate QFT circuit is viable for k down to: $k \approx \log_2(n)$ [6, 8].

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

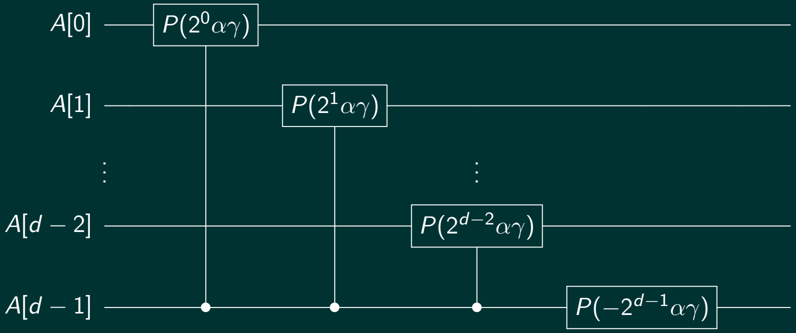
Optimization II: QFT Adder special case

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 [9].
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:

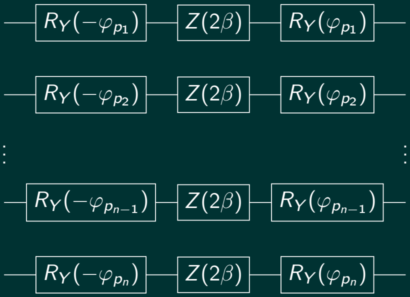
$$
\begin{aligned}
|p_i\rangle &:= \sqrt{1-p_i}\,|0\rangle + \sqrt{p_i}\,|1\rangle \\
|p_i^{\perp}\rangle &:= -\sqrt{p_i}\,|0\rangle + \sqrt{1-p_i}\,|1\rangle \\
|p\rangle &= |p_1\rangle \otimes |p_2\rangle \otimes \cdots |p_n\rangle \,.
\end{aligned}
$$

having the eigenstates:

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

Optimization IV: Hourglass mixers

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



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

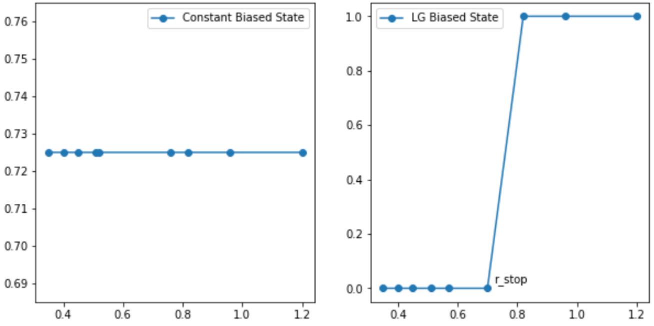
Another approach: Mimic Lazy-Greedy algorithm.

Lazy-Greedy: Sort choices by the efficiency ratio $\boxed{r_i = \frac{\lambda_i}{c_i}}$ and choose the most efficient ones up to C_{max} (With corresponding ratio r_{stop}).

It is easy and very greedy, unlike the constant approach.

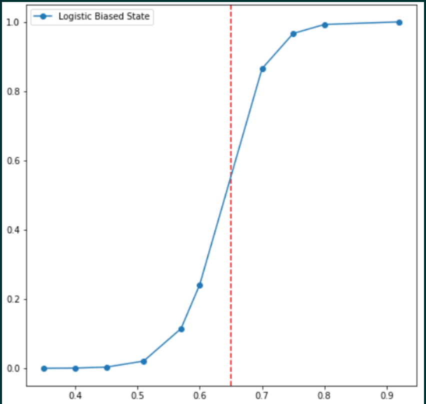
Optimization IV: Distribution combination

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:

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

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

Analytics: Measure efficiency

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

$$\frac{\sum_z \left[\left(R(z) - \sum_t \lambda_1^{(t)} \right) H(cost(z) \leq C_{max}) \right]}{N_{feasible} \cdot \left(\lambda_{opt} - \sum_t \lambda_1^{(t)} \right)} \tag{1}$$

where H(x) is the Heaviside step function.

Analytics: Precision comparison

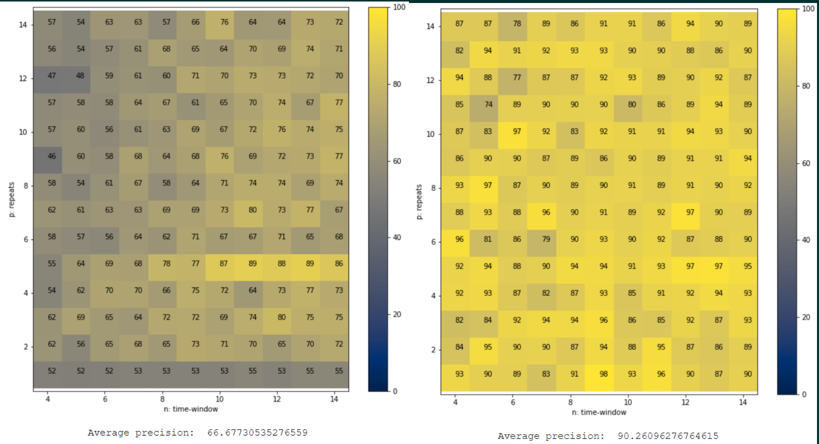


Figure: Precision for distributions $|+\rangle^{\otimes n}$ and **Logistic** ($k = 5$) [100%]

Analytics: Depth and Gates

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

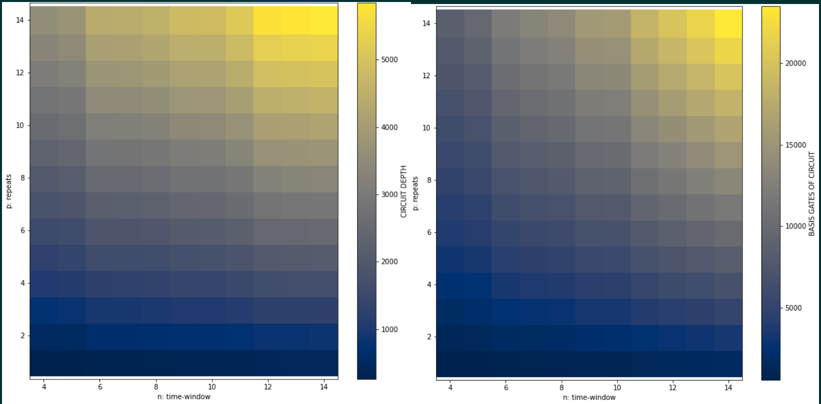


Figure: **Depth** and **basis gates** growing linearly for p and n

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

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

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

References II

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

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

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

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