A QCNN for Quantum State Preparation

Carnegie Vacation Scholarship

David Amorim

Weeks 7-8
(12/08/2024 - 23/08/2024)

Aims for the Week

The following aims were set at the last meeting (14/08/2024):

New Phase Encoding Approach

Investigate a new approach to phase encoding using linear piecewise phase functions without explicit function evaluation.

Handover

Hand over the slides, documentation, code and the poster for the Carnegie Trust.

Table of Contents

Preliminaries

- Consider an n-qubit register with computational basis states $|j\rangle = |j_0 j_1 ... j_{n-1}\rangle$ representing n-bit strings
- Let p of the register qubits be precision qubits so that

$$j = \sum_{k=0}^{n-1} j_k 2^{k-p} \tag{1}$$

- Consider a phase function Ψ over the domain $\mathcal{D} = \{j\}$ and construct an M-fold partition ($M = 2^m$, $m \leq n \in \mathbb{N}$) into equal sub-domains $\mathcal{D}_u$:

$$\mathcal{D} = \bigcup_{u=1}^{M} \mathcal{D}_u, \quad \mathcal{D}_u \cap \mathcal{D}_v = \emptyset, \quad |\mathcal{D}_u| = |\mathcal{D}_v| \tag{2}$$

- On each sub-domain, approximate Ψ using a linear function:

$$\Psi(j) = \alpha_u j + \beta_u, \quad j \in \mathcal{D}_u \tag{3}$$

Preliminaries

Aim

Construct an appropriate operator to transform

$$|j\rangle \mapsto e^{i\Psi(j)}\,|j\rangle \tag{4}$$

via the linear piecewise approximation

$$|j\rangle \mapsto e^{i(\alpha_u j+\beta_u)}\,|j\rangle \quad (j \in \mathcal{D}_u) \tag{5}$$

General Remarks

- The 2^m pairs of coefficients (α_u, β_u) require 2^m independent operators $\hat{O}_u$ to implement
- Each operator $\hat{O}_u$ will generally involve n controlled rotations
- The choice of $\hat{O}_u$ will generally involve all m control qubits
- Thus, it seems like a $\sim \mathcal{O}(2^m nm)$ complexity is to be expected for this algorithm [NO! A N-CONTROLLED OPERATION HAS A COST HIGHER THAN N!!] =¿ READ PAPERS (BOOKMARKED!): Lemma 7.5 / Cor 7.6 Barenco ' linear approximation: Lemma 7.8 Barenco ; linear with ancilla ?! Corollary 7.12
-
- While recursion cannot reduce the required number of independent operators it could simplify circuit structure

Phase Encoding within a Sub-domain

Aim 1

For $j \in \mathcal{D}_u$ construct an operator $\hat{O}_u$ such that $|j\rangle \mapsto e^{i(\alpha_u j + \beta_u)} |j\rangle$.

- Consider the single-qubit operators

$$\hat{P}^{(k)}(\varphi) = \begin{pmatrix} e^{i\varphi} & 0 \\ 0 & e^{i\varphi} \end{pmatrix}, \quad \hat{R}^{(k)}(\varphi) = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\varphi} \end{pmatrix} \tag{6}$$

  each acting on the kth qubit

- Then

$$\hat{O}_u \equiv \bigotimes_{k=0}^{n-1} \hat{P}^{(k)}(\beta_u/n) \hat{R}^{(k)} \left(\alpha_u 2^{k-p}\right) \tag{7}$$

  transforms

$$|j\rangle \mapsto \exp\left[i \left(\sum_{k=0}^{n-1} \alpha_u j_k 2^{k-p} + \beta_u \right) \right] |j\rangle = e^{i(\alpha_u j + \beta_u)} |j\rangle \tag{8}$$

Selecting the Subdomain

- It is straight-forward to construct $\hat{O}_u$ for each of the sub-domains $\mathcal{D}_u$
- More challenging is applying the correct $\hat{O}_u$ based on the sub-domain corresponding to each $|j\rangle$

Aim 2

Construct a system of controls such that $\hat{O}_u$ is applied to $|j\rangle$ if and only if $j \in \mathcal{D}_u$

Sample Case: $M = 2$

- Start with the simplest possible case, a 2-fold partition ($M = 2$):

$$j \in \begin{cases} \mathcal{D}_1 & j_0 = 0 \\ \mathcal{D}_2 & j_0 = 1 \end{cases} \qquad (9)$$

- Using an ancilla qubit, $\hat{O}_1$ is applied for $j \in \mathcal{D}_1$ and $\hat{O}_2$ for $j \in \mathcal{D}_2$
- The ancilla is required since the operation applied to $|j_0\rangle$ is conditional on $|j_0\rangle$ itself

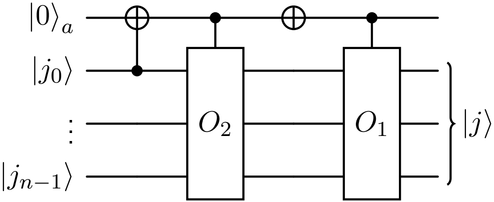

Figure 1: Circuit diagram for $M = 2$

Generalising the Approach

- The approach shown on the previous slide requires $1 \leq \log_2 M \leq n$ ancilla qubits
- The number of controls required is $\mathcal{O}(M \log Mn)$ as there are M operators, each controlled by all ancillas and each involving n controlled rotations
- For $M \sim 2^n$ the gate cost is exponential

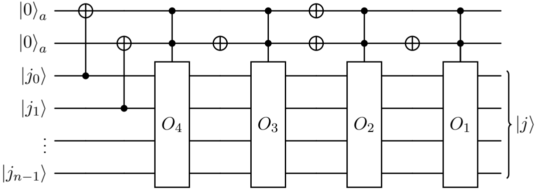

Figure 2: Circuit diagram for $M = 4$. Note that $j \in \mathcal{D}_u$ if $j_0 j_1 = u - 1$ (e.g. $j \in \mathcal{D}_3$ if $j_0 j_1 = 10$).

A Recursive Approach

- Since $M = 2^m$ for some $m \leq n \in \mathbb{N}$ we can view the partition of $\mathcal{D}$ as a recursive process, splitting the domain into halves m times
- Associate with each control qubit, g, two operators, $\hat{O}_0^{(g)}$ and $\hat{O}_1^{(g)}$ EACH ACTING ON THE REMAINING QUBITS! (CASCADE)
- Is it possible to construct

$$\hat{O}_u = \prod_{g=1}^{m} \hat{O}^{(g)} ?? \tag{10}$$

- DEFINE THE APPROPRIATE O AND U OPERATORS !!!

A Recursive Approach

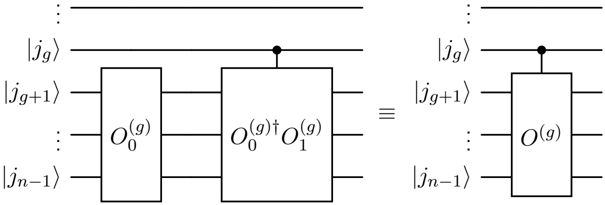

Figure 3: The controlled operator $\hat{O}^{(g)}$ $(0 \leq g \leq m-1)$

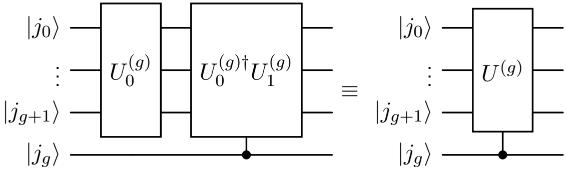

Figure 4: The controlled operator $\hat{U}^{(g)}$ $(1 \leq g \leq m-1)$

A Recursive Approach

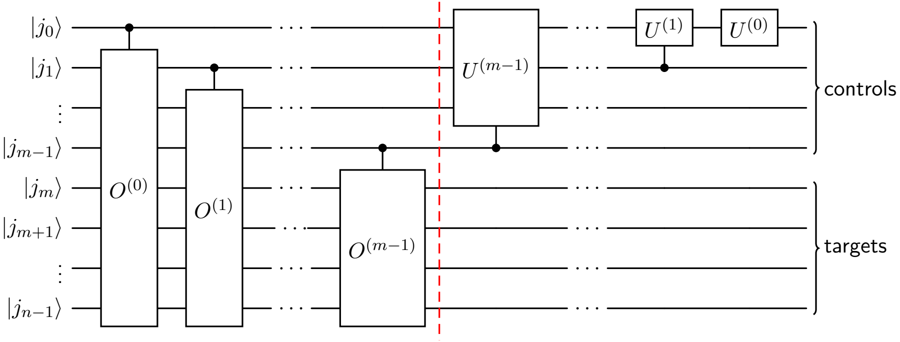

Figure 5: Cascaded controls: $2m - 1$ controlled operations

A Recursive Approach

- Consider

$$\hat{U}_u^{(k)} \equiv \hat{P}^{(k)}(\beta_u/n)\hat{R}^{(k)}(\alpha_u 2^{k-p}) \tag{11}$$

  and define

$$\hat{V}_u^{(k)} = \hat{U}_u^{(k)} \prod_{q=1}^{u-1} \hat{U}_q^{(k)\dagger} \tag{12}$$

  which is equivalent to a single rotation gate
- Use the notation from Barenco 1995 to denote a single-qubit operation $\hat{V}$ controlled by m qubits $\wedge_m(\hat{V})$

The Case $m = 2$ $(M = 4)$

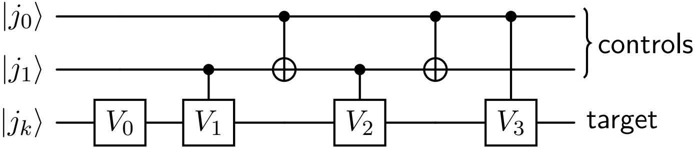

Figure 6: Control structure for $m = 2$ $(M = 4)$ with $2 \leq k < n$. The number of controlled operations is $2^{m+1} - 3 = 5$.

$(j_0 j_1)$	Operation	Equiv. $\hat{U}$	$(j_0 j_1)$	Operation	Equiv. $\hat{U}$
(00)	$\hat{V}_0$	$\hat{U}_0$	(10)	$\hat{V}_0\hat{V}_2\hat{V}_3$	$\hat{U}_2$
(01)	$\hat{V}_0\hat{V}_1\hat{V}_2$	$\hat{U}_1$	(11)	$\hat{V}_0\hat{V}_1\hat{V}_3$	$\hat{U}_3$

Table 1: Operations applied to $|j_k\rangle$ for various control states ORDER REVERSED!!

The Case $m = 3$ $(M = 8)$

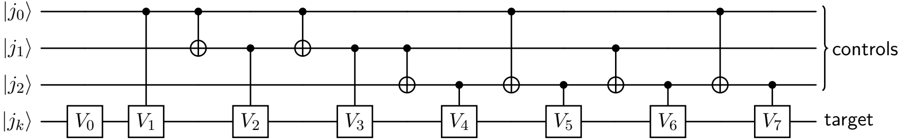

Figure 7: Control structure for $m = 3$ $(M = 8)$ with $3 \leq k < n$. The number of controlled operations is $2^{m+1} - 3 = 13$.

$(j_0j_1j_2)$	Operation	Equiv. $\hat{U}$	$(j_0j_1j_2)$	Operation	Equiv. $\hat{U}$
(000)	$\hat{V}_0$	$\hat{U}_0$	(100)	$\hat{V}_0\hat{V}_1\hat{V}_2\hat{V}_5\hat{V}_6$	$\hat{U}_4$
(001)	$\hat{V}_0\hat{V}_4\hat{V}_7$	$\hat{U}_1$	(101)	$\hat{V}_0\hat{V}_1\hat{V}_2\hat{V}_4\hat{V}_7$	$\hat{U}_5$
(010)	$\hat{V}_0\hat{V}_2\hat{V}_4\hat{V}_5$	$\hat{U}_2$	(110)	$\hat{V}_0\hat{V}_1\hat{V}_3\hat{V}_4\hat{V}_6$	$\hat{U}_6$
(011)	$\hat{V}_0\hat{V}_2\hat{V}_3\hat{V}_6\hat{V}_7$	$\hat{U}_3$	(111)	$\hat{V}_0\hat{V}_1\hat{V}_3\hat{V}_5\hat{V}_7$	$\hat{U}_7$

Table 2: Operations applied to $|j_k\rangle$ for various control states

- The control structure required to apply the appropriate $\hat{U}_u^{(k)}$ to the k-th target qubit requires $2^{m+1} - 3$ CX gates
- As there are $n - m$ target qubits this brings the CX count to $(n - m)(2^{m+1} - 3)$
- Note that the $\hat{U}_u^{(k)}$ are not directly implemented but rather are defined as the product of $\hat{V}_q^{(k)}$
- Since a product of rotation operators corresponds to a sum of rotation angles, this defines a linear system of 2^m equations (one for each $\hat{U}_u^{(k)}$) involving 2^m variables (the $\hat{V}_q^{(k)}$) and hence the $\hat{V}_q^{(k)}$ can be constructed from the $\hat{U}_u^{(k)}$
- Since an m-controlled operation requires m^2 (or m with ancillae) CX these will be avoided (using single-controlled operations only!) as will have 2^m directly !
- Now think about how to deal with controls themselves...

Comparison

- The current implementation uses $n_l \equiv m$ label qubits (with $2^{n_l} = M$) as well as n_c coefficient qubits
- The label operation (as described in Häner 2018), which initialises the label register into the state $|u\rangle$ if $j \in \mathcal{D}_u$ has a CX cost of
$$C_{\mathsf{Label}}(n, n_l) = \mathcal{O}(2^{n_l} n) \tag{13}$$
- The addition and multiplication operations, carried out via QFTs and requiring n_c qubits to store coefficient values, have gate costs of
$$C_{\mathsf{Add}}(n, n_c) = C_{\mathsf{Mult}}(n, n_c) = \mathcal{O}(n^2 + n_c^2) \tag{14}$$
- Loading the function coefficients into the register further comes with a cost of
$$C_{\mathsf{Load}}(n_c, n_l) = \mathcal{O}(n_c 2^{n_l} n_l^2) \tag{15}$$
with the n_l^2 term originating from n_l-controlled X gates
- Thus, in total the LPF gate cost is
$$C_{\mathsf{Total}}(n, n_l, n_c) = \mathcal{O}(n^2 + 2^{n_l}[n + n_l^2 n_c] + n_c^2) \tag{16}$$

Handling the Controls

- Requires a single ancilla
- To encode the phase on the lth control, apply a CX gate to the ancilla, controlled by $|j_l\rangle$; use the same control structure as before but with the ancilla as target ; at the end of the control circuit apply a SWAP between ancilla and $|j_l\rangle$; finally apply a CX on the ancilla, controlled by $|j_l\rangle$ to reset the ancilla to zero
- Thus, to encode the phase on each control requires the same control structure as before with an additional 5 CX gate (3 in SWAP)
- Thus, the m controls require $m(2^{m+2})$ CX giving the total algorithm a CX cost of

$$(n-m)(2^{m+1}-3)+m2^{m+2} = 2^{m+1}(n+m)-3(n-m) = n(2^{m+1}-3)+m \tag{17}$$

- Hence, for $n \ll m$ the CX gate cost is $\mathcal{O}(n2^m)$
- This is a quadratic speed-up in n as well as a significant reduction in the number of ancillae
- Note that cascading the above to all control qubits results in situations where a qubit with non zero phase acts as a control for a

Comments Sarah

For the other problem, yes this is exactly what I had in mind, this is the right idea. A couple of suggestions:

- I think that you can get rid of the ancillae entirely, and just condition directly on the first few (log M) qubits.
- I agree that you should be able to simplify a bit more by thinking about applying the operators recursively.
- Sadly it will always scale exponentially if we take $M \sim O(2^n)$. It would be helpful to consider breaking the circuit (conceptually, not literally...) into m = log M qubits (which will be the controls) and n-m remaining qubits (the targets). If m $\ll$ n, how does the complexity scale with m and n?
- Finally, how does the complexity of applying the phase in this way compare to the previous method of calculating the phase in an ancilla register, applying the phase, and then uncomputing the result in the ancilla.

Table of Contents

1 Phase Encoding

2 Handover

Handover

The code, documentation, slides, and poster are all available on GitHub:

`https://github.com/david-f-amorim/PQC_function_evaluation`

- The source code is found in the directory `pqcprep`
- The slides and poster are found in the directory `slides`
- The documentation is hosted externally here, which is also linked on GitHub